Specification Volume 0

# SPECIFICATION OF THE BLUETOOTH SYSTEM

Wireless Connections Made Easy



# Master Table of Contents & Compliance Requirements

**Covered Core Package version:**
**2.1 + EDR**
**Current Master TOC issued:**
**26 July 2007**

Bluetooth®



LifeScan, Inc. Ex. 1019
Page 1 of 1420
IPR2025-00103

## Revision History

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

## Contributors

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

## Web Site

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

## Disclaimer and Copyright Notice

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

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

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.



LifeScan, Inc. Ex. 1019
Page 2 of 1420
IPR2025-00103

THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

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

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

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

Copyright © 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007

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

*Third-party brands and names are the property of their respective owners.

LifeScan, Inc. Ex. 1019
Page 4 of 1420
IPR2025-00103

# MASTER TABLE OF CONTENTS

***This table of contents (TOC) covers the entire Bluetooth Specification.***
***In addition each volume has a TOC and each part of a volume is preceded by a detailed TOC.***



LifeScan, Inc. Ex. 1019
Page 5 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 6 of 1420
IPR2025-00103

# THE BLUETOOTH SPECIFICATION MASTER TABLE OF CONTENTS

In the following Master Table of Contents:

- The TOC for each Volume starts at the top of a page.
- The Volume number in black is followed by the Volume name in red.

**Note:** Each Volume is a self contained book which is published and updated separately and is equipped with a TOC of its own. However, this Master TOC is also revised as soon as any of the other Volumes are updated.

- A Volume contains one or more Parts (A, B, etc.); each Part can be viewed independently and has its own TOC.

Red or blue text on the following pages indicates hypertext links that take you directly to the indicated section, on condition that you have access to a complete specification.



LifeScan, Inc. Ex. 1019
Page 7 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 8 of 1420
IPR2025-00103

*Specification Volume 0*
**Master Table of Contents & Compliance Requirements**

**Part A**
**MASTER TABLE OF CONTENTS**

**The Bluetooth Specification Master Table of Contents .....7**

**Part B**
**BLUETOOTH COMPLIANCE REQUIREMENTS**

**Contents .....33**
**1 Introduction .....35**
**2 Scope .....36**
**3 Definitions.....37**
3.1 Types of Bluetooth Products .....37
**4 Core Configurations.....39**
4.1 Specification Naming Conventions .....39
4.2 EDR Configurations .....39

**Part C**
**APPENDIX**

**Contents .....43**
**1 Revision History.....45**
1.1 [Vol 0] Master TOC & Compliance Requirements.....45
1.2 [Vol 1] Architecture & Terminology Overview .....45
1.3 [Vol 2 & 3] Core System Package .....46
1.4 [Vol 4] Transport Layers .....48
**2 Contributors.....49**
2.1 [Vol 0] Master TOC & Compliance Requirements.....49
2.2 [Vol 1] Architecture &Terminology Overview .....49
2.3 [Vol 2] Core System Package, Controller.....52
2.4 [Vol 3] Core System Package, Host.....66
2.5 [Vol 4] Host Controller Interface [Transport Layer].....71



LifeScan, Inc. Ex. 1019
Page 9 of 1420
IPR2025-00103

*Specification Volume 1*
Architecture & Terminology Overview

Table of Contents ..... 5

Part A
ARCHITECTURE

Contents ..... 11

1 General Description ..... 13
1.1 Overview of Operation ..... 13
1.2 Nomenclature ..... 15

2 Core System Architecture ..... 20
2.1 Core Architectural Blocks ..... 23

3 Data Transport Architecture ..... 26
3.1 Core Traffic Bearers ..... 27
3.2 Transport Architecture Entities ..... 31
3.3 Physical Channels ..... 33
3.4 Physical Links ..... 38
3.5 Logical Links and Logical Transports ..... 40
3.6 L2CAP Channels ..... 49

4 Communication Topology ..... 50
4.1 Piconet Topology ..... 50
4.2 Operational Procedures and Modes ..... 52

5 Secure Simple Pairing Overview ..... 57
5.1 Security Goals ..... 57
5.2 Passive Eavesdropping Protection ..... 57
5.3 Man-In-The-Middle Protection ..... 58
5.4 Association Models ..... 59

Part B
ACRONYMS & ABBREVIATIONS

Contents ..... 64

1 List of Acronyms and Abbreviations ..... 65

Part C
CORE SPECIFICATION
CHANGE HISTORY

Contents ..... 75

1 Deprecated Features ..... 76



LifeScan, Inc. Ex. 1019
Page 10 of 1420
IPR2025-00103

**2 Changes from V1.1 to V1.2 ....77**
2.1 New Features....77
2.2 Structure Changes ....77
2.3 Deprecated Features list....77
2.4 Changes in Wording....78
2.5 Nomenclature Changes ....78
**3 Changes from V1.2 to V2.0 + EDR ....79**
3.1 New Features....79
3.2 Deprecated Features ....79
**4 Changes from v2.0 + EDR to v2.1 + EDR ....80**
4.1 New features....80
4.2 Deprecated Features ....80

**Part D**
**MIXING OF SPECIFICATION VERSIONS**

**Contents ....83**
**1 Mixing of Specification Versions ....84**
1.1 Features and their Types ....85

**Part E**
**IEEE LANGUAGE**

**Contents ....89**
**1 Use of IEEE Language ....91**
1.1 Shall ....91
1.2 Must ....92
1.3 Will ....92
1.4 Should....92
1.5 May ....92
1.6 Can ....93



LifeScan, Inc. Ex. 1019
Page 11 of 1420
IPR2025-00103

*Specification Volume 2*
**Core System Package**
**[Controller volume]**

**Table of Contents** ..... **5**

**Part A**
**RADIO SPECIFICATION**

**Contents** ..... **27**

**1 Scope** ..... **29**

**2 Frequency Bands and Channel Arrangement** ..... **31**

**3 Transmitter Characteristics** ..... **32**
3.1 Basic Rate ..... 33
3.2 Enhanced Data Rate ..... 35

**4 Receiver Characteristics** ..... **41**
4.1 Basic Rate ..... 41
4.2 Enhanced Data Rate ..... 43

**5 Appendix A** ..... **46**
5.1 Nominal Test Conditions ..... 46
5.2 Extreme Test Conditions ..... 47

**6 Appendix B** ..... **48**

**7 Appendix C** ..... **49**
7.1 Enhanced Data Rate Modulation Accuracy ..... 49

**Part B**
**BASEBAND SPECIFICATION**

**Contents** ..... **55**

**1 General Description** ..... **59**
1.1 Bluetooth Clock ..... 60
1.2 Bluetooth Device Addressing ..... 62
1.3 Access Codes ..... 63

**2 Physical Channels** ..... **64**
2.1 Physical Channel Definition ..... 65
2.2 Basic Piconet Physical Channel ..... 65
2.3 Adapted Piconet Physical Channel ..... 70
2.4 Page Scan Physical Channel ..... 71
2.5 Inquiry Scan Physical Channel ..... 75
2.6 Hop Selection ..... 77

**3 Physical Links** ..... **90**



LifeScan, Inc. Ex. 1019
Page 12 of 1420
IPR2025-00103

3.1 Link Supervision....90
**4 Logical Transports....91**
4.1 General....91
4.2 Logical Transport Address (LT_ADDR)....91
4.3 Synchronous Logical Transports....92
4.4 Asynchronous Logical Transport....92
4.5 Transmit/Receive Routines....93
4.6 Active Slave Broadcast Transport....98
4.7 Parked Slave Broadcast Transport....99
**5 Logical Links....100**
5.1 Link Control Logical Link (LC)....100
5.2 ACL Control Logical Link (ACL-C)....100
5.3 User Asynchronous/Isochronous Logical Link (ACL-U)....100
5.4 User Synchronous Data Logical Link (SCO-S)....101
5.5 User Extended Synchronous Data Logical Link (eSCO-S)....101
5.6 Logical Link Priorities....101
**6 Packets....102**
6.1 General Format....102
6.2 Bit Ordering....103
6.3 Access Code....104
6.4 Packet Header....109
6.5 Packet Types....111
6.6 Payload Format....121
6.7 Packet Summary....127
**7 Bitstream Processing....129**
7.1 Error Checking....130
7.2 Data Whitening....133
7.3 Error Correction....134
7.4 FEC Code: Rate 1/3....134
7.5 FEC Code: Rate 2/3....135
7.6 ARQ Scheme....136
7.7 Erroneous Synchronous Data Reporting....144
**8 Link Controller Operation....145**
8.1 Overview of States....145
8.2 Standby State....146
8.3 Connection Establishment Substates....146
8.4 Device Discovery Substates....155
8.5 Connection State....160
8.6 Active Mode....161
8.7 Sniff Mode....175
8.8 Hold Mode....178



LifeScan, Inc. Ex. 1019
Page 13 of 1420
IPR2025-00103

8.9 Park State ..... 179

**9 Audio ..... 187**

9.1 LOG PCM CODEC ..... 187
9.2 CVSD CODEC ..... 187
9.3 Error Handling ..... 190
9.4 General Audio Requirements ..... 190

**10 List of Figures ..... 191**

**11 List of Tables ..... 194**

**Appendix A: General Audio Recommendations ..... 195**
**Appendix B: Timers ..... 199**
**Appendix C: Recommendations for AFH Operation in Park, Hold, and Sniff ..... 201**

**Part C**
**LINK MANAGER PROTOCOL**

**Contents ..... 205**

**1 Introduction ..... 207**

**2 General Rules ..... 208**

2.1 Message Transport ..... 208
2.2 Synchronization ..... 208
2.3 Packet Format ..... 209
2.4 Transactions ..... 210
2.5 Error Handling ..... 211
2.6 Procedure Rules ..... 212
2.7 General Response Messages ..... 213
2.8 LMP Message Constraints ..... 213

**3 Device Features ..... 214**

3.1 General Description ..... 214
3.2 Feature Definitions ..... 214
3.3 Feature Mask Definition ..... 220
3.4 Link Manager Interoperability policy ..... 222

**4 Procedure Rules ..... 223**

4.1 Connection Control ..... 223
4.2 Security ..... 240
4.3 Informational Requests ..... 268
4.4 Role Switch ..... 274
4.5 Modes of Operation ..... 277
4.6 Logical Transports ..... 290
4.7 Test Mode ..... 298

**5 Summary ..... 303**

5.1 PDU Summary ..... 303



LifeScan, Inc. Ex. 1019
Page 14 of 1420
IPR2025-00103

5.2 Parameter Definitions ....313
5.3 LMP Encapsulated ....323
5.4 Default Values ....324

**6 List of Figures ....325**

**7 List of Tables ....329**

**Part D**
**ERROR CODES**

**Contents ....333**

**1 Overview of Error Codes ....335**
1.1 Usage Descriptions ....335
1.2 HCI Command Errors ....335
1.3 List of Error Codes ....336

**2 Error Code Descriptions ....338**
2.1 Unknown HCI Command (0X01) ....338
2.2 Unknown Connection Identifier (0X02) ....338
2.3 Hardware Failure (0X03) ....338
2.4 Page Timeout (0X04) ....338
2.5 Authentication Failure (0X05) ....338
2.6 PIN or key Missing (0X06) ....338
2.7 Memory Capacity Exceeded (0X07) ....338
2.8 Connection Timeout (0X08) ....339
2.9 Connection Limit Exceeded (0X09) ....339
2.10 Synchronous Connection Limit to a Device Exceeded (0X0A) 339
2.11 ACL Connection Already Exists (0X0B) ....339
2.12 Command Disallowed (0X0C) ....339
2.13 Connection Rejected due to Limited Resources (0X0D) ....339
2.14 Connection Rejected due to Security Reasons (0X0E) ....339
2.15 Connection Rejected due to Unacceptable BD_ADDR (0X0F)340
2.16 Connection Accept Timeout Exceeded (0X10) ....340
2.17 Unsupported Feature or Parameter Value (0X11) ....340
2.18 Invalid HCI Command Parameters (0X12) ....340
2.19 Remote User Terminated Connection (0X13) ....340
2.20 Remote Device Terminated Connection due to Low Resources (0X14) ....341
2.21 Remote Device Terminated Connection due to Power Off (0X15) ....341
2.22 Connection Terminated by Local Host (0X16) ....341
2.23 Repeated Attempts (0X17) ....341
2.24 Pairing not Allowed (0X18) ....341
2.25 Unknown LMP PDU (0X19) ....341



LifeScan, Inc. Ex. 1019
Page 15 of 1420
IPR2025-00103

2.26 Unsupported Remote Feature / Unsupported LMP Feature (0X1A) ..... 341
2.27 SCO Offset Rejected (0X1B) ..... 341
2.28 SCO Interval Rejected (0X1C) ..... 342
2.29 SCO Air Mode Rejected (0X1D) ..... 342
2.30 Invalid LMP Parameters (0X1E) ..... 342
2.31 Unspecified Error (0X1F) ..... 342
2.32 Unsupported LMP Parameter Value (0X20) ..... 342
2.33 Role Change Not Allowed (0X21) ..... 342
2.34 LMP Response Timeout (0X22) ..... 342
2.35 LMP Error Transaction Collision (0X23) ..... 343
2.36 LMP PDU Not Allowed (0X24) ..... 343
2.37 Encryption Mode Not Acceptable (0X25) ..... 343
2.38 Link Key Can Not be Changed (0X26) ..... 343
2.39 Requested Qos Not Supported (0X27) ..... 343
2.40 Instant Passed (0X28) ..... 343
2.41 Pairing with Unit Key Not Supported (0X29) ..... 343
2.42 Different Transaction Collision (0x2a) ..... 343
2.43 QoS Unacceptable Parameter (0X2C) ..... 343
2.44 QoS Rejected (0X2D) ..... 344
2.45 Channel Classification Not Supported (0X2E) ..... 344
2.46 Insufficient Security (0X2F) ..... 344
2.47 Parameter out of Mandatory Range (0X30) ..... 344
2.48 Role Switch Pending (0X32) ..... 344
2.49 Reserved Slot Violation (0X34) ..... 344
2.50 Role Switch Failed (0X35) ..... 344
2.51 Extended Inquiry Response Too Large (0x36) ..... 344
2.52 Simple Pairing Not Supported By Host (0X37) ..... 345
2.53 Host Busy–Pairing(0X38) ..... 345

**Part E**
**HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION**

**Contents ..... 349**

**1 Introduction ..... 357**
1.1 Lower Layers of the Bluetooth Software Stack ..... 357

**2 Overview of Host Controller Transport Layer ..... 359**

**3 Overview of Commands and Events ..... 360**
3.1 Generic Events ..... 361
3.2 Device Setup ..... 361
3.3 Controller Flow Control ..... 362
3.4 Controller Information ..... 362



LifeScan, Inc. Ex. 1019
Page 16 of 1420
IPR2025-00103

3.5 Controller Configuration ....363
3.6 Device Discovery ....364
3.7 Connection Setup ....366
3.8 Remote Information....368
3.9 Synchronous Connections ....369
3.10 Connection State....370
3.11 Piconet Structure....371
3.12 Quality of Service....372
3.13 Physical Links ....374
3.14 Host Flow Control....375
3.15 Link Information ....376
3.16 Authentication and Encryption ....377
3.17 Testing....380
3.18 Alphabetical List of Commands and Events ....381

**4 HCI Flow Control ....386**
4.1 Host to Controller Data Flow Control ....386
4.2 Controller to Host Data Flow Control ....387
4.3 Disconnection Behavior ....388
4.4 Command Flow Control ....388
4.5 Command Error Handling ....389

**5 HCI Data Formats ....390**
5.1 Introduction ....390
5.2 Data and Parameter Formats....390
5.3 Connection Handles....391
5.4 Exchange of HCI-Specific Information ....392

**6 HCI Configuration Parameters ....399**
6.1 Scan Enable....399
6.2 Inquiry Scan Interval ....399
6.3 Inquiry Scan Window ....400
6.4 Inquiry Scan Type ....400
6.5 Inquiry Mode ....400
6.6 Page Timeout....401
6.7 Connection Accept Timeout ....401
6.8 Page Scan Interval....402
6.9 Page Scan Window....402
6.10 Page Scan Period Mode (Deprecated) ....402
6.11 Page Scan Type....403
6.12 Voice Setting ....403
6.13 PIN Type ....404
6.14 Link Key ....404
6.15 Failed Contact Counter ....405
6.16 Hold Mode Activity ....405

6.17 Link Policy Settings ..... 406
6.18 Flush Timeout ..... 407
6.19 Num Broadcast Retransmissions ..... 407
6.20 Link Supervision Timeout ..... 408
6.21 Synchronous Flow Control Enable ..... 408
6.22 Local Name ..... 409
6.23 Extended Inquiry Response ..... 409
6.24 Erroneous Data Reporting ..... 409
6.25 Class Of Device ..... 410
6.26 Supported Commands ..... 410

**7 HCI Commands and Events ..... 416**
7.1 Link Control Commands ..... 416
7.2 Link Policy Commands ..... 475
7.3 Controller & Baseband Commands ..... 498
7.4 Informational Parameters ..... 580
7.5 Status Parameters ..... 590
7.6 Testing Commands ..... 600
7.7 Events ..... 606

**8 List of Figures ..... 664**

**9 List of Tables ..... 665**

**10 Appendix A: Deprecated Commands, Events and Configuration Parameters666**
10.1 Read Page Scan Mode Command ..... 667
10.2 Write Page Scan Mode Command ..... 668
10.3 Read Page Scan Period Mode Command ..... 669
10.4 Write Page Scan Period Mode Command ..... 670
10.5 Add SCO Connection Command ..... 671
10.6 Page Scan Mode Change Event ..... 673
10.7 Read Country Code Command ..... 674
10.8 Read Encryption Mode Command ..... 675
10.9 Write Encryption Mode Command ..... 675
10.10 Deprecated Parameters ..... 676

**Part F**
**MESSAGE SEQUENCE CHARTS**

**Contents ..... 681**

**1 Introduction ..... 683**
1.1 Notation ..... 683
1.2 Flow of Control ..... 684
1.3 Example MSC ..... 684

**2 Services Without Connection Request ..... 685**

2.1 Remote Name Request....685
2.2 One-time Inquiry....687
2.3 Periodic Inquiry....689

**3 ACL Connection Establishment and Detachment....691**
3.1 Connection Setup....692

**4 Optional Activities After ACL Connection Establishment....699**
4.1 Authentication Requested....699
4.2 Simple Pairing Message Sequence Charts....700
4.3 Link Supervision Timeout Changed Event....713
4.4 Set Connection Encryption....714
4.5 Change Connection Link Key....715
4.6 Change Connection Link Key with Encryption Pause and Resume....715
4.7 Master Link Key....717
4.8 Read Remote Supported Features....719
4.9 Read Remote Extended Features....719
4.10 Read Clock Offset....720
4.11 Role Switch on an Encrypted Link using Encryption Pause and Resume....720
4.12 Refreshing Encryption Keys....721
4.13 Read Remote Version Information....722
4.14 QOS Setup....722
4.15 Switch Role....723

**5 Synchronous Connection Establishment and Detachment....725**
5.1 Synchronous Connection Setup....725

**6 Sniff, Hold and Park....730**
6.1 Sniff Mode....730
6.2 Hold Mode....731
6.3 Park State....733

**7 Buffer Management, Flow Control....736**

**8 Loopback Mode....738**
8.1 Local Loopback Mode....738
8.2 Remote Loopback Mode....740

**9 List of Figures....742**

**Part G**
**SAMPLE DATA**

**Contents....747**

**1 Encryption Sample Data....749**
1.1 Generating Kc' from Kc,....749



LifeScan, Inc. Ex. 1019
Page 19 of 1420
IPR2025-00103

1.2 First Set of Sample Data ..... 752
1.3 Second Set of Sample Data ..... 760
1.4 Third Set of Samples ..... 768
1.5 Fourth Set of Samples ..... 776

**2 Frequency Hopping Sample Data ..... 784**
2.1 First set ..... 785
2.2 Second set ..... 791
2.3 Third set ..... 797

**3 Access Code Sample Data ..... 803**

**4 HEC and Packet Header Sample Data ..... 806**

**5 CRC Sample Data ..... 807**

**6 Complete Sample Packets ..... 808**
6.1 Example of DH1 Packet ..... 808
6.2 Example of DM1 Packet ..... 809

**7 Simple Pairing Sample Data ..... 810**
7.1 P-192 Sample Data ..... 810
7.2 Hash Functions Sample Data ..... 811

**8 Whitening Sequence Sample Data ..... 821**

**9 FEC Sample Data ..... 824**

**10 Encryption Key Sample Data ..... 825**
10.1 Four Tests of E1 ..... 825
10.2 Four Tests of E21 ..... 830
10.3 Three Tests of E22 ..... 832
10.4 Tests of E22 With Pin Augmenting ..... 834
10.5 Four Tests of E3 ..... 844

**Part H**
**SECURITY SPECIFICATION**

**Contents ..... 853**

**1 Security Overview ..... 855**
1.1 Pausing Encryption and Role Switch ..... 856
1.2 Change Connection Link Keys ..... 856
1.3 Periodically Refreshing Encryption Keys ..... 856

**2 Random Number Generation ..... 857**

**3 Key Management ..... 859**
3.1 Key Types ..... 859
3.2 Key Generation and Initialization ..... 861

**4 Encryption ..... 868**
4.1 Encryption Key Size Negotiation ..... 869

4.2 Encryption of Broadcast Messages....869
4.3 Encryption Concept....870
4.4 Encryption Algorithm....871
4.5 LFSR Initialization....874
4.6 Key Stream Sequence....877
**5 Authentication....878**
5.1 Repeated Attempts....880
**6 The Authentication And Key-Generating Functions....881**
6.1 The Authentication Function E1....881
6.2 The Functions Ar and A'r....883
6.3 E2-Key Generation Function for Authentication....885
6.4 E3-Key Generation Function for Encryption....887
**7 Secure Simple Pairing....888**
7.1 Phase 1: Public Key Exchange....889
7.2 Phase 2: Authentication Stage 1....890
7.3 Phase 3: Authentication Stage 2....896
7.4 Phase 4: Link Key Calculation....897
7.5 Phase 5: LMP Authentication and Encryption....897
7.6 Elliptic Curve Definition....897
7.7 Cryptographic Function Definitions....898
**8 List of Figures....902**
**9 List of Tables....903**



LifeScan, Inc. Ex. 1019
Page 21 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 22 of 1420
IPR2025-00103

*Specification Volume 3*
**Core System Package**
**[Host volume]**

**Table of Contents** ..... **5**

**Part A**
**LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION**

**Contents** ..... **15**

**1 Introduction** ..... **19**
1.1 L2CAP Features ..... 19
1.2 Assumptions ..... 23
1.3 Scope ..... 23
1.4 Terminology ..... 24

**2 General Operation** ..... **27**
2.1 Channel Identifiers ..... 27
2.2 Operation Between Devices ..... 27
2.3 Operation Between Layers ..... 29
2.4 Modes of Operation ..... 29

**3 Data Packet Format** ..... **31**
3.1 Connection-oriented Channel in Basic L2CAP Mode ..... 31
3.2 Connectionless Data Channel in Basic L2CAP Mode ..... 32
3.3 Connection-oriented Channel in Retransmission/Flow Control Modes 33

**4 Signaling Packet Formats** ..... **39**
4.1 Command Reject (code 0x01) ..... 41
4.2 Connection Request (code 0x02) ..... 42
4.3 Connection Response (code 0x03) ..... 44
4.4 Configuration Request (code 0x04) ..... 45
4.5 Configuration Response (code 0x05) ..... 48
4.6 Disconnection Request (code 0x06) ..... 50
4.7 Disconnection Response (code 0x07) ..... 51
4.8 Echo Request (code 0x08) ..... 51
4.9 Echo Response (code 0x09) ..... 52
4.10 Information Request (code 0x0A) ..... 52
4.11 Information Response (code 0x0B) ..... 53
4.12 Extended Feature Mask ..... 54

**5 Configuration Parameter Options** ..... **55**
5.1 Maximum Transmission Unit (MTU) ..... 55
5.2 Flush Timeout Option ..... 57



LifeScan, Inc. Ex. 1019
Page 23 of 1420
IPR2025-00103

5.3 Quality of Service (QoS) Option ..... 58
5.4 Retransmission and Flow Control Option ..... 62

**6 State Machine ..... 65**
6.1 General rules for the state machine: ..... 65
6.2 Timers events ..... 73

**7 General Procedures ..... 77**
7.1 Configuration Process ..... 77
7.2 Fragmentation and Recombination ..... 79
7.3 Encapsulation of SDUs ..... 81
7.4 Delivery of Erroneous L2CAP SDUs ..... 83
7.5 Operation with Flushing ..... 83
7.6 Connectionless Data Channel ..... 84

**8 Procedures for Flow Control and Retransmission ..... 85**
8.1 Information Retrieval ..... 85
8.2 Function of PDU Types for Flow Control and Retransmission ..... 85
8.3 Variables and Sequence Numbers ..... 87
8.4 Retransmission Mode ..... 91
8.5 Flow Control Mode ..... 97

**9 List of Figures ..... 101**

**10 List of Tables ..... 103**
**Appendix A: Configuration MSCs ..... 105**

**Part B**
**SERVICE DISCOVERY PROTOCOL (SDP)**

**Contents ..... 111**

**1 Introduction ..... 113**
1.1 General Description ..... 113
1.2 Motivation ..... 113
1.3 Requirements ..... 113
1.4 Non-requirements and Deferred Requirements ..... 114
1.5 Conventions ..... 114

**2 Overview ..... 115**
2.1 SDP Client-Server Interaction ..... 115
2.2 Service Record ..... 116
2.3 Service Attribute ..... 117
2.4 Service Class ..... 119
2.5 Searching for Services ..... 121
2.6 Browsing for Services ..... 122

**3 Data Representation ..... 125**

3.1 Data Element ....125
3.2 Data Element Type Descriptor ....125
3.3 Data Element Size Descriptor ....126
3.4 Data Element Examples....127

**4 Protocol Description ....128**
4.1 Transfer Byte Order ....128
4.2 Protocol Data Unit Format....128
4.3 Partial Responses and Continuation State....130
4.4 Error Handling ....130
4.5 ServiceSearch Transaction ....131
4.6 ServiceAttribute Transaction ....135
4.7 ServiceSearchAttribute Transaction....138

**5 Service Attribute Definitions ....142**
5.1 Universal Attribute Definitions ....142
5.2 ServiceDiscoveryServer Service Class Attribute Definitions....152
5.3 BrowseGroupDescriptor Service Class Attribute Definitions....153

**6 Security ....155**

**7 Appendix ....156**
**Appendix A – Background Information ....157**
**Appendix B – Example SDP Transactions ....158**

**Part C**
**GENERIC ACCESS PROFILE**

**Contents ....171**
**Foreword ....175**

**1 Introduction ....176**
1.1 Scope....176
1.2 Symbols and conventions ....176

**2 Profile overview....178**
2.1 Profile stack....178
2.2 Configurations and roles ....178
2.3 User requirements and scenarios ....179
2.4 Profile fundamentals ....180
2.5 Conformance ....180

**3 User interface aspects ....181**
3.1 The user interface level....181
3.2 Representation of Bluetooth parameters ....181
3.3 Pairing....184

**4 Modes....185**
4.1 Discoverability modes ....185



LifeScan, Inc. Ex. 1019
Page 25 of 1420
IPR2025-00103

4.2 Connectability modes ..... 189
4.3 Bondable modes ..... 191

**5 Security Aspects ..... 193**
5.1 Authentication ..... 193
5.2 Security modes ..... 194

**6 Idle Mode Procedures ..... 212**
6.1 General inquiry ..... 212
6.2 Limited inquiry ..... 213
6.3 Name discovery ..... 215
6.4 Device discovery ..... 217
6.5 Bonding ..... 218

**7 Establishment procedures ..... 222**
7.1 Link establishment ..... 222
7.2 Channel establishment ..... 225
7.3 Connection establishment ..... 227
7.4 Establishment of additional connection ..... 228

**8 Extended Inquiry Response Data Format ..... 229**
8.1 EIR Data Type Definitions ..... 230
8.2 Example Extended Inquiry Response ..... 233

**9 Definitions ..... 235**
9.1 General definitions ..... 235
9.2 Connection-related definitions ..... 235
9.3 Device-related definitions ..... 236
9.4 Procedure-related definitions ..... 237
9.5 Security-related definitions ..... 237

**10 Appendix A (Normative): Timers and constants ..... 239**

**11 Appendix B (Informative): Information Flows of Related Procedures ..... 240**
11.1 LMP-authentication ..... 240
11.2 LMP-pairing ..... 241
11.3 Service discovery ..... 242

**12 References ..... 243**

**Part D**
**TEST SUPPORT**

**Contents ..... 247**

**1 Test Methodology ..... 249**
1.1 Test Scenarios ..... 249
1.2 References ..... 258

**2 Test Control Interface (TCI) ..... 259**



LifeScan, Inc. Ex. 1019
Page 26 of 1420
IPR2025-00103

2.1 Introduction .....259
2.2 TCI Configurations .....260
2.3 TCI Configuration and Usage.....263



LifeScan, Inc. Ex. 1019
Page 27 of 1420
IPR2025-00103

*Specification Volume 4*
**Host Controller Interface**
**[Transport Layer]**

**Table of Contents** ..... 5

**Part A**
**UART TRANSPORT LAYER**

**Contents** ..... 11

**1 General** ..... 13

**2 Protocol** ..... 14

**3 RS232 Settings** ..... 15

**4 Error Recovery** ..... 16

**Part B**
**USB TRANSPORT LAYER**

**Contents** ..... 19

**1 Overview** ..... 21

**2 USB Endpoint Expectations** ..... 23
2.1 Descriptor Overview ..... 23
2.2 Control Endpoint Expectations ..... 28
2.3 Bulk Endpoints Expectations ..... 28
2.4 Interrupt Endpoint Expectations ..... 28
2.5 Isochronous Endpoints Expectations ..... 29

**3 Class Code** ..... 30

**4 Device Firmware Upgrade** ..... 31

**5 Limitations** ..... 32
5.1 Power Specific Limitations ..... 32
5.2 Other Limitations ..... 32

**Part C**
**SECURE DIGITAL (SD)**
**TRANSPORT LAYER**

**Contents** ..... 35

**1 Introduction** ..... 37

**2 Goals** ..... 38
2.1 Hardware Goals ..... 38
2.2 Software Goals ..... 38



LifeScan, Inc. Ex. 1019
Page 28 of 1420
IPR2025-00103

**3 Physical Interface Documents ..... 39**
**4 Communication ..... 40**
4.1 Overview ..... 40
**5 Appendix A - Acronyms and Abbreviations ..... 41**
**6 Appendix B - Related Documents ..... 42**
**7 Appendix C - Tests ..... 43**
7.1 Test Suite Structure ..... 43

**Part D**
**THREE-WIRE UART TRANSPORT LAYER**

**Contents ..... 47**
**1 General ..... 49**
**2 Overview ..... 50**
**3 Slip Layer ..... 51**
3.1 Encoding a Packet ..... 51
3.2 Decoding a Packet ..... 51
**4 Packet Header ..... 53**
4.1 Sequence Number ..... 53
4.2 Acknowledge Number ..... 54
4.3 Data Integrity Check Present ..... 54
4.4 Reliable Packet ..... 54
4.5 Packet Type ..... 54
4.6 Payload Length ..... 55
4.7 Packet Header Checksum ..... 55
**5 Data Integrity Check ..... 56**
5.1 16 Bit CCITT-CRC ..... 56
**6 Reliable Packets ..... 57**
6.1 Header Checksum Error ..... 57
6.2 Slip Payload Length Error ..... 57
6.3 Data Integrity Check Error ..... 57
6.4 Out Of Sequence Packet Error ..... 57
6.5 Acknowledgement ..... 58
6.6 Resending Packets ..... 58
6.7 Example Reliable Packet Flow ..... 58
**7 Unreliable Packets ..... 61**
7.1 Unreliable Packet Header ..... 61
7.2 Unreliable Packet Error ..... 61
**8 Link Establishment ..... 62**
8.1 Uninitialized State ..... 63



LifeScan, Inc. Ex. 1019
Page 29 of 1420
IPR2025-00103

8.2 Initialized State ..... 63
8.3 Active State ..... 63
8.4 Sync Message ..... 64
8.5 Sync Response Message ..... 64
8.6 Config Message ..... 64
8.7 Config Response Message ..... 65
8.8 Configuration Field ..... 65

**9 LOW POWER ..... 68**
9.1 Wakeup Message ..... 68
9.2 Woken Message ..... 68
9.3 Sleep Message ..... 69

**10 Out of Frame Control ..... 70**
10.1 Software Flow Control ..... 70

**11 Hardware Configuration ..... 71**
11.1 Wires ..... 71
11.2 Hardware Flow ..... 71

**12 Recommended Parameters ..... 72**
12.1 Timing Parameters ..... 72

**13 References ..... 73**



LifeScan, Inc. Ex. 1019
Page 30 of 1420
IPR2025-00103

# BLUETOOTH COMPLIANCE REQUIREMENTS

***This document specifies the requirements for Bluetooth compliance.***



LifeScan, Inc. Ex. 1019
Page 31 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 32 of 1420
IPR2025-00103

# CONTENTS

## BLUETOOTH COMPLIANCE REQUIREMENTS

**1 Introduction ..... 37**

**2 Scope ..... 38**

**3 Definitions ..... 39**

3.1 Types of Bluetooth Products ..... 39

**4 Core Configurations ..... 41**

4.1 Specification Naming Conventions ..... 41

4.2 EDR Configurations ..... 41



LifeScan, Inc. Ex. 1019
Page 33 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 34 of 1420
IPR2025-00103

# 1 INTRODUCTION

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

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



LifeScan, Inc. Ex. 1019
Page 35 of 1420
IPR2025-00103

# 2 SCOPE

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



LifeScan, Inc. Ex. 1019
Page 36 of 1420
IPR2025-00103

# 3 DEFINITIONS

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

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

## 3.1 TYPES OF BLUETOOTH PRODUCTS

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

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

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

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

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



LifeScan, Inc. Ex. 1019
Page 37 of 1420
IPR2025-00103

- *Bluetooth Controller Subsystem Product* – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in the Bluetooth Radio, Baseband, Link Manager and HCI parts of this Specification, but none of the Protocols and Profiles above HCI.
- *Bluetooth Profile Subsystem Product* – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in one or more of the profile specifications.

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

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

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



LifeScan, Inc. Ex. 1019
Page 38 of 1420
IPR2025-00103

# 4 CORE CONFIGURATIONS

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

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

## 4.1 SPECIFICATION NAMING CONVENTIONS

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

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

Examples of complete specification names including the core configuration names:

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

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

## 4.2 EDR CONFIGURATIONS

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

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

- A Bluetooth product shall support category 1 whenever it supports asynchronous transports for the profiles it incorporates.
- A Bluetooth product shall support category 2 whenever it supports asynchronous transports with multislot ACL packets for the profiles it incorporates.
- A Bluetooth product shall support category 3 whenever it supports eSCO synchronous transports for the profiles it incorporates.



LifeScan, Inc. Ex. 1019
Page 39 of 1420
IPR2025-00103

.

| No. | Transport Requirements | LMP Feature Bits Required | L2CAP Feature Bits Required |
|---|---|---|---|
| 1 | EDR for asynchronous transports (single slot) | Enhanced Data Rate ACL 2 Mbps mode (25)<br>Enhanced Data Rate ACL 3 Mbps mode (26) | None |
| 2 | EDR for asynchronous transports (multi-slot) | 3-slot Enhanced Data Rate ACL packets (39)<br>5-slot Enhanced Data Rate ACL packets (40) | None |
| 3 | EDR for synchronous transports | Enhanced Data Rate eSCO 2 Mbps mode (45)<br>Enhanced Data Rate eSCO 3 Mbps mode (46) | None |

*Table 4.1: EDR configuration requirements*

Note: No additional requirements are stated on the support of 2-EV5 and 3-EV5 packets.



LifeScan, Inc. Ex. 1019
Page 40 of 1420
IPR2025-00103

# APPENDIX



LifeScan, Inc. Ex. 1019
Page 41 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 42 of 1420
IPR2025-00103

# CONTENTS

**1 Revision History ..... 47**
1.1 [Vol 0] Master TOC & Compliance Requirements ..... 47
1.1.1 Bluetooth Compliance Requirements ..... 47
1.2 [Vol 1] Architecture & Terminology Overview ..... 47
1.3 [Vol 2 & 3] Core System Package ..... 48
1.4 [Vol 4] Transport Layers ..... 50
**2 Contributors ..... 51**
2.1 [Vol 0] Master TOC & Compliance Requirements ..... 51
2.1.1 Part B: Bluetooth Compliance Requirements ..... 51
2.2 [Vol 1] Architecture &Terminology Overview ..... 51
2.2.1 Part A: Architectural Overview ..... 51
2.2.2 Part B: Acronyms & Abbreviations ..... 52
2.2.3 Part C: Core Specification Change History ..... 53
2.2.4 Part D: Mixing of Specification Versions ..... 53
2.3 [Vol 2] Core System Package, Controller ..... 54
2.3.1 Part A: Radio Specification ..... 54
2.3.2 Part B: Baseband Specification ..... 55
2.3.3 Part C: Link Manager Protocol ..... 58
2.3.4 Part D: Error Codes ..... 60
2.3.5 Part E: Bluetooth Host Controller Interface Functional Specification ..... 61
2.3.6 Part F: Message Sequence Charts ..... 64
2.3.7 Part G: Sample Data ..... 65
2.3.8 Part H: Security Specification ..... 66
2.4 [Vol 3] Core System Package, Host ..... 68
2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification ..... 68
2.4.2 Part B: Service Discovery Protocol (SDP) ..... 70
2.4.3 Part C Generic Access Profile ..... 71
2.4.4 Part D: Test Support ..... 71
2.5 [Vol 4] Host Controller Interface [Transport Layer] ..... 73



LifeScan, Inc. Ex. 1019
Page 43 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 44 of 1420
IPR2025-00103

# 1 REVISION HISTORY

Public versions are marked with bold in the tables below.

Beginning with v1.2 of the Core System Package the core Bluetooth specification documents, protocols and profiles, were transferred to a new partitioning comprising volumes and individual profile specifications are each contained in an individual document instead of the two volumes (Core and Profiles) used in v1.1.

For more detailed information about changes between versions published before v1.2, please see the Appendix I “Revision History” in v1.1.

## 1.1 [VOL 0] MASTER TOC & COMPLIANCE REQUIREMENTS

### 1.1.1 Bluetooth Compliance Requirements

| Rev | Date | Comments |
|---|---|---|
| v2.1 + EDR | | No content changes. Updates to the Table of Contents. |
| v2.0 + EDR | Oct 15 2004 | This version of the specification is intended to be a separate Bluetooth Specification that has all the functional characteristics of the v1.2 Bluetooth Specification that adds the Enhanced Data Rate (EDR) feature which required changes to Volume 0, Part A, Master Table of Contents. |
| v1.2 | Nov 05 2003 | This Part was moved from the Core volume. No content changes been made to this document since v1.1 |

## 1.2 [VOL 1] ARCHITECTURE & TERMINOLOGY OVERVIEW

| Rev | Date | Comments |
|---|---|---|
| v2.1 + EDR | | Added definitions for new features: Encryption Pause Resume, Erroneous Data reporting, Extended Inquiry Response, Link Supervision Timeout Event, Packet Boundary Flag, Secure Simple Pairing, Sniff Subrating |
| v2.0 + EDR | Oct 15 2004 | This version of the specification is intended to be a separate Bluetooth Specification that has all the functional characteristics of the v1.2 Bluetooth Specification that adds the Enhanced Data Rate (EDR) feature which incorporates changes to Volume 1, Part B, Acronyms and Abbreviations. |
| v1.2 | Nov 05 2003 | New volume with informational content. This volume will always be updated in parallel with the Core System volumes |



LifeScan, Inc. Ex. 1019
Page 45 of 1420
IPR2025-00103

## 1.3 [VOL 2 & 3] CORE SYSTEM PACKAGE

| Rev | Date | Comments |
|---|---|---|
| v2.1 + EDR | | • New features added in 2.1 + EDR:<br>-Encryption Pause and Resume<br>-Erroneous Data Reporting<br>-Extended Inquiry Response<br>-Link Supervision Timeout Changed Event<br>-Non-Flushable Packet Boundary Flag<br>-Secure Simple Pairing<br>-Sniff Subrating<br>-Security Mode 4<br>• Updates to IEEE language in Volume 2, Part H, Security<br>• Errata for v2.0 + EDR |
| v2.0 + EDR | Aug 01 2004 | This version of the specification is intended to be a separate Bluetooth Specification. This specification was created by adding EDR and the errata. |
| v1.2 | Nov 05 2003 | New features added in v1.2:<br>- Architectural overview<br>- Faster connection<br>- Adaptive frequency hopping<br>- Extended SCO links<br>- Enhanced error detection and flow control<br>- Enhanced synchronization capability<br>- Enhanced flow specification<br>The Core System Package now comprises two volumes and the text has gone through a radical change both in terms of structure and nomenclature. The language is also more precise and is adapted to meet the IEEE standard.<br>The following parts are moved from the Core System Package to other<br>volumes or were deprecated:<br>RFCOMM [vol 7], Object Exchange (IrDA Interoperability) [vol 8],<br>TCS [vol 9], Interoperability Requirements for Bluetooth as a WAP Bearer [vol 6], HCI USB Transport Layer [vol4], HCI RS232 Transport Layer [vol 4], HCI UART Transport Layer [vol 4], Bluetooth Compliance Requirements [vol 0], Optional Paging Schemes [deprecated] |
| 1.1 | Feb 22 2001 | The specification was updated with Errata items previously published on the web site. The Bluetooth Assigned Numbers appendix was lifted out from the specification to allow continuous maintenance on the web site. |



LifeScan, Inc. Ex. 1019
Page 46 of 1420
IPR2025-00103

| Rev | Date | Comments |
|---|---|---|
| 1.0B | Dec. 1 1999 | The specification was updated with Errata items previously published on the web site and was revised from a linguistic point of view.<br>The following parts were added:<br>Interoperability Requirements for Bluetooth as a WAP Bearer,<br>Test Control Interface, Sample Data (appendix), Bluetooth Audio<br>(appendix), Baseband Timers (appendix) and Optional Paging Scheme (appendix) |
| 1.0a | July 26 1999 | The first version of the Bluetooth Specification published on the public web site.<br>Added part: Bluetooth Compliance Requirements. |
| 1.0 draft | July 5th 1999 | The following parts were added:<br>Service Discovery Protocol (SDP), Telephony Control Specification (TCS),<br>Bluetooth Assigned Numbers (appendix) and Message Sequence Charts (appendix) |
| 0.9 | April l30 1999 | The following parts were added:<br>IrDA Interoperability, HCI RS232 Transport Layer, HCI UART Transport Layer and Test Mode |
| 0.8 | Jan 21 1999 | The following parts were added:<br>Radio Specification, L2CAP, RFCOMM, HCI & HCI USB Transport Layer |
| 0.7 | Oct 19 1998 | This first version only included Baseband and Link Manager Protocol |



LifeScan, Inc. Ex. 1019
Page 47 of 1420
IPR2025-00103

## 1.4 [VOL 4] TRANSPORT LAYERS

| Rev | Date | Comments |
|---|---|---|
| **v2.1 + EDR** | Feb 22 2007 | Added this volume to the specification |



LifeScan, Inc. Ex. 1019
Page 48 of 1420
IPR2025-00103

# 2 CONTRIBUTORS

## 2.1 [VOL 0] MASTER TOC & COMPLIANCE REQUIREMENTS

### 2.1.1 Part B: Bluetooth Compliance Requirements

| BQRB (Editor) | |
|---|---|
| Wayne Park | 3Com Corporation |
| Lawrence Jones | ComBit, Inc. |
| Gary Robinson | IBM Corporation |
| Georges Seuron | IBM Corporation |
| Rick Jessop | Intel Corporation |
| John Webb | Intel Corporation |
| Bruce Littlefield | Lucent Technologies, Inc. |
| Brian A. Redding | Motorola, Inc. |
| Waldemar Hontscha | Nokia Corporation |
| Petri Morko | Nokia Corporation |
| Magnus Hansson | Telefonaktiebolaget LM Ericsson |
| Magnus Sommansson | Telefonaktiebolaget LM Ericsson |
| Göran Svennarp | Telefonaktiebolaget LM Ericsson |
| Warren Allen | Toshiba Corporation |
| John Shi | Toshiba Corporation |

## 2.2 [VOL 1] ARCHITECTURE &TERMINOLOGY OVERVIEW

### 2.2.1 Part A: Architectural Overview

| Ayse Findikli | Atheros |
|---|---|
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Henrik Hedlund | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Martin van der Zee | Ericsson |
| David Suvak | iAnywhere |
| Mattias Edlund | Infineon |
| Penny Chen | Intel |
| Selim Aissi | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |



LifeScan, Inc. Ex. 1019
Page 49 of 1420
IPR2025-00103

| | |
|---|---|
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| Brian Redding | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Thomas Müller | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Jürgen Schnitzler | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Guido Bertoni | ST |
| Jorgen van Parijs | ST |
| Tim Howes | Symbian |
| Michael Hasling | Tality |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |
| John Mersh | TTPCom |

### 2.2.2 Part B: Acronyms & Abbreviations

| | |
|---|---|
| Dan Sönnerstam | Pyramid Communication AB |
| Steve Koester | Bluetooth SIG, Inc. |



LifeScan, Inc. Ex. 1019
Page 50 of 1420
IPR2025-00103

### 2.2.3 Part C: Core Specification Change History

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Henrik Hedlund | CSR |
| Steven Singer | CSR |
| Paul Wright | CSR |
| David Suvak | iAnywhere |
| Mattias Edlund | Infineon |
| Rob Davies | Philips |
| Peter Hauser | Microsoft |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Jorgen van Parijs | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |
| John Mersh | TTPCom |

### 2.2.4 Part D: Mixing of Specification Versions

| | |
|---|---|
| David Suvak | iAnywhere |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Dave Suvak | iAnywhere |
| Mattias Edlund | Infineon |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Robin Heydon | CSR |
| Steven Singer | CSR |
| Peter Hauser | Microsoft |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Jorgen van Parijs | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |



LifeScan, Inc. Ex. 1019
Page 51 of 1420
IPR2025-00103

## 2.3 [VOL 2] CORE SYSTEM PACKAGE, CONTROLLER

### 2.3.1 Part A: Radio Specification

*Version 2.0 + EDR*

| | |
|---|---|
| Steven Hall | RF Micro Devices |
| Robert Young | CSR |
| Robert Kokke | Ericsson |
| Harald Kafemann | Nokia |
| Jukka Reunamäki | Nokia |
| Morton Gade | Digianswer |
| Mike Fitton | Toshiba |
| Oren Eliezer | Texas Instruments |
| Stephane Laurent-Michel | Tality |

*Version 1.2*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Morten Gade | Digianswer/Motorola |
| Henrik Hedlund | Ericsson |
| Stefan Agnani | Ericsson |
| Robert Kokke | Ericsson |
| Roland Hellfajer | Infineon |
| Thomas Müller | Nokia |
| Antonio Salloum | Philips |
| Joel Linsky | Silicon Wave |
| Steven Hall | Silicon Wave |
| Oren Eliezer | Texas Instruments |
| Mike Fitton | Toshiba |

*Previous versions*

| | |
|---|---|
| Steve Williams | 3Com Corporation |
| Todor V. Cooklov | Aware |
| Poul Hove Kristensen | Digianswer A/S |
| Kurt B. Fischer | Hyper Corporation |
| Kevin D. Marquess | Hyper Corporation |
| Troy Beukema | IBM Corporation |
| Brian Gaucher | IBM Corporation |
| Jeff Schiffer | Intel Corporation |
| James P. Gilb | Mobilian |
| Rich L. Ditch | Motorola, Inc. |
| Paul Burgess | Nokia Corporation |
| Olaf Joeressen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |

| | |
|---|---|
| Arto T. Palin | Nokia Corporation |
| Steven J. Shellhammer | Symbol |
| Sven Mattisson | Telefonaktiebolaget LM Ericsson |
| Lars Nord (Section Owner) | Telefonaktiebolaget LM Ericsson |
| Anders Svensson | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Allen Hotari | Toshiba Corporation |

### 2.3.2 Part B: Baseband Specification

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Dave Suvak | iAnywhere |
| Mattias Edlund | Infineon |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Oren Haggai | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |



LifeScan, Inc. Ex. 1019
Page 53 of 1420
IPR2025-00103

| | |
|---|---|
| Reinhard Meindl | NXP |
| Jorgen van Parijs | ST |
| Guido Bertoni | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |

*Version 2.0 + EDR*

| | |
|---|---|
| Steven Hall | RF Micro Devices |
| Robert Young | CSR |
| Robert Kokke | Ericsson |
| Harald Kafemann | Nokia |
| Joel Linsky | RF Micro Devices |
| Terry Bourk | RF Micro Devices |
| Arto Palin | Nokia |

*Version 1.2*

| | |
|---|---|
| P G Madhavan | Agere |
| Hongbing Gan | Bandspeed, Inc. |
| Tod Sizer | Bell Labs |
| Alexander Thoukydides | CSR |
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Kim Schneider | Digianswer/Motorola |
| Knud Dyring-Olsen | Digianswer/Motorola |
| Niels Nielsen | Digianswer/Motorola |
| Henrik Andersen | Digianswer/Motorola |
| Christian Gehrmann | Ericsson |
| Henrik Hedlund | Ericsson |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Rakesh Taori | Ericsson |
| Jaap Haartsen | Ericsson |
| Stefan Zürbes | Ericsson |
| Roland Hellfajer | Infineon |
| YC Maa | Integrated Programmable Communications, Inc. |
| HungKun Chen | Integrated Programmable Communications, Inc. |
| Steve McGowan | Intel |
| Adrian Stephens | Mobilian Corporation |
| Jim Lansford | Mobilian Corporation |
| Eric Meihofer | Motorola |
| Arto Palin | Nokia |
| Carmen Kühl | Nokia |
| Hannu Laine | Nokia |



LifeScan, Inc. Ex. 1019
Page 54 of 1420
IPR2025-00103

| | |
|---|---|
| Jürgen Schnitzler | Nokia |
| Päivi Ruuska | Nokia |
| Thomas Müller | Nokia |
| Antonio Salloum | Philips |
| Harmke de Groot | Philips |
| Marianne van de Casteelee | Philips |
| Rob Davies | Philips |
| Roland Matthijssen | Philips |
| Joel Linsky (section owner) | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Gary Schneider | Symbol Technologies, Inc. |
| Stephen J. Shellhammer | Symbol Technologies, Inc. |
| Michael Hasling | Tality |
| Amihai Kidron | Texas Instruments |
| Dan Michael | Texas Instruments |
| Eli Dekel | Texas Instruments |
| Jie Liang | Texas Instruments |
| Oren Eliezer | Texas Instruments |
| Tally Shina | Texas Instruments |
| Yariv Raveh | Texas Instruments |
| Anuj Batra | Texas Instruments |
| Katsuhiro Kinoshita | Toshiba |
| Toshiki Kizu | Toshiba |
| Yoshimitsu Shimojo | Toshiba |
| Charles Sturman | TTPCom |
| John Mersh | TTPCom |
| Sam Turner | TTPCom |
| Christoph Scholtz | University of Bonn |
| Simon Baatz | University of Bonn |

### *Previous versions*

| | |
|---|---|
| Kevin D. Marquess | Hyper Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Kris Fleming | Intel Corporation |
| James P. Gilb | Mobilian |
| David E. Cypher | NIST |
| Nada Golmie | NIST |
| Olaf Joeressen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Charlie Mellone | Motorola, Inc. |
| Harmke de Groot | Philips |
| Terry Bourk | Silicon Wave |
| Steven J. Shellhammer | Symbol |
| Jaap Haartsen | Telefonaktiebolaget LM Ericsson |
| Henrik Hedlund (Section Owner) | Telefonaktiebolaget LM Ericsson |



LifeScan, Inc. Ex. 1019
Page 55 of 1420
IPR2025-00103

| | |
|---|---|
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Joakim Persson | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Onn Haran | Texas Instruments |
| Thomas M. Siep | Texas Instruments |
| Ayse Findikli | Zeevo, Inc. |

*Previous versions [Encryption Sample Data, appendix]*

| | |
|---|---|
| Joel Linksy | RFMD |
| Thomas Müller | Nokia Corporation |
| Thomas Sander | Nokia Corporation |
| Joakim Persson (Section Owner) | Telefonaktiebolaget LM Ericsson |

*Previous versions [Bluetooth Audio, appendix]*

| | |
|---|---|
| Magnus Hansson | Telefonaktiebolaget LM Ericsson |
| Fisseha Mekuria | Telefonaktiebolaget LM Ericsson |
| Mats Omrin | Telefonaktiebolaget LM Ericsson |
| Joakim Persson (Section Owner) | Telefonaktiebolaget LM Ericsson |

*Previous versions [Baseband Timers, appendix]*

| | |
|---|---|
| David E. Cyper | NIST |
| Jaap Haartsen (Section Owner) | Telefonaktiebolaget LM Ericsson |
| Joakim Persson | Telefonaktiebolaget LM Ericsson |
| Ayse Findikli | Zeevo, Inc. |

### 2.3.3 Part C: Link Manager Protocol

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Henrik Hedlund | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Dave Suvak | iAnywhere |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Mattias Edlund | Infineon |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |

| | |
|---|---|
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Peter Hauser | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Jorgen van Parijs | ST |
| Guido Bertoni | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 2.0 + EDR*

| | |
|---|---|
| John Mersh | TTPCom Ltd. |
| Joel Linsky | RF Micro Devices |
| Harald Kafemann | Nokia |
| Simon Morris | CSR |

*Version 1.2*

| | |
|---|---|
| Jennifer Bray | CSR |
| Robin Heydon | CSR |
| Simon Morris | CSR |
| Alexander Thoukydides | CSR |
| Kim Schneider | Digianswer/Motorola |
| Knud Dyring-Olsen | Digianswer/Motorola |
| Henrik Andersen | Digianswer/Motorola |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Roland Hellfajer | Infineon |
| YC Maa | Integrated Programmable Communications, Inc. |
| Steve McGowan | Intel |



LifeScan, Inc. Ex. 1019
Page 57 of 1420
IPR2025-00103

| | |
|---|---|
| Tod Sizer | Lucent Technologies |
| Adrian Stephens | Mobilian |
| Jürgen Schnitzler | Nokia |
| Thomas Müller | Nokia |
| Carmen Kuhl | Nokia |
| Arto Palin | Nokia |
| Thomas Müller | Nokia |
| Roland Matthijssen | Philips |
| Rob Davies | Philips |
| Harmke de Groot | Philips |
| Antonio Salloum | Philips |
| Joel Linsky | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Yariv Raveh | Texas Instruments |
| Tally Shina | Texas Instruments |
| Amihai Kidron | Texas Instruments |
| Yoshimitsu Shimojo | Toshiba |
| Toshiki Kizu | Toshiba |
| John Mersh (section owner) | TTPCom |
| Sam Turner | TTPCom |

*Previous versions*

| | |
|---|---|
| Kim Schneider | Digianswer A/S |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Kris Fleming | Intel Corporation |
| David E. Cypher | NIST |
| Thomas Busse | Nokia Corporation |
| Julien Corthial | Nokia Corporation |
| Olaf Joeressen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Dong Nguyen | Nokia Corporation |
| Harmke de Groot | Philips |
| Terry Bourk | Silicon Wave |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Jaap Haartsen | Telefonaktiebolaget LM Ericsson |
| Tobias Melin (Section Owner) | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Onn Haran | Texas Instruments |
| John Mersh | TTPCom |

### 2.3.4 Part D: Error Codes

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |



LifeScan, Inc. Ex. 1019
Page 58 of 1420
IPR2025-00103

| | |
|---|---|
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Mattias Edlund | Infineon |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Guido Bertoni | ST |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Robin Heydon (section owner) | CSR |
| Roland Hellfajer | Infineon |
| Joel Linsky | Silicon Wave |
| John Mersh | TTPCom |

This part was earlier included in the LMP and HCI functional Specifications.

### 2.3.5 Part E: Bluetooth Host Controller Interface Functional Specification

*Version 2.1 + EDR*



LifeScan, Inc. Ex. 1019
Page 59 of 1420
IPR2025-00103

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Henrik Hedlund | CSR |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Paul Wright | CSR |
| Mattias Edlund | Infineon |
| David Suvak | iAnywhere |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Tim Howes | Symbian |
| Jorgen van Parijs | ST |
| Guido Bertoni | ST |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 2.0 + EDR*

| | |
|---|---|
| Neil Stewart | Tality UK, Ltd. |
| Joel Linsky | RF Micro Devices |



LifeScan, Inc. Ex. 1019
Page 60 of 1420
IPR2025-00103

| | |
|---|---|
| Robin Heydon | CSR |

*Version 1.2*

| | |
|---|---|
| Robin Heydon (section owner | CSR |
| Jennifer Bray | CSR |
| Alexander Thoukydides | CSR |
| Knud Dyring-Olsen | Digianswer/Motorola |
| Henrik Andersen | Digianswer/Motorola |
| Jan Åberg | Ericsson |
| Martin van der Zee | Ericsson |
| Don Liechty | Extended Systems |
| Kevin Marquess | Hyper Corp |
| Roland Hellfajer | Infineon |
| YC Maa | Integrated Programmable Communications, Inc. |
| Steve McGowan | Intel |
| Tod Sizer | Lucent Technologies |
| Tsuyoshi Okada | Matsushita Electric Industrial Co. Ltd |
| Andy Glass | Microsoft |
| Adrian Stephens | Mobilian |
| Jürgen Schnitzler | Nokia |
| Thomas Müller | Nokia |
| Rene Tischer | Nokia |
| Rob Davies | Philips |
| Antonio Salloum | Philips |
| Joel Linsky | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Len Ott | Socket Communications |
| Randy Erman | Taiyo Yuden |
| Yoshimitsu Shimojo | Toshiba |
| Toshiki Kizu | Toshiba |
| Katsuhiro Kinoshita | Toshiba |
| Sam Turner | TTPCom |
| John Mersh | TTPCom |

*Previous versions*

| | |
|---|---|
| Todor Cooklev | 3Com Corporation |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Akihiko Mizutani | IBM Corporation |
| Les Cline | Intel Corporation |
| Bailey Cross | Intel Corporation |
| Kris Fleming | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Jon Inouye | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |



LifeScan, Inc. Ex. 1019
Page 61 of 1420
IPR2025-00103

| | |
|---|---|
| Steve Lo | Intel Corporation |
| Vijay Suthar | Intel Corporation |
| Bruce P. Kraemer | Intersil |
| Greg Muchnik | Motorola, Inc. |
| David E. Cypher | NIST |
| Thomas Busse | Nokia Corporation |
| Julien Courthial | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Dong Nguyen | Nokia Corporation |
| Jürgen Schnitzler | Nokia Corporation |
| Fujio Watanabe | Nokia Corporation |
| Christian Zechlin | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Christian Johansson (Section Owner) | Telefonaktiebolaget LM Ericsson |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |
| Thomas M. Siep | Texas Instruments |
| Masahiro Tada | Toshiba Corporation |
| John Mersh | TTPCom |

### 2.3.6 Part F: Message Sequence Charts

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Henrik Hedlund | CSR |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Mattias Edlund | Infineon |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Motorola |



LifeScan, Inc. Ex. 1019
Page 62 of 1420
IPR2025-00103

| | |
|---|---|
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | Qualcomm |
| Terry Bourk | Qualcomm |
| Guido Bertoni | ST |
| Tim Howes | Symbian |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Robin Heydon (section owner) | CSR |
| Simon Morris | CSR |
| Jan Åberg | Ericsson |
| Christian Gehrmann | Ericsson |
| Joel Linsky | Silicon Wave |
| John Mersh | TTPCom |

*Previous versions*

| | |
|---|---|
| Todor Cooklev | 3Com Corporation |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Greg Muchnik | Motorola, Inc. |
| David E. Cypher | NIST |
| Thomas Busse | Nokia Corporation |
| Dong Nguyen (Section Owner) | Nokia Corporation |
| Fujio Watanabe | Nokia Corporation |
| Christian Johansson | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |

### 2.3.7 Part G: Sample Data

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |



LifeScan, Inc. Ex. 1019
Page 63 of 1420
IPR2025-00103

| | |
|---|---|
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Microsoft |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Guido Bertoni | ST |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Joel Linsky | Silicon Wave |

*Previous versions*

| | |
|---|---|
| Thomas Müller | Nokia Corporation |
| Thomas Sander | Nokia Corporation |
| Joakim Persson (Section Owner) | Telefonaktiebolaget LM Ericsson |

### 2.3.8 Part H: Security Specification

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Henrik Hedlund | CSR |



LifeScan, Inc. Ex. 1019
Page 64 of 1420
IPR2025-00103

| | |
|---|---|
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Microsoft |
| Mattias Edlund | Infineon |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Arto Palin | Nokia |
| Päivi Ruuska | Nokia |
| Dominique Everaere | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | QUALCOMM |
| Terry Bourk | QUALCOMM |
| Guido Bertoni | ST |
| Amihai Kidron | Texas Instruments |
| Eran Reuveni | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |



LifeScan, Inc. Ex. 1019
Page 65 of 1420
IPR2025-00103

## 2.4 [VOL 3] CORE SYSTEM PACKAGE, HOST

### 2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification

*Version 2.1 + EDR*

| | |
|---|---|
| Ayse Findikli | Atheros |
| Robert Hulvey | Broadcom |
| Shawn Ding | Broadcom |
| Robin Heydon | CSR |
| Simon Kingston | CSR |
| Steven Singer | CSR |
| Steven Wenham | CSR |
| Selim Aissi | Intel |
| Penny Chen | Intel |
| Dave Suvak | iAnywhere |
| Josh Benaloh | Microsoft |
| Andy Glass | Microsoft |
| Peter Hauser | Microsoft |
| Joby Lafky | Microsoft |
| Kristin Lauter | Microsoft |
| Dan Simon | Microsoft |
| Don Stanwyck | Microsoft |
| Yacov Yacobi | Microsoft |
| Gideon Yuval | Microsoft |
| Greg Muchnik | Microsoft |
| N Asokan | Nokia |
| Philip Ginzboorg | Nokia |
| Kanji Kerai | Nokia |
| Noel Lobo | Nokia |
| Kaisa Nyberg | Nokia |
| Dominique Everaere | NXP |
| Javier del Prado Pavon | NXP |
| Reinhard Meindl | NXP |
| Joel Linsky | QUALCOMM |
| Terry Bourk | QUALCOMM |
| Tim Howes | Symbian |
| Jorgen van Parijs | ST |
| Amihai Kidron | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Tom Siep | Bluetooth SIG Inc. |
| Carsten Andersen (section owner | CSR |
| Jennifer Bray | CSR |
| Jan Åberg | Ericsson |



LifeScan, Inc. Ex. 1019
Page 66 of 1420
IPR2025-00103

| | |
|---|---|
| Martin van der Zee | Ericsson |
| Sam Ravnborg | Ericsson |
| Stefan Agnani | Ericsson |
| Steve McGowan | Intel Corporation |
| Joby Lafky | Microsoft |
| Doron Holan | Microsoft |
| Andy Glass | Microsoft |
| Brian Redding | Motorola |
| Jürgen Schnitzler | Nokia |
| Thomas Müller | Nokia |
| Rob Davies | Philips |
| Terry Bourk | Silicon Wave |
| Michael Hasling | Tality |

### *Previous versions*

| | |
|---|---|
| Jon Burgess | 3Com Corporation |
| Paul Moran | 3Com Corporation |
| Doug Kogan | Extended Systems |
| Kevin D. Marquess | Hyper Corporation |
| Toru Aihara | IBM Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Uma Gadamsetty | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Jon Inouye | Intel Corporation |
| Steve C. Lo | Intel Corporation |
| Chunrong Zhu | Intel Corporation |
| Sergey Solyanik | Microsoft Corporation |
| David E. Cypher | NIST |
| Nada Golmie | NIST |
| Thomas Busse | Nokia Corporation |
| Rauno Makinen | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Petri Nykänen | Nokia Corporation |
| Peter Ollikainen | Nokia Corporation |
| Petri O. Nurminen | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Jaap Haartsen | Telefonaktiebolaget LM Ericsson |
| Elco Nijboer | Telefonaktiebolaget LM Ericsson |
| Ingemar Nilsson | Telefonaktiebolaget LM Ericsson |
| Stefan Runesson | Telefonaktiebolaget LM Ericsson |
| Gerrit Slot | Telefonaktiebolaget LM Ericsson |
| Johan Sörensen | Telefonaktiebolaget LM Ericsson |
| Goran Svennarp | Telefonaktiebolaget LM Ericsson |
| Mary A. DuVal | Texas Instruments |



LifeScan, Inc. Ex. 1019
Page 67 of 1420
IPR2025-00103

| | |
|---|---|
| Thomas M. Siep | Texas Instruments |
| Kinoshita Katsuhiro | Toshiba Corporation |

### 2.4.2 Part B: Service Discovery Protocol (SDP)

*Version 2.1 + EDR*

| | |
|---|---|
| Robin Heydon | CSR |
| Mattias Edlund | Infineon |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Joel Linsky | QUALCOMM |
| Terry Bourk | QUALCOMM |
| Amihai Kidron | Texas Instruments |
| Shimojo Yoshimitsu | Toshiba |

*Previous versions*

| | |
|---|---|
| Ned Plasson | 3Com Corporation |
| John Avery | Convergence |
| Jason Kronz | Convergence |
| Chatschik Bisdikian | IBM Corporation |
| Parviz Kermani | IBM Corporation |
| Brent Miller | IBM Corporation |
| Dick Osterman | IBM Corporation |
| Bob Pascoe | IBM Corporation |
| Jon Inouye | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Jay Eaglstun | Motorola, Inc. |
| Dale Farnsworth (Section Owner) | Motorola, Inc. |
| Jean-Michel Rosso | Motorola, Inc. |
| Jan Grönholm | Nokia Corporation |
| Kati Rantala | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Kazuaki Iwamura | Toshiba Corporation |



LifeScan, Inc. Ex. 1019
Page 68 of 1420
IPR2025-00103

### 2.4.3 Part C Generic Access Profile

*Version 2.1 + EDR*

| | |
|---|---|
| Robin Heydon | CSR |
| Mattias Edlund | Infineon |
| Päivi Ruuska | Nokia |
| Arto Palin | Nokia |
| Joel Linsky | QUALCOMM |
| Terry Bourk | QUALCOMM |
| Shimojo Yoshimitsu | Toshiba |

*Version 1.2*

| | |
|---|---|
| Jennifer Bray | CSR |
| Alexander Thoukydides | CSR |
| Christian Gehrmann | Ericsson |
| Henrik Hedlund | Ericsson |
| Jan Åberg | Ericsson |
| Stefan Agnani | Ericsson |
| Thomas Müller | Nokia |
| Joel Linsky | Silicon Wave |
| Terry Bourk | Silicon Wave |
| Katsuhiro Kinoshita | Toshiba |

*Previous versions*

| | |
|---|---|
| Ken Morley | 3Com Corporation |
| Chatschik Bisdikian | IBM Corporation |
| Jon Inouye | Intel Corporation |
| Brian Redding | Motorola, Inc. |
| David E. Cypher | NIST |
| Stephane Bouet | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Martin Roter | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Patric Lind (Section Owner) | Telefonaktiebolaget LM Ericsson |
| Erik Slotboom | Telefonaktiebolaget LM Ericsson |
| Johan Sörensen | Telefonaktiebolaget LM Ericsson |

### 2.4.4 Part D: Test Support

*Version 1.2*

| | |
|---|---|
| Emilio Mira Escartis | Cetecom |
| Robin Heydon | CSR |
| Jennifer Bray | CSR |
| Stefan Agnani (section owner) | Ericsson |
| Terry Bourk | Silicon Wave |



LifeScan, Inc. Ex. 1019
Page 69 of 1420
IPR2025-00103

| | |
|---|---|
| Joel Linsky | Silicon Wave |
| Michael Hasling | Tality |

*Previous versions [Test Mode]*

| | |
|---|---|
| Jeffrey Schiffer | Intel Corporation |
| David E. Cypher | NIST |
| Daniel Bencak | Nokia Corporation |
| Arno Kefenbaum | Nokia Corporation |
| Thomas Müller (Section Owner) | Nokia Corporation |
| Roland Schmale | Nokia Corporation |
| Fujio Watanabe | Nokia Corporation |
| Stefan Agnani | Telefonaktiebolaget LM Ericsson |
| Mårten Mattsson | Telefonaktiebolaget LM Ericsson |
| Tobias Melin | Telefonaktiebolaget LM Ericsson |
| Lars Nord | Telefonaktiebolaget LM Ericsson |
| Fredrik Töörn | Telefonaktiebolaget LM Ericsson |
| John Mersh | TTPCom |
| Ayse Findikli | Zeevo, Inc. |

*Previous versions [Test Control Interface]*

| | |
|---|---|
| Mike Feldman | Motorola, Inc. |
| Thomas Müller | Nokia Corporation |
| Stefan Agnani (Section Owner) | Telefonaktiebolaget LM Ericsson |
| Mårten Mattsson | Telefonaktiebolaget LM Ericsson |
| Dan Sönnerstam | Telefonaktiebolaget LM Ericsson |



LifeScan, Inc. Ex. 1019
Page 70 of 1420
IPR2025-00103

## 2.5 [VOL 4] HOST CONTROLLER INTERFACE [TRANSPORT LAYER]

| | |
|---|---|
| Toru Aihara | IBM Corporation |
| Edgar Kerstan | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Patrick Kane | Motorola, Inc. |
| Uwe Gondrum | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Christian Zechlin | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Sven Jerlhagen | Telefonaktiebolaget LM Ericsson |
| Christian Johansson | Telefonaktiebolaget LM Ericsson |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Lars Novak | Telefonaktiebolaget LM Ericsson |
| Masahiro Tada | Toshiba Corporation |
| Steve Ross | Digianswer A/S |
| Chatschik Bisdikian | IBM Corporation |
| Les Cline | Intel Corporation |
| Brad Hosler | Intel Corporation |
| John Howard | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Kosta Koeman | Intel Corporation |
| John McGrath | Intel Corporation |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Leonard Ott | Socket Communications |
| Rebecca O'Dell | Signia Technologies |
| Tsuyoshi Okada | Matsushita Electric |
| Robin Heydon | CSR |
| Toru Aihara | IBM Corporation |
| Edgar Kerstan | IBM Corporation |
| Nathan Lee | IBM Corporation |
| Kris Fleming | Intel Corporation |
| Robert Hunter | Intel Corporation |
| Patrick Kane | Motorola, Inc. |
| Uwe Gondrum | Nokia Corporation |
| Thomas Müller | Nokia Corporation |
| Christian Zechlin | Nokia Corporation |
| Johannes Elg | Telefonaktiebolaget LM Ericsson |
| Sven Jerlhagen | Telefonaktiebolaget LM Ericsson |
| Christian Johansson | Telefonaktiebolaget LM Ericsson |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Lars Novak | Telefonaktiebolaget LM Ericsson |
| Masahiro Tada | Toshiba Corporation |



LifeScan, Inc. Ex. 1019
Page 71 of 1420
IPR2025-00103

| | |
|---|---|
| Steve Ross | Digianswer A/S |
| Chatschik Bisdikian | IBM Corporation |
| Les Cline | Intel Corporation |
| Brad Hosler | Intel Corporation |
| John Howard | Intel Corporation |
| Srikanth Kambhatla | Intel Corporation |
| Kosta Koeman | Intel Corporation |
| John McGrath | Intel Corporation |
| Patrik Lundin | Telefonaktiebolaget LM Ericsson |
| Leonard Ott | Socket Communications |
| Rebecca O'Dell | Signia Technologies |
| Tsuyoshi Okada | Matsushita Electric |
| Robin Heydon | CSR |



LifeScan, Inc. Ex. 1019
Page 72 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 73 of 1420
IPR2025-00103





LifeScan, Inc. Ex. 1019
Page 74 of 1420
IPR2025-00103

Specification Volume 1

# SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Wireless Connections Made Easy



# Architecture & Terminology Overview

**Covered Core Package version:**
**2.1 + EDR**

Bluetooth®



LifeScan, Inc. Ex. 1019
Page 75 of 1420
IPR2025-00103

## Revision History

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

## Contributors

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

## Web Site

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

## Disclaimer and Copyright Notice

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

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

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.



LifeScan, Inc. Ex. 1019
Page 76 of 1420
IPR2025-00103

THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

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

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

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

Copyright © 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007

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

*Third-party brands and names are the property of their respective owners.

LifeScan, Inc. Ex. 1019
Page 78 of 1420
IPR2025-00103

# TABLE OF CONTENTS

**Part A**
**ARCHITECTURE**


**1 General Description ..... 13**
1.1 Overview of Operation ..... 13
1.2 Nomenclature ..... 15

**2 Core System Architecture ..... 20**
2.1 Core Architectural Blocks ..... 23
2.1.1 Channel manager ..... 23
2.1.2 L2CAP resource manager ..... 23
2.1.3 Device manager ..... 24
2.1.4 Link manager ..... 24
2.1.5 Baseband resource manager ..... 24
2.1.6 Link controller ..... 25
2.1.7 RF ..... 25

**3 Data Transport Architecture ..... 26**
3.1 Core Traffic Bearers ..... 27
3.1.1 Framed data traffic ..... 28
3.1.2 Unframed data traffic ..... 29
3.1.3 Reliability of traffic bearers ..... 29
3.2 Transport Architecture Entities ..... 31
3.2.1 Bluetooth generic packet structure ..... 31
3.3 Physical Channels ..... 33
3.3.1 Basic piconet channel ..... 34
3.3.2 Adapted piconet channel ..... 35
3.3.3 Inquiry scan channel ..... 36
3.3.4 Page scan channel ..... 37
3.4 Physical Links ..... 38
3.4.1 Links supported by the basic and adapted piconet physical channel 39
3.4.2 Links supported by the scanning physical channels ..... 40
3.5 Logical Links and Logical Transports ..... 40
3.5.1 Casting ..... 41
3.5.2 Scheduling and acknowledgement scheme ..... 42
3.5.3 Class of data ..... 42
3.5.4 Asynchronous connection-oriented (ACL) ..... 43
3.5.5 Synchronous connection-oriented (SCO) ..... 44




LifeScan, Inc. Ex. 1019
Page 79 of 1420
IPR2025-00103

3.5.6 Extended synchronous connection-oriented (eSCO) .... 44
3.5.7 Active slave broadcast (ASB) .... 45
3.5.8 Parked slave broadcast (PSB) .... 46
3.5.9 Logical links .... 47
3.5.10 User Asynchronous/Isochronous Logical Link (ACL-U) 48
3.5.11 User Synchronous/Extended Synchronous Logical Links (SCO-S/eSCO-S) .... 48
3.6 L2CAP Channels .... 49

**4 Communication Topology .... 50**
4.1 Piconet Topology .... 50
4.2 Operational Procedures and Modes .... 52
4.2.1 Inquiry (Discovering) Procedure .... 52
4.2.2 Paging (Connecting) Procedure .... 53
4.2.3 Connected mode .... 53
4.2.4 Hold mode .... 54
4.2.5 Sniff mode .... 54
4.2.6 Parked state .... 55
4.2.7 Role switch procedure .... 55
4.2.8 Enhanced Data Rate .... 56

**5 Secure Simple Pairing Overview .... 57**
5.1 Security Goals .... 57
5.2 Passive Eavesdropping Protection .... 57
5.3 Man-In-The-Middle Protection .... 58
5.4 Association Models .... 59
5.4.1 Numeric Comparison .... 59
5.4.2 Just Works .... 59
5.4.3 Out of Band .... 60
5.4.4 Passkey Entry .... 60
5.4.5 Association Model Overview .... 61

**Part B**
**ACRONYMS & ABBREVIATIONS**

**1 List of Acronyms and Abbreviations .... 65**

**Part C**
**CORE SPECIFICATION**
**CHANGE HISTORY**

**1 Deprecated Features .... 76**
**2 Changes from V1.1 to V1.2 .... 77**

2.1 New Features....77
2.2 Structure Changes....77
2.3 Deprecated Features list....77
2.4 Changes in Wording....78
2.5 Nomenclature Changes....78

**3 Changes from V1.2 to V2.0 + EDR....79**
3.1 New Features....79
3.2 Deprecated Features....79

**4 Changes from v2.0 + EDR to v2.1 + EDR....80**
4.1 New features....80
4.2 Deprecated Features....80

**Part D**
**MIXING OF SPECIFICATION VERSIONS**

**1 Mixing of Specification Versions....84**
1.1 Features and their Types....85

**Part E**
**IEEE LANGUAGE**

**1 Use of IEEE Language....90**
1.1 Shall....90
1.2 Must....91
1.3 Will....91
1.4 Should....91
1.5 May....91
1.6 Can....92



LifeScan, Inc. Ex. 1019
Page 81 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 82 of 1420
IPR2025-00103

# ARCHITECTURE



LifeScan, Inc. Ex. 1019
Page 83 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 84 of 1420
IPR2025-00103

# CONTENTS

**1 General Description ..... 13**
1.1 Overview of Operation ..... 13
1.2 Nomenclature ..... 15
**2 Core System Architecture ..... 20**
2.1 Core Architectural Blocks ..... 23
2.1.1 Channel manager ..... 23
2.1.2 L2CAP resource manager ..... 23
2.1.3 Device manager ..... 24
2.1.4 Link manager ..... 24
2.1.5 Baseband resource manager ..... 24
2.1.6 Link controller ..... 25
2.1.7 RF ..... 25
**3 Data Transport Architecture ..... 26**
3.1 Core Traffic Bearers ..... 27
3.1.1 Framed data traffic ..... 28
3.1.2 Unframed data traffic ..... 29
3.1.3 Reliability of traffic bearers ..... 29
3.2 Transport Architecture Entities ..... 31
3.2.1 Bluetooth generic packet structure ..... 31
3.3 Physical Channels ..... 33
3.3.1 Basic piconet channel ..... 34
3.3.2 Adapted piconet channel ..... 35
3.3.3 Inquiry scan channel ..... 36
3.3.4 Page scan channel ..... 37
3.4 Physical Links ..... 38
3.4.1 Links supported by the basic and adapted piconet physical channel ..... 39
3.4.2 Links supported by the scanning physical channels ..... 40
3.5 Logical Links and Logical Transports ..... 40
3.5.1 Casting ..... 41
3.5.2 Scheduling and acknowledgement scheme ..... 42
3.5.3 Class of data ..... 42
3.5.4 Asynchronous connection-oriented (ACL) ..... 43
3.5.5 Synchronous connection-oriented (SCO) ..... 44
3.5.6 Extended synchronous connection-oriented (eSCO) ..... 44
3.5.7 Active slave broadcast (ASB) ..... 45



LifeScan, Inc. Ex. 1019
Page 85 of 1420
IPR2025-00103

3.5.8 Parked slave broadcast (PSB) ..... 46
3.5.9 Logical links ..... 47
3.5.10 User Asynchronous/Isochronous Logical Link (ACL-U) 48
3.5.11 User Synchronous/Extended Synchronous Logical Links (SCO-S/eSCO-S) ..... 48
3.6 L2CAP Channels ..... 49

**4 Communication Topology ..... 50**
4.1 Piconet Topology ..... 50
4.2 Operational Procedures and Modes ..... 52
4.2.1 Inquiry (Discovering) Procedure ..... 52
4.2.2 Paging (Connecting) Procedure ..... 53
4.2.3 Connected mode ..... 53
4.2.4 Hold mode ..... 54
4.2.5 Sniff mode ..... 54
4.2.6 Parked state ..... 55
4.2.7 Role switch procedure ..... 55
4.2.8 Enhanced Data Rate ..... 56

**5 Secure Simple Pairing Overview ..... 57**
5.1 Security Goals ..... 57
5.2 Passive Eavesdropping Protection ..... 57
5.3 Man-In-The-Middle Protection ..... 58
5.4 Association Models ..... 59
5.4.1 Numeric Comparison ..... 59
5.4.2 Just Works ..... 59
5.4.3 Out of Band ..... 60
5.4.4 Passkey Entry ..... 60
5.4.5 Association Model Overview ..... 61



LifeScan, Inc. Ex. 1019
Page 86 of 1420
IPR2025-00103

# 1 GENERAL DESCRIPTION

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

The Bluetooth core system consists of an RF transceiver, baseband, 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 chapter of the specification provides an overview of the Bluetooth system architecture, communication topologies and data transport features. The text in this chapter of the specification should be treated as informational and used as a background and for context-setting.

## 1.1 OVERVIEW OF OPERATION

The Bluetooth RF (physical layer) 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 FHSS carriers. RF operation uses a shaped, binary frequency modulation to minimize transceiver complexity. The symbol rate is 1 Megasymbol per second (Ms/s) supporting the bit rate of 1 Megabit per second (Mb/s) or, with Enhanced Data Rate, a gross air bit rate of 2 or 3Mb/s. These modes are known as Basic Rate and Enhanced Data Rate respectively.

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 Bluetooth wireless technology.

Devices in a piconet use a specific frequency hopping pattern, which is algorithmically determined by certain fields in the Bluetooth address and 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 Bluetooth co-existence with static (non-hopping) ISM systems when these are co-located.

The physical channel is sub-divided into time units known as slots. Data is transmitted between Bluetooth devices in packets, that 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 transmis-



LifeScan, Inc. Ex. 1019
Page 87 of 1420
IPR2025-00103

sion or reception of packets. Bluetooth technology 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 Section 3.3 on page 33 to Section 3.6 on page 49 but are introduced here to aid the understanding of the remainder of this section.

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.

A control protocol for the baseband and physical layers is carried over logical links in addition to user data. This is the link manager protocol (LMP). Devices that are active in a piconet have a default asynchronous connection-oriented logical transport that is used to transport the LMP protocol signaling. For historical reasons this is known 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 Link Manager function uses LMP to control the operation of devices in the piconet and provide services to manage the lower architectural layers (radio layer and baseband layer). The LMP protocol is only carried on the default ACL logical transport and the default broadcast logical transport.

Above the baseband layer the L2CAP layer provides a channel-based abstraction to applications and services. It carries out segmentation and reassembly of application data and multiplexing and de-multiplexing 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 protocol may be carried on any logical link that supports the L2CAP protocol.



LifeScan, Inc. Ex. 1019
Page 88 of 1420
IPR2025-00103

## 1.2 NOMENCLATURE

Where the following terms appear in the specification they have the meaning given in Table 1.1 on page 15.

| | |
|---|---|
| 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. |
| Active Slave Broadcast (ASB) | The Active Slave Broadcast logical transport that is used to transport L2CAP user traffic to all active devices in the piconet. See Section 3.5.7 on page 45 |
| 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. |
| Bluetooth | Bluetooth is a wireless communication link, operating in the unlicensed ISM band at 2.4 GHz using a frequency hopping transceiver. It allows real-time AV and data communications between Bluetooth Hosts. The link protocol is based on time slots. |
| Bluetooth Baseband | The part of the Bluetooth system that specifies or implements the medium access and physical layer procedures to support the exchange of real-time voice, data information streams, and ad hoc networking between Bluetooth Devices. |
| Bluetooth Clock | A 28 bit clock internal to a Bluetooth controller sub-system that ticks every 312.5μs. The value of this clock defines the slot numbering and timing in the various physical channels. |
| Bluetooth Controller | A sub-system containing the Bluetooth RF, baseband, resource controller, link manager, device manager and a Bluetooth HCI. |
| Bluetooth Device | A Bluetooth Device is a device that is capable of short-range wireless communications using the Bluetooth system. |
| Bluetooth Device Address | A 48 bit address used to identify each Bluetooth device. |
| BD_ADDR | The Bluetooth Device Address, BD_ADDR, is used to identify a Bluetooth device. |
| Bluetooth HCI | The Bluetooth Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities. |

*Table 1.1: Nomenclature.*



LifeScan, Inc. Ex. 1019
Page 89 of 1420
IPR2025-00103

| | |
|---|---|
| Bluetooth Host | A Bluetooth Host is a computing device, peripheral, cellular telephone, access point to PSTN network or LAN, etc. A Bluetooth Host attached to a Bluetooth Controller may communicate with other Bluetooth Hosts attached to their Bluetooth Controllers as well. |
| Channel | Either a physical channel or an L2CAP channel, depending on the context. |
| 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 L2CAP channel. |
| Connectable device | A Bluetooth device in range that periodically listens on its page scan physical channel and will respond to a page on that channel. |
| Connected devices | Two Bluetooth devices in the same piconet and with a physical link between them. |
| 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.) |
| Connection | A connection between two peer applications or higher layer protocols mapped onto an L2CAP channel. |
| Connection establishment | A procedure for creating a connection mapped onto a channel. |
| Coverage area | The area where two Bluetooth devices can exchange messages with acceptable quality and performance. |
| Creation of a secure connection | A procedure of establishing a connection, including authentication and encryption. |
| 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). |
| Device discovery | A procedure for retrieving the Bluetooth device address, clock, class-of-device field and used page scan mode from discoverable devices. |
| Discoverable device | A Bluetooth device in range that periodically listens on an inquiry scan physical channel and will respond to an inquiry on that channel. Discoverable device are normally also connectable. |
| Inquiring device | A Bluetooth device that is carrying out the inquiry procedure. |
| Inquiry | A procedure where a Bluetooth device transmits inquiry messages and listens for responses in order to discover the other Bluetooth devices that are within the coverage area. |

*Table 1.1: Nomenclature.*



LifeScan, Inc. Ex. 1019
Page 90 of 1420
IPR2025-00103

| | |
|---|---|
| Inquiry scan | A procedure where a Bluetooth device listens for inquiry messages received on its inquiry scan physical channel. |
| Interoperability | The ability of two or more systems or components to exchange information and to use the information that has been exchanged. |
| Isochronous data | Information in a stream where each information entity in the stream is bound by a time relationship to previous and successive entities. |
| Known device | A Bluetooth device for which at least the BD_ADDR is stored. |
| L2CAP Channel | A logical connection on L2CAP level between two devices serving a single application or higher layer protocol. |
| L2CAP Channel establishment | A procedure for establishing a logical connection on L2CAP level. |
| Link establishment | A procedure for establishing the default ACL link and hierarchy of links and channels between devices. |
| Link | Shorthand for a logical link. |
| Link key | A secret key that is known by two devices and is used in order to authenticate each device to the other |
| LMP authentication | An LMP level procedure for verifying the identity of a remote device. |
| LMP pairing | A procedure that authenticates two devices and creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection. |
| Logical link | The lowest architectural level used to offer independent data transport services to clients of the Bluetooth system. |
| Logical transport | Used in Bluetooth to represent commonality between different logical links due to shared acknowledgement protocol and link identifiers. |
| Name discovery | A procedure for retrieving the user-friendly name (the Bluetooth device name) of a connectable device. |
| Packet | Format of aggregated bits that are transmitted on a physical channel. |
| 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 time-out occurs. |
| Page scan | A procedure where a device listens for page messages received on its page scan physical channel. |
| Paging device | A Bluetooth device that is carrying out the page procedure. |

*Table 1.1: Nomenclature.*



LifeScan, Inc. Ex. 1019
Page 91 of 1420
IPR2025-00103

| | |
|---|---|
| Paired device | A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase). |
| 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. |
| Passkey | A 6-digit number used to authenticate connections when Secure Simple Pairing is used. |
| Physical Channel | Characterized by synchronized occupancy of a sequence of RF carriers by one or more devices. A number of physical channel types exist with characteristics defined for their different purposes. |
| Physical Link | A Baseband-level connection between two devices established using paging. |
| 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. |
| Piconet Physical Channel | A Channel that is divided into time slots in which each slot is related to an RF hop frequency. Consecutive hops normally correspond to different RF hop frequencies and occur at a standard hop rate of 1600 hops/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 use. |
| Piconet Master | The device in a piconet whose Bluetooth Clock and Bluetooth Device Address are used to define the piconet physical channel characteristics. |
| Piconet Slave | Any device in a piconet that is not the Piconet Master, but is connected to the Piconet Master. |
| PIN | A user-friendly number that can be used to authenticate connections to a device before pairing has taken place. |
| PMP | A Participant in Multiple Piconets. A device that is concurrently a member of more than one piconet, which it achieves using time division multiplexing (TDM) to interleave its activity on each piconet physical channel. |
| The Parked Slave Broadcast (PSB) | The Parked Slave Broadcast logical transport that is used for communications between the master and parked devices. Section 3.5.8 on page 46. |
| Scatternet | Two or more piconets that include one or more devices acting as PMPs. |
| Service Layer Protocol | A protocol that uses an L2CAP channel for transporting PDUs. |

*Table 1.1: Nomenclature.*



LifeScan, Inc. Ex. 1019
Page 92 of 1420
IPR2025-00103

| | |
|---|---|
| Service discovery | Procedures for querying and browsing for services offered by or through another Bluetooth device. |
| Silent device | A Bluetooth device appears as silent to a remote device if it does not respond to inquiries made by the remote device. |
| Unknown device | A Bluetooth device for which no information (Bluetooth Device Address, link key or other) is stored. |

*Table 1.1: Nomenclature.*



LifeScan, Inc. Ex. 1019
Page 93 of 1420
IPR2025-00103

# 2 CORE SYSTEM ARCHITECTURE

The Bluetooth core system covers the four lowest layers and associated protocols defined by the Bluetooth specification as well as one common service layer protocol, the Service Discovery Protocol (SDP) and the overall profile requirements are specified in the Generic Access Profile (GAP). A complete Bluetooth application requires a number of additional service and higher layer protocols that are defined in the Bluetooth specification, but are not described here. The core system architecture is shown in Figure 2.1 on page 20 except for SDP that is not shown for clarity.

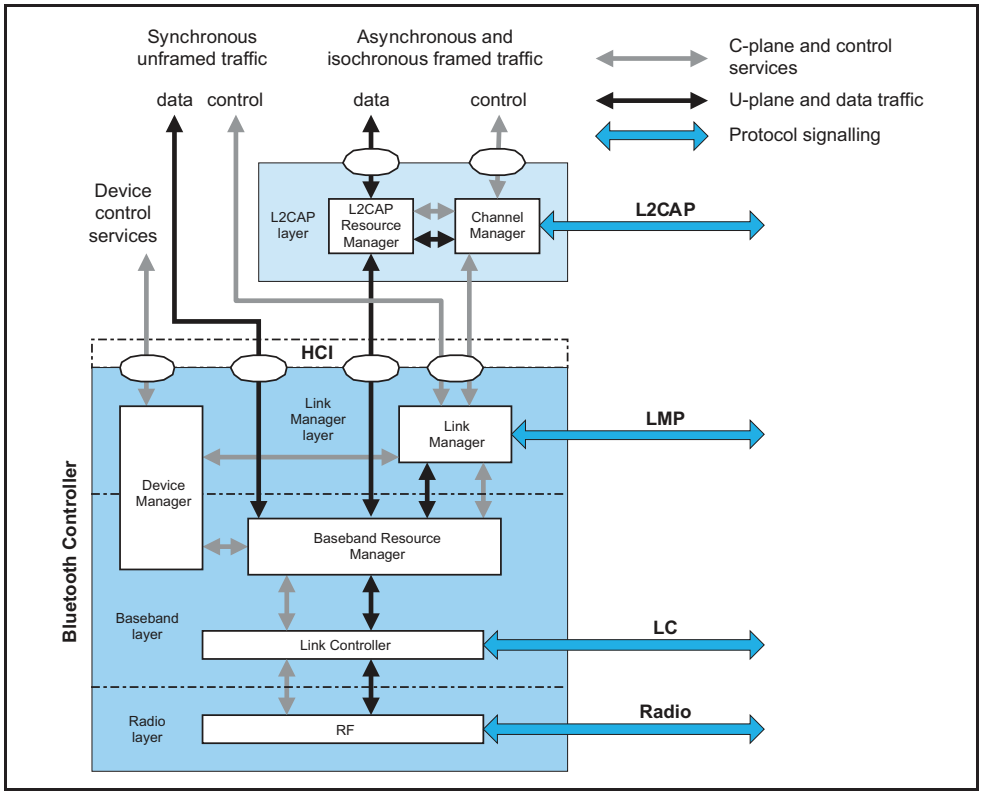



*Figure 2.1: Bluetooth core system architecture*

Figure 2.1 on page 20 shows the four lowest layers, each with its associated communication protocol. The lowest three layers are sometimes grouped into a subsystem known as the Bluetooth controller. This is a common implementation involving a standard physical communications interface between the Bluetooth controller and remainder of the Bluetooth system including the L2CAP, service and higher layers (known as the Bluetooth host.) Although this interface is optional the architecture is designed to allow for its existence and characteristics. The Bluetooth specification enables interoperability between independent Bluetooth systems by defining the protocol messages exchanged between equivalent layers, and also interoperability between independent



LifeScan, Inc. Ex. 1019
Page 94 of 1420
IPR2025-00103

Bluetooth subsystems by defining a common interface between Bluetooth controllers and Bluetooth hosts.

A number of functional blocks and the path of services and data between them are shown. The functional blocks shown in the diagram are informative; in general the Bluetooth specification does not define the details of implementations except where this is required for interoperability. Thus the functional blocks in Figure 2.1 on page 20 are shown in order to aid description of the system behavior. An implementation may be different from the system shown in Figure 2.1 on page 20.

Standard interactions are defined for all inter-device operation, where Bluetooth devices exchange protocol signaling according to the Bluetooth specification. The Bluetooth core system protocols are the Radio (RF) protocol, Link Control (LC) protocol, Link Manager (LM) protocol and Logical Link Control and Adaptation protocol (L2CAP), all of which are fully defined in subsequent parts of the Bluetooth specification. In addition the Service Discovery Protocol (SDP) is a service layer protocol required by all Bluetooth applications.

The Bluetooth core system offers services through a number of service access points that are shown in the diagram as ellipses. These services consist of the basic primitives that control the Bluetooth core system. The services can be split into three types. There are device control services that modify the behavior and modes of a Bluetooth device, transport control services that create, modify and release traffic bearers (channels and links), and 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.

A service interface to the Bluetooth controller subsystem is defined such that the Bluetooth controller may be considered a standard part. In this configuration the Bluetooth controller operates the lowest three layers and the L2CAP layer is contained with the rest of the Bluetooth application in a host system. The standard interface is called the Host to Controller Interface (HCI) and its service access points are represented by the ellipses on the upper edge of the Bluetooth controller subsystem in Figure 2.1 on page 20. Implementation of this standard service interface is optional.

As the Bluetooth architecture is defined with the possibility of separate host and controller communicating through an HCI, a number of general assumptions are made. The Bluetooth controller is assumed to have limited data buffering capabilities in comparison with the host. Therefore the L2CAP layer is expected to carry out some simple resource management when submitting L2CAP PDUs to the controller for transport to a peer device. This includes segmentation of L2CAP 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 Baseband layer provides the basic ARQ protocol in Bluetooth. The L2CAP layer 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.

Although these assumptions may not be required for embedded Bluetooth implementations that combine all layers in a single system, the general architectural and QoS models are defined with these assumptions in mind, in effect a lowest common denominator.

Automated conformance testing of implementations of the Bluetooth core system is required. This is achieved by allowing the tester to control the implementation through the RF interface, which is common to all Bluetooth systems, and through the Test Control Interface (TCI), which is only required for conformance testing.

The tester uses exchanges with the Implementation Under Test (IUT) through the RF interface to ensure the correct responses to requests from remote devices. The tester controls the IUT through the TCI to cause the IUT to originate exchanges through the RF interface so that these can also be verified as conformant.

The TCI uses a different command-set (service interface) for the testing of each architectural layer and protocol. A subset of the HCI command-set is used as the TCI service interface for each of the layers and protocols within the Bluetooth Controller subsystem. A separate service interface is used for testing the L2CAP layer and protocol. As an L2CAP service interface is not defined in the Bluetooth core specification it is defined separately in the Test Control Interface specification. Implementation of the L2CAP service interface is only required for conformance testing.



LifeScan, Inc. Ex. 1019
Page 96 of 1420
IPR2025-00103

## 2.1 CORE ARCHITECTURAL BLOCKS

This section describes the function and responsibility of each of the blocks shown in Figure 2.1 on page 20. An implementation is not required to follow the architecture described above, though every implementation shall conform to the protocol specifications described in subsequent parts of the Bluetooth specification, and shall implement the behavioral aspects of the system outlined below and specified in subsequent parts of the Bluetooth specification.

### 2.1.1 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 protocol 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 link manager to create new logical links (if necessary) and to configure these links to provide the required quality of service for the type of data being transported.

### 2.1.2 L2CAP resource manager

The L2CAP resource manager block is responsible for managing the ordering of submission of PDU fragments to the baseband and some relative scheduling between channels to ensure that L2CAP channels with QoS commitments are not denied access to the physical channel due to Bluetooth controller resource exhaustion. This is required because the architectural model does not assume that the Bluetooth 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 Bluetooth data transport model assumes well-behaved applications, and does not define how an implementation is expected to deal with this problem.



LifeScan, Inc. Ex. 1019
Page 97 of 1420
IPR2025-00103

### 2.1.3 Device manager

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

The device manager requests access to the transport medium from the baseband 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 local name, any stored link keys, and other functionality.

### 2.1.4 Link manager

The link manager 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 link manager achieves this by communicating with the link manager in remote Bluetooth devices using the Link Management Protocol (LMP.)

The LM protocol 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.

### 2.1.5 Baseband resource manager

The baseband resource manager is responsible for all access to the radio medium. 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 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 Bluetooth radio. This includes (for example) the normal exchange of data between connected devices over logical links, and logical transports, as well as the use of the radio medium to carry out inquiries, make connections, be discoverable or connectable, or to take readings from unused carriers during the use of adaptive frequency hopping 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



LifeScan, Inc. Ex. 1019
Page 98 of 1420
IPR2025-00103

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.

### 2.1.6 Link controller

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

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

### 2.1.7 RF

The RF block is responsible for transmitting and receiving packets of information on the physical channel. A control path between the baseband and the RF block allows the baseband 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 baseband into required formats.



LifeScan, Inc. Ex. 1019
Page 99 of 1420
IPR2025-00103

# 3 DATA TRANSPORT ARCHITECTURE

The Bluetooth data transport system follows a layered architecture. This description of the Bluetooth system describes the Bluetooth core transport layers up to and including L2CAP channels. All Bluetooth operational modes follow the same generic transport architecture, which is shown in Figure 3.1 on page 26.

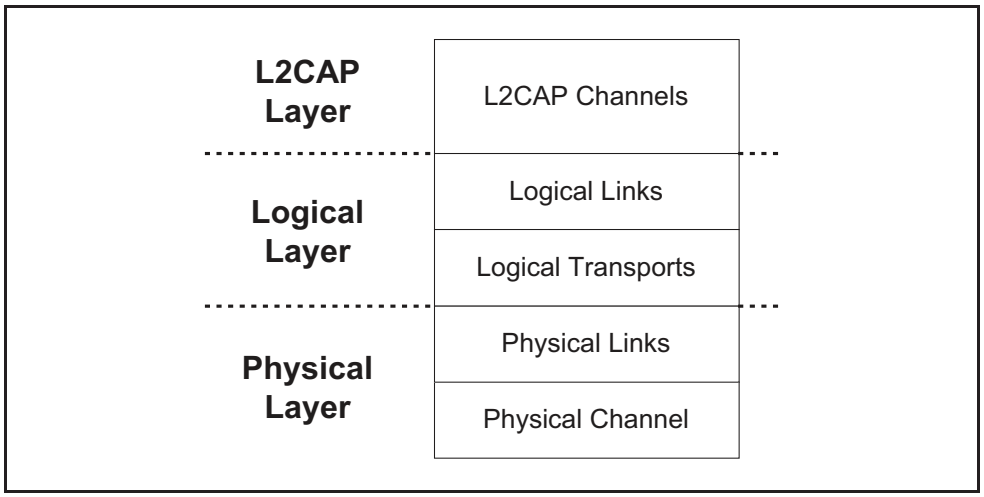



*Figure 3.1: Bluetooth generic data transport architecture*

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

Previous versions of the Bluetooth specification 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 their shared use of resources such as the LT_ADDR and acknowledgement/repeat request (ARQ) scheme. Hence the architecture is incapable of representing these logical transports with a single transport layer. The additional logical transport layer goes some way towards describing this behavior.



LifeScan, Inc. Ex. 1019
Page 100 of 1420
IPR2025-00103

## 3.1 CORE TRAFFIC BEARERS

The Bluetooth core system provides a number of standard traffic bearers for the transport of service protocol and application data. These are shown in Figure 3.2 on page 27 below (for ease of representation this is shown with higher layers to the left and lower layers to the right).

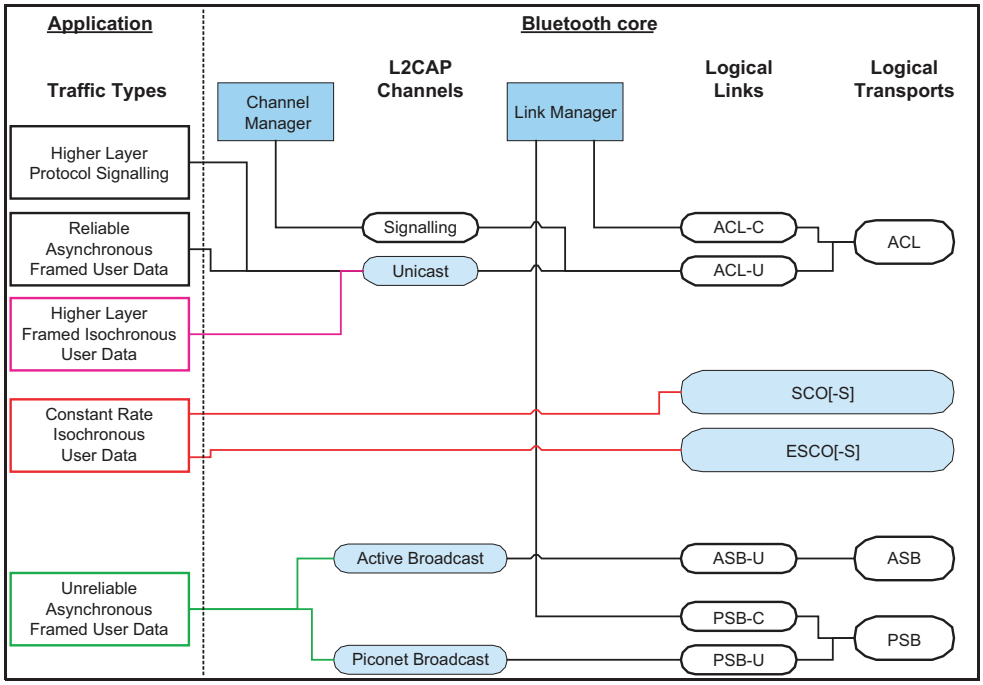



*Figure 3.2: Bluetooth traffic bearers*

The core traffic bearers that are available to applications are shown in Figure 3.2 on page 27 as the shaded rounded rectangles. The architectural layers that are defined to provide these services are described in Section 2 on page 20. A number of data traffic types are shown on the left of the diagram linked to the traffic bearers that are typically suitable for transporting that type of data traffic.

The logical links are named using the names of the associated logical transport and a suffix that indicates the type of data that is transported. (C for control links carrying LMP messages, U for L2CAP links carrying user data (L2CAP PDUs) and S for 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 protocol is being discussed, or the ACL-U logical link when the L2CAP layer is being discussed.

The mapping of application traffic types to Bluetooth core traffic bearers in Figure 3.2 on page 27 is based on matching the traffic characteristics with the



LifeScan, Inc. Ex. 1019
Page 101 of 1420
IPR2025-00103

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 Bluetooth 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 ASB-U or PSB-U logical links. This will probably be more efficient in terms of bandwidth (if the physical channel quality is not too degraded.) Use of alternative transport paths to those in Figure 3.2 on page 27 is only acceptable if the characteristics of the application traffic type are preserved.

Figure 3.2 on page 27 shows a number of application traffic types. These are used to classify the types of data that may be submitted to the Bluetooth core system. The original data traffic type may not be the same as the type that is submitted to the Bluetooth core system if an intervening process modifies it. For example, video data is generated at a constant rate but an intermediate coding process may alter this to variable rate, e.g. by MPEG4 encoding. For the purposes of the Bluetooth core system, only the characteristic of the submitted data is of interest.

### 3.1.1 Framed data traffic

The L2CAP layer 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 Bluetooth core system.)

Connection-oriented L2CAP channels may be created for transport of unicast (point-to-point) data between two Bluetooth 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 uni-directional. Unicast L2CAP channels may be uni-directional or bi-directional.

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 is isochronous, and therefore has a limited lifetime after which it becomes invalid, or that the data should be delivered within a given time period, or that the data is 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 baseband logical link, possibly multiplexing this onto the baseband logical link with other L2CAP channels with similar characteristics.



LifeScan, Inc. Ex. 1019
Page 102 of 1420
IPR2025-00103

### 3.1.2 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 is a pure stream, then it may avoid the use of L2CAP channels and make direct use of a baseband logical link.

The Bluetooth core system supports the direct transport of application data that is isochronous and of a constant rate (either bit-rate, or frame-rate for pre-framed data), using a SCO-S or eSCO-S logical link. These logical links reserve physical channel bandwidth and provide a constant rate transport locked to the piconet clock. Data is 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. Enhanced Data Rate operation is supported for eSCO, but not for SCO logical transports. SCO and eSCO logical transports do not support multiplexed logical links or any further layering within the Bluetooth 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 baseband, and 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 is isochronous and of a variable rate, then this may only be carried by the L2CAP unicast channel, and hence will be treated as framed data.

### 3.1.3 Reliability of traffic bearers

Bluetooth is a wireless communications system. In poor RF environments, this system should be considered inherently unreliable. To counteract this the system provides levels of protection at each layer. The baseband packet header uses forward error correcting (FEC) coding to allow error correction by the receiver and a header error check (HEC) to detect errors remaining after correction. Certain Baseband packet types include FEC for the payload. Furthermore, some Baseband packet types include a cyclic redundancy error check (CRC).

On ACL logical transports the results of the error detection algorithm are used to drive a simple acknowledgement/repeat request (ARQ) protocol. This provides an enhanced reliability by re-transmitting 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



LifeScan, Inc. Ex. 1019
Page 103 of 1420
IPR2025-00103

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 layer provides an additional level of error control that is designed to detect the occasional undetected errors in the baseband layer and request retransmission of the affected data. This provides the level of reliability required by typical Bluetooth 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, and so 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 such 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 layer.

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.



LifeScan, Inc. Ex. 1019
Page 104 of 1420
IPR2025-00103

## 3.2 TRANSPORT ARCHITECTURE ENTITIES

The Bluetooth transport architecture entities are shown in Figure 3.3 on page 31 and are described from the lowest layer upwards in the subsequent sections.

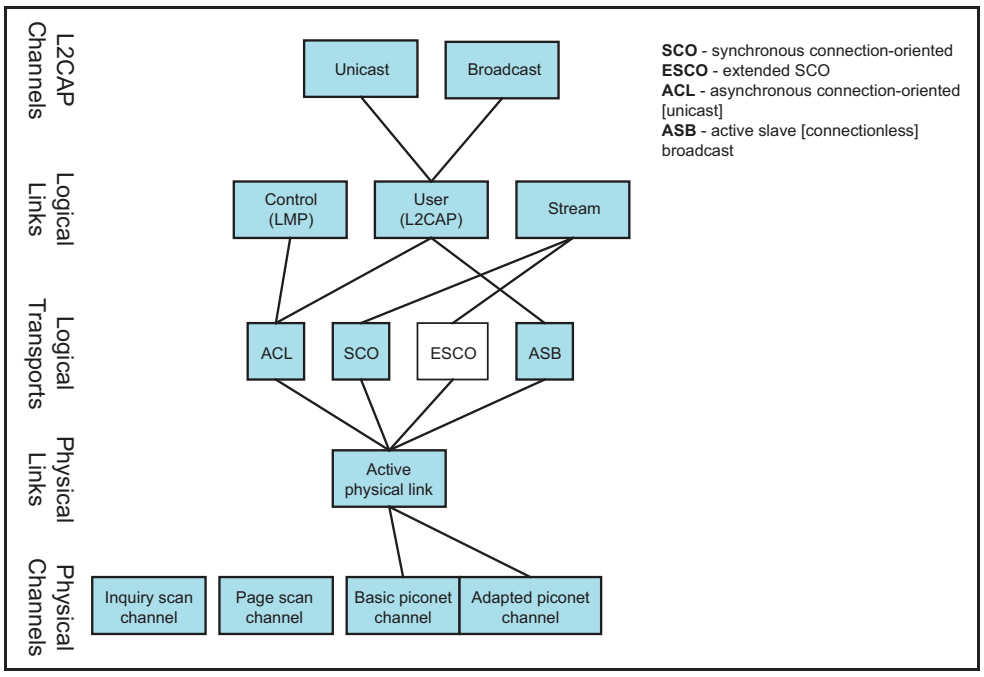



*Figure 3.3: Overview of transport architecture entities and hierarchy*

### 3.2.1 Bluetooth generic packet structure

The general packet structure nearly reflects the architectural layers found in the Bluetooth system. The packet structure is designed for optimal use in normal operation. It is shown in Figure 3.4 on page 31.

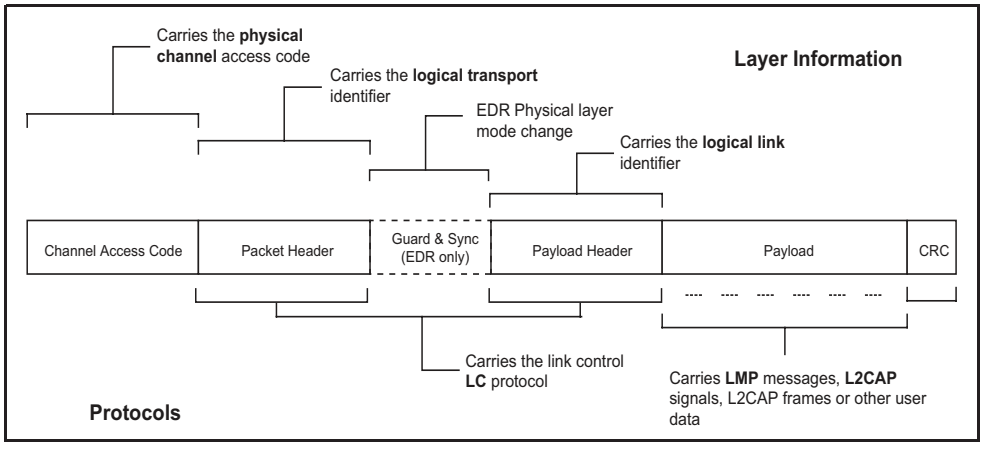



*Figure 3.4: Bluetooth packet structure*



LifeScan, Inc. Ex. 1019
Page 105 of 1420
IPR2025-00103

Packets normally only include 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 (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 channel access code. 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 Bluetooth packet structure that represents or contains information relating to physical links. This information is implied in the logical transport address (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 if the packet is addressed to the device and is used to route the packet internally.

The packet header also carries part of the link control (LC) protocol that is operated per logical transport (except for ACL and SCO transports that operate a shared LC protocol carried on either logical transport.)

The Enhanced Data Rate (EDR) packets have a guard time and synchronization sequence before the payload. This is a field used for physical layer change of modulation scheme.

The payload header is present in all packets on logical transports that support multiple logical links. The payload header includes a logical link identifier 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.

EDR packets have a trailer after the CRC.

The packet payload is used to transport the user data. The interpretation of this data is dependent on the logical transport and logical link identifiers. For ACL logical transports Link Manager Protocol (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.



LifeScan, Inc. Ex. 1019
Page 106 of 1420
IPR2025-00103

## 3.3 PHYSICAL CHANNELS

The lowest architectural layer in the Bluetooth system is the physical channel. A number of types of physical channel are defined. All Bluetooth physical channels are characterized by an RF frequency combined with temporal parameters and restricted by spatial considerations. For the basic and adapted piconet physical channels frequency hopping is used to change frequency periodically to reduce the effects of interference and for regulatory reasons.

Two Bluetooth devices use a shared physical channel for communication. To achieve this their transceivers need to be tuned to the same RF frequency 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 Bluetooth devices may be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth 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 channel access code is a property of the physical channel. The access code is always present at the start of every transmitted packet.

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

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

Whenever a Bluetooth 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.) The Bluetooth specification assumes that a device is only capable of connecting to one physical channel at any time. Advanced devices be may capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.

### 3.3.1 Basic piconet channel

#### *3.3.1.1 Overview*

The basic piconet channel is used for communication between connected devices during normal operation.

#### *3.3.1.2 Characteristics*

The basic piconet 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 Bluetooth device address of the master. The phase in the hopping sequence is determined by the Bluetooth clock of the master. All Bluetooth 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 Bluetooth clock of the piconet master. Packets are transmitted by Bluetooth 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 Bluetooth device address of the piconet.

On the basic piconet 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.

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 asynchronous connection-oriented 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 only used 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 onto these slots, providing there is a transmission starting in each of the slots. In the case where



LifeScan, Inc. Ex. 1019
Page 108 of 1420
IPR2025-00103

there is information from the parked slave broadcast (PSB) logical transport to be transmitted then this is transmitted in the beacon train slots and takes priority over any other logical transport.

#### *3.3.1.3 Topology*

A basic piconet channel may be shared by any number of Bluetooth 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 channel.

There is, however, a limitation on the number of logical transports that can be supported within a piconet. This means that although there is no theoretical limit to the number of Bluetooth devices that share a channel there is a limit to the number of these devices that can be actively involved in exchanging data with the master.

#### *3.3.1.4 Supported layers*

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

### 3.3.2 Adapted piconet channel

#### *3.3.2.1 Overview*

The adapted piconet channel differs from the basic piconet 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. The second way in which the adapted piconet channel differs from the basic piconet channel is that 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 channel uses the same timing and access code as the basic piconet channel, the two channels are often coincident. This provides a deliberate benefit as it allows slaves in either the basic piconet channel or the adapted piconet channel to adjust their synchronization to the master.

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



LifeScan, Inc. Ex. 1019
Page 109 of 1420
IPR2025-00103

### 3.3.3 Inquiry scan channel

#### *3.3.3.1 Overview*

In order for a device to be discovered an inquiry scan channel is used. A discoverable device listens for inquiry requests on its inquiry scan 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 channel frequencies in a pseudo-random fashion, sending an inquiry request on each frequency and listening for any response.

#### *3.3.3.2 Characteristics*

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

The access code used on the inquiry scan channel is taken from a reserved set of inquiry access codes that are shared by all Bluetooth 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 channels. As all of these channels share an identical hopping pattern, a device may concurrently occupy more than one inquiry scan channel if it is capable of concurrently correlating more than one access code.

A device using one of its inquiry scan channels remains passive until it receives an inquiry message on this channel from another Bluetooth device. This is identified by the appropriate inquiry access code. 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 Bluetooth devices it uses the inquiry scan channel of these devices in order 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 channel.

The device takes advantage of the fact that inquiry scan 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.

#### *3.3.3.3 Topology*

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.

#### *3.3.3.4 Supported layers*

During the exchange of packets between an inquiring and discoverable device it may be considered that a temporary physical link exists between these devices. However, the concept is quite irrelevant as it has no physical representation but is only implied by the brief transaction between the devices. No further architectural layers are considered to be supported.

### 3.3.4 Page scan channel

#### *3.3.4.1 Overview*

A connectable device (one that is prepared to accept connections) does so using a page scan channel. A connectable device listens for page requests on its page scan 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 channel frequencies in a pseudo-random fashion, sending a page request on each frequency and listening for any response.

#### *3.3.4.2 Characteristics*

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

A device using its page scan channel remains passive until it receives a page request from another Bluetooth device. This is identified by the page scan channel access code. 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 channel that is characterized by having the paging device as master.

In order for a device to connect to another Bluetooth device it uses the page scan channel of the target device in order to send page requests. If the paging device does not know the phase of the target device's page scan 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



LifeScan, Inc. Ex. 1019
Page 111 of 1420
IPR2025-00103

the paging device to cover all page scan frequencies in a reasonably short time period.

The paging device may have some knowledge of the target device's Bluetooth clock (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 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.

#### *3.3.4.3 Topology*

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.

#### *3.3.4.4 Supported layers*

During the exchange of packets between a paging and connectable device it may be considered that a temporary physical link exists between these devices. However, the concept is quite irrelevant as it has no physical representation but is only implied by the brief transaction between the devices. No further architectural layers are considered to be supported.

## 3.4 PHYSICAL LINKS

A physical link represents a baseband connection between Bluetooth 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 Bluetooth 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 Bluetooth device are used to identify a physical channel. However there is no subsequent part of the packet that directly identifies the physical link. Instead the physical link may be identified by association with the logical transport, as each logical transport is only received 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 LM protocol is used to adapt these properties. As the LM protocol 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 signaling.

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



LifeScan, Inc. Ex. 1019
Page 112 of 1420
IPR2025-00103

### 3.4.1 Links supported by the basic and adapted piconet physical channel

The basic and adapted piconet physical channels support a physical link which 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.

#### *3.4.1.1 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 ID 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. Logical transports that have defined scheduling characteristics are not affected by these modes and continue according to their pre-defined scheduling behavior. The default ACL logical transport and other links with undefined scheduling characteristics are subject to the mode of the active physical link.

#### *3.4.1.2 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 parked slave broadcast (PSB) logical transport is used to allow communication of a subset of LMP signaling and broadcast L2CAP to parked slaves. The PSB logical transport is closely associated with the beacon train.

A slave is parked (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 signaling 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 re-created. 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 parked slave broadcast logical transport (and associated transport) for each of these physical channels.

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

### 3.4.2 Links supported by the scanning physical channels

In the case of the inquiry scan and page scan 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.

## 3.5 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, acknowledgement/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 Bluetooth 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.

Logical transport identification and real-time (link control) signaling are carried in the packet header, and for some logical links identification is carried in the

payload header. Control signaling that does not require single slot response times is carried out using the LMP protocol.

Table 3.1 on page 41 lists all of the logical transport types, the supported logical link types, which type of physical links and physical channels can support them, and a brief description of the purpose of the logical transport.

| Logical transport | Links supported | Supported by | Overview |
|---|---|---|---|
| Asynchronous Connection-Oriented (ACL) | Control (LMP) ACL-C<br>User (L2CAP) ACL-U | Active physical link, basic or adapted physical channel | Reliable or time-bounded, bi-directional, point-to-point. |
| Synchronous Connection-Oriented (SCO) | Stream (unframed) SCO-S | Active physical link, basic or adapted physical channel | Bi-directional, symmetric, point-to-point, AV channels. Used for 64Kb/s constant rate data. |
| Extended Synchronous Connection-Oriented (eSCO) | Stream (unframed) eSCO-S | Active physical link, basic or adapted physical channel | Bi-directional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission. Used for constant rate data synchronized to the master Bluetooth clock. |
| Active slave broadcast (ASB) | User (L2CAP) ASB-U | Active physical link, basic or adapted physical channel. | Unreliable, uni-directional 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, uni-directional broadcast to all piconet devices. Used for LMP and L2CAP traffic to parked devices, and for access requests from parked devices. |

*Table 3.1: Logical transport types.*

The classification of each link type follows from a selection procedure within three categories.

### 3.5.1 Casting

The first category is that of casting. This may be either unicast or broadcast.

- *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, meaning that 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 zero or more receiver devices. Traffic is unidirectional, i e only sent from the source devices to the receiver devices. Broadcast links are connectionless, meaning that 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.

### 3.5.2 Scheduling and acknowledgement scheme

The second category relates to the scheduling and acknowledgement 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, though the default ACL link can be configured to operate in this fashion.

- *Synchronous links.* Synchronous links provide a method of associating the Bluetooth 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 has no time-based characteristics. The data is 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 has 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.)

### 3.5.3 Class of data

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

- *Control links.* Control links are only used for transporting LMP messages between two link managers. These links are invisible above the baseband 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



LifeScan, Inc. Ex. 1019
Page 116 of 1420
IPR2025-00103

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 signaling channel (on the default ACL-U logical link only) or framed user data submitted to user-instantiated L2CAP channels. L2CAP frames submitted to the baseband may be larger than the available baseband packets. A link control protocol 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 that has no inherent framing that should be preserved when delivering the data. Lost data may be replaced by padding at the receiver.

### 3.5.4 Asynchronous connection-oriented (ACL)

The asynchronous connection-oriented (ACL) logical transport is used to carry LMP and L2CAP control signaling and best effort asynchronous user data. The ACL logical transport uses a simple 1-bit ARQN/SEQN scheme to provide simple channel 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 (connects to the basic piconet physical channel). This default ACL is assigned a logical transport address (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 logical transports between the same master and slave. (This is for reasons of compatibility with earlier Bluetooth specifications.) 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 synchronous connection-oriented 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. Asynchronous traffic can be sent over an ACL logical transport configured for isochronous traffic by marking the asynchronous packets as non-automatically-flushable. This allows both isochronous and asynchronous traffic to be transferred at the same time to a single device.

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 parked 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 parked 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.

### 3.5.5 Synchronous connection-oriented (SCO)

The synchronous connection-oriented (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.

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 signaling on the default ACL when the physical channel bandwidth is fully occupied by SCOs. As SCOs carry different packet types to ACLs, 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 Bluetooth 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.

### 3.5.6 Extended synchronous connection-oriented (eSCO)

The extended synchronous connection-oriented (eSCO) logical transport is a symmetric or asymmetric, point-to-point link between the master and a specific



LifeScan, Inc. Ex. 1019
Page 118 of 1420
IPR2025-00103

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 a LT_ADDR that is unique within the piconet for the duration of the eSCO. eSCO-S links are created using LM signaling and follow scheduling rules similar to SCO-S links.

There are no further architectural layers defined by the Bluetooth 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.

### 3.5.7 Active slave broadcast (ASB)

The active slave broadcast 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 acknowledgement protocol and the traffic is uni-directional from the piconet master to the slaves. The ASB channel may be used for L2CAP group traffic (a legacy of the 1.1 specification), and is never used for L2CAP connection-oriented channels, L2CAP control signaling or LMP control signaling.

The ASB logical transport is inherently unreliable because of the lack of acknowledgement. To improve the reliability, each packet is transmitted a number of times. An identical sequence number 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



LifeScan, Inc. Ex. 1019
Page 119 of 1420
IPR2025-00103

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.)

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.

### 3.5.8 Parked slave broadcast (PSB)

The parked slave broadcast logical transport is used for communications between the master and slaves that are parked (have given up their default ACL logical transport.) The parked slave broadcast 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 (carrying baseband signaling.) 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 baseband 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



LifeScan, Inc. Ex. 1019
Page 120 of 1420
IPR2025-00103

baseband packets. If the user information occupies a single packet then the user information packet is repeated in each of the piconet channel beacon train slots.

If the user information occupies more than one baseband 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 (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 address 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 only connected to the piconet physical channel during the time that the PSB transport is being used.

### 3.5.9 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 logical link identifier (LLID) bits in the payload header of baseband 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 3.2 on page 27.) In particular the SCO and eSCO logical transports are only able to carry constant data rate streams, and these are uniquely identified by the LT_ADDR. Such logical transports only use packets that do not contain a payload header, as their length is known in advance, and no LLID is necessary.

### 3.5.10 ACL Control Logical Link (ACL-C)

The ACL Control Logical Link (ACL-C) is used to carry LMP signaling between devices in the piconet. The control link is only carried 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 (see below) link when carried on the same logical transport. The ACL-C link is always given priority over the ACL-U (see below) link when carried on the same logical transport.



LifeScan, Inc. Ex. 1019
Page 121 of 1420
IPR2025-00103

### 3.5.10 User Asynchronous/Isochronous Logical Link (ACL-U)

The user asynchronous/isochronous logical link (ACL-U) 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 baseband 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 baseband packet (the header is only required 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.)

### 3.5.11 User Synchronous/Extended Synchronous Logical Links (SCO-S/eSCO-S)

Synchronous (SCO-S) and extended synchronous (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 is 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 pre-defined 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.



LifeScan, Inc. Ex. 1019
Page 122 of 1420
IPR2025-00103

## 3.6 L2CAP CHANNELS

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 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 L2CAP, and each L2CAP channel endpoint on any device has a different CID.

L2CAP channels may be configured to provide an appropriate Quality of Service (QoS) to the application. 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 service data units (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 L2CAP layer. This option is selected by the application when the channel is established. L2CAP can also provide enhanced error detection and retransmission to (a) reduce the probability of undetected errors being passed to the application and (b) recover from loss of portions of the user data when the Baseband layer 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 baseband buffers, and also to operate a token based flow control procedure over the HCI, submitting fragments to the baseband only when allowed to do so. This may affect the scheduling algorithm.

# 4 COMMUNICATION TOPOLOGY

## 4.1 PICONET TOPOLOGY

Any time a Bluetooth 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 (which means that they are synchronized to a common clock and hopping sequence.) The common (piconet) clock is identical to the Bluetooth clock 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 clock and the master's Bluetooth device address. All other synchronized devices are referred to as slaves in the piconet. The terms master and slave are only used when describing these roles in a piconet.

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

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

A Bluetooth 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 Bluetooth device. The Bluetooth 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 the Bluetooth core specification.



LifeScan, Inc. Ex. 1019
Page 124 of 1420
IPR2025-00103

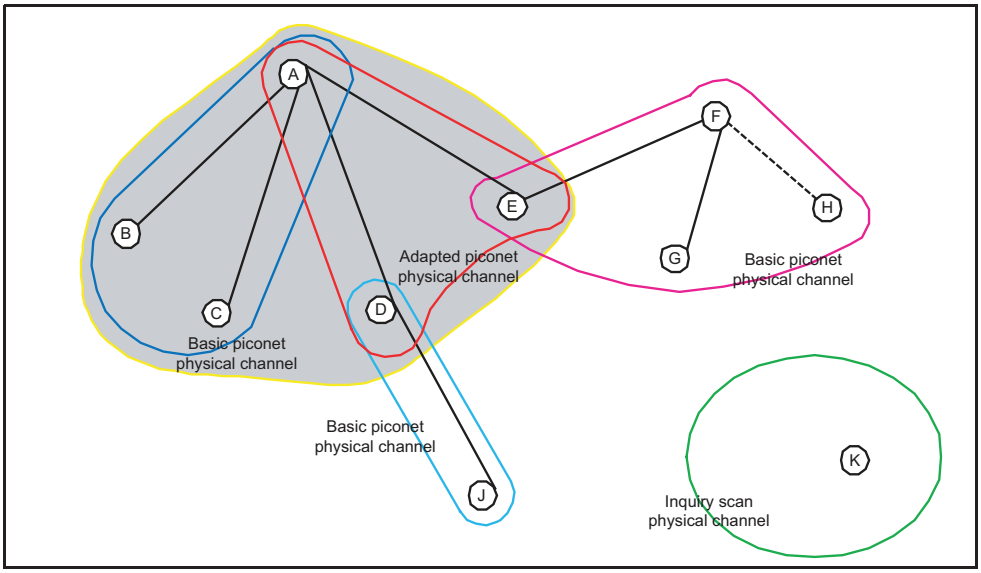



*Figure 4.1: Example Bluetooth topology*

In Figure 4.1 on page 51 an example 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: a) one piconet with device F as master (known as piconet F) and devices E, G and H as slaves and b) one piconet with device D as master (known as piconet D) and device J as slave.

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

Piconet D and piconet F are both using only a basic piconet physical channel (represented by the cyan and magenta enclosures respectively.) 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, and 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 Bluetooth devices. It is currently listening on its inquiry scan physical channel (represented by the green enclosure), awaiting an inquiry request from another device.



LifeScan, Inc. Ex. 1019
Page 125 of 1420
IPR2025-00103

## 4.2 OPERATIONAL PROCEDURES AND MODES

The typical operational mode of a Bluetooth device is to be connected to other Bluetooth devices (in a piconet) and exchanging data with that Bluetooth device. As Bluetooth 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 and therefore a device may be engaged in a number of these procedures and modes concurrently.

### 4.2.1 Inquiry (Discovering) Procedure

Bluetooth devices use the inquiry procedure to discover nearby devices, or to be discovered by devices in their locality.

The inquiry procedure is asymmetrical. A Bluetooth device that tries to find other nearby devices is known as an inquiring device and actively sends inquiry requests. Bluetooth devices that are available to be found are known as discoverable devices and 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 Bluetooth 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.

#### *4.2.1.1 Extended Inquiry Response*

An Extended Inquiry Response can be used to provide miscellaneous information during the inquiry response procedure. Data types are defined for such things as local name and supported services, information that otherwise would have to be obtained during a connection. A device that receives a local name and a list of supported services in an extended inquiry response does not have to connect to do a remote name request and an SDP service search, thereby shortening the time to useful information. It is recommended that a device includes all supported services and a significant minority part of its local name in the extended inquiry response.

The extended inquiry response procedure is backwards compatible with the standard inquiry response procedure.



LifeScan, Inc. Ex. 1019
Page 126 of 1420
IPR2025-00103

### 4.2.2 Paging (Connecting) Procedure

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

The connectable device uses a special physical channel to listen for connection request packets from the paging (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 Bluetooth 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.

### 4.2.3 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 Link Manager that exchanges Link Manager Protocol messages with the remote Bluetooth 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 signaling 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 Link Manager. The other link (ACL-U) transports the L2CAP signaling 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.

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



LifeScan, Inc. Ex. 1019
Page 127 of 1420
IPR2025-00103

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 only allowed 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 Link Manager on the master device restricts itself to the use of LMP messages that are allowed to be transported over the 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.

### 4.2.4 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 only active 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.

### 4.2.5 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 only affects 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.

Sniff subrating provides a mechanism for further reducing the active duty cycle, thereby enhancing the power-saving capability of sniff mode. Sniff subrating allows a Host to create a guaranteed access-like connection by specifying maximum transmit and receive latencies. This allows the basebands to optimize the low power performance without having to exit and re-enter sniff mode using Link Manager commands.



LifeScan, Inc. Ex. 1019
Page 128 of 1420
IPR2025-00103

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.

### 4.2.6 Parked state

A slave device may remain connected to a piconet but have its physical link in the parked 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 logical transport is inactive (or absent) then the devices may engage in activity on other physical channels, or enter reduced power mode.

### 4.2.7 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 only copies 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.



LifeScan, Inc. Ex. 1019
Page 129 of 1420
IPR2025-00103

### 4.2.8 Enhanced Data Rate

Enhanced Data Rate is a method of extending the capacity and types of Bluetooth packets for the purposes of increasing the maximum throughput, providing better support for multiple connections, and lowering power consumption, while the remainder of the architecture is unchanged.

Enhanced Data Rate may be selected as a mode that operates independently on each logical transport. Once enabled, the packet type bits in the packet header are interpreted differently from their meaning in Basic Rate mode. This different interpretation is clarified in conjunction with the logical transport address field in the header. The result of this interpretation allows the packet payload header and payload to be received and demodulated according to the packet type. Enhanced Data Rate can be enabled only for ACL-U, eSCO-S logical transports and cannot be enabled for ACL-C, SCO-S, and the broadcast logical transports.



LifeScan, Inc. Ex. 1019
Page 130 of 1420
IPR2025-00103

# 5 SECURE SIMPLE PAIRING OVERVIEW

The primary goal of Secure Simple Pairing is to simplify the pairing procedure for the user. Secondary goals are to maintain or improve the security in Bluetooth wireless technology. Since high levels of security and ease-of-use are often at opposite ends of the spectrum in many technologies and products, much care has been taken to maximize security while minimizing complexity from the end user's point of view.

## 5.1 SECURITY GOALS

Secure Simple Pairing has two security goals: protection against passive eavesdropping and protection against man-in-the-middle (MITM) attacks (active eavesdropping). It is a goal of Secure Simple Pairing to exceed the maximum security level provided by the use of a 16 alphanumeric PIN with the pairing algorithm used in Bluetooth Core Specification version 2.0 + EDR and earlier versions. Note that many Bluetooth devices compliant with Bluetooth Core Specification 2.0 + EDR and earlier versions use a 4-digit PIN or a fixed PIN of commonly known values significantly limiting the security on the link.

## 5.2 PASSIVE EAVESDROPPING PROTECTION

A strong link key coupled with a strong encryption algorithm is necessary to give the user protection against passive eavesdropping. The strength of the link key is based on the amount of entropy (or randomness) in its generation process which would not be known by an attacker. Using legacy pairing, the only source of entropy is the PIN which, in many use cases, is typically four digits either selected by the user or fixed for a given product. Therefore, if the pairing procedure and one authentication exchange is recorded one can run an exhaustive search to find the PIN in a very short amount of time on commonly available computing hardware. With Secure Simple Pairing, the recording attack becomes much harder as the attacker must have solved a hard problem in public key cryptography in order to derive the link key from the recorded information. This protection is independent of the length of the passkey or other numeric values that the user must handle. Secure Simple Pairing gives the same resistance against the recording and passive eavesdropping attacks even when the user is not required to do anything.

Secure Simple Pairing uses Elliptic Curve Diffie Hellman (ECDH) public key cryptography as a means to thwart passive eavesdropping attacks. ECDH provides a very high degree of strength against passive eavesdropping attacks but it may be subject to MITM attacks, which however, are much harder to perform in practice than the passive eavesdropping attack (see Section 5.3, “Man-In-The-Middle Protection,” on page 58).

Using the security protocols in the Bluetooth Core Specification version 2.0 + EDR and earlier with a 16 numeric digit PIN achieves about 53 bits of entropy



LifeScan, Inc. Ex. 1019
Page 131 of 1420
IPR2025-00103

whereas a 16 character alphanumeric, case sensitive PIN yields about 95 bits of entropy when the entire 62 character set is used ([0, … 9, 'A', … 'Z', 'a', … 'z']). Secure Simple Pairing has approximately 95 bits of entropy using the FIPS approved P192 elliptic curve which is at least as good as the entropy in Bluetooth Core Specification 2.0 + EDR and earlier using a 16 character, alphanumeric, case sensitive PIN. Secure Simple Pairing, therefore, exceeds the security requirements of the Bluetooth SIM Access Profile (SAP) which is the profile with the most stringent security requirements. ECDH cryptography was selected over standard Diffie Hellman (often referred to as DH76) since it is computationally less complex and less likely to exceed the low computational capacity in common Bluetooth Controllers.

## 5.3 MAN-IN-THE-MIDDLE PROTECTION

A man-in-the-middle (MITM) attack occurs when a user wants to connect two devices but instead of connecting directly with each other they unknowingly connect to a third (attacking) device that plays the role of the device they are attempting to pair with. The third device then relays information between the two devices giving the illusion that they are directly connected. The attacking device may even eavesdrop on communication between the two devices (known as active eavesdropping) and is able to insert and modify information on the connection. In this type of attack, all of the information exchanged between the two devices are compromised and the attacker may inject commands and information into each of the devices thus potentially damaging the function of the devices. Devices falling victim to the attack are capable of communicating only when the attacker is present. If the attacker is not active or out range, the two victim devices will not be able to communicate directly with each other and the user will notice it.

To prevent MITM attacks, Secure Simple Pairing offers two user assisted numeric methods: numerical comparison or passkey entry. If Secure Simple Pairing would use 16 decimal digit numbers, then the usability would be the same as using legacy pairing with 16 decimal digit PIN. The chance for a MITM to succeed inserting its own link keys in this case is a 1 in $10^{16} = 2^{53}$ pairing instances, which is an unnecessarily low probability.

Secure Simple Pairing protects the user from MITM attacks with a goal of offering a 1 in 1,000,000 chance that a MITM could mount a successful attack. The strength of the MITM protections was selected to minimize the user impact by using a six digit number for numerical comparison and Passkey entry. This level of MITM protection was selected since, in most cases, users can be alerted to the potential presence of a MITM attacker when the connection process fails as a result of a failed MITM attack. While most users feel that provided that they have not compromised their passkey, a 4-digit key is sufficient for authentication (i.e. bank card PIN codes), the use of six digits allows Secure Simple Pairing to be FIPS compliant and this was deemed to have little perceivable usability impact.

## 5.4 ASSOCIATION MODELS

Secure Simple Pairing uses four association models referred to as Numeric Comparison, Just Works, Out Of Band, and Passkey Entry. Each of these association models are described in more detail in the following sections.

The association model used is deterministic based on the I/O capabilities of the two devices.

### 5.4.1 Numeric Comparison

The Numeric Comparison association model is designed for scenarios where both devices are capable of displaying a six digit number and both are capable of having the user enter "yes" or "no". A good example of this model is the cell phone / PC scenario.

The user is shown a six digit number (from "000000" to "999999") on both displays and then asked whether the numbers are the same on both devices. If "yes" is entered on both devices, the pairing is successful.

The numeric comparison serves two purposes. First, since many devices do not have unique names, it provides confirmation to the user that the correct devices are connected with each other. Second, the numeric comparison provides protection against MITM attacks (see Section 5.3 on page 58).

Note that there is a significant difference from a cryptographic point of view between Numeric Comparison and the PIN entry model used by Bluetooth Core Specification and earlier versions. In the Numeric Comparison association model, the six digit number is an artifact of the security algorithm and not an input to it, as is the case in the Bluetooth security model. Knowing the displayed number is of no benefit in decrypting the encoded data exchanged between the two devices.

### 5.4.2 Just Works

The Just Works association model is primarily designed for scenarios where at least one of the devices does not have a display capable of displaying a six digit number nor does it have a keyboard capable of entering six decimal digits. A good example of this model is the cell phone/mono headset scenario where most headsets do not have a display.

The Just Works association model uses the Numeric Comparison protocol but the user is never shown a number and the application may simply ask the user to accept the connection (exact implementation is up to the end product manufacturer).

The Just Works association model provides the same protection as the Numeric Comparison association model against passive eavesdropping but offers no protection against the MITM attack.



LifeScan, Inc. Ex. 1019
Page 133 of 1420
IPR2025-00103

When compared against today's experience of a headset with a fixed PIN, the security level of the Just Works association model is considerably higher since a high degree of protection against passive eavesdropping is realized.

### 5.4.3 Out of Band

The Out of Band (OOB) association model is primarily designed for scenarios where an Out of Band mechanism is used to both discover the devices as well as to exchange or transfer cryptographic numbers used in the pairing process. In order to be effective from a security point of view, the Out of Band channel should provide different properties in terms of security compared to the Bluetooth radio channel. The Out of Band channel should be resistant to MITM attacks. If it is not, security may be compromised during authentication.

The user's experience differs a bit depending on the Out of Band mechanism. As an example, with a Near Field Communication (NFC) solution, the user(s) will initially touch the two devices together, and is given the option to pair the first device with the other device. If "yes" is entered, the pairing is successful. This is a single touch experience where the exchanged information is used in both devices. The information exchanged includes discovery information (such as the Bluetooth Device Address) as well as cryptographic information. One of the devices will use a Bluetooth Device Address to establish a connection with the other device. The rest of the exchanged information is used during authentication.

The OOB mechanism may be implemented as either read only or read/write. If one side is read only, a one-way authentication is performed. If both sides are read/write, a two-way authentication is performed.

The OOB protocol is selected only when the pairing process has been activated by previous OOB exchange of information and one (or both) of the device(s) gives OOB as the IO capabilities. The protocol uses the information which has been exchanged and simply asks the user to confirm connection.

The OOB association model supports any OOB mechanism where cryptographic information and the Bluetooth Device Address can be exchanged. The OOB association model does not support a solution where the user has activated a Bluetooth connection and would like to use OOB for authentication only.

### 5.4.4 Passkey Entry

The Passkey Entry association model is primarily designed for the scenario where one device has input capability but does not have the capability to display six digits and the other device has output capabilities. A good example of this model is the PC and keyboard scenario.

The user is shown a six digit number (from "000000" to "999999") on the device with a display, and is then asked to enter the number on the other



LifeScan, Inc. Ex. 1019
Page 134 of 1420
IPR2025-00103

device. If the value entered on the second device is correct, the pairing is successful. Note that there is a significant difference from a cryptographic point of view between Passkey Entry and the PIN entry model used by Bluetooth Core Specification 2.0 + EDR and earlier versions. In the Passkey Entry association model, the six digit number is independent of the security algorithm and not an input to it, as is the case in the 2.0 + EDR security model. Knowing the entered number is of no benefit in decrypting the encoded data exchanged between the two devices.

### 5.4.5 Association Model Overview

The following diagram shows Secure Simple Pairing from the point of view of the technology used for discovery and then the different association possibilities.

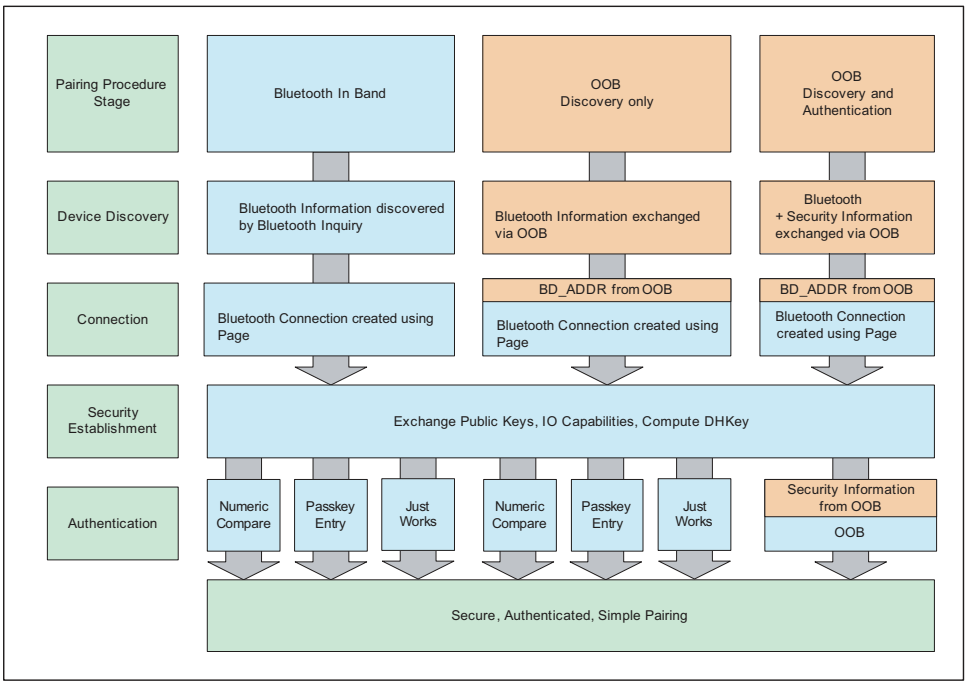



*Figure 5.1: Secure Simple Pairing Association Models*



LifeScan, Inc. Ex. 1019
Page 135 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 136 of 1420
IPR2025-00103

# ACRONYMS & ABBREVIATIONS



LifeScan, Inc. Ex. 1019
Page 137 of 1420
IPR2025-00103

# CONTENTS

1 List of Acronyms and Abbreviations ..... 65



LifeScan, Inc. Ex. 1019
Page 138 of 1420
IPR2025-00103

# 1 LIST OF ACRONYMS AND ABBREVIATIONS

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| 8DPSK | 8 phase Differential Phase Shift Keying | 3 Mbps modulation type used by Enhanced Data rate |
| π/4 DQPSK | pi/4 Rotated Differential Quaternary Phase Shift Keying | 2 Mbps modulation type used by Enhanced Data Rate |
| **A** | | |
| ACK | Acknowledge | |
| ACL | Asynchronous Connection-oriented [logical transport] | Reliable or time-bounded, bi-directional, point-to-point. |
| 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 | |
| ASB | Active Slave Broadcast [logical transport] | Unreliable, uni-directional broadcast to any devices synchronized with the physical channel. |
| ASB-U | ASB User [logical link] (L2CAP) | |
| **B** | | |
| BB | Baseband | |
| BCH | Bose, Chaudhuri & Hocquenghem | Type of code<br>The persons who discovered these codes in 1959 (H) and 1960 (B&C) |
| BD_ADDR | Bluetooth Device Address | |
| BER | Bit Error Rate | |
| BT | Bandwidth Time | |
| **C** | | |
| CAC | Channel Access Code | |
| CL | Connectionless | |
| CODEC | COder DECoder | |

*Table 1.1: Acronyms and Abbreviations*



LifeScan, Inc. Ex. 1019
Page 139 of 1420
IPR2025-00103

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| COF | Ciphering Offset | |
| CRC | Cyclic Redundancy Check | |
| CVSD | Continuous Variable Slope Delta Modulation | |
| **D** | | |
| DAC | Device Access Code | |
| DCE | Data Communication Equipment | |
| DCE | Data Circuit-Terminating Equipment | In serial communications, DCE refers to a device between the communication endpoints whose sole task is to facilitate the communications process; typically a modem |
| DCI | Default Check Initialization | |
| DEVM | Differential Error Vector Magnitude | Measure of modulation error used for Enhanced Data Rate transmitter testing |
| DH | Data-High Rate | Data packet type for high rate data |
| DIAC | Dedicated Inquiry Access Code | |
| DM | Data - Medium Rate | Data packet type for medium rate data |
| DPSK | Differential Phase Shift Keying | Generic description of Enhanced Data Rate modulation |
| DQPSK | Differential Quaternary Phase Shift Keying | Modulation type used by Enhanced Data Rate |
| DTE | Data Terminal Equipment | In serial communications, DTE refers to a device at the endpoint of the communications path; typically a computer or terminal. |
| DUT | Device Under Test | |
| DV | Data Voice | Data packet type for data and voice |
| **E** | | |
| ED | Erroneous Data | |
| EDR | Enhanced Data Rate | |
| EIR | Extended Inquiry Response | Host supplied information transmitted in the inquiry responses substate. |

*Table 1.1: Acronyms and Abbreviations*



LifeScan, Inc. Ex. 1019
Page 140 of 1420
IPR2025-00103

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| EIRP | Effective Isotropic Radiated Power | Equivalent power that an isotropic antenna must transmit to provide the same field power density |
| EPR | Encryption Pause Resume | Mode allowing the device to initiate the pause and resume encryption sequence |
| eSCO | Extended Synchronous Connection Oriented [logical transport] | Bi-directional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission. |
| eSCO-S | Stream eSCO (unframed) | used to support isochronous data delivered in a stream without framing |
| ETSI | European Telecommunications Standards Institute | |
| | | |
| **F** | | |
| FCC | Federal Communications Commission | |
| FEC | Forward Error Correction code | |
| FH | Frequency Hopping | |
| FHS | Frequency Hop Synchronization | |
| FIFO | First In First Out | |
| FM | Frequency Modulation | Modulation Type |
| **G** | | |
| GFSK | Gaussian Frequency Shift Keying | |
| GIAC | General Inquiry Access Code | |
| **H** | | |
| HCI | Host Controller Interface | |
| HEC | Header-Error-Check | |
| HID | Human Interface Device | |
| HV | High quality Voice | e.g. HV1 packet |
| HW | Hardware | |
| **I** | | |
| IAC | Inquiry Access Code | |

*Table 1.1: Acronyms and Abbreviations*



LifeScan, Inc. Ex. 1019
Page 141 of 1420
IPR2025-00103

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| IEEE | Institute of Electronic and Electrical Engineering | |
| IETF | Internet Engineering Task Force | |
| IP | Internet Protocol | |
| IrDA | Infra-red Data Association | |
| ISM | Industrial, Scientific, Medical | |
| IUT | Implementation Under Test | |
| | | |
| **L** | | |
| L2CAP | Logical Link Control and Adaptation Protocol | |
| LAP | Lower Address Part | |
| LC | Link Controller | Link Controller (or baseband) part of the Bluetooth protocol stack. Low level Baseband protocol handler |
| LC | Link Control [logical link] | The control logical links LC and ACL-C are used at the link control level and link manager level, respectively. |
| LCP | Link Control Protocol | |
| LFSR | Linear Feedback Shift Register | |
| LLID | Logical Link Identifier | |
| LM | Link Manager | |
| LMP | Link Manager Protocol | For LM peer to peer communication |
| LSB | Least Significant Bit | |
| LSTO | Link Supervision Timeout Event | Controller can send LSTO event to Host |
| LT_ADDR | Logical Transport ADDRess | |
| **M** | | |
| M | Master or Mandatory | |
| MAC | Medium Access Control | |
| Mbps | Million (Mega) bits per second | |
| MITM | Man-in-the-middle | |
| MMI | Man Machine Interface | |

*Table 1.1: Acronyms and Abbreviations*

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| MS | Mobile Station | |
| MSB | Most Significant Bit | |
| MSC | Message Sequence Chart | |
| MTU | Maximum Transmission Unit | |
| **N** | | |
| NAK | Negative Acknowledge | |
| NAP | Non-significant Address Part | |
| **O** | | |
| O | Optional | |
| OBEX | OBject EXchange protocol | |
| OCF | OpCode Command Field | |
| OGF | OpCode Group Field | |
| **P** | | |
| PBF | Packet Boundary Flag | The device supports the capability to correctly handle HCI ACL Data Packets |
| PCM | Pulse Coded Modulation | |
| PDU | Protocol Data Unit | a message |
| PIN | Personal Identification Number | |
| PM_ADDR | Parked Member Address | |
| PN | Pseudo-random Noise | |
| PPM | Part Per Million | |
| PPP | Point-to-Point Protocol | |
| PRBS | Pseudo Random Bit Sequence | |
| PRNG | Pseudo Random Noise Generation | |
| PSB | Parked Slave Broadcast [logical transport] | Unreliable, uni-directional broadcast to all piconet devices. |
| PSB-C | PSB Control [logical link] (LMP) | |
| PSB-U | PSB User [logical link] (L2CAP) | |
| PSK | Phase Shift Keying | Class of modulation types |
| PSTN | Public Switched Telephone Network | |

*Table 1.1: Acronyms and Abbreviations*

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| ptt | Packet Type Table | The ptt parameter is used to select the logical transport types via LMP. |
| **Q** | | |
| QoS | Quality of Service | |
| **R** | | |
| RAND | Random number | |
| RF | Radio Frequency | |
| RFC | Request For Comments | |
| RFCMode | Retransmission and Flow Control Mode | |
| RFCOMM | | Serial cable emulation protocol based on ETSI TS 07.10 |
| RMS | Root Mean Square | |
| RSSI | Received Signal Strength Indication | |
| RX | Receive | |
| **S** | | |
| S | Slave | |
| SAP | Service Access Points | |
| SAR | Segmentation and Reassembly | |
| SCO | Synchronous Connection-Oriented [logical transport] | Bi-directional, symmetric, point-to-point, AV channels. |
| SCO-S | Stream SCO (unframed) | |
| SCO-S | Synchronous logical link | used to support isochronous data delivered in a stream without framing |
| SD | Service Discovery | |
| SDP | Service Discovery Protocol | |
| SDU | Service Data Unit | |
| SEQN | Sequential Numbering scheme | |
| SRES | Signed Response | |
| SS | Supplementary Services | |
| SSI | Signal Strength Indication | |

*Table 1.1: Acronyms and Abbreviations*



LifeScan, Inc. Ex. 1019
Page 144 of 1420
IPR2025-00103

| Acronym or abbreviation | Writing out in full | Which means |
|---|---|---|
| SSP | Secure Simple Pairing | Simplifies the pairing process and improves Bluetooth security. |
| SSR | Sniff Subrating | A mode that defines the anchor points at which the master transmits to the slave. |
| SW | Software | |
| **T** | | |
| TBD | To Be Defined | |
| TC | Test Control | Test Control layer for the test interface |
| TCI | Test Control Interface | |
| TCP/IP | Transport Control Protocol/Internet Protocol | |
| TCS | Telephony Control protocol Specification | |
| TDD | Time-Division Duplex | |
| TX | Transmit | |
| **U** | | |
| UAP | Upper Address Part | |
| UART | Universal Asynchronous receiver Transmitter | |
| UI | User Interface | |
| UI | Unnumbered Information | |
| ULAP | Upper and Lower Address Parts | |
| USB | Universal Serial Bus | |
| UUID | Universal Unique Identifier | |
| **W** | | |
| WAP | Wireless Application Protocol | |

*Table 1.1: Acronyms and Abbreviations*



LifeScan, Inc. Ex. 1019
Page 145 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 146 of 1420
IPR2025-00103

# CORE SPECIFICATION CHANGE HISTORY



LifeScan, Inc. Ex. 1019
Page 147 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 148 of 1420
IPR2025-00103

# CONTENTS

**1 Deprecated Features ..... 76**

**2 Changes from V1.1 to V1.2 ..... 77**

2.1 New Features ..... 77

2.2 Structure Changes ..... 77

2.3 Deprecated Features list ..... 77

2.4 Changes in Wording ..... 78

2.5 Nomenclature Changes ..... 78

**3 Changes from V1.2 to V2.0 + EDR ..... 79**

3.1 New Features ..... 79

3.2 Deprecated Features ..... 79

**4 Changes from v2.0 + EDR to v2.1 + EDR ..... 80**

4.1 New features ..... 80

4.2 Deprecated Features ..... 80



LifeScan, Inc. Ex. 1019
Page 149 of 1420
IPR2025-00103

# 1 DEPRECATED FEATURES

Some features have been deemed no longer useful and have been deprecated. The term *deprecation* does not mean that they are no longer allowed, but that they are no longer recommended as the best way of performing a given function.

Features that have been deprecated are not included in this specification or corresponding test documentation. Deprecated features may be implemented using the latest qualifiable version of the Bluetooth core specification that specifies the deprecated features.



LifeScan, Inc. Ex. 1019
Page 150 of 1420
IPR2025-00103

# 2 CHANGES FROM V1.1 TO V1.2

## 2.1 NEW FEATURES

Several new features were introduced in the Bluetooth Core Specification 1.2. The major areas of improvement are:

- Architectural overview
- Faster connection
- Adaptive frequency hopping
- Extended SCO links
- Enhanced error detection and flow control
- Enhanced synchronization capability
- Enhanced flow specification

The feature descriptions are incorporated into the existing text in different core parts, described in Volumes 2 and 3.

## 2.2 STRUCTURE CHANGES

The Bluetooth Core Specification 1.2 was significantly restructured for better consistency and readability. The most important structure changes have been performed in Baseband, LMP, HCI and L2CAP. The text in these sections has been rearranged to provide:

- Presentation of the information in a more logical progression
- Removal of redundant text and requirements
- Consolidation of baseband related requirements (for example, the *Baseband Timers* and *Bluetooth Audio* sections into the Baseband Specification)
- Alignment of the specification with the new architecture and terminology presented in the Architecture Overview (see “IEEE Language” on page 87 [vol. 1]).

In addition, new text and requirements were added for the new features as well as many changes throughout the specification to update the text to use IEEE language (see “IEEE Language” on page 87 [vol. 1]).

## 2.3 DEPRECATED FEATURES LIST

Features deprecated in version 1.2 are:

- The use of Unit Keys for security
- Optional Paging schemes
- 23 channel hopping sequence



LifeScan, Inc. Ex. 1019
Page 151 of 1420
IPR2025-00103

- Page scan period mode and associated commands

## 2.4 CHANGES IN WORDING

Two general classes of changes to the wording of the Bluetooth Specification have been done for version 1.2. They are a formalization of the language by using conventions established by the Institute of Electrical and Electronic Engineers (IEEE), and a regularization of Bluetooth wireless technology-specific terms. Many portions of the version 1.1 specification use imprecise or inaccurate terms to describe attributes of the protocol. A more accurate terminology described in Part E was introduced into the version 1.2 specification and have been applied in future versions.

## 2.5 NOMENCLATURE CHANGES

The nomenclature in Bluetooth 1.2 was also updated due to new concepts that are introduced together with the new features and the new architecture. See “Architecture” on page 9.



LifeScan, Inc. Ex. 1019
Page 152 of 1420
IPR2025-00103

# 3 CHANGES FROM V1.2 TO V2.0 + EDR

## 3.1 NEW FEATURES

The Bluetooth Core Specification version 2.0 + EDR introduces Enhanced Data Rate (EDR). EDR provides a set of additional packet types that use the new 2 Mbps and 3 Mbps modes.

In addition to EDR a set of errata provided in ESR02 has been incorporated into this version and revised to include changes caused by the addition of EDR.

These additions are incorporated into the existing text in different core parts described in Volumes 2 and 3.

## 3.2 DEPRECATED FEATURES

The only feature deprecated in version 2.0 + EDR is the Page Scan Period Mode and associated commands (based on Erratum 694 which is also included in ESR02).



LifeScan, Inc. Ex. 1019
Page 153 of 1420
IPR2025-00103

# 4 CHANGES FROM V2.0 + EDR TO V2.1 + EDR

## 4.1 NEW FEATURES

Several new features are introduced in Bluetooth Core Specification 2.1 + EDR. The major areas of improvement are:

- Erroneous Data Reporting
- Encryption Pause and Resume
- Extended Inquiry Response
- Link Supervision Timeout Changed Event
- Non-Automatically-Flushable Packet Boundary Flag
- Secure Simple Pairing
- Sniff Subrating
- Security Mode 4

## 4.2 DEPRECATED FEATURES

No features were deprecated in v2.1 + EDR.



LifeScan, Inc. Ex. 1019
Page 154 of 1420
IPR2025-00103

# MIXING OF SPECIFICATION VERSIONS



LifeScan, Inc. Ex. 1019
Page 155 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 156 of 1420
IPR2025-00103

# CONTENTS

**1 Mixing of Specification Versions .....84**

1.1 Features and their Types .....85



LifeScan, Inc. Ex. 1019
Page 157 of 1420
IPR2025-00103

# 1 MIXING OF SPECIFICATION VERSIONS

This part describes how different versions of the Core System Packages can be mixed in Bluetooth implementations. The Core System Packages consist of a Controller Package (see volume 2) and a Host Package (see volume 3).

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

| Category | Description |
|---|---|
| Type 1 | Feature that exists below HCI and cannot be configured via HCI |
| Type 2 | Feature that exists below HCI and can be configured/enabled via HCI |
| Type 3 | Feature that exists below and above HCI and requires HCI command/ events to function |
| Type 4 | Feature that exists only above HCI |

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

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

## 1.1 FEATURES AND THEIR TYPES

The following table lists the features and their types.

| Feature | Version | Type |
|---|---|---|
| Basic AFH operation | V1.2 | 1 |
| Enhanced inquiry | V1.2 | 1 |
| Configuration of AFH (setting channels and enabling/disabling channel assessment) | V1.2 | 2 |
| Enhanced synchronization capability | V1.2 | 2 |
| Interlaced inquiry scan | V1.2 | 2 |
| Interlaced page scan | V1.2 | 2 |
| Broadcast encryption | V1.2 | 2 |
| Enhanced flow specification and flush time-out | V1.2 | 3 |
| Extended SCO links | V1.2 | 3 |
| Inquiry Result with RSSI | V1.2 | 3 |
| L2CAP flow and error control | V1.2 | 4 |
| 2 Mbps EDR | V2.0 + EDR | 2 |
| 3 Mbps EDR | V2.0 + EDR | 2 |
| 3 slot packets in EDR | V2.0 + EDR | 2 |
| 5 slot packets in EDR | V2.0 + EDR | 2 |
| 2 Mbps eSCO | V2.0 + EDR | 2* |
| 3 Mbps eSCO | V2.0 + EDR | 2* |
| 3 slot packets for EDR eSCO | V2.0 + EDR | 2* |
| Erroneous Data Reporting | V2.1 + EDR | 3 |
| Extended Inquiry Response | V2.1 + EDR | 3 |
| Encryption Pause and Resume | V2.1 + EDR | 1 |
| Link Supervision Timeout Changed Event | V2.1 + EDR | 3 |
| Non-Flushable Packet Boundary Flag | V2.1 + EDR | 3 |
| Sniff subrating | V2.1+ EDR | 3 |
| Secure Simple Pairing | V21.+ EDR | 3 |

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



LifeScan, Inc. Ex. 1019
Page 159 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 160 of 1420
IPR2025-00103

# IEEE LANGUAGE



LifeScan, Inc. Ex. 1019
Page 161 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 162 of 1420
IPR2025-00103

# CONTENTS

**11 Use of IEEE Language .....91**
1.1 Shall .....91
1.2 Must .....92
1.3 Will .....92
1.4 Should .....92
1.5 May .....92
1.6 Can .....93



LifeScan, Inc. Ex. 1019
Page 163 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 164 of 1420
IPR2025-00103

# 1 USE OF IEEE LANGUAGE

One of the purposes of this terminology is to make it easy for the reader to identify text that describes requirements as opposed to background information. The general term for text that describes attributes that are required for proper implementation of Bluetooth wireless technology is normative. The general term for language that provides background and context for normative text is informative. These terms are used in various sections to clarify implementation requirements.

Many portions of the Bluetooth Specification use imprecise or inaccurate terms to describe attributes of the protocol. This subsection describes the correct usage of key terms that indicate degree of requirements for processes and data structures. The information here was derived from the Institute of Electrical and Electronic Engineers (IEEE) Style Guide, see http://standards.ieee.org/guides/style/.

The following list is a summary of the terms to be discussed in more detail below:

| | |
|---|---|
| *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 -- only used 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> |

For clarity of the definition of those terms, the following sections document why and how they are used. For these sections only, the IEEE terms are italicized to indicate their use as a noun. Uses and examples of the use of the terms in this section are underlined.

## 1.1 SHALL

The word *shall* is used to indicate mandatory requirements that shall be followed in order to conform to the specification and from which no deviation is permitted.



LifeScan, Inc. Ex. 1019
Page 165 of 1420
IPR2025-00103

There is a strong implication that the presence of the word *shall* indicates a testable requirement. All testable requirements shall be reflected in the Protocol Implementation Conformance Statement (PICS). In turn, all PICS indicators should be reflected in the Test Cases (TCs) either directly or indirectly.

A direct reference is a specific test for the attribute cited in the text. For example, a minimum value for a given parameter may be an entry in the TCs. Indirect test coverage may be appropriate if the existence of the attribute is requisite for passing a higher level test.

## 1.2 MUST

The word *must* shall not be used when stating mandatory requirements. *Must* is used only to describe unavoidable situations and is seldom appropriate for the text of a Specification.

An example of an appropriate use of the term *must* is: “the Bluetooth radios must be in range of each other to communicate”.

## 1.3 WILL

The use of the word *will* shall not be used when stating mandatory requirements. The term *will* is only used in statements of fact. As with the term *must*, *will* is not generally applicable to the description of a protocol. An example of appropriate use of will is: “when power is removed from the radio, it can be assumed that communications will fail”

## 1.4 SHOULD

*Should* equals *is recommended that*. The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable without mentioning or excluding others. Alternatively it may indicate that a certain course of action is preferred but not necessarily required. Finally, in the negative form, it indicates a certain course of action is deprecated but not prohibited.

In the Bluetooth Specification the term designates an optional attribute that may require an entry in the PICS.

Explicit specification of alternatives should be done when using *should*.

## 1.5 MAY

The word *may* is used to indicate a course of action permissible within the limits of the specification. The term *may* equals *is permitted*. This is generally used when there is a single, optional attribute described, but multiple alternatives may be cited.



LifeScan, Inc. Ex. 1019
Page 166 of 1420
IPR2025-00103

The use of *may* implies an optional condition in the PICS and therefore may need to be reflected in the corresponding test cases.

## 1.6 CAN

The word *can* is used for statements of possibility and capability, whether material, physical, or causal. The term *can* equals *is able to*.

The term *can* shall be used only in informative text. It describes capabilities by virtue of the rules established by normative text.



LifeScan, Inc. Ex. 1019
Page 167 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 168 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 169 of 1420
IPR2025-00103



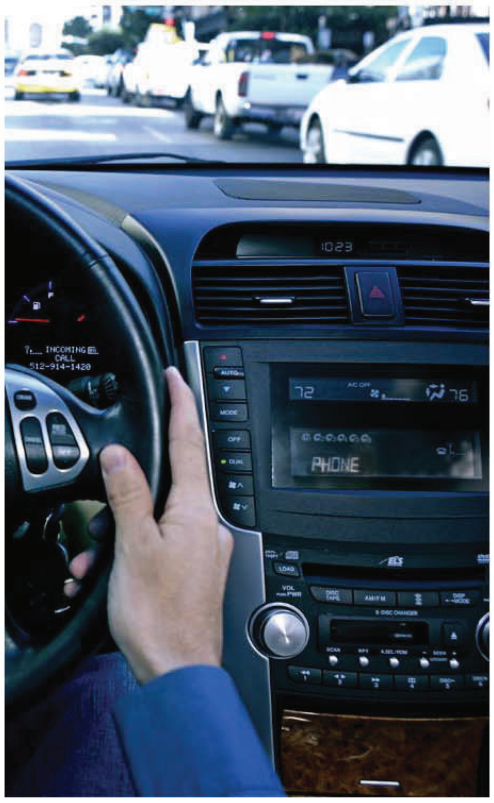


LifeScan, Inc. Ex. 1019
Page 170 of 1420
IPR2025-00103

Specification Volume 2

# SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Wireless Connections Made Easy




# Core System Package

## [Controller volume]

**Covered Core Package version:**
**2.1 + EDR**

Bluetooth®



LifeScan, Inc. Ex. 1019
Page 171 of 1420
IPR2025-00103

## Revision History

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

## Contributors

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

## Web Site

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

## Disclaimer and Copyright Notice

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

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

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.



LifeScan, Inc. Ex. 1019
Page 172 of 1420
IPR2025-00103

THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

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

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

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

Copyright © 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007

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

*Third-party brands and names are the property of their respective owners.

LifeScan, Inc. Ex. 1019
Page 174 of 1420
IPR2025-00103

# TABLE OF CONTENTS

**Part A**
**RADIO SPECIFICATION**

**Contents** ..... **27**
**1 Scope** ..... **29**
**2 Frequency Bands and Channel Arrangement** ..... **31**
**3 Transmitter Characteristics** ..... **32**
3.1 Basic Rate ..... 33
3.1.1 Modulation Characteristics ..... 33
3.1.2 Spurious Emissions ..... 34
3.1.3 Radio Frequency Tolerance ..... 35
3.2 Enhanced Data Rate ..... 35
3.2.1 Modulation Characteristics ..... 35
3.2.2 Spurious Emissions ..... 38
3.2.3 Radio Frequency Tolerance ..... 39
3.2.4 Relative Transmit Power ..... 40
**4 Receiver Characteristics** ..... **41**
4.1 Basic Rate ..... 41
4.1.1 Actual Sensitivity Level ..... 41
4.1.2 Interference Performance ..... 41
4.1.3 Out-of-Band Blocking ..... 42
4.1.4 Intermodulation Characteristics ..... 42
4.1.5 Maximum Usable Level ..... 43
4.1.6 Receiver Signal Strength Indicator ..... 43
4.1.7 Reference Signal Definition ..... 43
4.2 Enhanced Data Rate ..... 43
4.2.1 Actual Sensitivity Level ..... 43
4.2.2 BER Floor Performance ..... 43
4.2.3 Interference Performance ..... 43
4.2.4 Maximum Usable Level ..... 44
4.2.5 Out-of-Band and Intermodulation Characteristics ..... 45
4.2.6 Reference Signal Definition ..... 45
**5 Appendix A** ..... **46**
5.1 Nominal Test Conditions ..... 46
5.1.1 Nominal temperature ..... 46
5.1.2 Nominal power source ..... 46



LifeScan, Inc. Ex. 1019
Page 175 of 1420
IPR2025-00103

5.2 Extreme Test Conditions ..... 47
5.2.1 Extreme temperatures ..... 47
5.2.2 Extreme power source voltages ..... 47
**6 Appendix B ..... 48**
**7 Appendix C ..... 49**
7.1 Enhanced Data Rate Modulation Accuracy ..... 49

**Part B**
**BASEBAND SPECIFICATION**

**Contents ..... 55**
**1 General Description ..... 59**
1.1 Bluetooth Clock ..... 60
1.2 Bluetooth Device Addressing ..... 62
1.2.1 Reserved addresses ..... 62
1.3 Access Codes ..... 63
**2 Physical Channels ..... 64**
2.1 Physical Channel Definition ..... 65
2.2 Basic Piconet Physical Channel ..... 65
2.2.1 Master-slave definition ..... 65
2.2.2 Hopping characteristics ..... 66
2.2.3 Time slots ..... 66
2.2.4 Piconet clocks ..... 67
2.2.5 Transmit/receive timing ..... 67
2.3 Adapted Piconet Physical Channel ..... 70
2.3.1 Hopping characteristics ..... 70
2.4 Page Scan Physical Channel ..... 71
2.4.1 Clock estimate for paging ..... 71
2.4.2 Hopping characteristics ..... 71
2.4.3 Paging procedure timing ..... 72
2.4.4 Page response timing ..... 73
2.5 Inquiry Scan Physical Channel ..... 75
2.5.1 Clock for inquiry ..... 75
2.5.2 Hopping characteristics ..... 75
2.5.3 Inquiry procedure timing ..... 75
2.5.4 Inquiry response timing ..... 75
2.6 Hop Selection ..... 77
2.6.1 General selection scheme ..... 77
2.6.2 Selection kernel ..... 81
2.6.3 Adapted hop selection kernel ..... 84



LifeScan, Inc. Ex. 1019
Page 176 of 1420
IPR2025-00103

2.6.4 Control word ....85
**3 Physical Links ....90**
3.1 Link Supervision ....90
**4 Logical Transports ....91**
4.1 General ....91
4.2 Logical Transport Address (LT_ADDR) ....91
4.3 Synchronous Logical Transports ....92
4.4 Asynchronous Logical Transport ....92
4.5 Transmit/Receive Routines ....93
4.5.1 TX Routine ....93
4.5.2 RX routine ....96
4.5.3 Flow control ....97
4.6 Active Slave Broadcast Transport ....98
4.7 Parked Slave Broadcast Transport ....99
4.7.1 Parked member address (PM_ADDR) ....99
4.7.2 Access request address (AR_ADDR) ....99
**5 Logical Links ....100**
5.1 Link Control Logical Link (LC) ....100
5.2 ACL Control Logical Link (ACL-C) ....100
5.3 User Asynchronous/Isochronous Logical Link (ACL-U) ....100
5.3.1 Pausing the ACL-U logical link ....101
5.4 User Synchronous Data Logical Link (SCO-S) ....101
5.5 User Extended Synchronous Data Logical Link (eSCO-S) ....101
5.6 Logical Link Priorities ....101
**6 Packets ....102**
6.1 General Format ....102
6.1.1 Basic Rate ....102
6.1.2 Enhanced Data Rate ....102
6.2 Bit Ordering ....103
6.3 Access Code ....104
6.3.1 Access code types ....104
6.3.2 Preamble ....105
6.3.3 Sync word ....105
6.3.4 Trailer ....108
6.4 Packet Header ....109
6.4.1 LT_ADDR ....109
6.4.2 TYPE ....109
6.4.3 FLOW ....110
6.4.4 ARQN ....110



LifeScan, Inc. Ex. 1019
Page 177 of 1420
IPR2025-00103

6.4.5 SEQN ..... 110
6.4.6 HEC ..... 110
6.5 Packet Types ..... 111
6.5.1 Common packet types ..... 112
6.5.2 SCO packets ..... 116
6.5.3 eSCO packets ..... 117
6.5.4 ACL packets ..... 119
6.6 Payload Format ..... 121
6.6.1 Synchronous Data Field ..... 121
6.6.2 Asynchronous data field ..... 123
6.7 Packet Summary ..... 127
**7 Bitstream Processing ..... 129**
7.1 Error Checking ..... 130
7.1.1 HEC generation ..... 130
7.1.2 CRC generation ..... 131
7.2 Data Whitening ..... 133
7.3 Error Correction ..... 134
7.4 FEC Code: Rate 1/3 ..... 134
7.5 FEC Code: Rate 2/3 ..... 135
7.6 ARQ Scheme ..... 136
7.6.1 Unnumbered ARQ ..... 136
7.6.2 Retransmit filtering ..... 139
7.6.3 Flushing Payloads ..... 142
7.6.4 Multi-slave considerations ..... 142
7.6.5 Broadcast packets ..... 143
7.7 Erroneous Synchronous Data Reporting ..... 144
**8 Link Controller Operation ..... 145**
8.1 Overview of States ..... 145
8.2 Standby State ..... 146
8.3 Connection Establishment Substates ..... 146
8.3.1 Page scan substate ..... 146
8.3.2 Page substate ..... 148
8.3.3 Page response substates ..... 151
8.4 Device Discovery Substates ..... 155
8.4.1 Inquiry scan substate ..... 156
8.4.2 Inquiry substate ..... 157
8.4.3 Inquiry Response Substate ..... 158
8.5 Connection State ..... 160
8.6 Active Mode ..... 161



LifeScan, Inc. Ex. 1019
Page 178 of 1420
IPR2025-00103

8.6.1 Polling in the active mode ....162
8.6.2 SCO....162
8.6.3 eSCO....164
8.6.4 Broadcast scheme....166
8.6.5 Role switch ....167
8.6.6 Scatternet ....169
8.6.7 Hop sequence switching ....170
8.6.8 Channel classification and channel map selection....173
8.6.9 Power Management ....174
8.7 Sniff Mode ....175
8.7.1 Sniff Transition Mode....176
8.7.2 Sniff Subrating....177
8.8 Hold Mode....178
8.9 Park State....179
8.9.1 Beacon train ....179
8.9.2 Beacon access window ....182
8.9.3 Parked slave synchronization....183
8.9.4 Parking ....184
8.9.5 Master-initiated unparking ....185
8.9.6 Slave-initiated unparking ....185
8.9.7 Broadcast scan window....186
8.9.8 Polling in the park state ....186

**9 Audio ....187**
9.1 LOG PCM CODEC....187
9.2 CVSD CODEC ....187
9.3 Error Handling....190
9.4 General Audio Requirements....190
9.4.1 Signal levels ....190
9.4.2 CVSD audio quality ....190

**10 List of Figures....191**

**11 List of Tables ....194**
**Appendix A: General Audio Recommendations ....195**
**Appendix B: Timers ....199**
**Appendix C: Recommendations for AFH Operation in Park, Hold, and Sniff ....201**

**Part C**
**LINK MANAGER PROTOCOL**

**Contents ....205**



LifeScan, Inc. Ex. 1019
Page 179 of 1420
IPR2025-00103

**1 Introduction ..... 207**

**2 General Rules ..... 208**

2.1 Message Transport ..... 208
2.2 Synchronization ..... 208
2.3 Packet Format ..... 209
2.4 Transactions ..... 210
2.4.1 LMP Response Timeout ..... 211
2.5 Error Handling ..... 211
2.5.1 Transaction collision resolution ..... 212
2.6 Procedure Rules ..... 212
2.7 General Response Messages ..... 213
2.8 LMP Message Constraints ..... 213

**3 Device Features ..... 214**

3.1 General Description ..... 214
3.2 Feature Definitions ..... 214
3.3 Feature Mask Definition ..... 220
3.4 Link Manager Interoperability policy ..... 222

**4 Procedure Rules ..... 223**

4.1 Connection Control ..... 223
4.1.1 Connection establishment ..... 223
4.1.2 Detach ..... 224
4.1.3 Power control ..... 225
4.1.4 Adaptive frequency hopping ..... 227
4.1.5 Channel classification ..... 230
4.1.6 Link supervision ..... 232
4.1.7 Channel quality driven data rate change (CQDDR) ..... 233
4.1.8 Quality of service (QoS) ..... 234
4.1.9 Paging scheme parameters ..... 236
4.1.10 Control of multi-slot packets ..... 237
4.1.11 Enhanced Data Rate ..... 237
4.1.12 Encapsulated LMP PDUs ..... 239
4.2 Security ..... 240
4.2.1 Authentication ..... 240
4.2.2 Pairing ..... 242
4.2.3 Change link key ..... 245
4.2.4 Change current link key type ..... 246
4.2.5 Encryption ..... 248
4.2.6 Request supported encryption key size ..... 255
4.2.7 Secure Simple Pairing ..... 255



LifeScan, Inc. Ex. 1019
Page 180 of 1420
IPR2025-00103

4.3 Informational Requests ....268
4.3.1 Timing accuracy ....268
4.3.2 Clock offset ....270
4.3.3 LMP version ....270
4.3.4 Supported features ....271
4.3.5 Name request ....273
4.4 Role Switch ....274
4.4.1 Slot offset ....274
4.4.2 Role switch ....275
4.5 Modes of Operation ....277
4.5.1 Hold mode ....277
4.5.2 Park state ....279
4.5.3 Sniff mode ....286
4.6 Logical Transports ....290
4.6.1 SCO logical transport ....290
4.6.2 eSCO logical transport ....293
4.7 Test Mode ....298
4.7.1 Activation and deactivation of test mode ....298
4.7.2 Control of test mode ....299
4.7.3 Summary of test mode PDUs ....299

**5 Summary ....303**
5.1 PDU Summary ....303
5.2 Parameter Definitions ....313
5.3 LMP Encapsulated ....323
5.4 Default Values ....324

**6 List of Figures ....325**

**7 List of Tables ....329**

**Part D**
**ERROR CODES**

**Contents ....333**

**1 Overview of Error Codes ....335**
1.1 Usage Descriptions ....335
1.2 HCI Command Errors ....335
1.3 List of Error Codes ....336

**2 Error Code Descriptions ....338**
2.1 Unknown HCI Command (0X01) ....338
2.2 Unknown Connection Identifier (0X02) ....338
2.3 Hardware Failure (0X03) ....338



LifeScan, Inc. Ex. 1019
Page 181 of 1420
IPR2025-00103

2.4 Page Timeout (0X04) ..... 338
2.5 Authentication Failure (0X05) ..... 338
2.6 PIN or key Missing (0X06) ..... 338
2.7 Memory Capacity Exceeded (0X07) ..... 338
2.8 Connection Timeout (0X08) ..... 339
2.9 Connection Limit Exceeded (0X09) ..... 339
2.10 Synchronous Connection Limit to a Device Exceeded (0X0A) 339
2.11 ACL Connection Already Exists (0X0B) ..... 339
2.12 Command Disallowed (0X0C) ..... 339
2.13 Connection Rejected due to Limited Resources (0X0D) ..... 339
2.14 Connection Rejected due to Security Reasons (0X0E) ..... 339
2.15 Connection Rejected due to Unacceptable BD_ADDR (0X0F) 340
2.16 Connection Accept Timeout Exceeded (0X10) ..... 340
2.17 Unsupported Feature or Parameter Value (0X11) ..... 340
2.18 Invalid HCI Command Parameters (0X12) ..... 340
2.19 Remote User Terminated Connection (0X13) ..... 340
2.20 Remote Device Terminated Connection due to Low Resources (0X14) 341
2.21 Remote Device Terminated Connection due to Power Off (0X15). 341
2.22 Connection Terminated by Local Host (0X16) ..... 341
2.23 Repeated Attempts (0X17) ..... 341
2.24 Pairing not Allowed (0X18) ..... 341
2.25 Unknown LMP PDU (0X19) ..... 341
2.26 Unsupported Remote Feature / Unsupported LMP Feature (0X1A) 341
2.27 SCO Offset Rejected (0X1B) ..... 341
2.28 SCO Interval Rejected (0X1C) ..... 342
2.29 SCO Air Mode Rejected (0X1D) ..... 342
2.30 Invalid LMP Parameters (0X1E) ..... 342
2.31 Unspecified Error (0X1F) ..... 342
2.32 Unsupported LMP Parameter Value (0X20) ..... 342
2.33 Role Change Not Allowed (0X21) ..... 342
2.34 LMP Response Timeout (0X22) ..... 342
2.35 LMP Error Transaction Collision (0X23) ..... 343
2.36 LMP PDU Not Allowed (0X24) ..... 343
2.37 Encryption Mode Not Acceptable (0X25) ..... 343
2.38 Link Key Can Not be Changed (0X26) ..... 343
2.39 Requested Qos Not Supported (0X27) ..... 343
2.40 Instant Passed (0X28) ..... 343
2.41 Pairing with Unit Key Not Supported (0X29) ..... 343
2.42 Different Transaction Collision (0x2a) ..... 343
2.43 QoS Unacceptable Parameter (0X2C) ..... 343

2.44 QoS Rejected (0X2D) ....344
2.45 Channel Classification Not Supported (0X2E) ....344
2.46 Insufficient Security (0X2F) ....344
2.47 Parameter out of Mandatory Range (0X30) ....344
2.48 Role Switch Pending (0X32) ....344
2.49 Reserved Slot Violation (0X34) ....344
2.50 Role Switch Failed (0X35) ....344
2.51 Extended Inquiry Response Too Large (0x36) ....344
2.52 Simple Pairing Not Supported By Host (0X37) ....345
2.53 Host Busy–Pairing(0X38) ....345

**Part E**
**HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION**

**Contents ....349**

**1 Introduction ....357**
1.1 Lower Layers of the Bluetooth Software Stack ....357

**2 Overview of Host Controller Transport Layer ....359**

**3 Overview of Commands and Events ....360**
3.1 Generic Events ....361
3.2 Device Setup ....361
3.3 Controller Flow Control ....362
3.4 Controller Information ....362
3.5 Controller Configuration ....363
3.6 Device Discovery ....364
3.7 Connection Setup ....366
3.8 Remote Information ....368
3.9 Synchronous Connections ....369
3.10 Connection State ....370
3.11 Piconet Structure ....371
3.12 Quality of Service ....372
3.13 Physical Links ....374
3.14 Host Flow Control ....375
3.15 Link Information ....376
3.16 Authentication and Encryption ....377
3.17 Testing ....380
3.18 Alphabetical List of Commands and Events ....381

**4 HCI Flow Control ....386**
4.1 Host to Controller Data Flow Control ....386
4.2 Controller to Host Data Flow Control ....387
4.3 Disconnection Behavior ....388
4.4 Command Flow Control ....388

4.5 Command Error Handling ..... 389
**5 HCI Data Formats ..... 390**
5.1 Introduction ..... 390
5.2 Data and Parameter Formats ..... 390
5.3 Connection Handles ..... 391
5.4 Exchange of HCI-Specific Information ..... 392
5.4.1 HCI Command Packet ..... 392
5.4.2 HCI ACL Data Packets ..... 394
5.4.3 HCI Synchronous Data Packets ..... 396
5.4.4 HCI Event Packet ..... 398
**6 HCI Configuration Parameters ..... 399**
6.1 Scan Enable ..... 399
6.2 Inquiry Scan Interval ..... 399
6.3 Inquiry Scan Window ..... 400
6.4 Inquiry Scan Type ..... 400
6.5 Inquiry Mode ..... 400
6.6 Page Timeout ..... 401
6.7 Connection Accept Timeout ..... 401
6.8 Page Scan Interval ..... 402
6.9 Page Scan Window ..... 402
6.10 Page Scan Period Mode (Deprecated) ..... 402
6.11 Page Scan Type ..... 403
6.12 Voice Setting ..... 403
6.13 PIN Type ..... 404
6.14 Link Key ..... 404
6.15 Failed Contact Counter ..... 405
6.16 Hold Mode Activity ..... 405
6.17 Link Policy Settings ..... 406
6.18 Flush Timeout ..... 407
6.19 Num Broadcast Retransmissions ..... 407
6.20 Link Supervision Timeout ..... 408
6.21 Synchronous Flow Control Enable ..... 408
6.22 Local Name ..... 409
6.23 Extended Inquiry Response ..... 409
6.24 Erroneous Data Reporting ..... 409
6.25 Class Of Device ..... 410
6.26 Supported Commands ..... 410
**7 HCI Commands and Events ..... 416**
7.1 Link Control Commands ..... 416
7.1.1 Inquiry Command ..... 416
7.1.2 Inquiry Cancel Command ..... 418



LifeScan, Inc. Ex. 1019
Page 184 of 1420
IPR2025-00103

7.1.3 Periodic Inquiry Mode Command ....419
7.1.4 Exit Periodic Inquiry Mode Command ....422
7.1.5 Create Connection Command ....423
7.1.6 Disconnect Command ....426
7.1.7 Create Connection Cancel Command ....427
7.1.8 Accept Connection Request Command ....429
7.1.9 Reject Connection Request Command ....431
7.1.10 Link Key Request Reply Command ....432
7.1.11 Link Key Request Negative Reply Command ....434
7.1.12 PIN Code Request Reply Command ....435
7.1.13 PIN Code Request Negative Reply Command ....437
7.1.14 Change Connection Packet Type Command ....438
7.1.15 Authentication Requested Command ....441
7.1.16 Set Connection Encryption Command ....443
7.1.17 Change Connection Link Key Command ....444
7.1.18 Master Link Key Command ....445
7.1.19 Remote Name Request Command ....446
7.1.20 Remote Name Request Cancel Command ....448
7.1.21 Read Remote Supported Features Command ....449
7.1.22 Read Remote Extended Features Command ....450
7.1.23 Read Remote Version Information Command ....451
7.1.24 Read Clock Offset Command ....452
7.1.25 Read LMP Handle Command ....453
7.1.26 Setup Synchronous Connection Command ....455
7.1.27 Accept Synchronous Connection Request Command 459
7.1.28 Reject Synchronous Connection Request Command .463
7.1.29 IO Capability Request Reply Command ....464
7.1.30 User Confirmation Request Reply Command ....467
7.1.31 User Confirmation Request Negative Reply Command ....468
7.1.32 User Passkey Request Reply Command ....469
7.1.33 User Passkey Request Negative Reply Command ....470
7.1.34 Remote OOB Data Request Reply Command ....471
7.1.35 Remote OOB Data Request Negative Reply Command ....473
7.1.36 IO Capability Request Negative Reply Command ....474
7.2 Link Policy Commands ....475
7.2.1 Hold Mode Command ....475



LifeScan, Inc. Ex. 1019
Page 185 of 1420
IPR2025-00103

7.2.2 Sniff Mode Command ..... 477
7.2.3 Exit Sniff Mode Command ..... 480
7.2.4 Park State Command ..... 481
7.2.5 Exit Park State Command ..... 483
7.2.6 QoS Setup Command ..... 484
7.2.7 Role Discovery Command ..... 487
7.2.8 Switch Role Command ..... 488
7.2.9 Read Link Policy Settings Command ..... 490
7.2.10 Write Link Policy Settings Command ..... 491
7.2.11 Read Default Link Policy Settings Command ..... 493
7.2.12 Write Default Link Policy Settings Command ..... 494
7.2.13 Flow Specification Command ..... 495
7.2.14 Sniff Subrating Command ..... 497
7.3 Controller & Baseband Commands ..... 498
7.3.1 Set Event Mask Command ..... 499
7.3.2 Reset Command ..... 502
7.3.3 Set Event Filter Command ..... 503
7.3.4 Flush Command ..... 508
7.3.5 Read PIN Type Command ..... 510
7.3.6 Write PIN Type Command ..... 511
7.3.7 Create New Unit Key Command ..... 512
7.3.8 Read Stored Link Key Command ..... 513
7.3.9 Write Stored Link Key Command ..... 514
7.3.10 Delete Stored Link Key Command ..... 516
7.3.11 Write Local Name Command ..... 517
7.3.12 Read Local Name Command ..... 518
7.3.13 Read Connection Accept Timeout Command ..... 519
7.3.14 Write Connection Accept Timeout Command ..... 520
7.3.15 Read Page Timeout Command ..... 521
7.3.16 Write Page Timeout Command ..... 522
7.3.17 Read Scan Enable Command ..... 523
7.3.18 Write Scan Enable Command ..... 524
7.3.19 Read Page Scan Activity Command ..... 525
7.3.20 Write Page Scan Activity Command ..... 527
7.3.21 Read Inquiry Scan Activity Command ..... 528
7.3.22 Write Inquiry Scan Activity Command ..... 529
7.3.23 Read Authentication Enable Command ..... 529
7.3.24 Write Authentication Enable Command ..... 530

7.3.25 Read Class of Device Command .....532
7.3.26 Write Class of Device Command .....532
7.3.27 Read Voice Setting Command .....534
7.3.28 Write Voice Setting Command .....535
7.3.29 Read Automatic Flush Timeout Command.....536
7.3.30 Write Automatic Flush Timeout Command.....537
7.3.31 Read Num Broadcast Retransmissions Command.....538
7.3.32 Write Num Broadcast Retransmissions Command .....539
7.3.33 Read Hold Mode Activity Command .....540
7.3.34 Write Hold Mode Activity Command.....541
7.3.35 Read Transmit Power Level Command.....542
7.3.36 Read Synchronous Flow Control Enable Command...544
7.3.37 Write Synchronous Flow Control Enable Command ...545
7.3.38 Set Controller To Host Flow Control Command .....546
7.3.39 Host Buffer Size Command.....547
7.3.40 Host Number Of Completed Packets Command.....548
7.3.41 Read Link Supervision Timeout Command.....551
7.3.42 Write Link Supervision Timeout Command .....553
7.3.43 Read Number Of Supported IAC Command.....555
7.3.44 Read Current IAC LAP Command .....556
7.3.45 Write Current IAC LAP Command.....557
7.3.46 Set AFH Host Channel Classification Command .....559
7.3.47 Read Inquiry Scan Type Command .....560
7.3.48 Write Inquiry Scan Type Command.....561
7.3.49 Read Inquiry Mode Command .....562
7.3.50 Write Inquiry Mode Command.....563
7.3.51 Read Page Scan Type Command.....564
7.3.52 Write Page Scan Type Command .....565
7.3.53 Read AFH Channel Assessment Mode Command .....566
7.3.54 Write AFH Channel Assessment Mode Command .....567
7.3.55 Read Extended Inquiry Response Command .....568
7.3.56 Write Extended Inquiry Response Command .....569
7.3.57 Refresh Encryption Key.....570
7.3.58 Read Simple Pairing Mode Command .....571
7.3.59 Write Simple Pairing Mode Command .....572
7.3.60 Read Local OOB Data Command .....573
7.3.61 Read Inquiry Response Transmit Power Level Command .....574



LifeScan, Inc. Ex. 1019
Page 187 of 1420
IPR2025-00103

7.3.62 Write Inquiry Transmit Power Level Command ..... 575
7.3.63 Send Keypress Notification Command ..... 576
7.3.64 Read Default Erroneous Data Reporting ..... 577
7.3.65 Write Default Erroneous Data Reporting ..... 578
7.3.66 Enhanced Flush Command ..... 579
7.4 Informational Parameters ..... 580
7.4.1 Read Local Version Information Command ..... 580
7.4.2 Read Local Supported Commands Command ..... 583
7.4.3 Read Local Supported Features Command ..... 584
7.4.4 Read Local Extended Features Command ..... 585
7.4.5 Read Buffer Size Command ..... 587
7.4.6 Read BD_ADDR Command ..... 589
7.5 Status Parameters ..... 590
7.5.1 Read Failed Contact Counter Command ..... 590
7.5.2 Reset Failed Contact Counter Command ..... 592
7.5.3 Read Link Quality Command ..... 593
7.5.4 Read RSSI Command ..... 594
7.5.5 Read AFH Channel Map Command ..... 596
7.5.6 Read Clock Command ..... 598
7.6 Testing Commands ..... 600
7.6.1 Read Loopback Mode Command ..... 600
7.6.2 Write Loopback Mode Command ..... 601
7.6.3 Enable Device Under Test Mode Command ..... 604
7.6.4 Write Simple Pairing Debug Mode Command ..... 604
7.7 Events ..... 606
7.7.1 Inquiry Complete Event ..... 606
7.7.2 Inquiry Result Event ..... 607
7.7.3 Connection Complete Event ..... 609
7.7.4 Connection Request Event ..... 610
7.7.5 Disconnection Complete Event ..... 612
7.7.6 Authentication Complete Event ..... 613
7.7.7 Remote Name Request Complete Event ..... 614
7.7.8 Encryption Change Event ..... 615
7.7.9 Change Connection Link Key Complete Event ..... 616
7.7.10 Master Link Key Complete Event ..... 617
7.7.11 Read Remote Supported Features Complete Event ..... 618
7.7.12 Read Remote Version Information Complete Event ..... 619
7.7.13 QoS Setup Complete Event ..... 620



LifeScan, Inc. Ex. 1019
Page 188 of 1420
IPR2025-00103

7.7.14 Command Complete Event ....622
7.7.15 Command Status Event....623
7.7.16 Hardware Error Event....624
7.7.17 Flush Occurred Event....624
7.7.18 Role Change Event ....625
7.7.19 Number Of Completed Packets Event ....626
7.7.20 Mode Change Event....627
7.7.21 Return Link Keys Event....629
7.7.22 PIN Code Request Event ....630
7.7.23 Link Key Request Event ....631
7.7.24 Link Key Notification Event....632
7.7.25 Loopback Command Event ....634
7.7.26 Data Buffer Overflow Event ....634
7.7.27 Max Slots Change Event....635
7.7.28 Read Clock Offset Complete Event....636
7.7.29 Connection Packet Type Changed Event....637
7.7.30 QoS Violation Event ....639
7.7.31 Page Scan Repetition Mode Change Event....640
7.7.32 Flow Specification Complete Event ....641
7.7.33 Inquiry Result with RSSI Event ....643
7.7.34 Read Remote Extended Features Complete Event ....645
7.7.35 Synchronous Connection Complete Event....646
7.7.36 Synchronous Connection Changed event....649
7.7.37 Sniff Subrating Event....650
7.7.38 Extended Inquiry Result Event ....652
7.7.39 Encryption Key Refresh Complete ....654
7.7.40 IO Capability Request Event ....655
7.7.41 IO Capability Response Event ....656
7.7.42 User Confirmation Request Event....657
7.7.43 User Passkey Request Event....658
7.7.44 Remote OOB Data Request Event....658
7.7.45 Simple Pairing Complete Event....659
7.7.46 Link Supervision Timeout Changed Event ....660
7.7.47 Enhanced Flush Complete Event....661
7.7.48 User Passkey Notification Event ....661
7.7.49 Keypress Notification Event ....662
7.7.50 Remote Host Supported Features Notification Event..662

**8 List of Figures....664**

**9 List of Tables ..... 665**

**10 Appendix A: Deprecated Commands, Events and Configuration Parameters ..... 666**

10.1 Read Page Scan Mode Command ..... 667
10.2 Write Page Scan Mode Command ..... 668
10.3 Read Page Scan Period Mode Command ..... 669
10.4 Write Page Scan Period Mode Command ..... 670
10.5 Add SCO Connection Command ..... 671
10.6 Page Scan Mode Change Event ..... 673
10.7 Read Country Code Command ..... 674
10.8 Read Encryption Mode Command ..... 675
10.9 Write Encryption Mode Command ..... 675
10.10 Deprecated Parameters ..... 676
10.10.1 Authentication Enable ..... 676
10.10.2 Encryption Mode ..... 676
10.10.3 Page Scan Mode ..... 677

**Part F**
**MESSAGE SEQUENCE CHARTS**

**Contents ..... 681**

**1 Introduction ..... 683**
1.1 Notation ..... 683
1.2 Flow of Control ..... 684
1.3 Example MSC ..... 684

**2 Services Without Connection Request ..... 685**
2.1 Remote Name Request ..... 685
2.2 One-time Inquiry ..... 687
2.3 Periodic Inquiry ..... 689

**3 ACL Connection Establishment and Detachment ..... 691**
3.1 Connection Setup ..... 692

**4 Optional Activities After ACL Connection Establishment ..... 699**
4.1 Authentication Requested ..... 699
4.2 Simple Pairing Message Sequence Charts ..... 700
4.2.1 Optional OOB Information Collection ..... 700
4.2.2 Enable Simple Pairing ..... 701
4.2.3 Connection Establishment ..... 701
4.2.4 L2CAP Connection Request for a Secure Service ..... 701
4.2.5 Optional OOB Information Transfer ..... 702
4.2.6 Start Simple Pairing ..... 702
4.2.7 IO Capability Exchange ..... 703

4.2.8 Public Key Exchange ....703
4.2.9 Authentication....704
4.2.10 Numeric Comparison....704
4.2.11 Numeric Comparison Failure on Initiating Side ....705
4.2.12 Numeric Comparison Failure on Responding Side ....706
4.2.13 Passkey Entry ....707
4.2.14 Passkey Entry Failure on Responding Side ....708
4.2.15 Passkey Entry Failure on Initiator Side ....709
4.2.16 Out of Band ....709
4.2.17 OOB Failure on Initiator Side ....710
4.2.18 DHKey Checks ....711
4.2.19 Calculate Link Key....711
4.2.20 Enable Encryption ....712
4.2.21 L2CAP Connection Response....713
4.3 Link Supervision Timeout Changed Event ....713
4.4 Set Connection Encryption....714
4.5 Change Connection Link Key....715
4.6 Change Connection Link Key with Encryption Pause and Resume....715
4.7 Master Link Key ....717
4.8 Read Remote Supported Features ....719
4.9 Read Remote Extended Features....719
4.10 Read Clock Offset ....720
4.11 Role Switch on an Encrypted Link using Encryption Pause and Resume....720
4.12 Refreshing Encryption Keys....721
4.13 Read Remote Version Information ....722
4.14 QOS Setup....722
4.15 Switch Role ....723
**5 Synchronous Connection Establishment and Detachment ....725**
5.1 Synchronous Connection Setup....725
**6 Sniff, Hold and Park ....730**
6.1 Sniff Mode ....730
6.2 Hold Mode....731
6.3 Park State....733
**7 Buffer Management, Flow Control....736**
**8 Loopback Mode....738**
8.1 Local Loopback Mode....738
8.2 Remote Loopback Mode....740
**9 List of Figures....742**

Part G
SAMPLE DATA

Contents ..... 747

1 Encryption Sample Data ..... 749
1.1 Generating Kc' from Kc, ..... 749
1.2 First Set of Sample Data ..... 752
1.3 Second Set of Sample Data ..... 760
1.4 Third Set of Samples ..... 768
1.5 Fourth Set of Samples ..... 776
2 Frequency Hopping Sample Data ..... 784
2.1 First set ..... 785
2.2 Second set ..... 791
2.3 Third set ..... 797
3 Access Code Sample Data ..... 803
4 HEC and Packet Header Sample Data ..... 806
5 CRC Sample Data ..... 807
6 Complete Sample Packets ..... 808
6.1 Example of DH1 Packet ..... 808
6.2 Example of DM1 Packet ..... 809
7 Simple Pairing Sample Data ..... 810
7.1 P-192 Sample Data ..... 810
7.1.1 Data Set 1 ..... 810
7.1.2 Data Set 2 ..... 810
7.1.3 Data Set 3 ..... 810
7.1.4 Data Set 4 ..... 810
7.1.5 Data Set 5 ..... 810
7.1.6 Data Set 6 ..... 811
7.1.7 Data Set 7 ..... 811
7.1.8 Data Set 8 ..... 811
7.1.9 Data Set 9 ..... 811
7.1.10 Data Set 10 ..... 811
7.2 Hash Functions Sample Data ..... 811
7.2.1 f1() ..... 812
7.2.2 g() ..... 813
7.2.3 f2() ..... 813
7.2.4 f3() ..... 813
8 Whitening Sequence Sample Data ..... 821
9 FEC Sample Data ..... 824



LifeScan, Inc. Ex. 1019
Page 192 of 1420
IPR2025-00103

**10 Encryption Key Sample Data ....825**
10.1 Four Tests of E1 ....825
10.2 Four Tests of E21 ....830
10.3 Three Tests of E22 ....832
10.4 Tests of E22 With Pin Augmenting ....834
10.5 Four Tests of E3 ....844

**Part H**
**SECURITY SPECIFICATION**

**Contents ....853**

**1 Security Overview ....855**
1.1 Pausing Encryption and Role Switch ....856
1.2 Change Connection Link Keys ....856
1.3 Periodically Refreshing Encryption Keys ....856

**2 Random Number Generation ....857**

**3 Key Management ....859**
3.1 Key Types ....859
3.2 Key Generation and Initialization ....861
3.2.1 Generation of initialization key, ....862
3.2.2 Authentication ....862
3.2.3 Generation of a unit key ....862
3.2.4 Generation of a combination key ....863
3.2.5 Generating the encryption key ....864
3.2.6 Point-to-multipoint configuration ....865
3.2.7 Modifying the link keys ....865
3.2.8 Generating a master key ....866

**4 Encryption ....868**
4.1 Encryption Key Size Negotiation ....869
4.2 Encryption of Broadcast Messages ....869
4.3 Encryption Concept ....870
4.4 Encryption Algorithm ....871
4.4.1 The operation of the cipher ....873
4.5 LFSR Initialization ....874
4.6 Key Stream Sequence ....877

**5 Authentication ....878**
5.1 Repeated Attempts ....880

**6 The Authentication And Key-Generating Functions ....881**
6.1 The Authentication Function E1 ....881
6.2 The Functions Ar and A'r ....883



LifeScan, Inc. Ex. 1019
Page 193 of 1420
IPR2025-00103

6.2.1 The round computations ..... 883
6.2.2 The substitution boxes “e” and “l” ..... 883
6.2.3 Key scheduling ..... 884
6.3 E2-Key Generation Function for Authentication ..... 885
6.4 E3-Key Generation Function for Encryption ..... 887
**7 Secure Simple Pairing ..... 888**
7.1 Phase 1: Public Key Exchange ..... 889
7.2 Phase 2: Authentication Stage 1 ..... 890
7.2.1 Authentication Stage 1: Numeric Comparison Protocol ..... 890
7.2.2 Authentication Stage 1: Out of Band Protocol ..... 892
7.2.3 Authentication Stage 1: Passkey Entry Protocol ..... 894
7.3 Phase 3: Authentication Stage 2 ..... 896
7.4 Phase 4: Link Key Calculation ..... 897
7.5 Phase 5: LMP Authentication and Encryption ..... 897
7.6 Elliptic Curve Definition ..... 897
7.7 Cryptographic Function Definitions ..... 898
7.7.1 The Simple Pairing Commitment Function f1 ..... 898
7.7.2 The Simple Pairing Numeric Verification Function g ..... 899
7.7.3 The Simple Pairing Key Derivation Function f2 ..... 899
7.7.4 The Simple Pairing Check Function f3 ..... 900
**8 List of Figures ..... 902**
**9 List of Tables ..... 903**

# RADIO SPECIFICATION



LifeScan, Inc. Ex. 1019
Page 195 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 196 of 1420
IPR2025-00103

# CONTENTS

**1 Scope ..... 29**

**2 Frequency Bands and Channel Arrangement ..... 31**

**3 Transmitter Characteristics ..... 32**

3.1 Basic Rate ..... 33

3.1.1 Modulation Characteristics ..... 33

3.1.2 Spurious Emissions ..... 34

3.1.3 Radio Frequency Tolerance ..... 35

3.2 Enhanced Data Rate ..... 35

3.2.1 Modulation Characteristics ..... 35

3.2.2 Spurious Emissions ..... 38

3.2.3 Radio Frequency Tolerance ..... 39

3.2.4 Relative Transmit Power ..... 40

**4 Receiver Characteristics ..... 41**

4.1 Basic Rate ..... 41

4.1.1 Actual Sensitivity Level ..... 41

4.1.2 Interference Performance ..... 41

4.1.3 Out-of-Band Blocking ..... 42

4.1.4 Intermodulation Characteristics ..... 42

4.1.5 Maximum Usable Level ..... 43

4.1.6 Receiver Signal Strength Indicator ..... 43

4.1.7 Reference Signal Definition ..... 43

4.2 Enhanced Data Rate ..... 43

4.2.1 Actual Sensitivity Level ..... 43

4.2.2 BER Floor Performance ..... 43

4.2.3 Interference Performance ..... 43

4.2.4 Maximum Usable Level ..... 44

4.2.5 Out-of-Band and Intermodulation Characteristics ..... 45

4.2.6 Reference Signal Definition ..... 45

**5 Appendix A ..... 46**

5.1 Nominal Test Conditions ..... 46

5.1.1 Nominal temperature ..... 46

5.1.2 Nominal power source ..... 46

5.2 Extreme Test Conditions ..... 47

5.2.1 Extreme temperatures ..... 47

5.2.2 Extreme power source voltages ..... 47

**6 Appendix B ..... 48**

**7 Appendix C ..... 49**

7.1 Enhanced Data Rate Modulation Accuracy ..... 49



LifeScan, Inc. Ex. 1019
Page 198 of 1420
IPR2025-00103

# 1 SCOPE

Bluetooth devices operate in the unlicensed 2.4 GHz ISM (Industrial Scientific Medical) band. A frequency hop transceiver is applied to combat interference and fading.

Two modulation modes are defined. A mandatory mode, called Basic Rate, uses a shaped, binary FM modulation to minimize transceiver complexity. An optional mode, called Enhanced Data Rate, uses PSK modulation and has two variants: $\pi$/4-DQPSK and 8DPSK. The symbol rate for all modulation schemes is 1 Ms/s. The gross air data rate is 1 Mbps for Basic Rate, 2 Mbps for Enhanced Data Rate using $\pi$/4-DQPSK and 3 Mbps for Enhanced Data Rate using 8DPSK.

For full duplex transmission, a Time Division Duplex (TDD) scheme is used in both modes. This specification defines the requirements for a Bluetooth radio for the Basic Rate and Enhanced Data Rate modes.

Requirements are defined for two reasons:

- Provide compatibility between radios used in the system
- Define the quality of the system

The Bluetooth radio shall fulfil the stated requirements under the operating conditions specified in Appendix A and Appendix B. The radio parameters shall be measured according to the methods described in the RF Test Specification.

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

**Europe:**

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

**Japan:**

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



LifeScan, Inc. Ex. 1019
Page 199 of 1420
IPR2025-00103

**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), Industry Canada (Canada)

# 2 FREQUENCY BANDS AND CHANNEL ARRANGEMENT

The Bluetooth system operates in the 2.4 GHz ISM band. This frequency band is 2400 - 2483.5 MHz.

| Regulatory Range | RF Channels |
|---|---|
| 2.400-2.4835 GHz | f=2402+k MHz, k=0,…,78 |

*Table 2.1: Operating frequency bands*

RF channels are spaced 1 MHz and are ordered in channel number k as shown in Table 2.1. In order to comply with out-of-band regulations in each country, a guard band is used at the lower and upper band edge.

| Lower Guard Band | Upper Guard Band |
|---|---|
| 2 MHz | 3.5 MHz |

*Table 2.2: Guard Bands*



LifeScan, Inc. Ex. 1019
Page 201 of 1420
IPR2025-00103

# 3 TRANSMITTER CHARACTERISTICS

The requirements stated in this section are given as power levels at the antenna connector of the Bluetooth 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 301 489-17and FCC part 15 shall be compensated for.

The device is classified into three power classes.

| Power Class | Maximum Output Power (Pmax) | Nominal Output Power | Minimum Output Power[1] | Power Control |
|---|---|---|---|---|
| 1 | 100 mW (20 dBm) | N/A | 1 mW (0 dBm) | Pmin<+4 dBm to Pmax<br>Optional:<br>Pmin[2] to Pmax |
| 2 | 2.5 mW (4 dBm) | 1 mW (0 dBm) | 0.25 mW (-6 dBm) | Optional:<br>Pmin[2)] to Pmax |
| 3 | 1 mW (0 dBm) | N/A | N/A | Optional:<br>Pmin[2)] to Pmax |

*Table 3.1: Power classes*

1. Minimum output power at maximum power setting.
2. The lower power limit Pmin<-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 optimizes the output power in a physical link with LMP commands (see Link Manager Protocol). It is done by measuring RSSI and reporting back if the transmission power shall be increased or decreased if possible.

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

Link Manager Protocol Section 4.1.3 on page 225. 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 Section 4.1.5 on page 43. This can cause the receiving device to fail to respond. It may therefore be useful to page at Class 2 or 3 power in addition to paging at power class 1.

Devices shall not exceed the maximum allowed transmit power levels set by controlling regulatory bodies. The maximum allowed transmit power level may depend upon the modulation mode.

## 3.1 BASIC RATE

### 3.1.1 Modulation Characteristics

The Modulation is GFSK (Gaussian Frequency Shift Keying) with a bandwidth-bit period product BT=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.

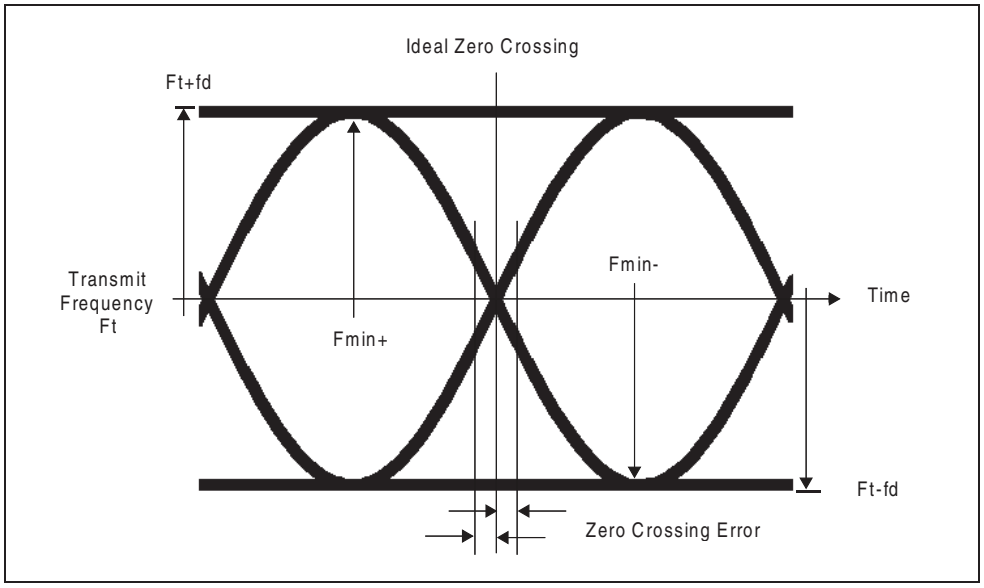



*Figure 3.1: GFSK parameters definition.*

For each transmission, the minimum frequency deviation, Fmin = min{|Fmin+|, Fmin-}, 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.



LifeScan, Inc. Ex. 1019
Page 203 of 1420
IPR2025-00103

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

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

#### *3.1.2.1 In-band spurious emission*

Within the ISM band the transmitter shall pass a spectrum mask, given in Table 3.2. 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.

| Frequency offset | Transmit Power |
|---|---|
| ± 500 kHz | -20 dBc |
| 2MHz (\|M-N\| = 2) | -20 dBm |
| 3MHz or greater (\|M-N\| ≥ 3) | -40 dBm |

*Table 3.2: Transmit Spectrum mask.*

**Note:** If the output power is less than 0dBm then, wherever appropriate, the FCC's 20 dB relative requirement overrules the absolute adjacent channel power requirement stated in the above table.

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

### 3.1.3 Radio Frequency Tolerance

The transmitted initial center frequency shall be within ±75 kHz from $F_c$. 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 3.3. The different packets are defined in the Baseband Specification.

| Duration of Packet | Frequency Drift |
|---|---|
| Max length one slot packet | ±25 kHz |
| Max length three slot packet | ±40 kHz |
| Max length five slot packet | ±40 kHz |
| Maximum drift rate[1] | 400 Hz/µs |

*Table 3.3: Maximum allowable frequency drifts in a packet.*

1. The maximum drift rate is allowed anywhere in a packet.

## 3.2 ENHANCED DATA RATE

A key characteristic of the Enhanced Data Rate mode is that the modulation scheme is changed within the packet. The access code and packet header, as defined in Table 6.1 in the Baseband Specification, are transmitted with the Basic Rate 1 Mbps GFSK modulation scheme, whereas the subsequent synchronization sequence, payload, and trailer sequence are transmitted using the Enhanced Data Rate PSK modulation scheme.

### 3.2.1 Modulation Characteristics

During access code and packet header transmission the Basic Rate GFSK modulation scheme shall be used. During the transmission of the synchronization sequence, payload, and trailer sequence a PSK type of modulation with a data rate of 2 Mbps or optionally 3 Mbps shall be used. The following subsections specify the PSK modulation for this transmission.

#### *3.2.1.1 Modulation Method Overview*

The PSK modulation format defined for the 2 Mbps transmission shall be π/4 rotated differential encoded quaternary phase shift keying (π/4-DQPSK).

The PSK modulation format defined for the 3 Mbps transmission shall be differential encoded 8-ary phase shift keying (8DPSK).



LifeScan, Inc. Ex. 1019
Page 205 of 1420
IPR2025-00103

The modulation shall employ square-root raised cosine pulse shaping to generate the equivalent lowpass information-bearing signal v(t). The output of the transmitter shall be a bandpass signal that can be represented as

$$S(t) = Re\left[v(t)e^{j2\pi F_c t}\right] \qquad \textit{(EQ 1)}$$

with $F_c$ denoting the carrier frequency.

### *3.2.1.2 Differential Phase Encoding*

For the M-ary modulation, the binary data stream $\{b_n\}$, n=1,2,3, …N, shall be mapped onto a corresponding sequence {Sk}, k=1,2, …N/$\log_2$(M) of complex valued signal points. M=4 applies for 2 Mbps and M=8 applies for 3 Mbps. Gray coding shall be applied as shown in Table 3.4 and Table 3.5. In the event that the length of the binary data stream N is not an integer multiple of log2(M), the last symbol of the sequence $\{S_k\}$ shall be formed by appending data zeros to the appropriate length. The signal points $S_k$ shall be defined by:

$$S_k = S_{k-1}e^{j\varphi_k} \qquad k = 1, 2, ..N/\log_2(M) \qquad \text{(EQ 2)}$$

$$S_0 = e^{j\phi} \qquad \text{with } \phi \in [0, 2\pi) \qquad \text{(EQ 3)}$$

The relationship between the binary input $b_k$ and the phase $\phi_k$ shall be as defined in Table 3.4 for the 2 Mbps transmission and in Table 3.5 for the 3 Mbps transmission.

| $b_{2k-1}$ | $b_{2k}$ | $\varphi_k$ |
|---|---|---|
| 0 | 0 | $\pi/4$ |
| 0 | 1 | $3\pi/4$ |
| 1 | 1 | $-3\pi/4$ |
| 1 | 0 | $-\pi/4$ |

Table 3.4: $\pi/4$-DQPSK mapping.



LifeScan, Inc. Ex. 1019
Page 206 of 1420
IPR2025-00103

| $b_{3k-2}$ | $b_{3k-1}$ | $b_{3k}$ | $\varphi_k$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | π/4 |
| 0 | 1 | 1 | π/2 |
| 0 | 1 | 0 | 3π/4 |
| 1 | 1 | 0 | π |
| 1 | 1 | 1 | -3π/4 |
| 1 | 0 | 1 | -π/2 |
| 1 | 0 | 0 | -π/4 |

*Table 3.5: 8DPSK mapping.*

### *3.2.1.3 Pulse Shaping*

The lowpass equivalent information-bearing signal v(t) shall be generated according to

$$v(t) = \sum_k S_k p(t - kT) \qquad \text{(EQ 4)}$$

in which the symbol period T shall be 1μs.

The frequency spectrum P(f), which corresponds to the square-root raised cosine pulse p(t) of the pulse shaping filter is:

$$|P(f)| = \begin{cases} 1 & 0 \le |f| \le \frac{1-\beta}{2T} \\ \sqrt{\frac{1}{2}\left(1 - \sin\left(\frac{\pi(2|f|T - 1)}{2\beta}\right)\right)} & \frac{1-\beta}{2T} \le |f| \le \frac{1+\beta}{2T} \\ 0 & \text{elsewhere} \end{cases} \qquad \text{(EQ 5)}$$

The roll off factor β shall be 0.4.

### *3.2.1.4 Modulation Accuracy*

The measurement of modulation accuracy utilizes differential error vector magnitude (DEVM) with tracking of the carrier frequency drift. The definition of DEVM and the derivation of the RMS and peak measures of DEVM are given in Appendix C.



LifeScan, Inc. Ex. 1019
Page 207 of 1420
IPR2025-00103

The DEVM shall be measured over the synchronization sequence and payload portions of the packet, but not the trailer symbols. For each modulation method and each measurement carrier frequency, DEVM measurement is made over a total of 200 non-overlapping blocks, where each block contains 50 symbols.

The transmitted packets shall be the longest supported packet type for each modulation method, as defined in Table 6.9 and Table 6.10 in the Baseband part.

The DEVM is measured using a square-root raised cosine filter, with a roll-off of 0.4 and a 3 dB bandwidth of ±500 kHz.

3.2.1.4.1 RMS DEVM

The RMS DEVM for any of the measured blocks shall not exceed 0.20 for π/4-DQPSK and 0.13 for 8DPSK.

3.2.1.4.2 99% DEVM

The 99% DEVM (defined as the DEVM value for which 99% of measured symbols have a lower DEVM) shall not exceed 0.30 for π/4-DQPSK and 0.20 for 8DPSK.

3.2.1.4.3 Peak DEVM

The Peak DEVM shall not exceed 0.35 for π/4-DQPSK and 0.25 for 8DPSK.

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

#### *3.2.2.1 In-band Spurious Emission*

Within the ISM band the power spectral density of the transmitter shall comply with the following requirements when sending pseudo random data. All power measurements shall use a 100 kHz bandwidth with maximum hold. The power measurements between 1 MHz and 1.5 MHz from the carrier shall be at least 26 dB below the maximum power measurement up to 500 kHz from the carrier. The adjacent channel power for channels at least 2 MHz from the carrier is defined as the sum of the power measurements over a 1 MHz channel and shall not exceed -20 dBm for the second adjacent channels and -40 dBm for the third and subsequent adjacent channels. These requirements shall apply to

the transmitted signal from the start of the guard time to the end of the power down ramp. The spectral mask is illustrated in Figure 3.2.

Exceptions are allowed in up to 3 bands of 1 MHz width centered on a frequency which is an integer multiple of 1 MHz. They shall comply with an absolute value of –20 dBm.

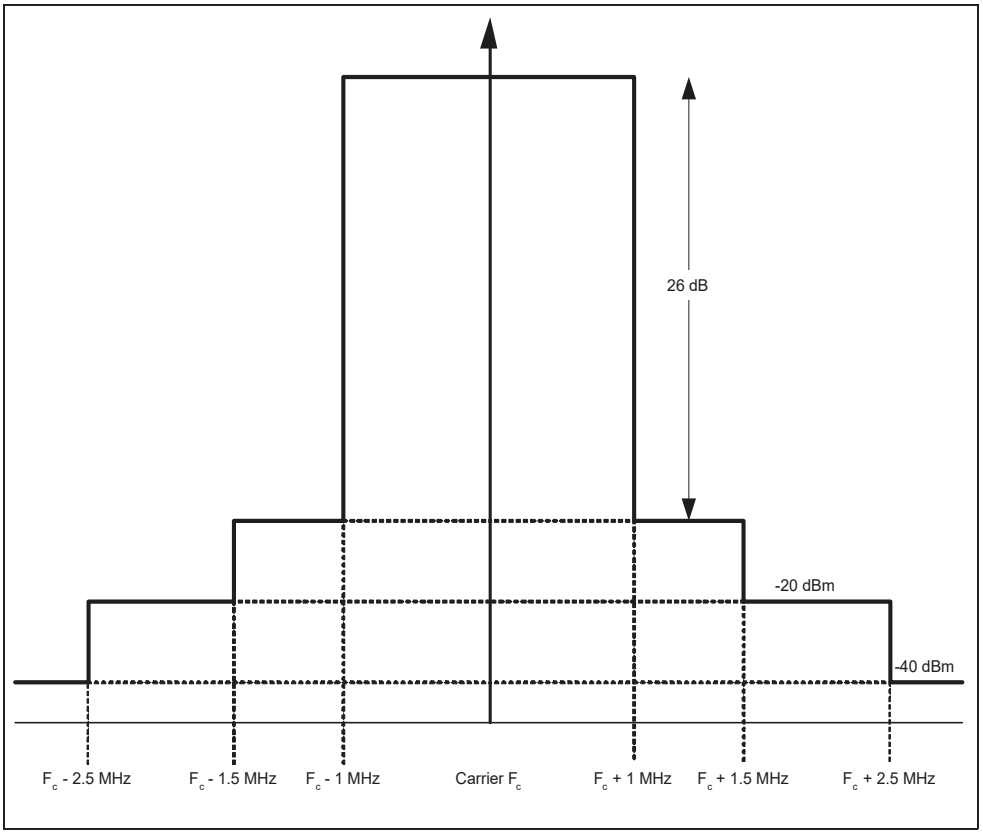



*Figure 3.2: Transmitter spectrum mask*

### 3.2.3 Radio Frequency Tolerance

The same carrier frequencies $F_c$ as used for Basic Rate transmissions shall be used for the Enhanced Data Rate transmissions. The transmitted initial center frequency accuracy shall be within ±75 kHz from $F_c$. The maximum excursion from $F_c$ (frequency offset plus drift) shall not exceed ±75 kHz.

The initial frequency accuracy is defined as being the frequency accuracy before any information is transmitted.

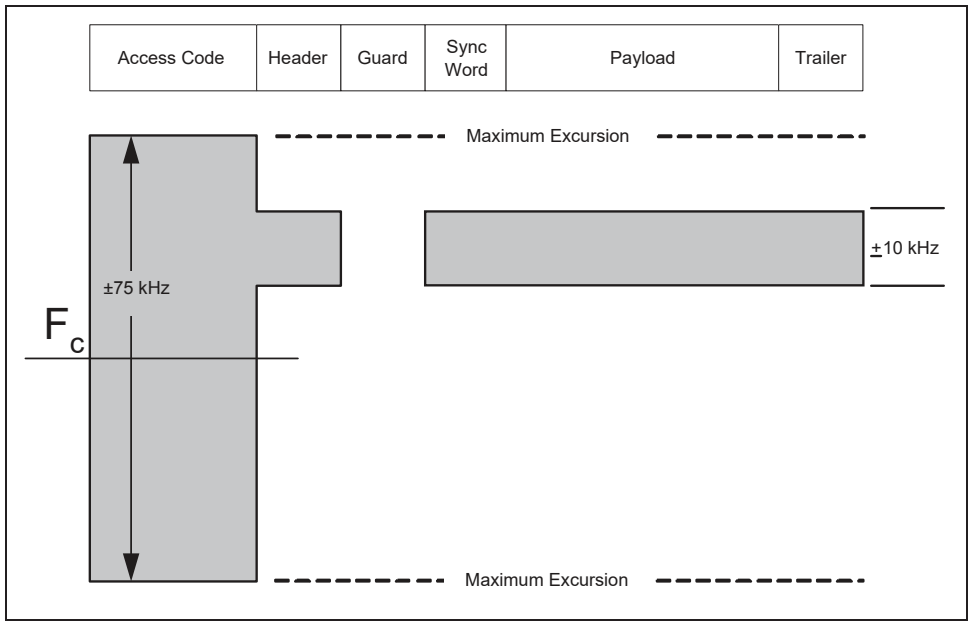



*Figure 3.3: Carrier frequency mask*

The requirements on accuracy and stability are illustrated by Figure 3.3 for the Enhanced Data Rate packet format defined in Baseband Specification. The higher frequency accuracy requirement shall start at the first symbol of the header. The maximum drift over the header, synchronization sequence and payload shall be ±10 kHz.

### 3.2.4 Relative Transmit Power

The requirement on the relative power of the GFSK and PSK portions of the Enhanced Data Rate packet is defined as follows. The average power level during the transmission of access code and header is denoted as $P_{DFSK}$ and the average power level during the transmission of the synchronization sequence and the payload is denoted as $P_{DFSK}$.The following inequalities shall be satisfied independently for every Enhanced Data Rate packet transmitted:

$(P_{GFSK} - 4\ dB) < P_{DPSK} < (P_{GFSK} + 1\ dB)$

# 4 RECEIVER CHARACTERISTICS

The receiver characteristics shall be measured using loopback as defined in Section 1, “Test Methodology,” on page 249.

The reference sensitivity level referred to in this chapter is -70 dBm.

## 4.1 BASIC RATE

### 4.1.1 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 Bluetooth transmitter compliant to the transmitter specification specified in Section 3 on page 32.

### 4.1.2 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 band 2400-2483.5 MHz, the out-of-band blocking specification (see Section 4.1.3 on page 42) shall apply. The interfering signal shall be Bluetooth-modulated (see Section 4.1.7 on page 43). The BER shall be ≤0.1% for all the signal-to-interference ratios listed in Table 4.1:

| 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[1] [2], $C/I_{Image}$ | -9 dB |
| Adjacent (1 MHz) interference to in-band image frequency, $C/I_{Image\pm 1MHz}$ | -20 dB |

*Table 4.1: Interference performance*

1. In-band image frequency
2. 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 4.1 are applicable to the same channel, the more relaxed specification applies.

These specifications are only to be tested at nominal temperature conditions with a device receiving on one RF channel and transmitting on a second RF channel;



LifeScan, Inc. Ex. 1019
Page 211 of 1420
IPR2025-00103

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.

### 4.1.3 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 fulfil the following requirements:

| Interfering Signal Frequency | Interfering Signal Power Level |
|---|---|
| 30 MHz - 2000 MHz | -10 dBm |
| 2000 - 2399 MHz | -27 dBm |
| 2484 – 3000 MHz | -27 dBm |
| 3000 MHz – 12.75 GHz | -10 dBm |

*Table 4.2: Out-of-band suppression (or rejection) requirements.*

24 exceptions are permitted which are dependent upon the given RF channel and are centered at a frequency which is an integer multiple of 1 MHz. For at least 19 of these spurious response frequencies, a reduced interference level of at least -50dBm is allowed in order to achieve the required BER=0.1% performance, whereas for a maximum of 5 of the spurious frequencies the interference level may be assumed arbitrarily lower.

### 4.1.4 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 a frequency $f_1$ with a power level of –39 dBm.
- A Bluetooth modulated signal (see Section 4.1.7 on page 43) shall be at $f_2$ with a power level of -39 dBm.

Frequencies f0, f1 and f2 shall be chosen such 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).

### 4.1.5 Maximum Usable Level

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

### 4.1.6 Receiver Signal Strength Indicator

If a device supports Receive Signal Strength Indicator (RSSI) the accuracy shall be +/- 6 dBm.

### 4.1.7 Reference Signal Definition

A Bluetooth modulated interfering signal shall be defined as:

Modulation = GFSK

Modulation index = 0.32±1%

BT= 0.5±1%

Bit Rate = 1 Mbps ±1 ppm

Modulating Data for wanted signal = PRBS9

Modulating Data for interfering signal = PRBS 15

Frequency accuracy better than ±1 ppm.

## 4.2 ENHANCED DATA RATE

### 4.2.1 Actual Sensitivity Level

The actual sensitivity level shall be defined as the input level for which a raw bit error rate (BER) of 0.01% is met. The requirement for a Bluetooth $\pi$/4-DQPSK and 8DPSK Enhanced Data Rate receiver shall be an actual sensitivity level of –70 dBm or better. The receiver shall achieve the –70 dBm sensitivity level with any Bluetooth transmitter compliant to the Enhanced Data Rate transmitter specification as defined in Section 3.2.

### 4.2.2 BER Floor Performance

The receiver shall achieve a BER less than 0.001% at 10 dB above the reference sensitivity level.

### 4.2.3 Interference Performance

The interference performance for co-channel and adjacent 1 MHz and 2 MHz channels shall be measured with the wanted signal 10 dB above the reference sensitivity level. On all other frequencies the wanted signal shall be 3 dB above the reference sensitivity level. The requirements in this section shall only apply if the frequency of the interferer is inside of the band 2400-2483.5 MHz.



LifeScan, Inc. Ex. 1019
Page 213 of 1420
IPR2025-00103

The interfering signal for co-channel interference shall be similarly modulated as the desired signal. The interfering signal for other channels shall be equivalent to a nominal Bluetooth Basic Rate GFSK transmitter. The interfering signal shall carry random data.

A BER of 0.1% or better shall be achieved for the signal to interference ratios defined in Table 4.3.

| Frequency of Interference | π/4-DQPSK Ratio | 8DPSK Ratio |
|---|---|---|
| Co-Channel interference, $C/I_{co\text{-}channel}$ | 13 dB | 21 dB |
| Adjacent (1 MHz) interference[1], $C/I_{1MHz}$ | 0 dB | 5 dB |
| Adjacent (2MHz) interference[1], $C/I_{2MHz}$ | -30 dB | -25 dB |
| Adjacent (≥3MHz) interference[1] | -40 dB | -33 dB |
| Image frequency interference[1,2,3], $C/I_{Image}$ | -7 dB | 0 dB |
| Adjacent (1 MHz) interference to in-band image frequency[1,2,3], $C/I_{Image \pm 1MHz}$ | -20 dB | -13 dB |

*Table 4.3: Interference Performance*

1. If two adjacent channel specifications from Table 4.3 are applicable to the same channel, the more relaxed specification applies.
2. In-band image frequency.
3. If the image frequency is not equal to n*1 MHz, then the image reference frequency is defined as the closest n*1 MHz frequency.

These specifications are only to be tested at nominal temperature conditions with a receiver hopping on one frequency; this means that the synthesizer may change frequency between receive slot and transmit slot, but always returns to the same receive frequency.

Frequencies where the requirements are not met are called spurious response frequencies. Five spurious response frequencies are allowed at frequencies with a distance of ≥2 MHz from the wanted signal. On these spurious response frequencies a relaxed interference requirement C/I = -15 dB for π/4-DQPSK and C/I = -10 dB for 8DPSK shall be met.

### 4.2.4 Maximum Usable Level

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

### 4.2.5 Out-of-Band and Intermodulation Characteristics

Note: The Basic Rate out-of-band blocking and intermodulation requirements ensure adequate Enhanced Data Rate performance, and therefore there are no specific requirements for Enhanced Data Rate.

### 4.2.6 Reference Signal Definition

A 2 Mbps Bluetooth signal used as "wanted" or "interfering signal" is defined as:

Modulation = $\pi$/4-DQPSK
Symbol Rate = 1 Msym/s ± 1 ppm
Frequency accuracy better than ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
RMS Differential Error Vector Magnitude < 5%
Average power over the GFSK and DPSK portions of the packet shall be equal to within +/- 1 dB

A 3 Mbps Bluetooth signal used as "wanted" or "interfering signal" is defined as:

Modulation = 8DPSK
Symbol Rate = 1 Msym/s ± 1 ppm
Frequency accuracy better than ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
RMS Differential Error Vector Magnitude < 5%
Average power over the GFSK and DPSK portions of the packet shall be equal to within +/- 1 dB



LifeScan, Inc. Ex. 1019
Page 215 of 1420
IPR2025-00103

# 5 APPENDIX A

## 5.1 NOMINAL TEST CONDITIONS

### 5.1.1 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.

### 5.1.2 Nominal power source

#### *5.1.2.1 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.

#### *5.1.2.2 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 (6V, 12V, etc.).

#### *5.1.2.3 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.

## 5.2 EXTREME TEST CONDITIONS

### 5.2.1 Extreme temperatures

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

- 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.

### 5.2.2 Extreme power source voltages

Tests at extreme power source voltages specified below 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.

#### *5.2.2.1 Mains voltage*

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

#### *5.2.2.2 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 (6V, 12V etc.)

#### *5.2.2.3 Power sources using other types of batteries*

The lower extreme test voltage for equipment with power sources using the following types of battery, shall be

a) for Leclanché, alkaline, or lithium type battery: 0.85 times the nominal voltage of the battery

b) for mercury or nickel-cadmium types of battery: 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.

#### *5.2.2.4 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.

# 6 APPENDIX B

The Basic Rate radio parameters shall be tested in the following conditions

| Parameter | Temperature | Power source |
|---|---|---|
| Output Power | ETC | ETC |
| Power control | NTC | 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 |
| Receiver Signal Strength Indicator | NTC | NTC |

ETC = Extreme Test Conditions
NTC = Nominal Test Conditions

The Enhanced Data Rate radio parameters shall be tested in the following conditions

| Parameter | Temperature | Power source |
|---|---|---|
| Modulation accuracy | ETC | ETC |
| Carrier frequency stability | ETC | ETC |
| In-band spurious emissions | ETC | ETC |
| Relative transmit power | ETC | ETC |
| Sensitivity | ETC | ETC |
| BER floor sensitivity | NTC | NTC |
| Interference Performance | NTC | NTC |
| Maximum usable level | NTC | NTC |

ETC = Extreme Test Conditions
NTC = Nominal Test Conditions



LifeScan, Inc. Ex. 1019
Page 218 of 1420
IPR2025-00103

# 7 APPENDIX C

## 7.1 ENHANCED DATA RATE MODULATION ACCURACY

The Enhanced Data Rate modulation accuracy is defined by the differential error vector, being the difference between the vectors representing consecutive symbols of the transmitted signal, after passing the signal through a specified measurement filter, sampling it at the symbol rate with an optimum sampling phase and compensating it for carrier frequency error and for the ideal carrier phase changes. The magnitude of the normalized differential error vector is called the Differential Error Vector Magnitude (DEVM). The objective of the DEVM is to estimate the modulation errors that would be perceived by a differential receiver.

In an ideal transmitter, the input bit sequence $\{b_j\}$ is mapped onto a complex valued symbol sequence $\{S_k\}$. Subsequently, this symbol sequence is transformed into a baseband signal S(t) by means of a pulse-shaping filter.

In an actual transmitter implementation, the bit sequence $\{b_j\}$ generates a baseband equivalent transmitted signal Y(t). This signal Y(t) contains, besides the desired component S(t), multiple distortion components. This is illustrated in Figure 7.1.

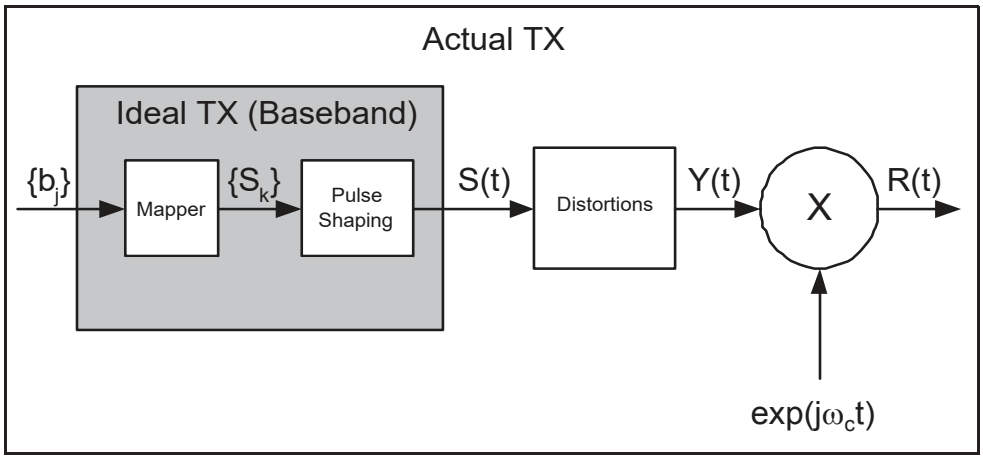



*Figure 7.1: TX model.*

Let Z(t) be the output of the measurement filter after first compensating the received signal for the initial center frequency error, $\omega_i$, of the received packet, i.e. the output after down conversion and filtering the transmit signal R(t) (see Figure 7.2).The measurement filter is defined by a square-root raised cosine shaping filter with a roll-off factor equal to 0.4 and 3 dB bandwidth of ±500 kHz.

Let $\{Z_k(\varepsilon)\}$ be the sequence of samples obtained by sampling the signal Z(t) with a sampling period equal to the symbol period T and a sampling phase equal to $\varepsilon$ such that $Z_k(\varepsilon)=Z((k+\varepsilon)T)$. Note that this sequence $\{Z_k(\varepsilon)\}$ would coincide with the symbol sequence $\{S_k\}$ if no distortion is present and the correct timing phase $\varepsilon$ is chosen.

To reflect the behavior of a typical differential receiver, the sample sequence $\{Z_k(\varepsilon)\}$ should be compensated for carrier frequency drift. Therefore, the sequence $\{Z_k(\varepsilon)\}$ is multiplied by a factor $W^{-k}$ in which $W = e^{j\omega T}$ accounts for the frequency offset $\omega$. A constant value of $\omega$ is used for each DEVM block of N = 50 symbols, but $\omega$ may vary between DEVM blocks (note that the values of $\omega$ can be used to measure carrier frequency drift.

In addition, $\{Z_k(\varepsilon)\}$ is compensated for the ideal phase changes between symbols by multiplying it with the complex conjugate of the symbol sequence $\{S_k\}$. However, it is not necessary to compensate $\{Z_k(\varepsilon)\}$ for initial carrier phase or output power of the transmitter.

Let $\{Q_k(\varepsilon,\omega)\}$ denote the compensated sequence $\{Z_k(\varepsilon)\}$, where the ideal phase changes have been removed and $\varepsilon$ and $\omega$ are chosen optimally to minimize the DEVM, (i.e. remove time and frequency uncertainty). For a transmitter with no distortions other than a constant frequency error, $\{Q_k(\varepsilon,\omega)\}$ is a complex constant that depends on the initial carrier phase and the output power of the transmitter.

The differential error sequence $\{E_k(\varepsilon,\omega)\}$ is defined as the difference between $\{Q_k(\varepsilon,\omega)\}$ and $\{Q_{k-1}(\varepsilon,\omega)\}$. This reflects the modulation errors that would be perceived by a differential receiver. For a transmitter with no distortions other than a constant frequency error, $\{E_k(\varepsilon,\omega)\}$ is zero.

The definitions of the DEVM measures are based upon this differential error sequence $\{E_k(\varepsilon,\omega)\}$. The generation of the error sequence is depicted in Figure 7.2.

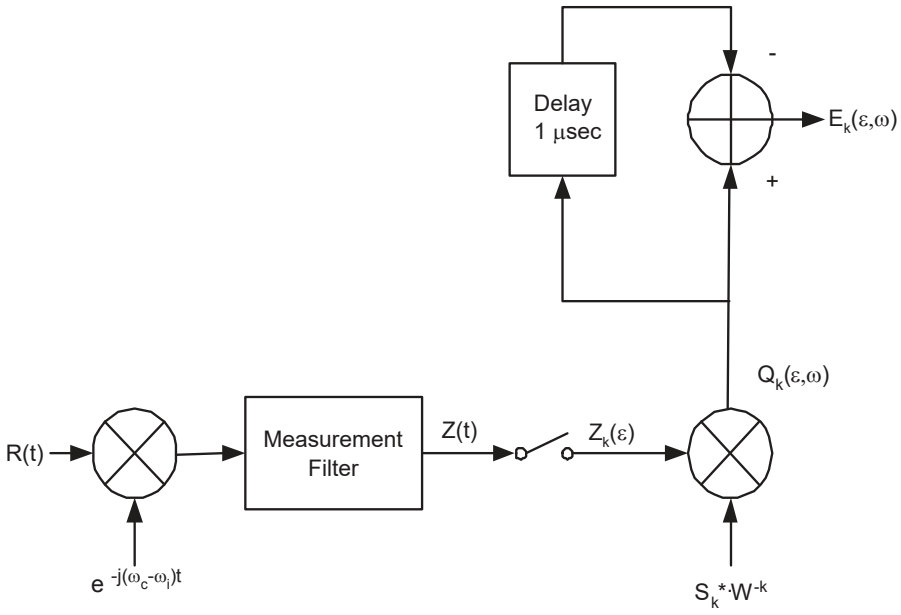



*Figure 7.2: Error sequence generation.*

### RMS DEVM

The root mean squared DEVM (RMS DEVM) computed over N = 50 symbols is defined as:

$$RMS\ DEVM = \min_{\varepsilon,\omega} \left\{ \sqrt{\sum_{k=1}^{N} \left|E_k(\varepsilon,\omega)\right|^2 \Big/ \sum_{k=1}^{N} \left|Q_k(\varepsilon,\omega)\right|^2} \right\}$$

As can be seen from the expression above, the RMS DEVM is the square-root of the normalized power of the error.



LifeScan, Inc. Ex. 1019
Page 221 of 1420
IPR2025-00103

### Peak DEVM

The DEVM at symbol k is defined as

$$DEVM\,(k) = \sqrt{\frac{\left|E_k\,(\varepsilon_0, \omega_0)\right|^2}{\sum_{j=1}^{N}\left|Q_j\,(\varepsilon_0, \omega_0)\right|^2 / N}} \qquad \text{(EQ 6)}$$

where $\varepsilon_0$ and $\omega_0$ are the values for $\varepsilon$ and $\omega$ used to calculate the RMS DEVM.

The peak DEVM is defined as:

$$Peak\ DEVM = \max_{k}\ \{\mathrm{DEVM}\ (k)\} \qquad \text{(EQ 7)}$$

# BASEBAND SPECIFICATION

***This document describes the specification of the Bluetooth link controller which carries out the baseband protocols and other low-level link routines.***



LifeScan, Inc. Ex. 1019
Page 223 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 224 of 1420
IPR2025-00103

# CONTENTS

**1 General Description ..... 59**
1.1 Bluetooth Clock ..... 60
1.2 Bluetooth Device Addressing ..... 62
1.2.1 Reserved addresses ..... 62
1.3 Access Codes ..... 63
**2 Physical Channels ..... 64**
2.1 Physical Channel Definition ..... 65
2.2 Basic Piconet Physical Channel ..... 65
2.2.1 Master-slave definition ..... 65
2.2.2 Hopping characteristics ..... 66
2.2.3 Time slots ..... 66
2.2.4 Piconet clocks ..... 67
2.2.5 Transmit/receive timing ..... 67
2.3 Adapted Piconet Physical Channel ..... 70
2.3.1 Hopping characteristics ..... 70
2.4 Page Scan Physical Channel ..... 71
2.4.1 Clock estimate for paging ..... 71
2.4.2 Hopping characteristics ..... 71
2.4.3 Paging procedure timing ..... 72
2.4.4 Page response timing ..... 73
2.5 Inquiry Scan Physical Channel ..... 75
2.5.1 Clock for inquiry ..... 75
2.5.2 Hopping characteristics ..... 75
2.5.3 Inquiry procedure timing ..... 75
2.5.4 Inquiry response timing ..... 75
2.6 Hop Selection ..... 77
2.6.1 General selection scheme ..... 77
2.6.2 Selection kernel ..... 81
2.6.3 Adapted hop selection kernel ..... 84
2.6.4 Control word ..... 85
**3 Physical Links ..... 90**
3.1 Link Supervision ..... 90
**4 Logical Transports ..... 91**
4.1 General ..... 91
4.2 Logical Transport Address (LT_ADDR) ..... 91
4.3 Synchronous Logical Transports ..... 92
4.4 Asynchronous Logical Transport ..... 92



LifeScan, Inc. Ex. 1019
Page 225 of 1420
IPR2025-00103

4.5 Transmit/Receive Routines ..... 93
4.5.1 TX Routine ..... 93
4.5.2 RX routine ..... 96
4.5.3 Flow control ..... 97
4.6 Active Slave Broadcast Transport ..... 98
4.7 Parked Slave Broadcast Transport ..... 99
4.7.1 Parked member address (PM_ADDR) ..... 99
4.7.2 Access request address (AR_ADDR) ..... 99

**5 Logical Links ..... 100**
5.1 Link Control Logical Link (LC) ..... 100
5.2 ACL Control Logical Link (ACL-C) ..... 100
5.3 User Asynchronous/Isochronous Logical Link (ACL-U) ..... 100
5.3.1 Pausing the ACL-U logical link ..... 101
5.4 User Synchronous Data Logical Link (SCO-S) ..... 101
5.5 User Extended Synchronous Data Logical Link (eSCO-S) ..... 101
5.6 Logical Link Priorities ..... 101

**6 Packets ..... 102**
6.1 General Format ..... 102
6.1.1 Basic Rate ..... 102
6.1.2 Enhanced Data Rate ..... 102
6.2 Bit Ordering ..... 103
6.3 Access Code ..... 104
6.3.1 Access code types ..... 104
6.3.2 Preamble ..... 105
6.3.3 Sync word ..... 105
6.3.4 Trailer ..... 108
6.4 Packet Header ..... 109
6.4.1 LT_ADDR ..... 109
6.4.2 TYPE ..... 109
6.4.3 FLOW ..... 110
6.4.4 ARQN ..... 110
6.4.5 SEQN ..... 110
6.4.6 HEC ..... 110
6.5 Packet Types ..... 111
6.5.1 Common packet types ..... 112
6.5.2 SCO packets ..... 116
6.5.3 eSCO packets ..... 117
6.5.4 ACL packets ..... 119
6.6 Payload Format ..... 121



LifeScan, Inc. Ex. 1019
Page 226 of 1420
IPR2025-00103

6.6.1 Synchronous Data Field ....121
6.6.2 Asynchronous data field ....123
6.7 Packet Summary ....127

**7 Bitstream Processing ....129**
7.1 Error Checking ....130
7.1.1 HEC generation ....130
7.1.2 CRC generation ....131
7.2 Data Whitening ....133
7.3 Error Correction ....134
7.4 FEC Code: Rate 1/3 ....134
7.5 FEC Code: Rate 2/3 ....135
7.6 ARQ Scheme ....136
7.6.1 Unnumbered ARQ ....136
7.6.2 Retransmit filtering ....139
7.6.3 Flushing Payloads ....142
7.6.4 Multi-slave considerations ....142
7.6.5 Broadcast packets ....143
7.7 Erroneous Synchronous Data Reporting ....144

**8 Link Controller Operation ....145**
8.1 Overview of States ....145
8.2 Standby State ....146
8.3 Connection Establishment Substates ....146
8.3.1 Page scan substate ....146
8.3.2 Page substate ....148
8.3.3 Page response substates ....151
8.4 Device Discovery Substates ....155
8.4.1 Inquiry scan substate ....156
8.4.2 Inquiry substate ....157
8.4.3 Inquiry Response Substate ....158
8.5 Connection State ....160
8.6 Active Mode ....161
8.6.1 Polling in the active mode ....162
8.6.2 SCO ....162
8.6.3 eSCO ....164
8.6.4 Broadcast scheme ....166
8.6.5 Role switch ....167
8.6.6 Scatternet ....169
8.6.7 Hop sequence switching ....170
8.6.8 Channel classification and channel map selection ....173



LifeScan, Inc. Ex. 1019
Page 227 of 1420
IPR2025-00103

8.6.9 Power Management ..... 174
8.7 Sniff Mode ..... 175
8.7.1 Sniff Transition Mode ..... 176
8.7.2 Sniff Subrating ..... 177
8.8 Hold Mode ..... 178
8.9 Park State ..... 179
8.9.1 Beacon train ..... 179
8.9.2 Beacon access window ..... 182
8.9.3 Parked slave synchronization ..... 183
8.9.4 Parking ..... 184
8.9.5 Master-initiated unparking ..... 185
8.9.6 Slave-initiated unparking ..... 185
8.9.7 Broadcast scan window ..... 186
8.9.8 Polling in the park state ..... 186

**9 Audio ..... 187**
9.1 LOG PCM CODEC ..... 187
9.2 CVSD CODEC ..... 187
9.3 Error Handling ..... 190
9.4 General Audio Requirements ..... 190
9.4.1 Signal levels ..... 190
9.4.2 CVSD audio quality ..... 190

**10 List of Figures ..... 191**

**11 List of Tables ..... 194**
**Appendix A: General Audio Recommendations ..... 195**
**Appendix B: Timers ..... 199**
**Appendix C: Recommendations for AFH Operation in Park, Hold,and Sniff ..... 201**



LifeScan, Inc. Ex. 1019
Page 228 of 1420
IPR2025-00103

# 1 GENERAL DESCRIPTION

This part specifies the normal operation of a Bluetooth baseband.

The Bluetooth system provides a point-to-point connection or a point-to-multi-point connection, see (a) and (b) in Figure 1.1 on page 59. In a point-to-point connection the physical channel is shared between two Bluetooth devices. In a point-to-multipoint connection, the physical channel is shared among several Bluetooth devices. Two or more devices sharing the same physical channel form a *piconet*. One Bluetooth 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 a parked 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 1.1 on page 59. Each piconet only has a single master, however, slaves can participate in different piconets on a time-division multiplex 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.

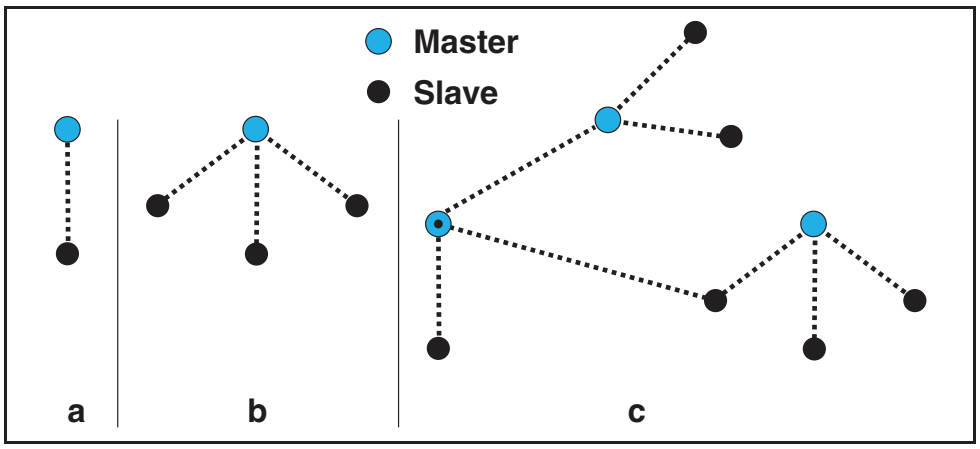



*Figure 1.1: Piconets with a single slave operation (a), a multi-slave operation (b) and a scatternet operation (c).*

Data is transmitted over the air in packets. Two modulation modes are defined: a mandatory mode called Basic Rate and an optional mode called Enhanced Data Rate. The symbol rate for all modulation schemes is 1 Ms/s. The gross air data rate is 1 Mbps for Basic Rate. Enhanced Data Rate has a primary modulation mode that provides a gross air data rate of 2 Mbps, and a secondary modulation mode that provides a gross air data rate of 3 Mbps.

The general Basic Rate packet format is shown in Figure 1.2. Each packet consists of 3 entities: the access code, the header, and the payload.

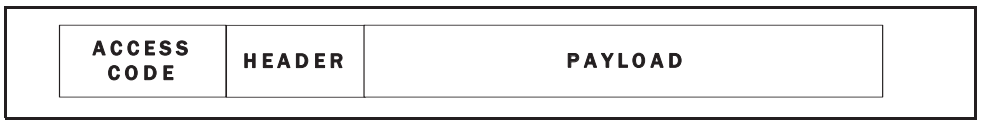



*Figure 1.2: Standard Basic Rate packet format.*

The general Enhanced Data Rate packet format is shown in Figure 1.3. Each packet consists of 6 entities: the access code, the header, the guard period, the synchronization sequence, the Enhanced Data Rate payload and the trailer. The access code and header use the same modulation scheme as for Basic Rate packets while the synchronization sequence, the Enhanced Data Rate payload and the trailer use the Enhanced Data Rate modulation scheme. The guard time allows for the transition between the modulation schemes.

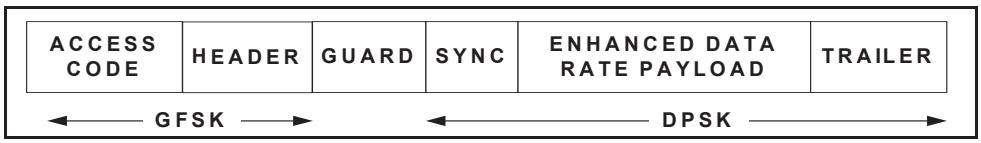



*Figure 1.3: Standard Enhanced Data Rate packet format*

## 1.1 BLUETOOTH CLOCK

Every Bluetooth device shall have a native clock that shall be derived from a free running system clock. For synchronization with other devices, offsets are used that, when added to the native clock, provide temporary Bluetooth clocks that are mutually synchronized. It should be noted that the Bluetooth clock has no relation to the time of day; it may therefore be initialized to any value. The clock has a cycle of about a day. If the 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 Bluetooth 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 1.4 on page 60.

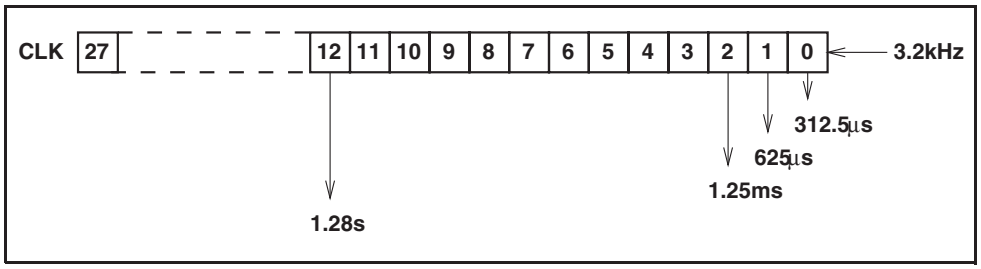



*Figure 1.4: Bluetooth clock.*



LifeScan, Inc. Ex. 1019
Page 230 of 1420
IPR2025-00103

In the different modes and states a device can reside in, 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 in Park, Hold, and Sniff modes the native clock may be driven by a low power oscillator (LPO) with worst case accuracy (+/-250ppm). Otherwise, the native clock shall be driven by the reference crystal oscillator with worst case accuracy of +/-20ppm.

See Section 2.2.4 on page 67 for the definition of CLK and Section 2.4.1 on page 71 for the definition of CLKE.

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



LifeScan, Inc. Ex. 1019
Page 231 of 1420
IPR2025-00103

## 1.2 BLUETOOTH DEVICE ADDRESSING

Each Bluetooth device shall be allocated a unique 48-bit Bluetooth 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: non-significant address part consisting of 16 bits

The LAP and UAP form the significant part of the BD_ADDR. The bit pattern in Figure 1.5 is an example BD_ADDR.

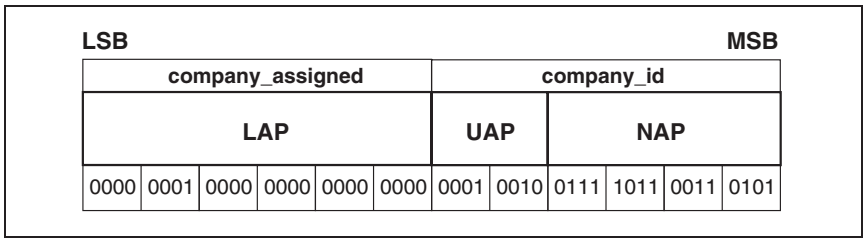



*Figure 1.5: Format of BD_ADDR.*

The BD_ADDR may take any values except the 64 reserved LAP values for general and dedicated inquiries (see Section 1.2.1 on page 62).

### 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, the remaining 63 LAPs are reserved for dedicated inquiry of specific classes of devices (see Assigned Numbers on the web site[1]). 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).

1. https://www.bluetooth.org/assigned_numbers

## 1.3 ACCESS CODES

In the Bluetooth system all transmissions over the physical channel begin with an access code. Three different access codes are defined, see also Section 6.3.1 on page 104:

- 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 device access code is used during **page**, **page scan** and **page response** substates and shall be derived from the paged device's BD_ADDR. The channel access code is used in the **CONNECTION** state and forms the beginning of all packets exchanged on the piconet physical channel. The channel access code shall be derived from the LAP of the master's BD_ADDR. Finally, the inquiry access code 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 signaling.



LifeScan, Inc. Ex. 1019
Page 233 of 1420
IPR2025-00103

# 2 PHYSICAL CHANNELS

The lowest architectural layer in the Bluetooth system is the physical channel. A number of types of physical channels are defined. All Bluetooth 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.

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 frequency 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 Bluetooth devices may be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth 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 channel access code is a property of the physical channel. The access code is always present at the start of every transmitted packet.

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

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

Whenever a Bluetooth 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 only be capable of connection to one physical channel at a time, however, advanced devices may be capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.



LifeScan, Inc. Ex. 1019
Page 234 of 1420
IPR2025-00103

## 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 Bluetooth device address and the selected hopping sequence. The phase in the hopping sequence is determined by the Bluetooth clock. 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 Section 2.6 on page 77).The maximum hop rate is 1600 hops/s in the **CONNECTION** state and the maximum is 3200 hops/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

## 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 Section 2.3 on page 70.

### 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 Section 8.5 on page 160)

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 Section 8.6.5 on page 167.



LifeScan, Inc. Ex. 1019
Page 235 of 1420
IPR2025-00103

### 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 the Bluetooth clock and BD_ADDR of the master. When the piconet is established, the master clock is communicated to the slaves. Each slave shall add an offset to its native clock to synchronize with the master clock. 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 and is described in Section 2.6 on page 77.

### 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 most significant 27 bits of the Bluetooth 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 alternately transmit, see Figure 2.1 on page 66. The packet start shall be aligned with the slot start. Packets may extend over up to five time slots.

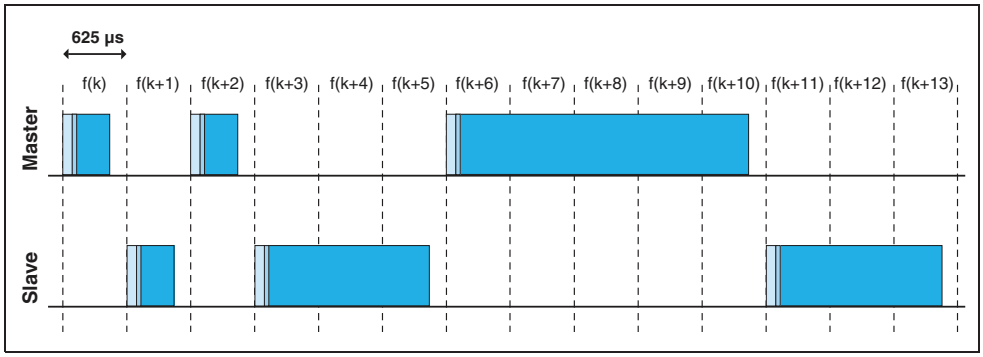



*Figure 2.1: Multi-slot packets*

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



LifeScan, Inc. Ex. 1019
Page 236 of 1420
IPR2025-00103

### 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 native clock CLKN (see Section 1.1 on page 60) by adding an offset, see Figure 2.2 on page 67. The offset shall be zero for the master since CLK is identical to its own native clock CLKN. Each slave shall add an appropriate offset to its CLKN such that the CLK 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 such that CLK is approximately CLKN of the master.

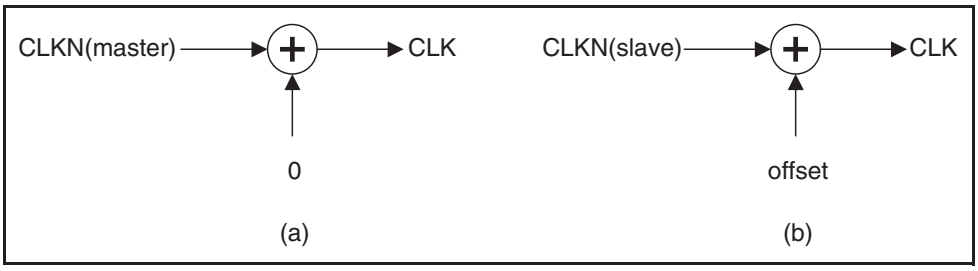



*Figure 2.2: Derivation of CLK in master (a) and in slave (b).*

### 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 2.1 on page 66.

All timing diagrams shown in this chapter 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 the equation:

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

where $T_N$ is the nominal slot length (625 μs), $j_k$ denotes jitter ($|j_k| \leq 1$ μs) at the start of slot $k$, and, $d_k$, 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



LifeScan, Inc. Ex. 1019
Page 237 of 1420
IPR2025-00103

arbitrary but fixed constant. For hold, Park, and sniff the drift and jitter parameters specified in Link Manager Protocol [Part C] Section 4.3.1 on page 268 apply.

### *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 2.3 on page 68. In Figure 2.3 and Figure 2.4 the channel hopping frequencies are indicated by *f(k)* where k is the time slot number. After transmission, a return packet is expected N × 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 250ms.

During the beginning of the RX cycle, the access correlator shall search for the correct channel access code 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 non-recoverable header error is detected, or a non-recoverable payload error is detected.

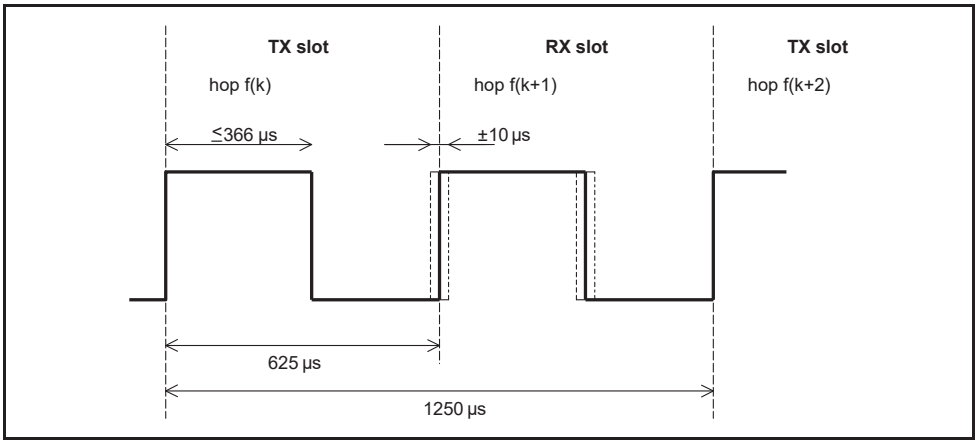



*Figure 2.3: RX/TX cycle of master transceiver in normal mode for single-slot packets.*

Each master transmission shall be derived from bit 2 of the Master's native Bluetooth clock, thus the current transmission will be scheduled Mx1250μs 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



LifeScan, Inc. Ex. 1019
Page 238 of 1420
IPR2025-00103

master TX timing shall be derived from the master's native Bluetooth clock, and thus it will not be affected by time drifts in the slave(s).

Slaves maintain an estimate of the master's native clock by adding a timing offset to the slave's native clock (see Section 2.2.4 on page 67). 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 channel access code is required to synchronize the slave, slave RX timing can be corrected with any packet sent in the master-to-slave transmission slot.

The slave's TX/RX timing is shown in Figure 2.4 on page 69. The slave's transmission shall be scheduled N × 625µs 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 Section 8.6 on page 161) and is not able to receive any valid channel access codes 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.

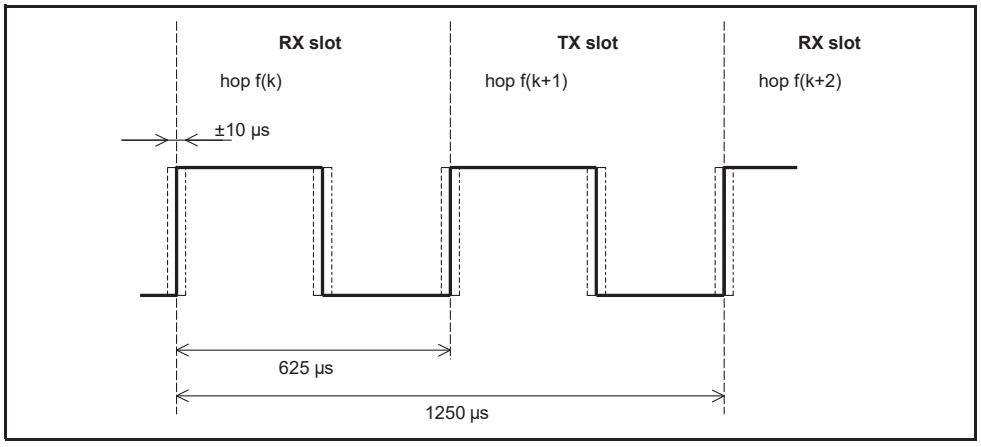



*Figure 2.4: RX/TX cycle of slave transceiver in normal mode for single-slot packets.*

### 2.2.5.2 Piconet physical channel re-synchronization

In the piconet physical channel, a slave may loose synchronization if it does not receive a packet from the master at least every 250ms (or less if the low power clock is used). This may occur in sniff, hold, and Park in a scatternet or due to interference. When re-synchronizing 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 2.5 on page 70. 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.



LifeScan, Inc. Ex. 1019
Page 239 of 1420
IPR2025-00103

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 are transmitted by the master during slave re-synchronization.

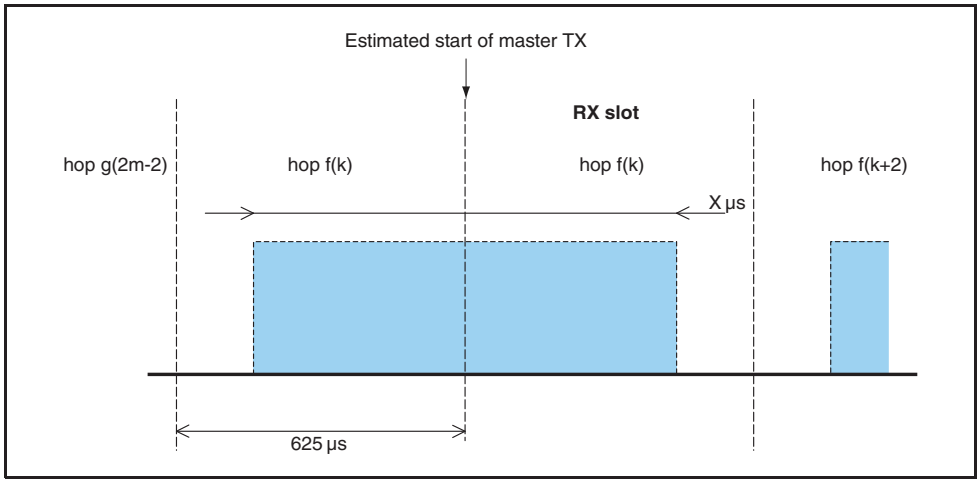



*Figure 2.5: RX timing of slave returning from hold mode.*

## 2.3 ADAPTED PICONET PHYSICAL CHANNEL

### 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 Section 2.6 on page 77.

Adapted piconet physical channels can be used for connected devices that have adaptive frequency hopping (AFH) enabled. There are two distinctions between basic and adapted piconet physical channels. The first is the same channel mechanism that makes the slave frequency the same as the preceding master transmission. The second aspect is that the adapted piconet physical channel may be based on less than the full 79 frequencies of the basic piconet physical channel.



LifeScan, Inc. Ex. 1019
Page 240 of 1420
IPR2025-00103

## 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).

### 2.4.1 Clock estimate for paging

A paging device uses an estimate of the native clock 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 2.6 on page 71. 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.

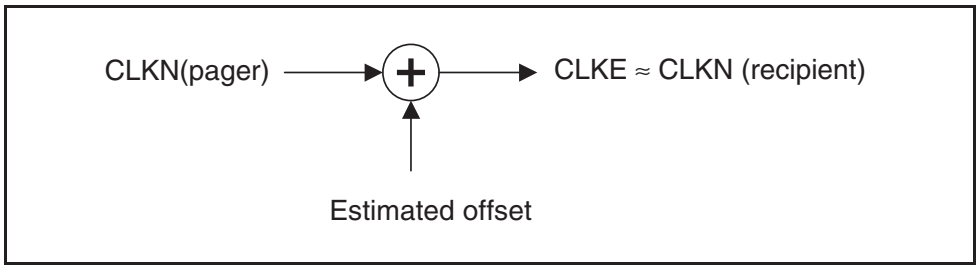



*Figure 2.6: Derivation of CLKE.*

### 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 channel shall be determined by the native Bluetooth clock of the scanning device. The frequency hopping sequence is determined by the Bluetooth 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 Section 2.6 on page 77.



LifeScan, Inc. Ex. 1019
Page 241 of 1420
IPR2025-00103

### 2.4.3 Paging procedure timing

During the paging procedure, the master shall transmit paging messages (see Table 8.3 on page 151) corresponding to the slave to be connected. Since the paging message is a very short packet, the hop rate is 3200 hops/s. In a single TX slot interval, the paging device shall transmit on two different hop frequencies. In Figure 2.7 through Figure 2.11, *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 2.7 on page 72. 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.

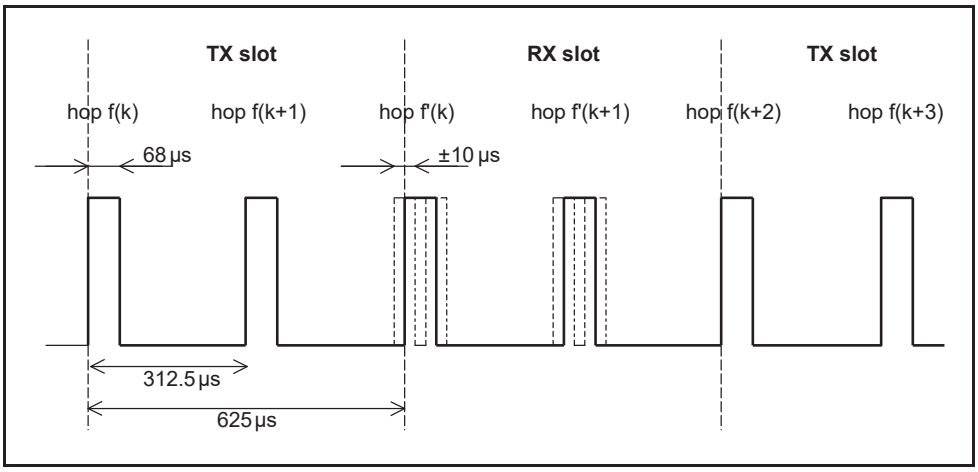



*Figure 2.7: RX/TX cycle of transceiver in PAGE mode.*



LifeScan, Inc. Ex. 1019
Page 242 of 1420
IPR2025-00103

### 2.4.4 Page response timing

At connection setup a master page response packet is transmitted from the master to the slave (see Table 8.3 on page 151). 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 2.8 on page 73, 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)* the page message was received in.

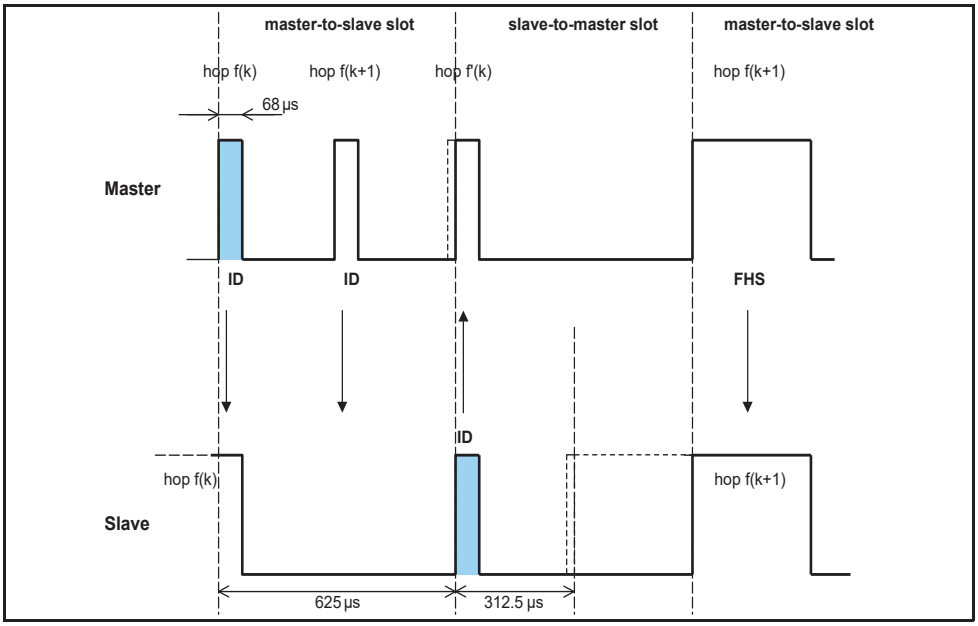



*Figure 2.8: Timing of page response packets on successful page in first half slot*

In Figure 2.9 on page 74, 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)* the page message was received in.

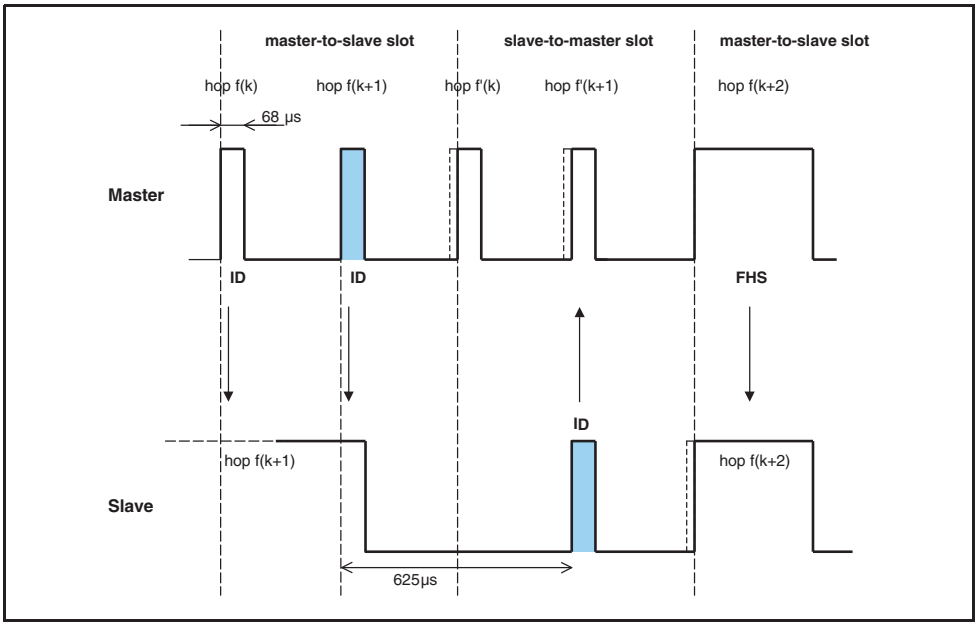



*Figure 2.9: Timing of page response packets on successful page in second half slot*

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). That is, 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.



LifeScan, Inc. Ex. 1019
Page 244 of 1420
IPR2025-00103

## 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.

### 2.5.1 Clock for inquiry

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

### 2.5.2 Hopping characteristics

The inquiry scan 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 channel is determined by the native Bluetooth clock of the scanning device while the frequency hopping sequence is determined by the general inquiry access code.

The inquiry scan physical channel uses the inquiry, inquiry response, and inquiry scan hopping sequences described in Section 2.6 on page 77.

### 2.5.3 Inquiry procedure timing

During the inquiry procedure, the master shall transmit inquiry messages with the general or dedicated inquiry access code. The timing for inquiry is the same as for paging (see Section 2.4.3 on page 72).

### 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 8.5 on page 160). This packet contains information necessary for the inquiring master to page the slave (see definition of the FHS packet in Section 6.5.1.4 on page 113) and follows 625μs after the receipt of the inquiry message. If the slave transmits an extended inquiry response packet, it shall be transmitted1250μs after the start of the inquiry response packet.

In Figure 2.10 and Figure 2.11, *f(k)* is used for the frequencies of the inquiry hopping sequence and f'(k) denotes the corresponding inquiry response sequence frequency. The inquiry response packet and the extended inquiry response packet are 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.



LifeScan, Inc. Ex. 1019
Page 245 of 1420
IPR2025-00103

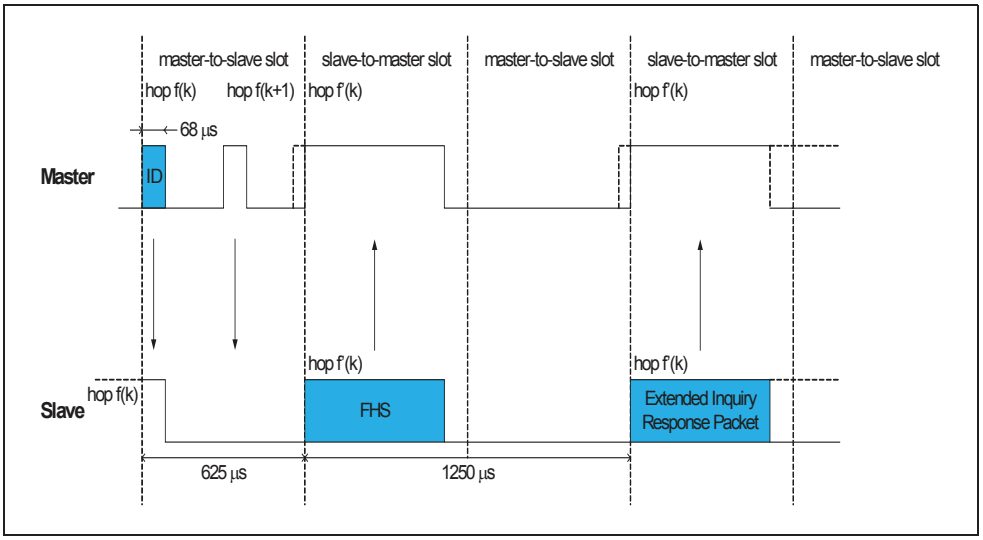



*Figure 2.10: Timing of inquiry response packets on successful inquiry in first half slot*

When the inquiry message received by the slave was the second in the master-to-slave slot the inquiry response packet and the extended inquiry response packet are received by the master at the hop frequency *f'(k+1)*.

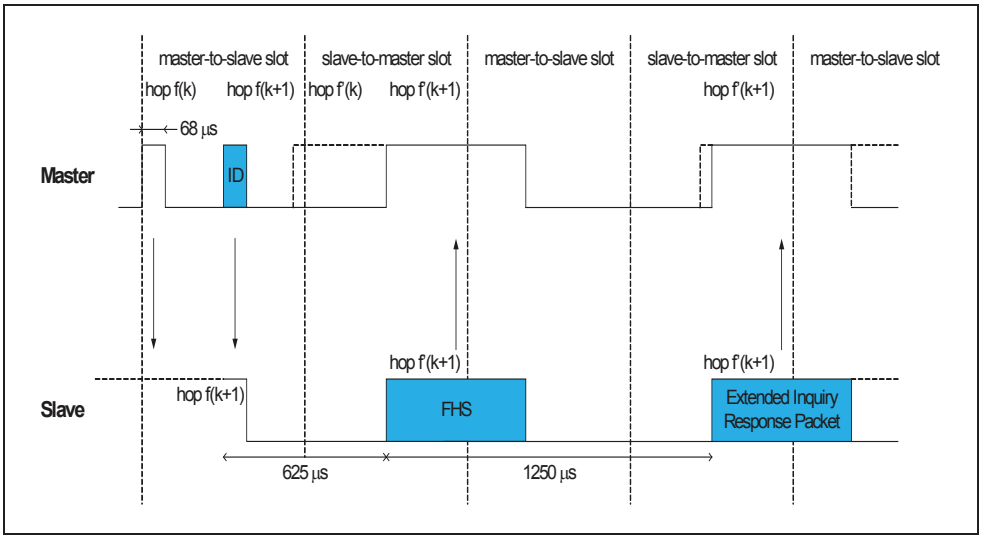



*Figure 2.11: Timing of inquiry response packets on successful inquiry in second half slot*



LifeScan, Inc. Ex. 1019
Page 246 of 1420
IPR2025-00103

## 2.6 HOP SELECTION

Bluetooth devices shall use the hopping kernel as defined in the following sections.

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 adaptive frequency hopping (AFH). These sequences are:

- 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 only used in place of the basic channel hopping sequence. All other hopping sequences are not affected by hop sequence adaptation.

### 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 2.12 on page 78. The mapping from the input to a particular RF channel index is performed in the selection box.

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 2.2 on page 86). N and $k_{offset}$ are defined in Section 2.6.4 on page 85.



LifeScan, Inc. Ex. 1019
Page 247 of 1420
IPR2025-00103

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 Bluetooth 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 HCI Create Connection Command [Part E] Section 7.1.5 on page 423). 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 Section 1.2.1 on page 62) is used for generating a frequency hop sequence, the UAP shall be replaced by the default check initialization (DCI, see Section 7.1 on page 130). 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 Section 2.6.3 on page 84.

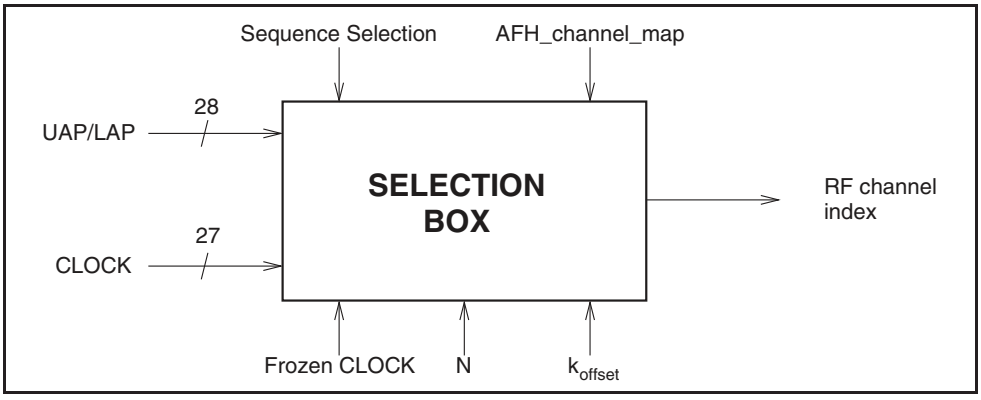



*Figure 2.12: General block diagram of hop selection scheme.*



LifeScan, Inc. Ex. 1019
Page 248 of 1420
IPR2025-00103

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 2.1 on page 31 in the Radio Specification.

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 2.13 on page 79.

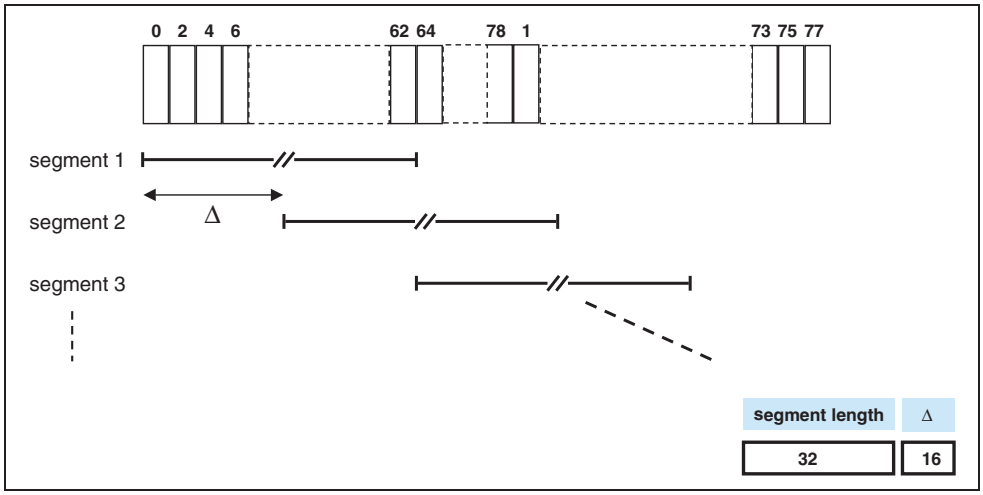



*Figure 2.13: Hop selection scheme in CONNECTION state.*

The RF frequency shall remain fixed for the duration of the packet. The RF frequency for the packet shall be derived from the Bluetooth clock value in the first slot of the packet. The RF frequency in the first slot after a multi-slot packet shall use the frequency as determined by the Bluetooth clock value for that slot. Figure 2.14 on page 80 illustrates the hop definition on single- and multi-slot packets.

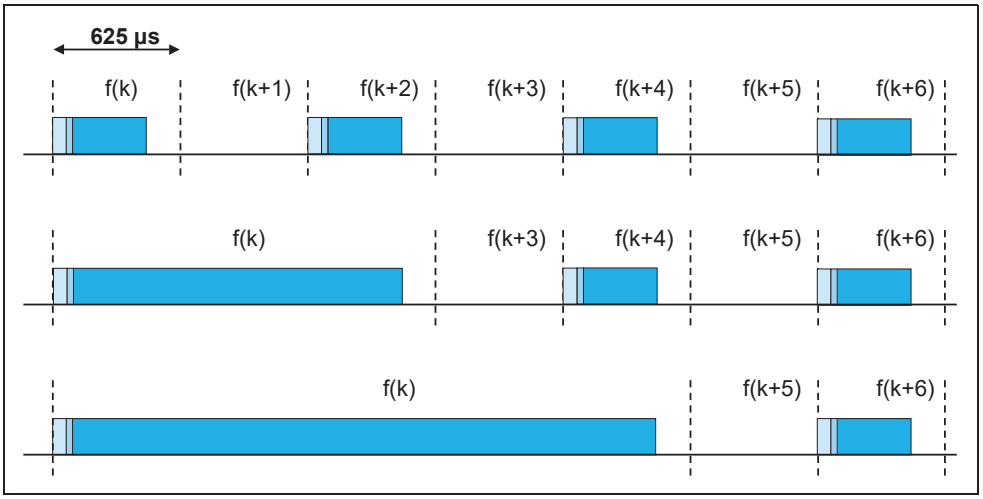



*Figure 2.14: Single- and multi-slot packets.*

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 *AFH_channel_map* input. The adapted sequence has similar statistical properties to the non-adapted hop sequence. 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 2.15 on page 80. The same channel mechanism shall be used whenever the adapted channel hopping sequence is selected.

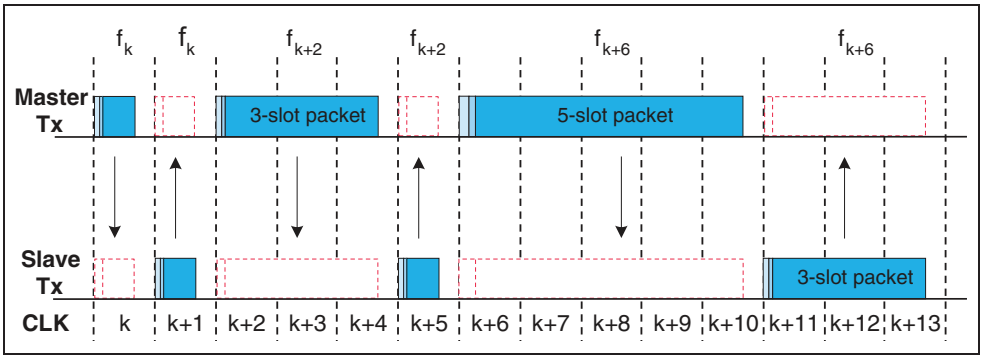



*Figure 2.15: Example of the same channel mechanism.*



LifeScan, Inc. Ex. 1019
Page 250 of 1420
IPR2025-00103

### 2.6.2 Selection kernel

The basic hop selection kernel shall be as shown in Figure 2.16 on page 81 and is used for the page, page response, inquiry, inquiry response and basic channel hopping selection kernels. In these substates the AFH_channel_map input is unused. The adapted channel hopping selection kernel is described in Section 2.6.3 on page 84. The X input determines the phase in the 32-hop segment, whereas Y1 and Y2 selects between master-to-slave and slave-to-master. The inputs A to D determine the ordering within the segment, the inputs E and F determine the mapping onto 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.

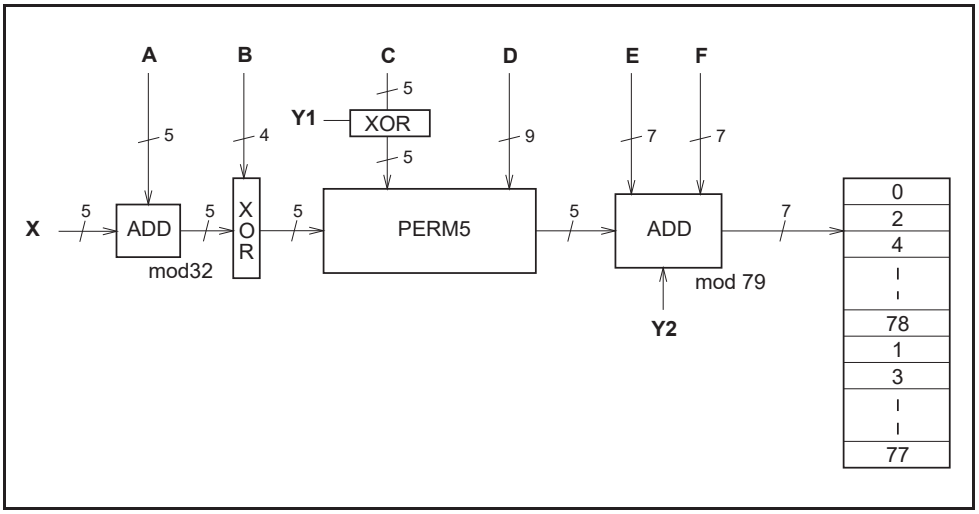



*Figure 2.16: Block diagram of the basic hop selection kernel for the hop system.*

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 chapter, the notation $A_i$ is used for bit i of the BD_ADDR.

#### *2.6.2.1 First addition operation*

The first addition operation only adds a constant to the phase and applies a modulo 32 operation. For the page hopping sequence, the first addition is redundant since it only changes 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.



LifeScan, Inc. Ex. 1019
Page 251 of 1420
IPR2025-00103

### 2.6.2.2 XOR operation

Let Z' denote the output of the first addition. In the XOR operation, the four LSBs of Z' are modulo-2 added to the address bits $A_{22\text{-}19}$. The operation is illustrated in Figure 2.17 on page 82.

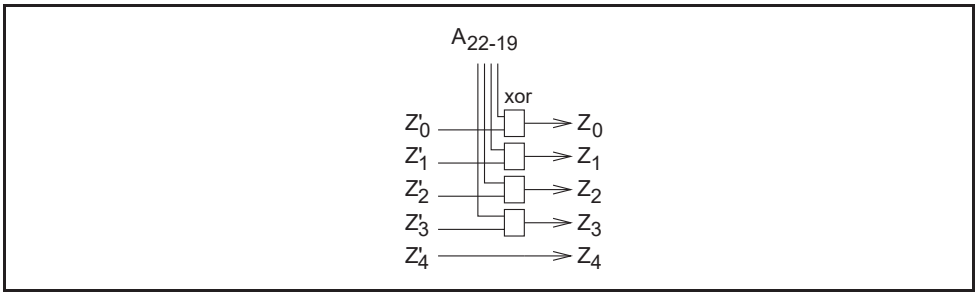



*Figure 2.17: XOR operation for the hop system.*

### 2.6.2.3 Permutation operation

The permutation operation involves the switching from 5 inputs to 5 outputs for the hop system, controlled by the control word. The permutation or switching box shall be as shown in Figure 2.18 on page 83. It consists of 7 stages of butterfly operations. The control of the butterflies by the control signals P is shown in Table 2.1. $P_{0\text{-}8}$ corresponds to $D_{0\text{-}8}$, and, $P_{i+9}$ corresponds to $C_i \oplus Y1$ for $i = 0 \ldots 4$ in Figure 2.16.

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

*Table 2.1: Control of the butterflies for the hop system*

The Z input is the output of the XOR operation as described in the previous section. The butterfly operation can be implemented with multiplexers as depicted in Figure 2.19 on page 83.



LifeScan, Inc. Ex. 1019
Page 252 of 1420
IPR2025-00103

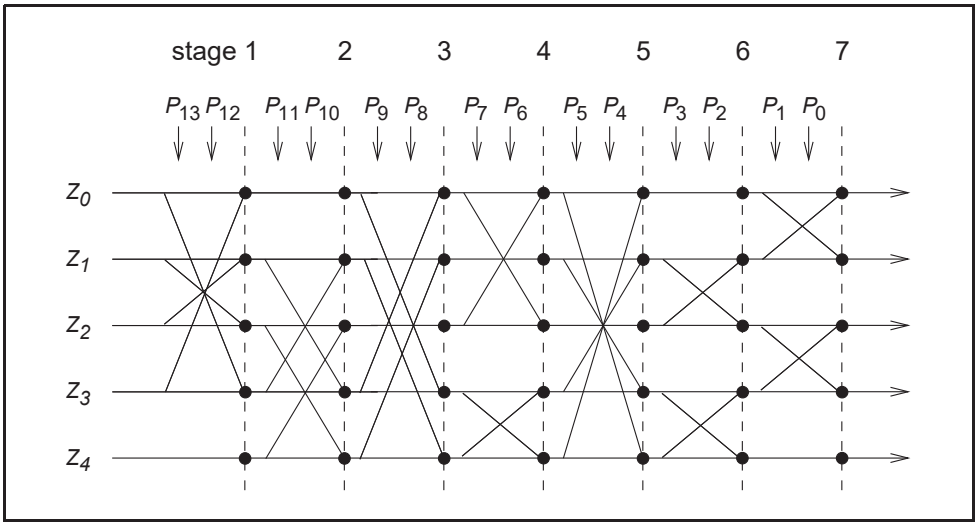



*Figure 2.18: Permutation operation for the hop system.*

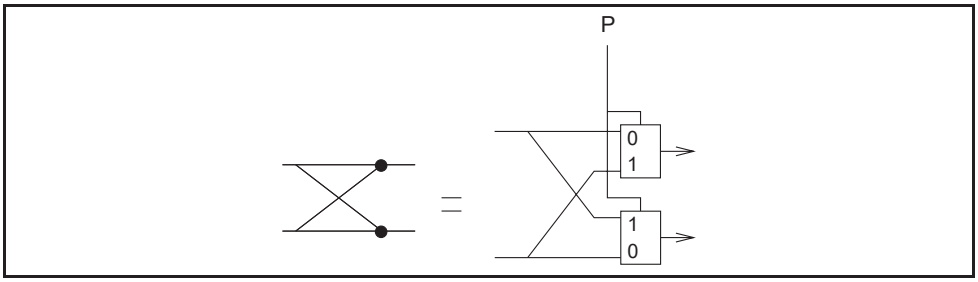



*Figure 2.19: Butterfly implementation.*

### *2.6.2.4 Second addition operation*

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

### *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.

### 2.6.3 Adapted hop selection kernel

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

The inputs to the adapted hop selection kernel are the same as for the basic hop system kernel except that the input AFH_channel_map (defined in Link Manager Protocol [Part C] Section 5.2 on page 313) 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 Section 2.3.1 on page 70.

Adaptation of the hopping sequence is achieved through two additions to the basic channel hopping sequence according to Figure 2.16 on page 81:

- *Unused* RF channels are re-mapped uniformly onto *used* RF channels. That is, 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 Section 2.6.1 on page 77).

#### *2.6.3.1 Channel re-mapping function*

When the adapted hop selection kernel is selected, the basic hop selection kernel according to Figure 2.16 on page 81 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 re-mapped by the re-mapping 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 (non-adapted) 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.

A block diagram of the re-mapping mechanism is shown in Figure 2.20 on page 85. The re-mapping function is a post-processing step to the selection kernel from Figure 2.16 on page 81, denoted as ‘Hop selection of the basic hop’. 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 either be in the set of *used* RF channels or in the set of *unused* RF channels.



LifeScan, Inc. Ex. 1019
Page 254 of 1420
IPR2025-00103

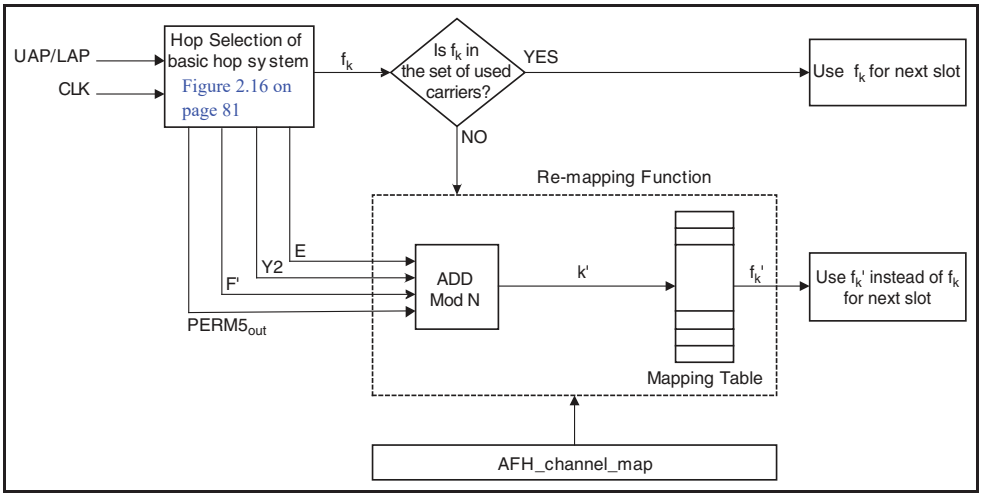



*Figure 2.20: Block diagram of adaptive hop selection mechanism*

When an unused RF channel is generated by the basic hop selection mechanism, it is re-mapped 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:

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

where *F'* is defined in Table 2.2 on page 86. The index *k'* is then used to select the re-mapped 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 2.16 on page 81 with all the *unused* RF channels removed).

### 2.6.4 Control word

In the following section $X_{j-i}$, *i*<*j*, will denote bits *i*, *i*+1,...,*j* of the bit vector X. By convention, $X_0$ is the least significant bit 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 2.16 on page 81 and Figure 2.20 on page 85. During paging and inquiry, the inputs A to E use the address values as given in the corresponding columns of Table 2.2 on page 86. In addition, the inputs X, Y1 and Y2 are used. The F and F' inputs 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 Section 8.5 on page 160), the inputs A, C and D shall be derived from the address bits being bit-wise XORed with the



LifeScan, Inc. Ex. 1019
Page 255 of 1420
IPR2025-00103

clock bits as shown in the “Connection state” column of Table 2.2 on page 86 (the two most significant bits, MSBs, are XORed together, the two second MSBs are XORed together, etc.).

| | Page scan / Interlaced Page Scan / Inquiry scan / Interlaced Inquiry Scan | Page/Inquiry | Master/Slave page response and Inquiry response | Connection state |
|---|---|---|---|---|
| X | $CLKN_{16-12}$ / $(CLKN_{16-12}+16) \bmod 32$ / $Xir_{4-0}$ / $Xir_{4-0}+16) \bmod 32$ | $Xp_{4-0}/Xi_{4-0}$ | $Xprm_{4-0}$/ $Xprs_{4-0}$/ $Xir_{4-0}$ | $CLK_{6-2}$ |
| Y1 | 0 | $CLKE_1/CLKN_1$ | $CLKE_1/CLKN_1/1$ | $CLK_1$ |
| Y2 | 0 | $32 \times CLKE_1/$ $32 \times CLKN_1$ | $32 \times CLKE_1$ / $32 \times CLKN_1$ / $32 \times 1$ | $32 \times CLK_1$ |
| A | $A_{27-23}$ | $A_{27-23}$ | $A_{27-23}$ | $A_{27-23} \oplus CLK_{25-21}$ |
| B | $A_{22-19}$ | $A_{22-19}$ | $A_{22-19}$ | $A_{22-19}$ |
| C | $A_{8,6,4,2,0}$ | $A_{8,6,4,2,0}$ | $A_{8,6,4,2,0}$ | $A_{8,6,4,2,0} \oplus CLK_{20-16}$ |
| D | $A_{18-10}$ | $A_{18-10}$ | $A_{18-10}$ | $A_{18-10} \oplus CLK_{15-7}$ |
| 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 | $16 \times CLK_{27-7} \bmod 79$ |
| F’ | n/a | n/a | n/a | $16 \times CLK_{27-7} \bmod N$ |

*Table 2.2: Control for hop system.*

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

### *2.6.4.1 Page scan and inquiry scan hopping sequences*

When the sequence selection input is set to page scan, the Bluetooth 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 inquiry access code to use depending on the purpose of the inquiry.

### 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 2.16. There are 32 possible paging frequencies within each 1.28 second 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,

$$Xp = [CLKE_{16-12} + k_{offset} + (CLKE_{4-2,0} - CLKE_{16-12}) \bmod 16] \bmod 32, \quad \text{(EQ 2)}$$

where

$$k_{offset} = \begin{cases} 24 & \text{A-train,} \\ 8 & \text{B-train.} \end{cases} \quad \text{(EQ 3)}$$

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).

### 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 native clock CLKN and the master's clock estimate CLKE, the four 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 native clock shall *not* be stopped; it is merely the values of the bits used for creating the X-input that are kept fixed for a while. A frozen value is denoted by an asterisk (*) in the discussion below.

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

$$Xprs = [CLKN^*_{16-12} + N] \bmod 32, \quad \text{(EQ 4)}$$



LifeScan, Inc. Ex. 1019
Page 257 of 1420
IPR2025-00103

The counter $N$ shall be set to zero in the slot where the slave acknowledges the page (see Figure 8.3 on page 152 and Figure 8.4 on page 152). 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 X-input 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.

#### *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 estimated slave clock 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 **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 X-input 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 X-input is:

$$\begin{aligned} Xprm = [&CLKE^*_{16-12} + k_{offset}{}^* + \\ &(CLKE^*_{4-2,0} - CLKE^*_{16-12}) \bmod 16 + N] \bmod 32, \end{aligned} \qquad \text{(EQ 5)}$$

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.

#### *2.6.4.5 Inquiry hopping sequence*

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

$$Xi = [CLKN_{16-12} + k_{offset} + (CLKN_{4-2,0} - CLKN_{16-12}) \bmod 16] \bmod 32, \qquad \text{(EQ 6)}$$

where $k_{offset}$ is defined by (EQ 3) on page 87. The initial choice of the offset is arbitrary.

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



LifeScan, Inc. Ex. 1019
Page 258 of 1420
IPR2025-00103

#### 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 X-input. The clock input shall not be frozen, thus the following equation apply:

$$Xir = [CLKN_{16-12} + N] \mod 32, \quad \text{(EQ 7)}$$

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 $X_{ir}$ value used for the extended inquiry response packet shall be the same $X_{ir}$ value as calculated for the immediately preceding FHS packet.

The GIAC LAP and the four 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.

#### 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 master clock, CLK. The address bits shall be derived from the Bluetooth device address of the master.



LifeScan, Inc. Ex. 1019
Page 259 of 1420
IPR2025-00103

# 3 PHYSICAL LINKS

A physical link represents a baseband 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 common properties of physical links are:

- Power control (see Link Manager Protocol Section 4.1.3 on page 225)
- Link supervision (see Section 3.1 on page 90 and Link Manager Protocol Section 4.1.6 on page 232)
- Encryption (see Security [Part H] Section 4 on page 868 and Link Manager Protocol [Part C] Section 4.2.5 on page 248)
- Channel quality-driven data rate change (see Link Manager Protocol Section 4.1.7 on page 233)
- Multi-slot packet control (see Link Manager Protocol Section 4.1.10 on page 237)

## 3.1 LINK SUPERVISION

A connection can break down due to various reasons such as a device moving out of range, encountering 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 logical transport address (see Section 4.2 on page 91) or parked member address (see Section 4.7.1 on page 99) 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 Section 4.2 on page 91) on the physical link, the 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 Link Manager. Its value shall be chosen so that the supervision timeout will be longer than hold and sniff periods. Link supervision of a parked slave shall be done by unparking and re-parking the slave.



LifeScan, Inc. Ex. 1019
Page 260 of 1420
IPR2025-00103

# 4 LOGICAL TRANSPORTS

## 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.

The ASB logical transport is used by a master to communicate with active slaves. The PSB logical transport is used by a master to communicate with parked slaves.

## 4.2 LOGICAL TRANSPORT ADDRESS (LT_ADDR)

Each slave active in a piconet is assigned a primary 3-bit logical transport address (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 only accept packets with matching primary or secondary LT_ADDR and broadcast packets. The LT_ADDR is carried in the packet header (see Section 6.4 on page 109). The LT_ADDR shall only be valid 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.



LifeScan, Inc. Ex. 1019
Page 261 of 1420
IPR2025-00103

The primary LT_ADDR shall be assigned by the master to the slave when the slave is activated. This is either at connection establishment, or 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.

## 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.

## 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 every slave. If there is no data to be sent on the ACL logical transport and no polling is required, no transmission is required.

## 4.5 TRANSMIT/RECEIVE ROUTINES

This section describes the way to use the packets as defined in Section 6 in order to support the traffic on the ACL, SCO and eSCO logical transports. Both single-slave and multi-slave configurations are considered. In addition, the use of buffers for the TX and RX routines are described.

*The TX and RX routines described in sections 4.5.1 and 4.5.2 are informative only.*

### 4.5.1 TX Routine

The TX routine is carried out separately for each asynchronous and synchronous link. Figure 4.1 on page 93 shows the asynchronous and synchronous buffers as used in the TX routine. In this figure, only a single TX asynchronous buffer and a single TX synchronous buffer are shown. In the master, there is a separate TX asynchronous buffer for each slave. In addition there may be one or more TX synchronous buffers for each synchronous slave (different SCO or eSCO logical transports may either reuse the same TX synchronous buffer, or each have their own TX synchronous buffer). Each TX buffer consists of two FIFO registers: one **current** register which can be accessed and read by the Link Controller in order to compose the packets, and one **next** register that can be accessed by the Baseband Resource Manager to load new information. The positions of the switches S1 and S2 determine which register is current and which register is next; the switches are controlled by the Link Controller. The switches at the input and the output of the FIFO registers can never be connected to the same register simultaneously.

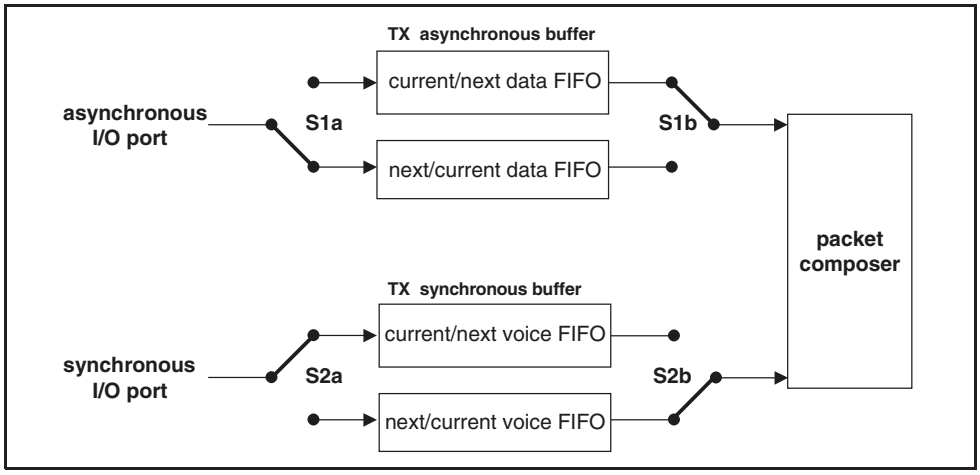



*Figure 4.1: Functional diagram of TX buffering.*

Of the packets common on the ACL and SCO logical transports (**NULL**, **POLL and DM1**) only the **DM1** packet carries a payload that is exchanged between the Link Controller and the Link Manager; this common packet makes use of the asynchronous buffer. All ACL packets make use of the asynchronous



LifeScan, Inc. Ex. 1019
Page 263 of 1420
IPR2025-00103

buffer. All SCO and eSCO packets make use of the synchronous buffer except for the **DV** packet where the synchronous data part is handled by the synchronous buffer and the data part is handled by the asynchronous buffer. In the next sections, the operation for ACL traffic, SCO traffic, eSCO traffic, and combined data-voice traffic on the SCO logical transport are described.

#### *4.5.1.1 ACL traffic*

In the case of asynchronous data only the TX ACL buffer in Figure 4.1 on page 93 has to be considered. In this case, only packet types **DM** or **DH** are used, and these can have different lengths. The length is indicated in the payload header. The selection of **DM** or **DH** packets should depend on the quality of the link. See [Part C] Section 4.1.7 on page 233.

The default packet type in pure data traffic is **NULL** (see Section 6.5.1.2 on page 113). This means that, if there is no data to be sent (the data traffic is asynchronous, and therefore pauses occur in which no data is available) or no slaves need to be polled, **NULL** packets are sent instead – in order to send link control information to the other device (e.g. ACK/STOP information for received data). When no link control information is available (either no need to acknowledge and/or no need to stop the RX flow) no packet is sent at all.

The TX routine works as follows. The Baseband Resource Manager loads new data information in the register to which the switch S1a points. Next, it gives a command to the Link Controller, which forces the switch S1 to change (both S1a and S1b switch synchronously). When the payload needs to be sent, the packet composer reads the current register and, depending on the packet type, builds a payload which is appended to the channel access code and the header and is subsequently transmitted. In the response packet (which arrives in the following RX slot if it concerned a master transmission, or may be postponed until some later RX slot if it concerned a slave transmission), the result of the transmission is reported back. In case of an ACK, the switch S1 changes position; if a NAK (explicit or implicit) is received instead, the switch S1 will not change position. In that case, the same payload is retransmitted at the next TX occasion.

As long as the Baseband Resource Manager keeps loading the registers with new information, the Link Controller will automatically transmit the payload; in addition, retransmissions are performed automatically in case of errors. The Link Controller will send **NULL** or nothing when no new data is loaded. If no new data has been loaded in the **next** register, during the last transmission, the packet composer will be pointing to an empty register after the last transmission has been acknowledged and the **next** register becomes the **current** register. If new data is loaded in the **next** register, a **flush** command is required to switch the S1 switch to the proper register. As long as the Baseband Resource Manager keeps loading the data and type registers before each TX slot, the data is automatically processed by the Link Controller since the S1 switch is controlled by the ACK information received in response. However, if the traffic from the Baseband Resource Manager is interrupted once and a default packet is sent instead, a **flush** command is necessary to continue the flow in the Link Controller.

The **flush** command can also be used in case of time-bounded (isochronous) data. In case of a bad link, many retransmissions are necessary. In certain applications, the data is time-bounded: if a payload is retransmitted all the time because of link errors, it may become outdated, and the system might decide to continue with more recent data instead and skip the payload that does not come through. This is accomplished by the **flush** command as well. With the **flush**, the switch S1 is forced to change and the Link Controller is forced to consider the next data payload and overrules the ACK control. Any ACL type of packet can be used to send data or link control information to any other ACL slave.

#### *4.5.1.2 SCO traffic*

On the SCO logical transport only **HV** and **DV** packet types are used, See Section 6.5.2 on page 116. The synchronous port may continuously load the **next** register in the synchronous buffer. The S2 switches are changed according to the $T_{sco}$ interval. This $T_{sco}$ interval is negotiated between the master and the slave at the time the SCO logical transport is established.

For each new SCO slot, the packet composer reads the **current** register after which the S2 switch is changed. If the SCO slot has to be used to send control information with high priority concerning a control packet between the master and the SCO slave, or a control packet between the master and any other slave, the packet composer will discard the SCO information and use the control information instead. This control information shall be sent in a DM1 packet. Data or link control information may also be exchanged between the master and the SCO slave by using the **DV** or **DM1** packets.

#### *4.5.1.3 Mixed data/voice traffic*

In Section 6.5.2 on page 116, a **DV** packet has been defined that can support both data and voice simultaneously on a single SCO logical transport. When the TYPE is **DV**, the Link Controller reads the data register to fill the data field and the voice register to fill the voice field. Thereafter, the switch S2 is changed. However, the position of S1 depends on the result of the transmission as on the ACL logical transport: only if an ACK has been received will the S1 switch change its position. In each **DV** packet, the voice information is new, but the data information might be retransmitted if the previous transmission failed. If there is no data to be sent, the SCO logical transport will automatically change from **DV** packet type to the current **HV** packet type used before the mixed data/voice transmission. Note that a **flush** command is necessary when the data stream has been interrupted and new data has arrived.

Combined data-voice transmission can also be accomplished by using a separate ACL logical transport in addition to the SCO logical transport(s) if channel capacity permits this.



LifeScan, Inc. Ex. 1019
Page 265 of 1420
IPR2025-00103

#### *4.5.1.4 eSCO Traffic*

On the eSCO logical transport only **EV**, **POLL** and **NULL** packet types are used, see Section 6.5.3 on page 117. The synchronous port may continuously load the next register in the synchronous buffer. The S2 switches are changed according to the $T_{eSCO}$ interval. This $T_{eSCO}$ interval is negotiated between the master and the slave at the time the eSCO logical transport is established.

For each new eSCO slot, the packet composer reads the current register after which the S2 switch is changed. If the eSCO slot has to be used to send control information with high priority concerning a control packet between the master and the eSCO slave, or an ACL packet between the master and any other slave, the packet composer will discard the eSCO information and use the control information instead. Control information to the eSCO slave is sent in a DM1 packet on the primary LT_ADDR.

#### *4.5.1.5 Default packet types*

On the ACL links, the default type is always **NULL** both for the master and the slave. This means that if no user information needs to be sent, either a **NULL** packet is sent if there is **ACK** or **STOP** information, or no packet is sent at all. The **NULL** packet can be used by the master to allocate the next slave-to-master slot to a certain slave (namely the one addressed). However, the slave is not forced to respond to the **NULL** packet from the master. If the master requires a response, it sends a **POLL** packet.

The SCO and eSCO packet types are negotiated at the LM level when the SCO or eSCO logical transport is established. The agreed packet type is also the default packet type for the reserved SCO or eSCO slots.

### 4.5.2 RX routine

The RX routine is carried out separately for the ACL logical transport and the synchronous logical transports. However, in contrast to the master TX asynchronous buffer, a single RX buffer is shared among all slaves. For the synchronous buffer, how the different synchronous logical transports are distinguished depends on whether extra synchronous buffers are required or not. Figure 4.2 on page 97 shows the asynchronous and synchronous buffers as used in the RX routine. The RX asynchronous buffer consists of two FIFO registers: one register that can be accessed and loaded by the Link Controller with the payload of the latest RX packet, and one register that can be accessed by the Baseband Resource Manager to read the previous payload. The RX synchronous buffer also consists of two FIFO registers: one register which is filled with newly arrived voice information, and one register which can be read by the voice processing unit.



LifeScan, Inc. Ex. 1019
Page 266 of 1420
IPR2025-00103

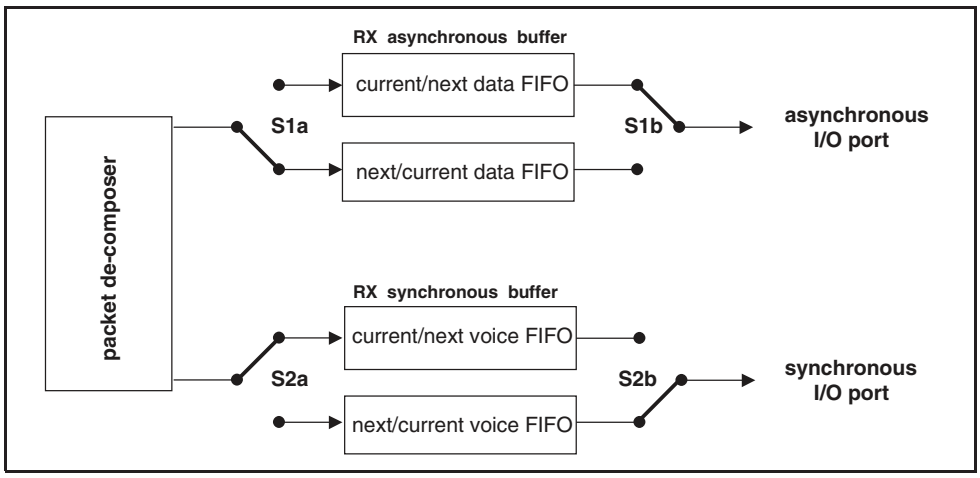



*Figure 4.2: Functional diagram of RX buffering*

Since the TYPE indication in the header (see Section 6.4.2 on page 109) of the received packet indicates whether the payload contains data and/or voice, the packet de-composer can automatically direct the traffic to the proper buffers. The switch S1 changes every time the Baseband Resource Manager reads the old register. If the next payload arrives before the RX register is emptied, a STOP indication is included in the packet header of the next TX packet that is returned. The STOP indication is removed again as soon as the RX register is emptied. The SEQN field is checked before a new ACL payload is stored into the asynchronous register (flush indication in LLID and broadcast messages influence the interpretation of the SEQN field see Section 7.6 on page 136).

The S2 switch is changed every $T_{SCO}$ or $T_{eSCO}$ for SCO and eSCO respectively. If, due to errors in the header, no new synchronous payload arrives, the switch still changes. The synchronous data processing unit then processes the synchronous data to account for the missing parts.

### 4.5.3 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.



LifeScan, Inc. Ex. 1019
Page 267 of 1420
IPR2025-00103

#### *4.5.3.1 Destination control*

As long as data can not be received, a STOP indication shall be transmitted which 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 Baseband 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 can not receive new information, it may still continue to transmit information: the flow control shall be separate for each direction.

#### *4.5.3.2 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 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 is present in the TX ACL buffers.

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

## 4.6 ACTIVE SLAVE BROADCAST TRANSPORT

The active slave broadcast 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 acknowledgement protocol and the traffic is uni-directional from the piconet master to the slaves. The ASB logical transport may only be used for L2CAP group traffic and shall never be used for L2CAP connection-oriented channels, L2CAP control signaling or LMP control signaling.

The ASB logical transport is unreliable. To improve reliability somewhat each packet is transmitted a number of times. An identical sequence number 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.



LifeScan, Inc. Ex. 1019
Page 268 of 1420
IPR2025-00103

## 4.7 PARKED SLAVE BROADCAST TRANSPORT

The parked slave broadcast 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 baseband signaling).

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

### 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 parked member address (PM_ADDR). This latter address is an 8-bit member address that separates the parked slaves. The PM_ADDR shall only be valid 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 [Part C] Section 4.5.2 on page 279).

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

### 4.7.2 Access request address (AR_ADDR)

The access request address (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 Section 8.9.6 on page 185. The AR_ADDR shall be assigned to the slave when it enters the **PARK** state and shall only be valid 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.



LifeScan, Inc. Ex. 1019
Page 269 of 1420
IPR2025-00103

# 5 LOGICAL LINKS

Five logical links are defined:

- Link Control (LC)
- ACL Control (ACL-C)
- User Asynchronous/Isochronous (ACL-U)
- User Synchronous (SCO-S)
- User Extended Synchronous (eSCO-S)

The control logical links LC and ACL-C are used at the link control level and link manager level, respectively. The ACL-U logical link is used to carry either asynchronous or isochronous user information. The SCO-S, and eSCO-S logical links are 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 logical link ID, 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, it 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.

## 5.1 LINK CONTROL LOGICAL LINK (LC)

The LC control 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 packet header.

## 5.2 ACL CONTROL LOGICAL LINK (ACL-C)

The ACL-C logical link shall carry control information exchanged between the link managers of the master and the slave(s). The ACL-C logical link shall use DM1or DV packets. DV packets shall only be used on the ACL-C link if the ACL-C message is less than or equal to 9 bytes and an HV1 synchronous logical transport is in use. The ACL-C logical link is indicated by the LLID code 11 in the payload header.

## 5.3 USER ASYNCHRONOUS/ISOCHRONOUS LOGICAL LINK (ACL-U)

The ACL-U logical link shall carry L2CAP asynchronous and isochronous user data. These messages may be transmitted in one or more baseband 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.



LifeScan, Inc. Ex. 1019
Page 270 of 1420
IPR2025-00103

### 5.3.1 Pausing the ACL-U logical link

When paused by LM, the Link Controller transmits the current packet with ACL-U information, if any, until an ACK is received or, optionally, until an explicit NACK 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 was paused after an ACK, the next sequence number shall be used on the next packet. If the ACL-U was paused after a NAK, the same sequence number shall be used on the next packet and the un-acknowledged packet shall be transmitted once the ACL-U logical link is un-paused.

When the ACL-U logical link is un-paused by LM, the Link Controller may resume transmitting packets with ACL-U information.

## 5.4 USER SYNCHRONOUS DATA LOGICAL LINK (SCO-S)

The SCO-S logical link carries transparent synchronous user data. This logical link is carried over the synchronous logical transport SCO.

## 5.5 USER EXTENDED SYNCHRONOUS DATA LOGICAL LINK (eSCO-S)

The eSCO-S logical link also carries transparent synchronous user data. This logical link is carried over the extended synchronous logical transport eSCO.

## 5.6 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 SCO-S and eSCO-S logical links but opportunities for interleaving the logical links should be taken.



LifeScan, Inc. Ex. 1019
Page 271 of 1420
IPR2025-00103

# 6 PACKETS

Bluetooth devices shall use the packets as defined in the following sections.

## 6.1 GENERAL FORMAT

### 6.1.1 Basic Rate

The general packet format of Basic Rate packets is shown in Figure 6.1 on page 102. Each packet consists of 3 entities: the access code, the header, and the payload. In the figure, the number of bits per entity is indicated.

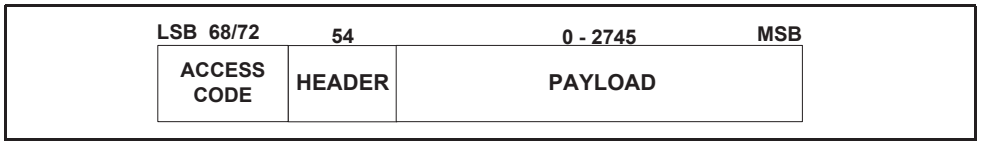



*Figure 6.1: General Basic Rate packet format.*

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 shortened access code only (see ID packet on page 116)
- the access code and the packet header
- the access code, the packet header and the payload.

### 6.1.2 Enhanced Data Rate

The general format of Enhanced Data Rate packets is shown in Figure 6.2 on page 102. The access code and packet header are identical in format and modulation to Basic Rate packets. Enhanced Data Rate packets have a guard time and synchronization sequence following the packet header. Following the payload are two trailer symbols. The guard time, synchronization sequence and trailer are defined in section 6.6.

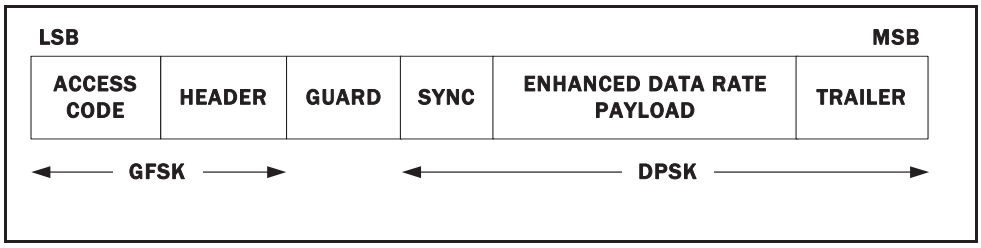



*Figure 6.2: General enhanced data rate packet format*

## 6.2 BIT ORDERING

The bit ordering when defining packets and messages in the *Baseband Specification*, follows the *Little Endian* format. The following rules apply:

- The *least significant bit* (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 baseband 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:

$$b_0 b_1 b_2 = 110$$

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



LifeScan, Inc. Ex. 1019
Page 273 of 1420
IPR2025-00103

## 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 signaling 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 6.3 on page 104. For details see Section 6.3.1 on page 104.

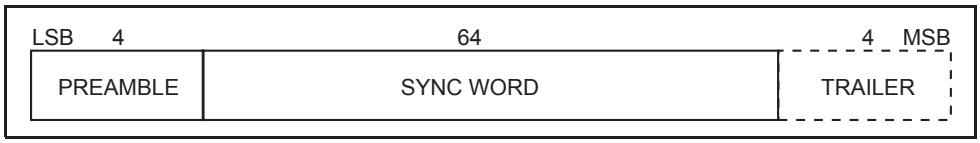



*Figure 6.3: Access code format*

### 6.3.1 Access code types

The different access code types use different Lower Address Parts (LAPs) to construct the sync word. The LAP field of the BD_ADDR is explained in Section 1.2 on page 62. A summary of the different access code types is in Table 6.1 on page 104.

| Code type | LAP | Code length | Comments |
|---|---|---|---|
| CAC | Master | 72 | See also Section 1.3 on page 63 |
| DAC | Paged device | 68/72[1] | |
| GIAC | Reserved | 68/72* | |
| DIAC | Dedicated | 68/72* | |

*Table 6.1: Summary of access code types.*

1. length 72 is only used 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 of length 68 bits.



LifeScan, Inc. Ex. 1019
Page 274 of 1420
IPR2025-00103

### 6.3.2 Preamble

The preamble is a fixed zero-one pattern of 4 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 6.4 on page 105.




*Figure 6.4: Preamble*

### 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.

#### *6.3.3.1 Synchronization word definition*

The sync words are based on a (64,30) expurgated block code with an overlay (bit-wise 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:

1. Generate information sequence;
2. XOR this with the "information covering" part of the PN overlay sequence;
3. Generate the codeword;
4. XOR the codeword with all 64 bits of the PN overlay sequence;

The information sequence is generated by appending 6 bits to the 24 bit LAP (step 1). 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 2 the information is pre-scrambled by XORing it with the bits $p_{34} \cdots p_{63}$ of the PN sequence (defined in Section 6.3.3.2 on page 108). After generating the codeword (step 3), the complete PN sequence is XORed to the



LifeScan, Inc. Ex. 1019
Page 275 of 1420
IPR2025-00103

codeword (step 4). This step de-scrambles the information part of the codeword. At the same time the parity bits of the codeword 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 is removed. The principle is depicted in Figure 6.5 on page 106

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 PRNG (see Figure 6.6 on page 108), 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. We denote the 24 bit lower address part (LAP) of the Bluetooth device address by $a(D) = a_0 + a_1 D + \ldots + a_{23} D^{23}$ ($a_0$ is the LSB of the Bluetooth device address).

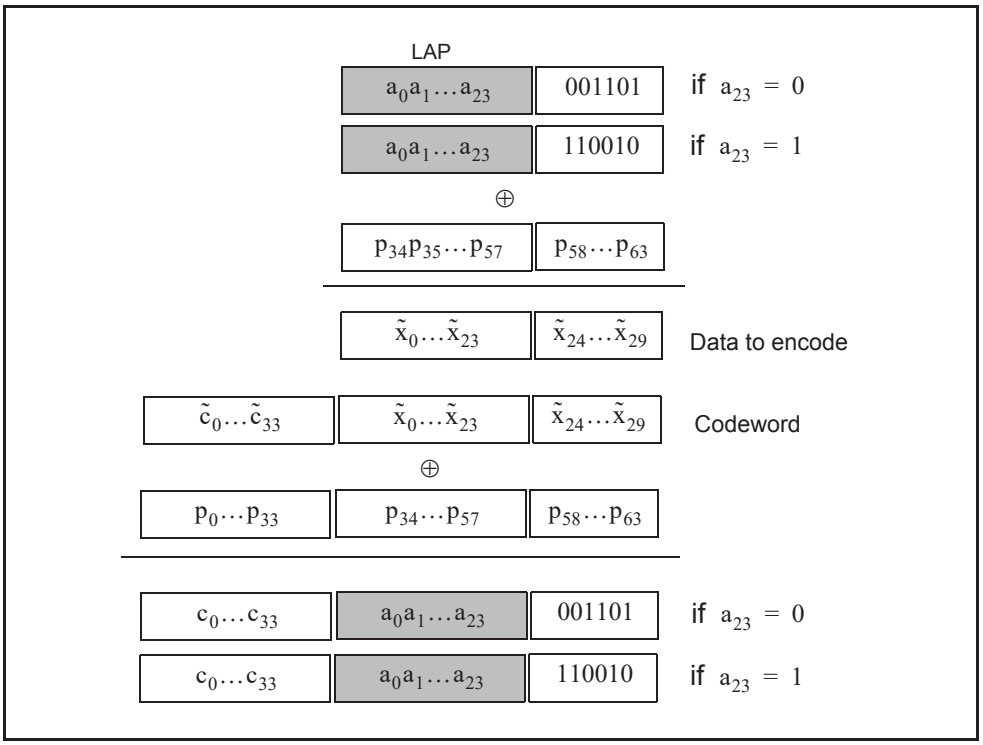



*Figure 6.5: Construction of the sync word.*

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



LifeScan, Inc. Ex. 1019
Page 276 of 1420
IPR2025-00103

$$g(D) = 260534236651, \quad \text{(EQ 8)}$$

the left-most bit corresponds to the high-order ($g_{34}$) coefficient. The DC-free four bit sequences 0101 and 1010 can be written

$$\begin{cases} F_0(D) = D + D^3, \\ F_1(D) = 1 + D^2, \end{cases} \quad \text{(EQ 9)}$$

respectively. Furthermore,

$$\begin{cases} B_0(D) = D^2 + D^3 + D^5, \\ B_1(D) = 1 + D + D^4, \end{cases} \quad \text{(EQ 10)}$$

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

1. Format the 30 information bits to encode:

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

2. Add the information covering part of the PN overlay sequence:

$$\tilde{x}(D) = x(D) + p_{34} + p_{35}D + \ldots + p_{63}D^{29}.$$

3. Generate parity bits of the (64,30) expurgated block code:[1]

$$\tilde{c}(D) = D^{34}\tilde{x}(D) \mod g(D).$$

4. Create the codeword:

$$\tilde{s}(D) = D^{34}\tilde{x}(D) + \tilde{c}(D).$$

5. Add the PN sequence:

$$s(D) = \tilde{s}(D) + p(D).$$

6. Append the (DC-free) preamble and trailer:

$$y(D) = F_{c_0}(D) + D^4 s(D) + D^{68}F_{a_{23}}(D).$$

1. $x(D)$ mod $y(D)$ denotes the remainder when $x(D)$ is divided by $y(D)$.



LifeScan, Inc. Ex. 1019
Page 277 of 1420
IPR2025-00103

#### *6.3.3.2 Pseudo-random noise sequence generation*

To generate the PN sequence the primitive polynomial $h(D) = 1 + D + D^3 + D^4 + D^6$ shall be used. The LFSR and its starting state are shown in Figure 6.6 on page 108. The PN sequence generated (including the extra terminating zero) becomes (hexadecimal notation) 83848D96BBCC54FC. The LFSR output starts with the left-most bit of this PN sequence. This corresponds to $p'(D)$ of the previous section. Thus, using the reciprocal $p(D)$ as overlay gives the 64 bit sequence:

$$\mathbf{p} = 3F2A33DD69B121C1, \quad \text{(EQ 11)}$$

where the left-most 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 right-most bit.

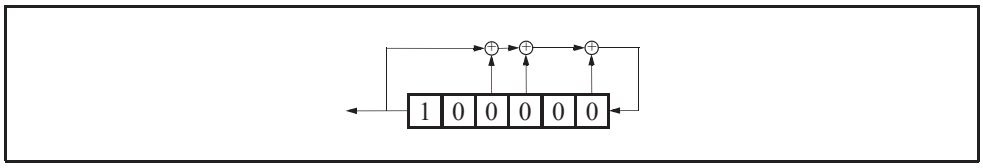



*Figure 6.6: LFSR and the starting state to generate $p'(D)$.*

### 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 three MSBs of the syncword 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 6.7 on page 108.

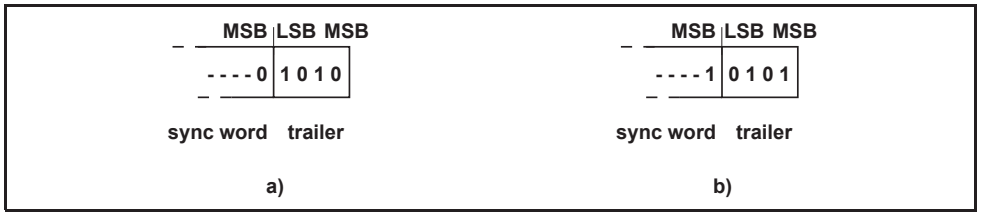



*Figure 6.7: Trailer in CAC when MSB of sync word is 0 (a), and when MSB of sync word is 1 (b).*



LifeScan, Inc. Ex. 1019
Page 278 of 1420
IPR2025-00103

## 6.4 PACKET HEADER

The header contains link control (LC) information and consists of 6 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 6.8 on page 109, and is encoded with a rate 1/3 FEC (not shown but described in Section 7.4 on page 134) resulting in a 54-bit header. The LT_ADDR and TYPE fields shall be sent LSB first.

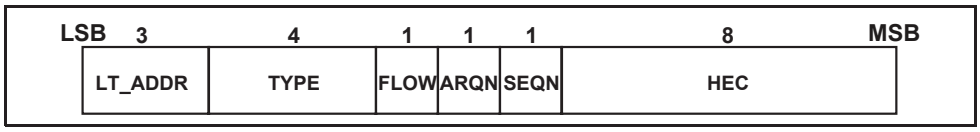



*Figure 6.8: Header format.*

### 6.4.1 LT_ADDR

The 3-bit LT_ADDR field contains the logical transport address for the packet (see Section 4.2 on page 91). 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.

### 6.4.2 TYPE

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 logical transport address 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. Second, it shall be determined whether Enhanced Data Rate has been enabled for the logical transport (ACL or eSCO) indicated by LT_ADDR. It can then 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 6.2 on page 112). This allows the non-addressed receivers to refrain from listening to the channel for the duration of the remaining slots. In Section 6.5 on page 111, each packet type is described in more detail.



LifeScan, Inc. Ex. 1019
Page 279 of 1420
IPR2025-00103

### 6.4.3 FLOW

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 only affects ACL packets. Packets including only link control information (ID, POLL, and NULL packets), SCO packets or eSCO packets can still 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 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.

### 6.4.4 ARQN

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 acknowledge ACK or negative acknowledge NAK. See Section 7.6 on page 136 for initialization and usage of this bit.

### 6.4.5 SEQN

The SEQN bit provides a sequential numbering scheme to order the data packet stream. See Section 7.6.2 on page 139 for initialization and usage of the SEQN bit. For broadcast packets, a modified sequencing method is used, see Section 7.6.5 on page 143.

### 6.4.6 HEC

Each header has a header-error-check to check the header integrity. The HEC is an 8-bit word (generation of the HEC is specified in Section 7.1.1 on page 130). 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 and extended inquiry response packets sent in **inquiry response**, the default check initialization (DCI, see Section 1.2.1 on page 62) 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 Section 7.1 on page 130.

## 6.5 PACKET TYPES

The packets used on the piconet are related to the logical transports they are used in. Three logical transports with distinct packet types are defined (see Section 4 on page 91): the SCO logical transport, the eSCO logical transport, and 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 6.2 on page 112 summarizes the packets defined for the SCO, eSCO, and ACL logical transport types.

All packet types with a payload shall use GFSK modulation unless specified otherwise in the following sections.

ACL logical transports Enhanced Data Rate packet types are explicitly selected via LMP using the *packet_type_table* (ptt) parameter. eSCO Enhanced Data Rate packet types are selected when the eSCO logical transport is established.



LifeScan, Inc. Ex. 1019
Page 281 of 1420
IPR2025-00103

.

| Segment | TYPE code $b_3b_2b_1b_0$ | Slot occupancy | SCO logical transport (1 Mbps) | eSCO logical transport (1 Mbps) | eSCO logical transport (2-3 Mbps) | ACL logical transport (1 Mbps) ptt=0 | ACL logical transport (2-3 Mbps) ptt=1 |
|---|---|---|---|---|---|---|---|
| 1 | 0000 | 1 | NULL | NULL | NULL | NULL | NULL |
| | 0001 | 1 | POLL | POLL | POLL | POLL | POLL |
| | 0010 | 1 | FHS | reserved | reserved | FHS | FHS |
| | 0011 | 1 | DM1 | reserved | reserved | DM1 | DM1 |
| 2 | 0100 | 1 | undefined | undefined | undefined | DH1 | 2-DH1 |
| | 0101 | 1 | HV1 | undefined | undefined | undefined | undefined |
| | 0110 | 1 | HV2 | undefined | 2-EV3 | undefined | undefined |
| | 0111 | 1 | HV3 | EV3 | 3-EV3 | undefined | undefined |
| | 1000 | 1 | DV | undefined | undefined | undefined | 3-DH1 |
| | 1001 | 1 | undefined | undefined | undefined | AUX1 | AUX1 |
| 3 | 1010 | 3 | undefined | undefined | undefined | DM3 | 2-DH3 |
| | 1011 | 3 | undefined | undefined | undefined | DH3 | 3-DH3 |
| | 1100 | 3 | undefined | EV4 | 2-EV5 | undefined | undefined |
| | 1101 | 3 | undefined | EV5 | 3-EV5 | undefined | undefined |
| 4 | 1110 | 5 | undefined | undefined | undefined | DM5 | 2-DH5 |
| | 1111 | 5 | undefined | undefined | undefined | DH5 | 3-DH5 |

*Table 6.2: Packets defined for synchronous and asynchronous logical transport types.*

### 6.5.1 Common packet types

There are five common kinds of packets. In addition to the types listed in segment 1 of the previous table, the ID packet is also a common packet type but is not listed in segment 1 because it does not have a packet header.

#### *6.5.1.1 ID packet*

The identity or ID packet consists of the device access code (DAC) or inquiry access code (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.



LifeScan, Inc. Ex. 1019
Page 282 of 1420
IPR2025-00103

#### 6.5.1.2 NULL packet

The NULL packet has no payload and consists of the channel access code 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 may not have to be acknowledged.

#### 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 acknowledgement 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.

#### 6.5.1.4 FHS packet

The FHS packet is a special control packet containing, among other things, the Bluetooth 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 6.9 on page 113 illustrates the format and contents of the FHS payload. The payload consists of eleven fields. The FHS packet is used in page master response, inquiry response and in 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 than 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 channel has been established, or when an existing piconet changes to a new piconet.

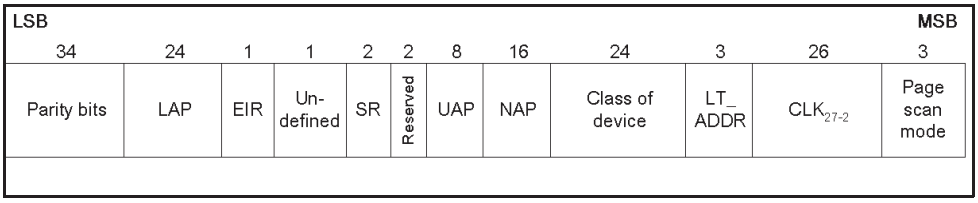



*Figure 6.9: Format of the FHS payload.*

Each field is described in more detail below:



LifeScan, Inc. Ex. 1019
Page 283 of 1420
IPR2025-00103

| | |
|---|---|
| **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 Section 1.2 on page 62. |
| **LAP** | This 24-bit field shall contain the lower address part of the device that sends the FHS packet. |
| **EIR** | This bit shall indicate that an extended inquiry response packet may follow. See Section 8.4.3 on page 158. |
| **Undefined** | This 1-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 6.4 and Table 8.1 on page 147 |
| **Reserved** | This 2-bit field shall be set to 10. |
| **UAP** | This 8-bit field shall contain the upper address part of the device that sends the FHS packet. |
| **NAP** | This 16-bit field shall contain the non-significant address part of the device that sends the FHS packet (see also Section 1.2 on page 62 for LAP, UAP, and NAP). |
| **Class of device** | This 24-bit field shall contain the class of device of the device that sends the FHS packet. The field is defined in Bluetooth Assigned Numbers. |
| **LT_ADDR** | This 3-bit field shall contain the logical transport address 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. |
| **$CLK_{27-2}$** | This 26-bit field shall contain the value of the native clock 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 1.25ms (two-slot interval). For each 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 default by the sender of the FHS packet. The interpretation of the page scan mode is illustrated in Table 6.5. |

*Table 6.3: Description of the FHS payload*



LifeScan, Inc. Ex. 1019
Page 284 of 1420
IPR2025-00103

The device sending the FHS shall set the SR bits according to Table 6.4.

| SR bit format $b_1b_0$ | SR mode |
|---|---|
| 00 | R0 |
| 01 | R1 |
| 10 | R2 |
| 11 | reserved |

*Table 6.4: Contents of SR field*

The device sending the FHS shall set the page scan mode bits according to Table 6.5.

| 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 |

*Table 6.5: Contents of page scan mode field*

The LAP, UAP, and NAP together form the 48-bit Bluetooth Device Address of the device that sends the FHS packet. Using the parity bits and the LAP, the recipient can directly construct the channel access code of the sender of the FHS packet.

When initializing the HEC and CRC for the FHS packet of inquiry response, the UAP shall be the DCI.

#### *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 6.2 on page 112). However, it may also carry regular user data. Since the DM1 packet can be regarded as an ACL packet, it will be discussed in Section 6.5.4 on page 119.



LifeScan, Inc. Ex. 1019
Page 285 of 1420
IPR2025-00103

### 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 section, but not on the synchronous data section. The data section of DV packets shall be retransmitted. 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 64kb/s speech transmission but may be used for transparent synchronous data.

#### *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.

#### *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.

#### *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.

#### *6.5.2.4 DV packet*

The DV packet is a combined data - voice packet. The DV packet shall only be used 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 6.10. 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; that is, 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 be acknowledged before the SCO logical transport is terminated, the asynchronous data field shall be retransmitted in a DM1 packet.

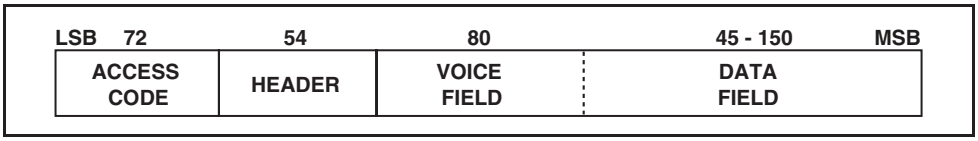



*Figure 6.10: DV packet format*



LifeScan, Inc. Ex. 1019
Page 286 of 1420
IPR2025-00103

### 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 acknowledgement of proper reception is received within the retransmission window. eSCO packets may be routed to the synchronous I/O port. Three eSCO packet types (EV3, EV4, EV5) are defined for Basic Rate operation and four additional eSCO packet types (2-EV3, 3-EV3, 2-EV5, 3-EV5) for Enhanced Data Rate operation. The eSCO packets may be used for 64kb/s speech transmission as well as transparent data at 64kb/s and other rates.

#### *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 re-negotiated.

#### *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 re-negotiated.

#### *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 re-negotiated.

#### *6.5.3.4 2-EV3 packet*

The **2-EV3** packet is similar to the EV3 packet except that the payload is modulated using $\pi/4$-DQPSK. It has between 1 and 60 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 2-EV3 packet covers 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 re-negotiated.



LifeScan, Inc. Ex. 1019
Page 287 of 1420
IPR2025-00103

#### *6.5.3.5 2-EV5 packet*

The **2-EV5** packet is similar to the EV5 packet except that the payload is modulated using π/4-DQPSK. It has between 1 and 360 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 2-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 re-negotiated.

#### *6.5.3.6 3-EV3 packet*

The **3-EV3** packet is similar to the EV3 packet except that the payload is modulated using 8DPSK. It has between 1 and 90 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 3-EV3 packet covers 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 re-negotiated.

#### *6.5.3.7 3-EV5 packet*

The **3-EV5** packet is similar to the EV5 packet except that the payload is modulated using 8DPSK. It has between 1 and 540 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 3-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 re-negotiated.



LifeScan, Inc. Ex. 1019
Page 288 of 1420
IPR2025-00103

### 6.5.4 ACL packets

ACL packets are used on the asynchronous logical transport. The information carried may be user data or control data.

Seven packet types are defined for Basic Rate operation: DM1, DH1, DM3, DH3, DM5, DH5 and AUX1. Six additional packets are defined for Enhanced Data Rate operation: 2-DH1, 3-DH1, 2-DH3, 3-DH3, 2-DH5 and 3-DH5.

#### *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 6.12 on page 123. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

#### *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.

#### *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 6.13 on page 124. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

#### *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.

#### *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



LifeScan, Inc. Ex. 1019
Page 289 of 1420
IPR2025-00103

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).

#### *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.

#### *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.

#### *6.5.4.8 2-DH1 packet*

This packet is similar to the DH1 packet except that the payload is modulated using $\pi$/4-DQPSK. The 2-DH1 packet has between 2 and 56 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 2-DH1 packet occupies a single time slot.

#### *6.5.4.9 2-DH3 packet*

This packet is similar to the DH3 packet except that the payload is modulated using $\pi$/4-DQPSK. The 2-DH3 packet has between 2 and 369 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 2-DH3 packet may occupy up to three time slots.

#### *6.5.4.10 2-DH5 packet*

This packet is similar to the DH5 packet except that the payload is modulated using $\pi$/4-DQPSK. The 2-DH5 packet has between 2 and 681 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 2-DH5 packet may occupy up to five time slots.

#### *6.5.4.11 3-DH1 packet*

This packet is similar to the DH1 packet except that the payload is modulated using 8DPSK. The 3-DH1 packet has between 2 and 85 information bytes

(including the 2-byte payload header) plus a 16-bit CRC code. The 3-DH1 packet occupies a single time slot.

#### *6.5.4.12 3-DH3 packet*

This packet is similar to the DH3 packet except that the payload is modulated using 8DPSK. The 3-DH3 packet has between 2 and 554 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 3-DH3 packet may occupy up to three time slots.

#### *6.5.4.13 3-DH5 packet*

This packet is similar to the DH5 packet except that the payload is modulated using 8DPSK. The 3-DH5 packet has between 2 and 1023 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 3-DH5 packet may occupy up to five time slots.

## 6.6 PAYLOAD FORMAT

In the payload, two fields are distinguished: the synchronous data field and the asynchronous data field. The ACL packets only have the asynchronous data field and the SCO and eSCO packets only have the synchronous data field – with the exception of the DV packets which have both.

### 6.6.1 Synchronous Data Field

In SCO, which is only supported in Basic Rate mode, the synchronous data field has a fixed length and consists only of the synchronous data body portion. No payload header is present.

In Basic Rate eSCO, the synchronous data field consists of two segments: a synchronous data body and a CRC code. No payload header is present.

In Enhanced Data Rate eSCO, the synchronous data field consists of five segments: a guard time, a synchronization sequence, a synchronous data body, a CRC code and a trailer. No payload header is present.

**1. Enhanced Data Rate Guard Time**

For Enhanced Data Rate packets the guard time is defined as the period starting at the end of the last GFSK symbol of the header and ending at the start of the reference symbol of the synchronization sequence. The length of the guard time shall be between 4.75 µsec and 5.25 µsec.



LifeScan, Inc. Ex. 1019
Page 291 of 1420
IPR2025-00103

**2. Enhanced Data Rate Synchronization Sequence**

For Enhanced Data Rate packets the symbol timing at the start of the synchronization sequence shall be within ±¼ µsec of the symbol timing of the last GFSK symbol of the packet header. The length of the synchronization sequence is 11 µsec (11 DPSK symbols) and consists of a reference symbol (with arbitrary phase) followed by ten DPSK symbols.

The phase changes between the DPSK symbols (shown in Synchronization sequence) shall be

$$\{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}\} = \{3\pi/4, -3\pi/4, 3\pi/4, -3\pi/4, 3\pi/4, -3\pi/4, -3\pi/4, 3\pi/4, 3\pi/4, 3\pi/4\} \quad \text{(EQ 12)}$$

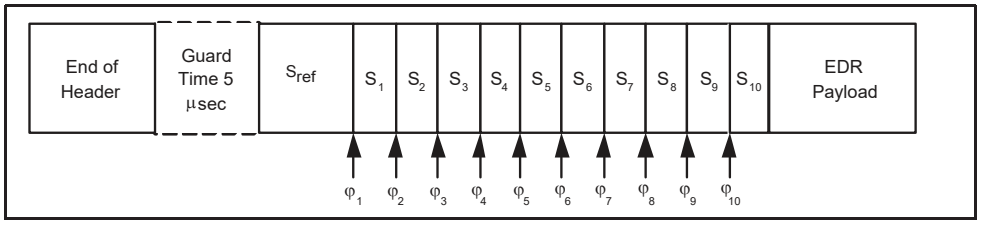



*Figure 6.11: Synchronization sequence*

Sref is the reference symbol. $\varphi_1$ is the phase change between the reference symbol and the first DPSK symbol $S_1$. $\varphi_k$ is the phase change between the k-1$^{th}$ symbol $S_{k-1}$ and the k$^{th}$ symbol $S_k$

Note: the synchronization sequence may be generated using the modulator by pre-pending the data with bits that generate the synchronization sequence.

For $\pi$/4-DQPSK, the bit sequence used to generate the synchronization sequence is 0,1,1,1,0,1,1,1,0,1,1,1,1,1,0,1,0,1,0,1.

For 8DPSK, the bit sequence used to generate the synchronization sequence is 0,1,0,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,1,1,1,1,0,1,0,0,1,0,0,1,0.

**3. 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 re-negotiated. The synchronous data body length may be different for each direction of the eSCO logical transport.

**4. CRC code**

The 16-bit CRC in the payload is generated as specified in Section 7.1 on page 130. The 8-bit UAP of the master is used to initialize the CRC generator.

Only the Synchronous data body segment is used to generate the CRC code.



LifeScan, Inc. Ex. 1019
Page 292 of 1420
IPR2025-00103

**5. Enhanced Data Rate Trailer**

For Enhanced Data Rate packets, two trailer symbols shall be added to the end of the payload. The trailer bits shall be all zero, i.e. {00, 00} for the π/4-DQPSK and {000, 000} for the 8DPSK.

### 6.6.2 Asynchronous data field

Basic rate 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).

Enhanced Data Rate ACL packets have an asynchronous data field consisting of six segments: a guard time, a synchronization sequence, a payload header, a payload body, a CRC and a trailer.

**1. Enhanced Data Rate Guard time**

This is the same as defined for the Synchronous data field in Section 6.6.1.

**2. Enhanced Data Rate Synchronization sequence**

This is the same as defined for the Synchronous data field in Section 6.6.1.

**3. Payload header**

The payload header is one or two bytes long. Basic rate packets in segments one and two have a 1-byte payload header; Basic Rate packets in segments three and four and all Enhanced Data Rate packets 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 10 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 five bits into the next byte. The remaining three 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 6.12 on page 123 and Figure 6.13 on page 124.

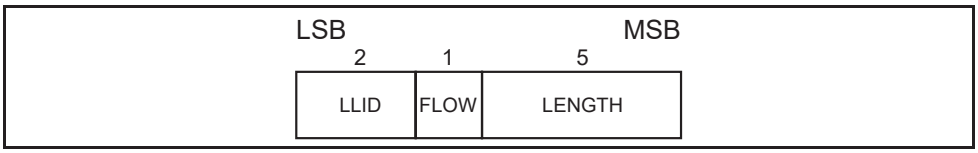



*Figure 6.12: Payload header format for Basic Rate single-slot ACL packets.*



LifeScan, Inc. Ex. 1019
Page 293 of 1420
IPR2025-00103

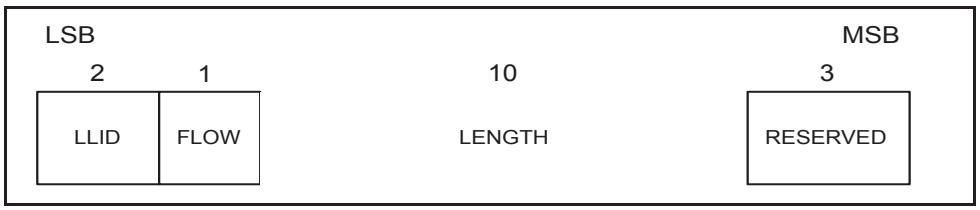



*Figure 6.13: Payload header format for multi-slot ACL packets and all EDR ACL packets.*

The LLID field shall be transmitted first, the length field last. In Table 6.6 on page 124, more details about the contents of the LLID field are listed.

| **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 |

*Table 6.6: Logical link LLID field contents*

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. The first fragment of an L2CAP message sent over HCI is identified by having a Packet_Boundary_Flag value of 00 or 10 both of which are mapped to LLID code 10. 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 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 ($7 \times 256$ 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 baseband 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.



LifeScan, Inc. Ex. 1019
Page 295 of 1420
IPR2025-00103

The Baseband 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 Section 6.4.3 on page 110). 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 6.7 on page 126. On the ACL-C logical link, no flow control is applied and the payload FLOW bit shall always be set to one.

| 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 |

*Table 6.7: Use of payload header flow bit on the logical links.*

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 6.12 and Figure 6.13, the MSB of the length field in a 1-byte header is the last (right-most) 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.

**4. 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.

**5. CRC code generation**

The 16-bit cyclic redundancy check code in the payload is generated as specified in Section 7.1 on page 130. 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 response** substate, the UAP of the slave is used. For the FHS packet and the extended inquiry response packet sent in **inquiry response** substate, the DCI (see Section 1.2.1 on page 62) is used. For all other packets, the UAP of the master is used.

Only the Payload header and Payload body segments are used to generate the CRC code.

**6. Enhanced Data Rate Trailer**

This is the same as defined for the Synchronous data field in section 6.6.1.



LifeScan, Inc. Ex. 1019
Page 296 of 1420
IPR2025-00103

## 6.7 PACKET SUMMARY

A summary of the packets and their characteristics is shown in Table 6.8, Table 6.9 and Table 6.10. The payload represents the packet payload excluding FEC, CRC, and payload header.

| Type | Payload (bytes) | FEC | CRC | Symmetric Max. Rate | Asymmetric Max. 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 6.8: Link control packets*

| Type | Payload Header (bytes) | User Payload (bytes) | FEC | CRC | Symmetric Max. Rate (kb/s) | Asymmetric Max. 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 |
| 2-DH1 | 2 | 0-54 | no | yes | 345.6 | 345.6 | 345.6 |
| 2-DH3 | 2 | 0-367 | no | yes | 782.9 | 1174.4 | 172.8 |
| 2-DH5 | 2 | 0-679 | no | yes | 869.1 | 1448.5 | 115.2 |
| 3-DH1 | 2 | 0-83 | no | yes | 531.2 | 531.2 | 531.2 |
| 3-DH3 | 2 | 0-552 | no | yes | 1177.6 | 1766.4 | 235.6 |
| 3-DH5 | 2 | 0-1021 | no | yes | 1306.9 | 2178.1 | 177.1 |

*Table 6.9: ACL packets*



LifeScan, Inc. Ex. 1019
Page 297 of 1420
IPR2025-00103

| Type | Payload Header (bytes) | User Payload (bytes) | FEC | CRC | Symmetric Max. 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[1] | 1 D | 10+(0-9) D | 2/3 D | yes D | 64.0+57.6 D |
| EV3 | na | 1-30 | No | Yes | 96 |
| EV4 | na | 1-120 | 2/3 | Yes | 192 |
| EV5 | na | 1-180 | No | Yes | 288 |
| 2-EV3 | na | 1-60 | No | Yes | 192 |
| 2-EV5 | na | 1-360 | No | Yes | 576 |
| 3-EV3 | na | 1-90 | No | Yes | 288 |
| 3-EV5 | na | 1-540 | No | Yes | 864 |

*Table 6.10: Synchronous packets*

1. Items followed by 'D' relate to data field only.



LifeScan, Inc. Ex. 1019
Page 298 of 1420
IPR2025-00103

# 7 BITSTREAM PROCESSING

Bluetooth devices shall use the bitstream processing schemes as defined in the following sections.

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 7.1 on page 129 shows the processes carried out for the packet header both at the transmit and the receive side. All header bit processes are mandatory.

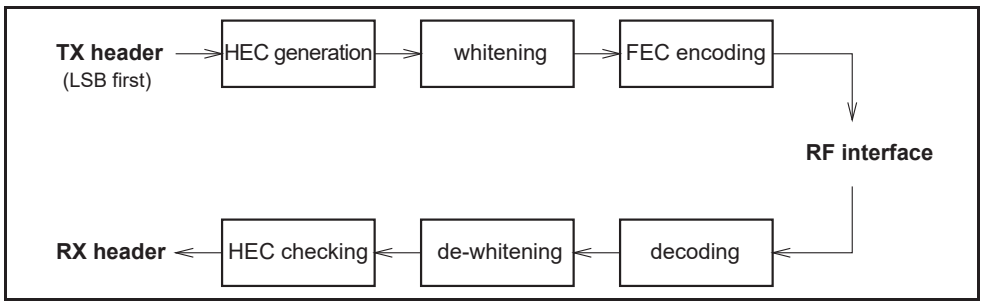



*Figure 7.1: Header bit processes.*

Figure 7.2 on page 129 shows the processes that may be carried out on the payload. In addition to the processes defined for the packet header, encryption can be applied on the payload. Only whitening and de-whitening, as explained in Section 7.2 on page 133, are mandatory for every payload; all other processes are optional and depend on the packet type (see Section 6.6 on page 121) and whether encryption is enabled. In Figure 7.2 on page 129, optional processes are indicated by dashed blocks.

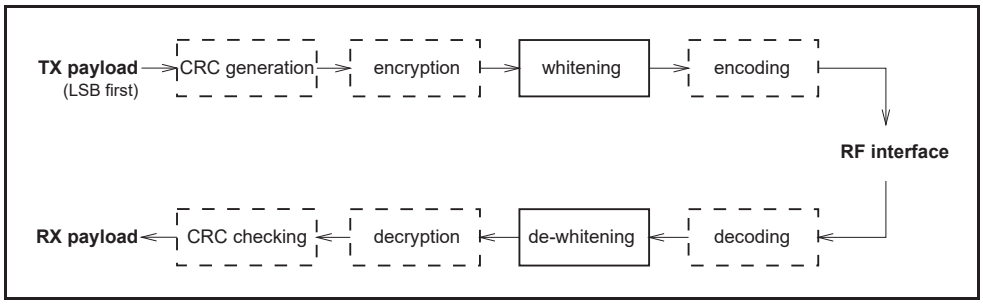



*Figure 7.2: Payload bit processes.*



LifeScan, Inc. Ex. 1019
Page 299 of 1420
IPR2025-00103

## 7.1 ERROR CHECKING

The packet can be checked for errors or wrong delivery using the channel access code, 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 channel access code 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 and the extended inquiry response 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 7.5 on page 131 and Figure 7.8 on page 132. 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).

### 7.1.1 HEC generation

The HEC generating LFSR is depicted in Figure 7.3 on page 130. 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 pre-loaded with the 8-bit UAP such that the LSB of the UAP (denoted $UAP_0$) goes to the left-most shift register element, and, $UAP_7$ goes to the right-most element. The initial state of the HEC LFSR is depicted in Figure 7.4 on page 131. 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, etc.).

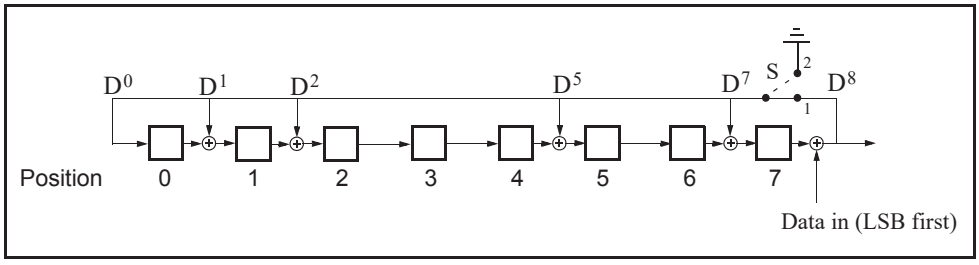



*Figure 7.3: The LFSR circuit generating the HEC.*

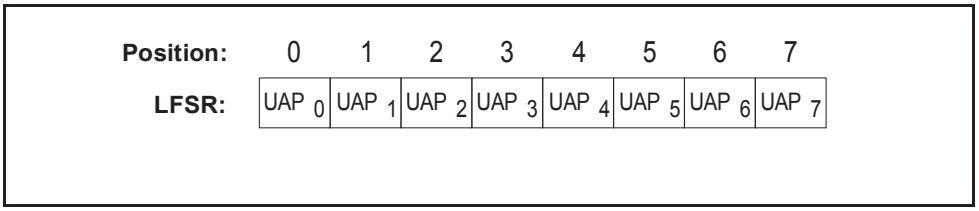



*Figure 7.4: Initial state of the HEC generating circuit.*

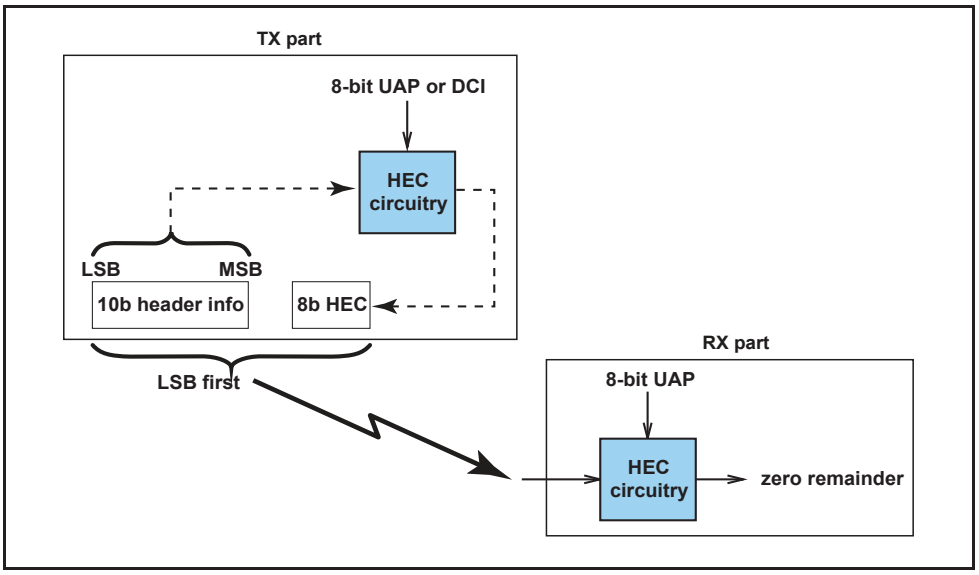



*Figure 7.5: HEC generation and checking.*

### 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 7.6 on page 132). For this case, the 8 left-most bits shall be initially loaded with the 8-bit UAP ($UAP_0$ to the left and $UAP_7$ to the right) while the 8 right-most bits shall be reset to zero. The initial state of the 16 bit LFSR is specified in Figure 7.7 on page 132. The switch S shall be set in position 1 while the data is 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, etc.).



LifeScan, Inc. Ex. 1019
Page 301 of 1420
IPR2025-00103

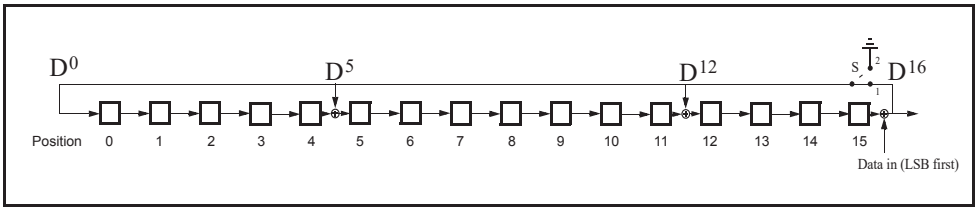



*Figure 7.6: The LFSR circuit generating the CRC.*

| 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 7.7: Initial state of the CRC generating circuit.*

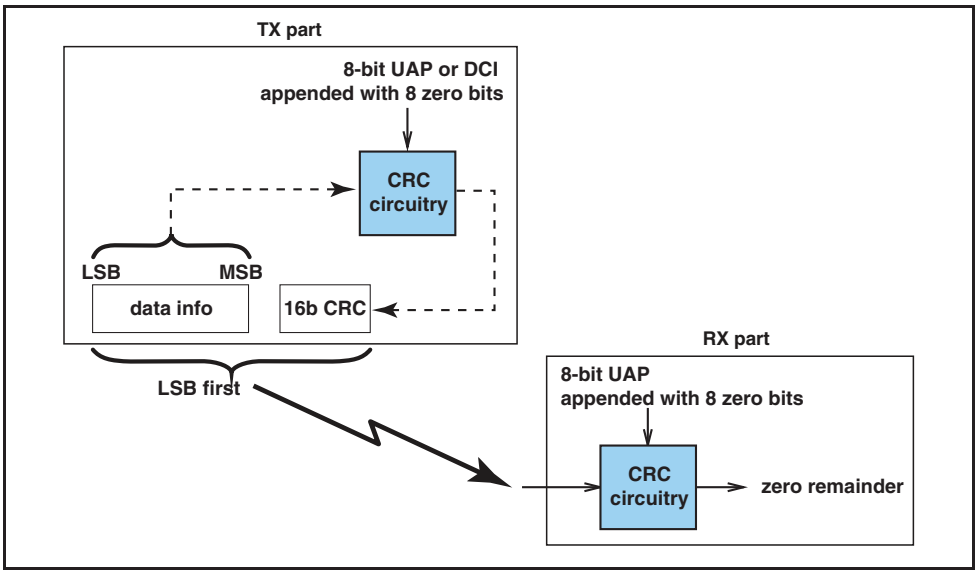



*Figure 7.8: CRC generation and checking.*

## 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 linear feedback shift register shown in Figure 7.9 on page 133. Before each transmission, the shift register shall be initialized with a portion of the master Bluetooth clock, CLK6-1, extended with an MSB of value one. This initialization shall be carried out with CLK1 written to position 0, CLK2 written to position 1, etc. Exceptions are the FHS packet sent during inquiry response or page response, and the extended inquiry response packet sent during inquiry response, where initialization of the whitening register shall be carried out differently. Instead of the master clock, the X-input used in the inquiry or page response (depending on current state) routine shall be used, see Table 2.2. The 5-bit value shall be extended with two MSBs of value 1. During register initialization, the LSB of X (i.e., X0) shall be written to position 0, X1 shall be written to position 1, etc.

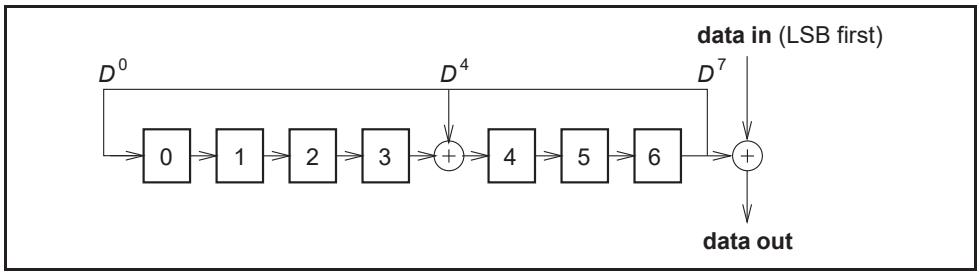



*Figure 7.9: Data whitening LFSR.*

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 re-initialization 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.

For Enhanced Data Rate packets, whitening shall not be applied to the guard, synchronization and trailer portions of the Enhanced Data Rate packets. During the periods where whitening is not applied the LFSR shall be paused.



LifeScan, Inc. Ex. 1019
Page 303 of 1420
IPR2025-00103

## 7.3 ERROR CORRECTION

There are three error correction schemes defined for Bluetooth:

- 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 reasonably error-free environment, FEC gives unnecessary overhead that reduces the throughput. Therefore, the packet definitions given in Section 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 section. This matter is discussed in Section 9.3 on page 190.

## 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 7.10 on page 134. The 3-times repetition code is used for the entire header, as well as for the synchronous data field in the **HV1** packet.

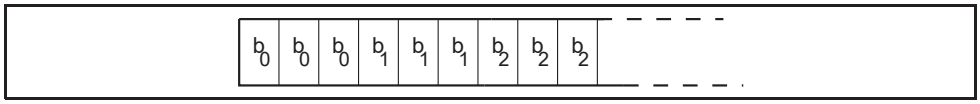



*Figure 7.10: Bit-repetition encoding scheme.*



LifeScan, Inc. Ex. 1019
Page 304 of 1420
IPR2025-00103

## 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. The LFSR generating this code is depicted in Figure 7.11 on page 135. 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 five 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 codeword. This code can correct all single errors and detect all double errors in each codeword. 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. 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.

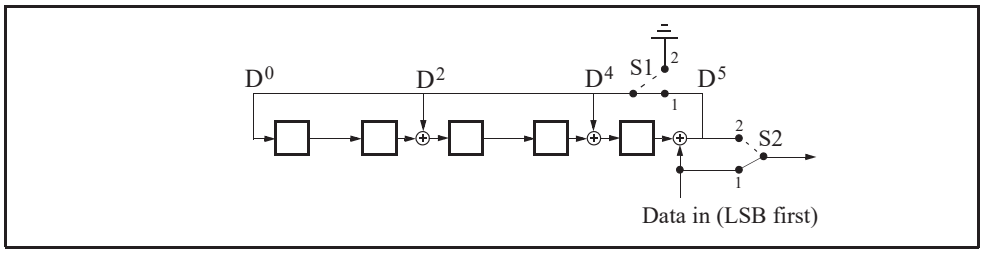



*Figure 7.11: LFSR generating the (15,10) shortened Hamming code.*



LifeScan, Inc. Ex. 1019
Page 305 of 1420
IPR2025-00103

## 7.6 ARQ SCHEME

With an automatic repeat request scheme, **DM, DH** the data field of **DV** packets, and **EV** packets shall be transmitted until acknowledgement of a successful reception is returned by the destination (or timeout is exceeded). The acknowledgement information shall be included in the header of the return packet. The ARQ scheme is only used 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.

### 7.6.1 Unnumbered ARQ

Bluetooth 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 only be affected by data packets containing CRC and empty slots. As shown in Figure 7.12 on page 137, upon successful reception of a CRC packet, the ARQN bit shall be set to ACK. If, in any receive slot in the slave, or, in a receive slot in the master following transmission of a packet, one of these events applies:

1. no access code is detected,
2. the HEC fails,
3. the CRC fails,

then the ARQN bit shall be set to NAK. In eSCO the ARQN bit may be set to ACK even when the CRC on an EV packet has failed thus enabling delivery of erroneous packets.

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 Section 7.6.2.2 on page 140. 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.

The ARQ bit in the extended inquiry response packet is not used. The bit shall be set to zero and ignored upon receipt.

Broadcast packets shall be checked on errors using the CRC, but no ARQ scheme shall be applied. Broadcast packets shall never be acknowledged.

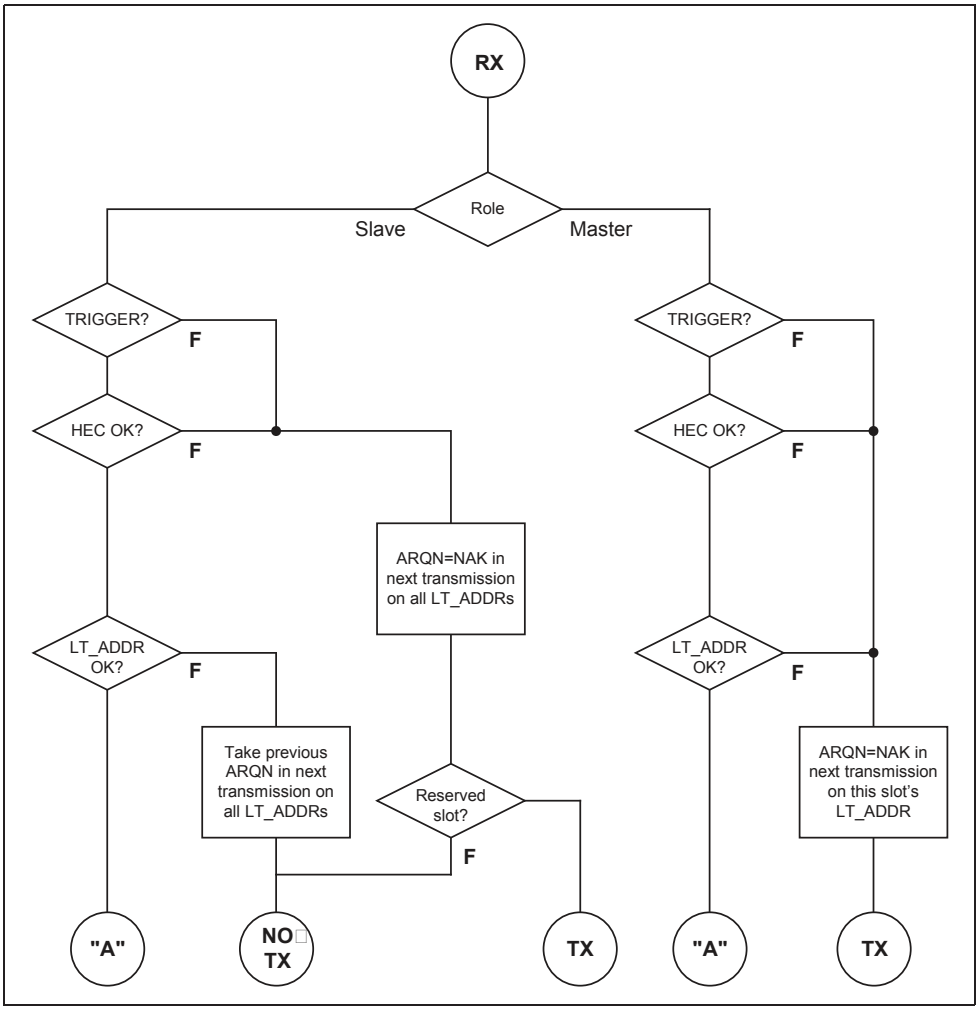



*Figure 7.12: Stage 1 of the receive protocol for determining the ARQN bit.*



LifeScan, Inc. Ex. 1019
Page 307 of 1420
IPR2025-00103

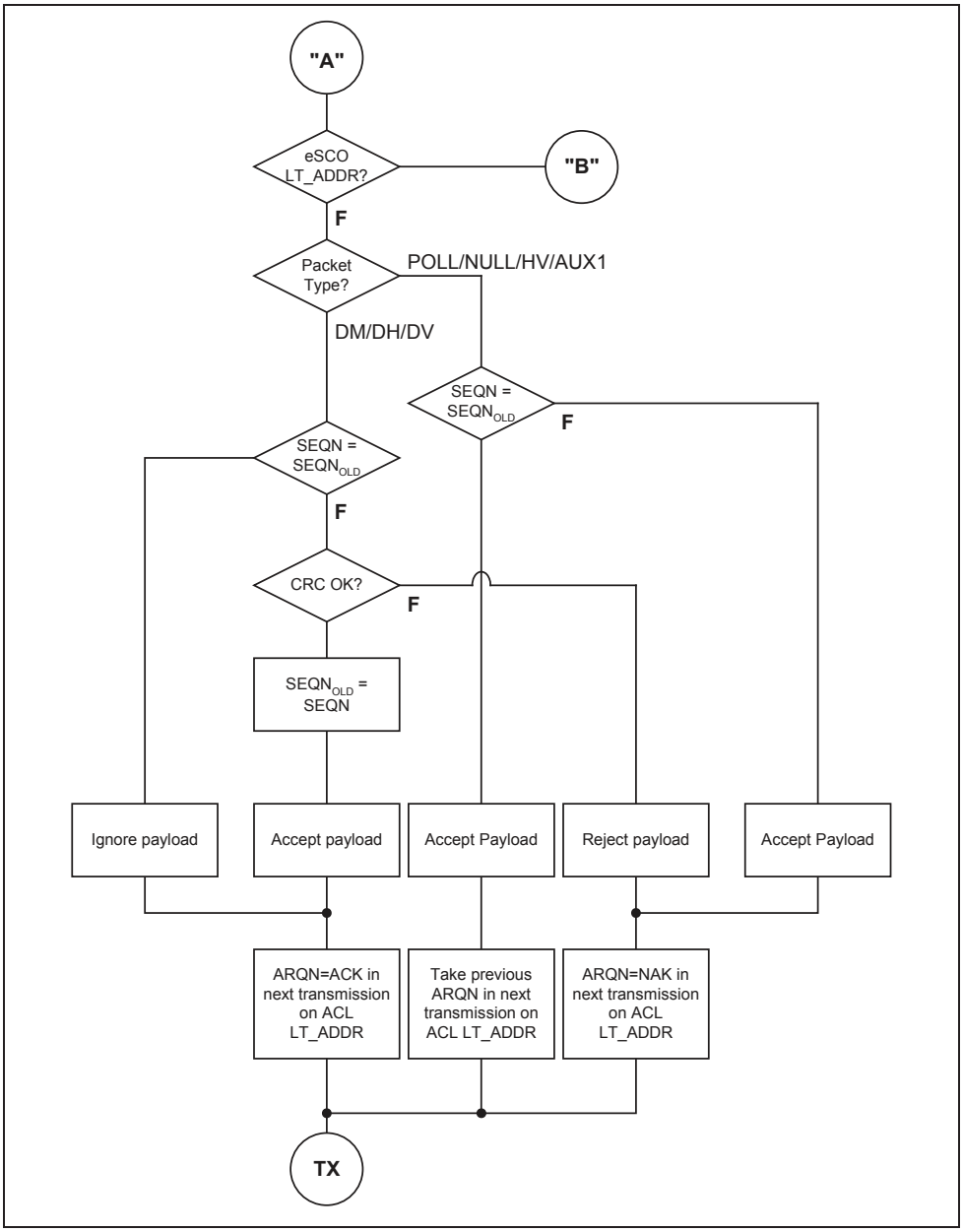



*Figure 7.13: Stage 2 (ACL) of the receive protocol for determining the ARQN bit.*



LifeScan, Inc. Ex. 1019
Page 308 of 1420
IPR2025-00103

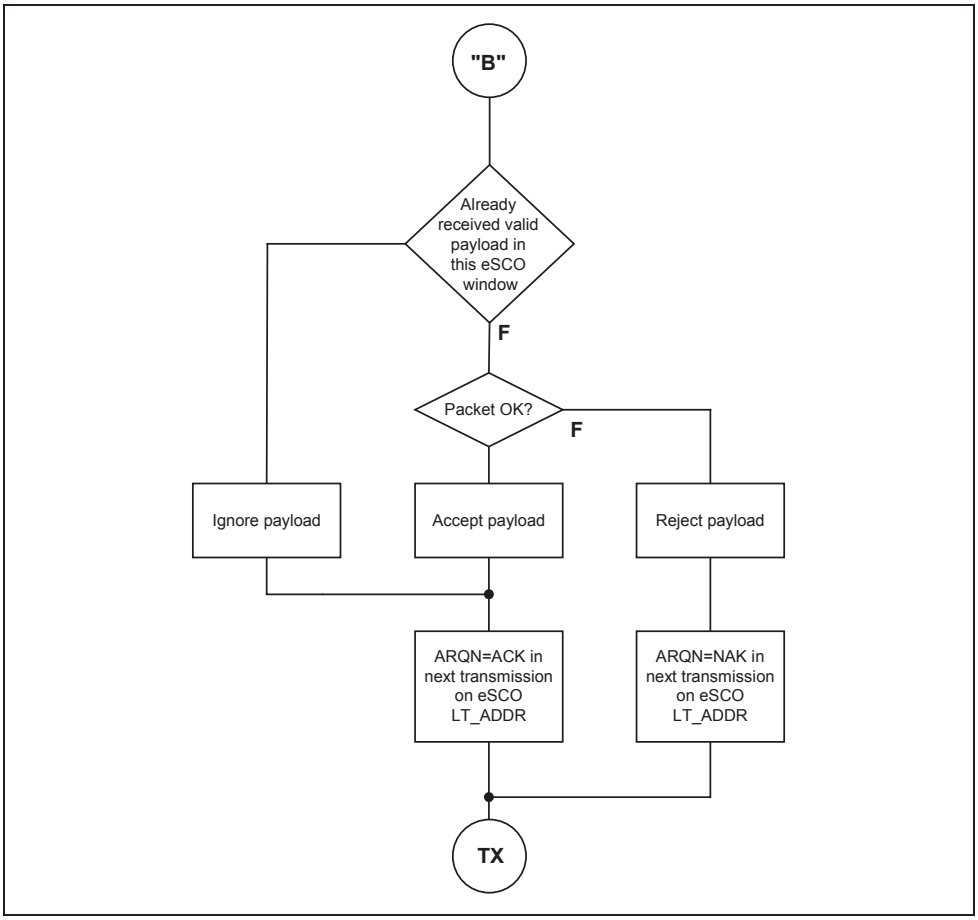



*Figure 7.14: Stage 2 (eSCO) of the receive protocol for determining the ARQN bit.*

### 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 Baseband Resource Manager. Note that CRC data payloads can be carried only by **DM**, **DH**, **DV** or **EV** packets.

#### *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, or 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 the following sections.

#### *7.6.2.2 ACL and SCO retransmit filtering*

The SEQN bit shall only be affected by the CRC data packets as shown in Figure 7.15. 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 Section 7.6.3 on page 142), and it has not received an acknowledgement for the current packet, it shall replace the current packet with an ACL-U continuation packet with the same sequence number 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.

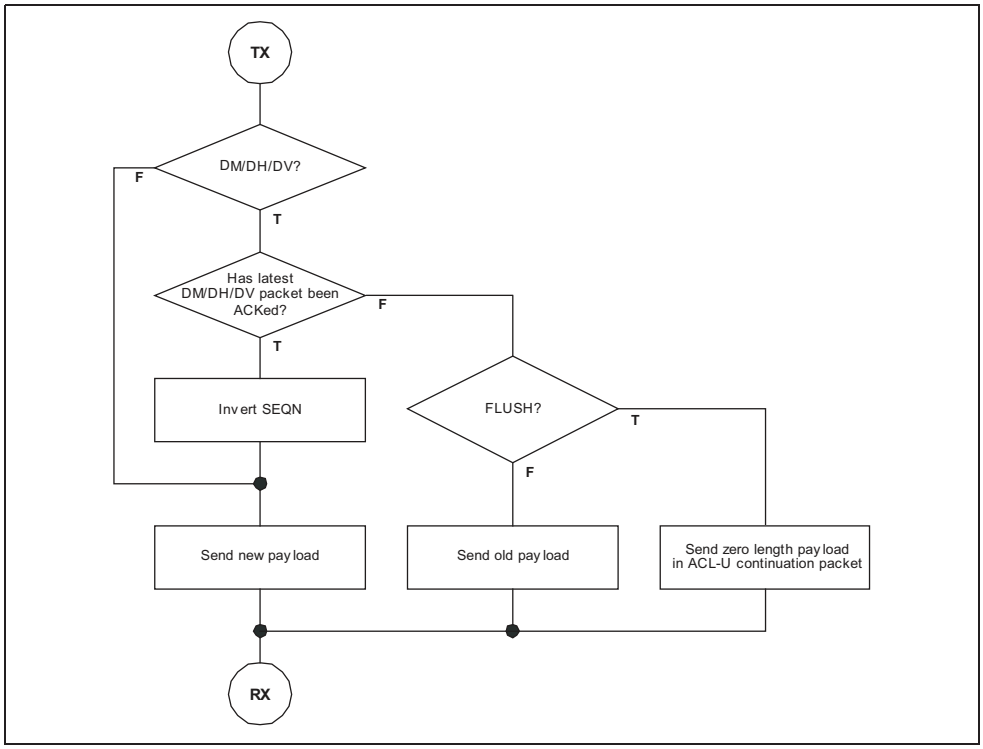



*Figure 7.15: Transmit filtering for packets with CRC.*

#### *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.

#### *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.

#### *7.6.2.5 Extended Inquiry Response retransmit filtering*

The SEQN bit in the extended inquiry response packet is not used. The bit shall be set to zero and ignored upon receipt.

#### *7.6.2.6 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.

### 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 Section 6.6). Flushing will not necessarily result in a change in the SEQN bit value, see the previous section.

The Flush Timeout defines a maximum period after which all segments of the ACL-U packet with a Packet_Boundary_Flag value of 10 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. If the First segment of an ACL-U packet has a Packet_Boundary_Flag value of 00, it is non-automatically-flushable and shall not cause the Flush Timeout to start. After the Flush timeout has expired the Link Controller may continue transmissions according to the procedure described in Section 7.6.2.2 on page 140, however the Baseband Resource Manager shall not continue the transmission of the ACL-U packet to the Link Controller. If the Baseband 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 a First segment is received. When the complete ACL-U packet has been flushed, the Link Manager shall continue transmission of the next ACL-U packet for the ACL logical transport. The default Flush Timeout shall be infinite, i.e. re-transmissions 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. Reliable data shall be sent over a channel with a finite flush timeout by marking reliable packets as non-automatically-flushable.

In eSCO, packets shall be automatically flushed at the end of the eSCO window.

### 7.6.4 Multi-slave considerations

In a piconet with multiple logical transports, the master shall carry out the ARQ protocol independently on each logical transport.



LifeScan, Inc. Ex. 1019
Page 312 of 1420
IPR2025-00103

### 7.6.5 Broadcast packets

Broadcast packets are packets transmitted by the master to all the slaves simultaneously (see Section 8.6.4) If multiple hop sequences are being used each transmission may only be received 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; the FHS packet and the extended inquiry response packet are the only packets which may have an all-zero LT_ADDR but are not broadcast packets). Broadcast packets shall not be acknowledged (at least not at the LC level).

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 7.16 on page 144. Optionally, a broadcast packet may be transmitted $N_{BC}$ + 1 times. Note: $N_{BC}$=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 Section 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 sequence number. The SEQN of the first broadcast packet with a CRC shall be set to SEQN = 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 sequence number. 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 acknowledgement of broadcast messages and there is no end packet indication, it is important to receive the start packets correctly. 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 Section 6.6 on page 121. Only repetitions of the L2CAP continuation packets shall be filtered out.

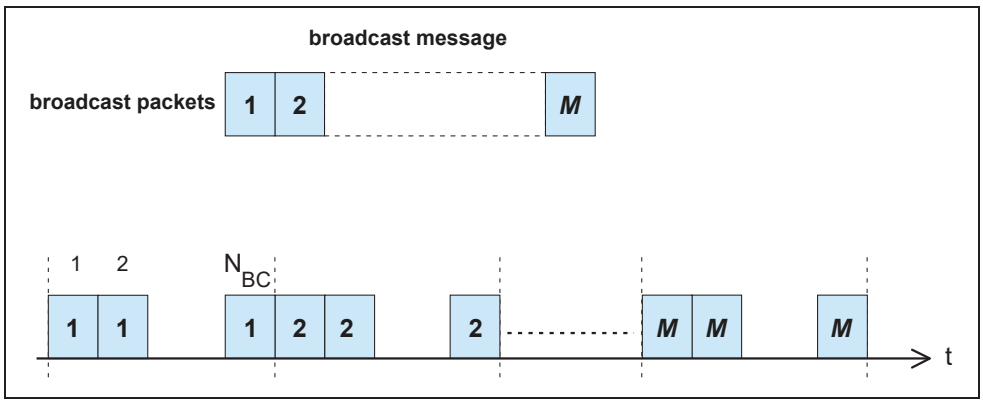



*Figure 7.16: Broadcast repetition scheme*

## 7.7 ERRONEOUS SYNCHRONOUS DATA REPORTING

Erroneous data reporting may be enabled for synchronous links. When enabled, synchronous data shall be processed as follows:

- If, during an (e)SCO interval, an eSCO packet was received with valid HEC and valid CRC or a SCO packet was received with valid HEC, the received payload data shall be sent to the upper-edge of the Controller with a "good data" indication.
- If, during an eSCO interval, eSCO packets with valid HEC were received, but none of them had a valid CRC, the best known data available (e.g. data derived from the received data or the actual payload data that was received with a CRC error) shall be sent to the upper-edge of the Controller with a "data with possible errors" indication.
- If, during an (e)SCO interval, no SCO or eSCO packet with valid HEC has been received, a "lost data" indication shall be sent to the upper-edge of the Controller.

# 8 LINK CONTROLLER OPERATION

This section 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.

## 8.1 OVERVIEW OF STATES

Figure 8.1 on page 145 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 link manager are used, or internal signals in the link controller are used (such as the trigger signal from the correlator and the timeout signals).

.

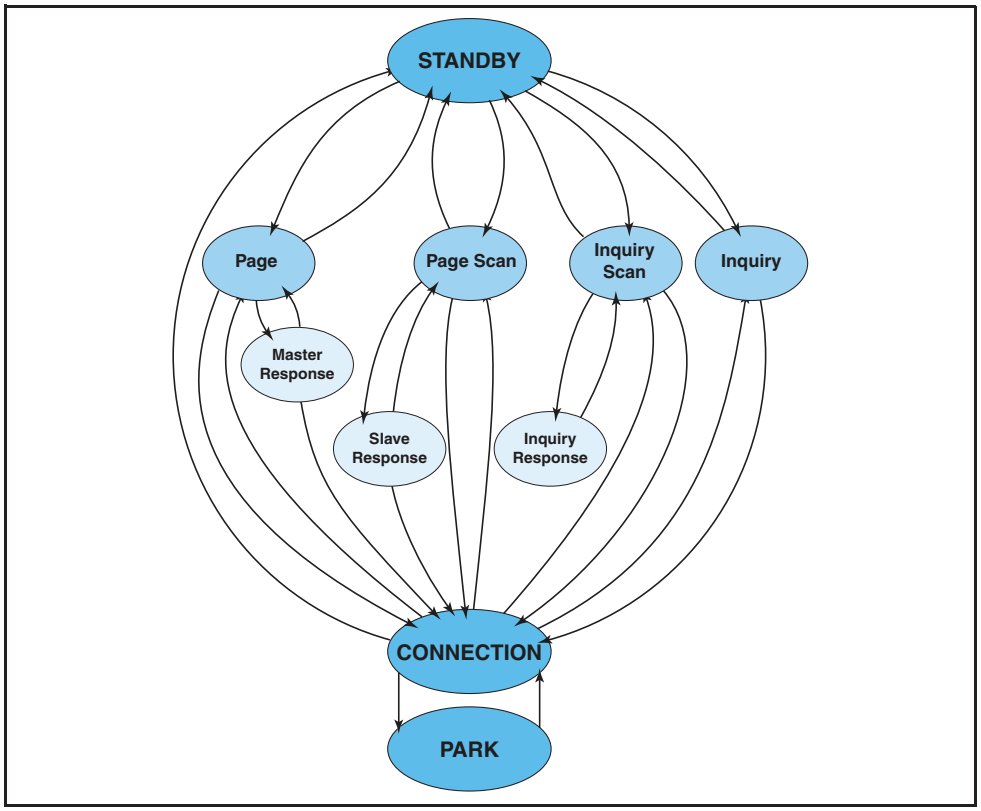



*Figure 8.1: State diagram of link controller.*



LifeScan, Inc. Ex. 1019
Page 315 of 1420
IPR2025-00103

## 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 native clock 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 page or inquiry itself.

## 8.3 CONNECTION ESTABLISHMENT SUBSTATES

In order to establish new connections the paging procedure is used. Only the Bluetooth device address is required to set up a connection. Knowledge about the clock, obtained from the inquiry procedure (see Section 8.4 on page 155) 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.

### 8.3.1 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 $T_{w\_page\_scan}$ (11.25ms default, see HCI [Part E] Section 7.3.20 on page 527), 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 device access code (DAC). The scan window shall be long enough to completely scan 16 page frequencies.

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 Bluetooth device address, see Section 2.6.4.1 on page 86. The phase in the sequence shall be determined by $CLKN_{16-12}$ of the device's native clock; that is, every 1.28s 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 Section 8.3.3.1 on page 152. 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 Spars(79), see Section 2.6.4.3 on page 87 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, Park, or Sniff (see Section 8.8 on page 178 and Section 8.9 on page 179). 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 should be used requiring the least amount of capacity (**HV3** packets). 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 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 (22.5ms) is recommended; if two SCO links are present using **HV3** packets and $T_{SCO}$=6 slots or two eSCO links are present using **EV3** packets and $T_{ESCO}$=6 slots, a total scan window of at least 54 slots (33.75ms) 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 1.28s, or the scan interval is maximal 2.56s. These three cases shall determine the behavior of the paging device; that is, whether the paging device shall use R0, R1 or R2, see also Section 8.3.2 on page 148. Table 8.1 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.

| SR mode | $T_{page\ scan}$ |
|---|---|
| R0 | ≤ 1.28s and = $T_{w\ page\ scan}$ |
| R1 | ≤ 1.28s |
| R2 | ≤ 2.56s |
| Reserved | - |

*Table 8.1: Relationship between scan interval, and paging modes R0, R1, and R2.*



LifeScan, Inc. Ex. 1019
Page 317 of 1420
IPR2025-00103

### 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 device access code (DAC) in different hop channels. Since the Bluetooth 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 frequency. Therefore, it transmits a train of identical page scan messages at different hop frequencies and listens in between the transmit intervals until it receives a response from the slave.

The page procedure in the master consists of a number of steps. First, the Host 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 Section 2.6.4.2 on page 87. 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 estimate CLKE of the slave's Bluetooth clock, the master can predict on which hop channel the slave starts page scanning.

The estimate of the Bluetooth 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 Section 2.2.5, see also Figure 2.7 on page 72. In the next TX slot, the master shall transmit on two hop frequencies different from the former ones. Note: The hop rate is increased to 3200 hops/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),...,f(k),...,f(k+7)

When the difference between the Bluetooth clocks of the master and the slave is between -8x1.28 s and +7x1.28 s, one of the frequencies used by the master will be the hop frequency the slave will listen to. 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 Npage >= 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 Bluetooth clocks of the master and the slave is less than -8x1.28 s or larger than +7x1.28 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),...,f(k-9),f(k+8)...,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 on hold or park. However, the synchronous connections shall not be disturbed by the page. This means that the 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 which 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 8.2. Here it has been assumed that the **HV3** packet 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.



LifeScan, Inc. Ex. 1019
Page 319 of 1420
IPR2025-00103

.

| 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 | $N_{page} \geq 256$ | $N_{page} \geq 512$ | $N_{page} \geq 768$ |

*Table 8.2: Relationship between train repetition, and paging modes R0, R1 and R2 when synchronous links are present.*

The construction of the page train shall be independent of the presence of synchronous links; that is, synchronous packets are sent on the reserved slots but shall not affect the hop frequencies used in the unreserved slots, see Figure 8.2 on page 150.

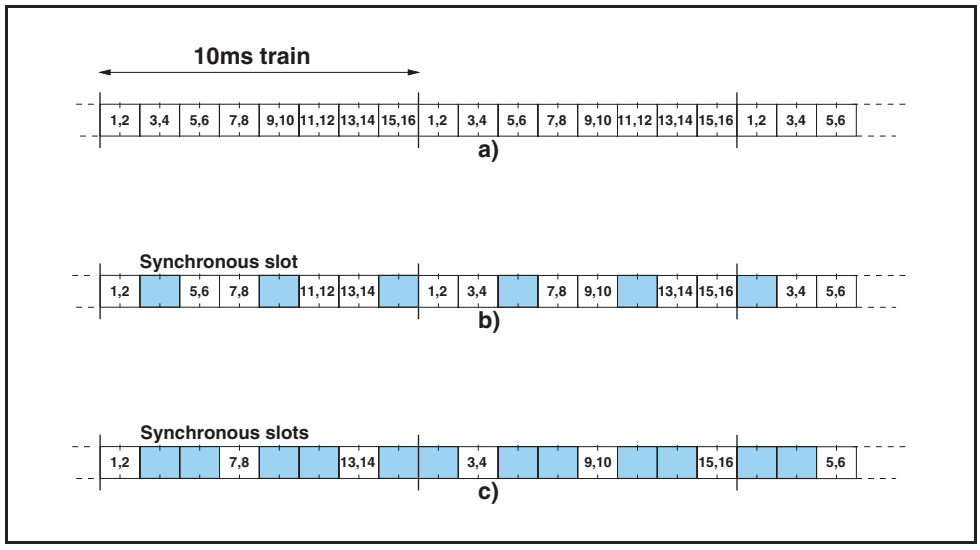



*Figure 8.2: Conventional page (a), page while one synchronous link present (b), page while two synchronous links present (c).*



LifeScan, Inc. Ex. 1019
Page 320 of 1420
IPR2025-00103

### 8.3.3 Page response substates

When a page message is successfully received by the slave, there is a coarse FH 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 channel access code, use the same channel hopping sequence, and 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 only valid during the time the piconet exists). The channel access code and channel hopping sequence shall be derived from the Bluetooth device address (BD_ADDR) of the master. The timing shall be determined by the master clock. An offset shall be added to the slave's native clock to temporarily synchronize the slave clock to the master clock. At start-up, the master parameters are transmitted from the master to the slave. The messaging between the master and the slave at start-up is specified in this section.

The initial messaging between master and slave is shown in Table 8.3 on page 151 and in Figure 8.3 on page 152 and Figure 8.4 on page 152. 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.

| 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 |

*Table 8.3: Initial messaging during start-up.*

In step 1 (see Table 8.3 on page 151), 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** 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** in step 3. Note that during the initial message exchange, all parame-



LifeScan, Inc. Ex. 1019
Page 321 of 1420
IPR2025-00103

ters are derived from the slave's device address, and that only the page hopping and page response hopping sequences are used (are also derived from the slave's device address). Note that when the master and slave enter the response states, their clock input to the page and page response hop selection is frozen as is described in Section 2.6.4.3 on page 87.

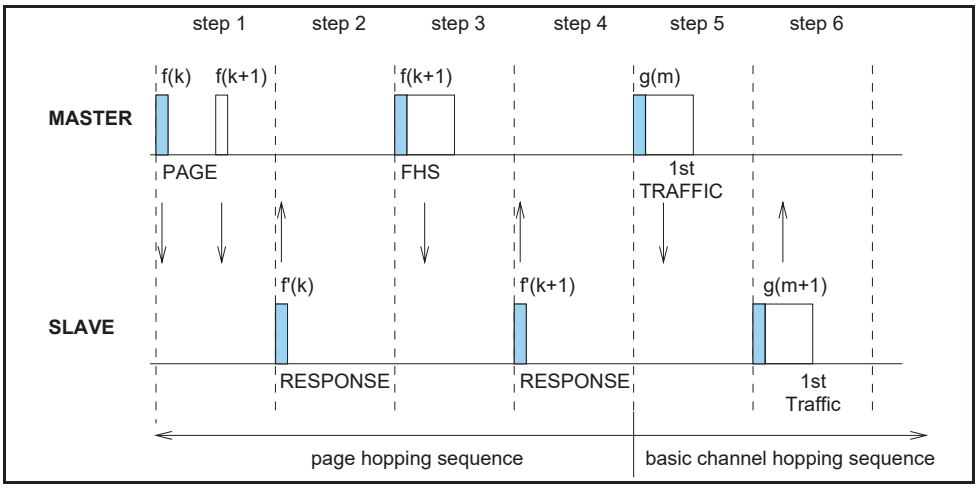



*Figure 8.3: Messaging at initial connection when slave responds to first page message.*

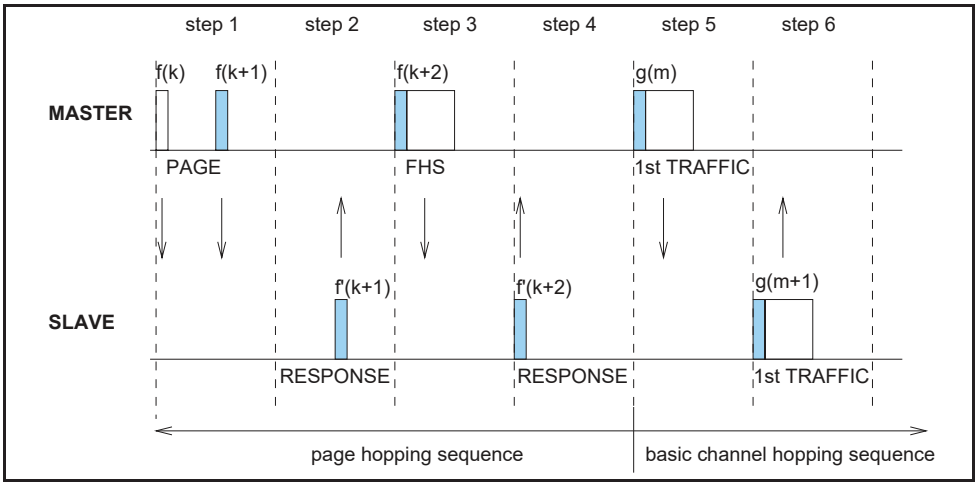



*Figure 8.4: Messaging at initial connection when slave responds to second page message.*

### *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 device access code) in step 2. This response message shall be the slave's device access code. 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 fre-

quency 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 8.4 on page 152, 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 Section 2.4.4 on page 73.

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 channel access code and clock as received from the **FHS** packet. Only the 26 MSBs of the master clock 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 Baseband 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 channel access code. The slave 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 acknowledgement, the master and the slave shall return to **page** and **page scan** substates, respectively. See Section 8.5 on page 160

#### 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 real-time Bluetooth clock, the master's BD_ADDR, the BCH parity bits, and the class of device. The **FHS** packet contains all information to construct the channel access code without requiring a mathematical derivation from the master's Bluetooth 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** 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 like shown in Figure 8.4 on page 152 and not 625 μs after the received packet as is usual in the piconet physical channel RX/TX timing, see also Section 2.4.4 on page 73.

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 device access code. 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 Baseband 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 channel access code and the master clock. The lower clock bits $CLK_0$ and $CLK_1$ shall be reset to zero at the start of the **FHS** packet transmission and 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 Section 2.6.4.7 on page 89. 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 Section 8.5 on page 160. This packet shall be sent within *newconnectionTO* number of slots after reception of the FHS packet acknowledgement. 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.

## 8.4 DEVICE DISCOVERY SUBSTATES

In order to discover other devices a device shall enter **inquiry** substate. In this substate, it shall repeatedly transmit the inquiry message (ID packet, see Section 6.5.1.1 on page 112) 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, shall regularly enter the **inquiry scan** substate to respond to inquiry messages. The following sections describe the mes sage exchange and contention resolution during inquiry response. 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 Bluetooth device addresses and clocks of all devices that respond to the inquiry message. In addition, the discovering device also collects extended information (e.g. local name and supported services) from devices that respond with an extended inquiry response packet. It can then, if desired, make a connection to any one of the discovered devices by means of the previously described page procedure.

The inquiry message broadcast by the source does not contain any information about the source. However, it may indicate which class of devices should respond. There is one general inquiry access code (GIAC) to inquire for any device, and a number of dedicated inquiry access codes (DIAC) that only inquire for a certain type of device. The inquiry access codes are derived from reserved Bluetooth device addresses and are further described in Section 6.3.1.



LifeScan, Inc. Ex. 1019
Page 325 of 1420
IPR2025-00103

### 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 device access code, the receiver shall scan for the inquiry access code 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 $T_{w\_inquiry\_scan}$ (11.25ms default, see HCI [Part E] Section 7.3.22 on page 529). The standard scan is performed at a single hop frequency as defined by $Xir_{4-0}$ (see Section 2.6.4.6 on page 89). 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] mod 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 the native clock of the device carrying out the **inquiry scan**; the phase changes every 1.28s.

Instead of, or in addition to, the general inquiry access code, the device may scan for one or more dedicated inquiry access codes. 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, Hold, or Park. 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 should be used requiring the least amount of capacity (**HV3** packets). 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 one eSCO logical transport is present using EV3 packets and $T_{ESCO}$=6 slots, a total scan window of at least 36 slots (22.5ms) is recommended; if two SCO links are present using HV3 packets and $T_{SCO}$=6 slots or two eSCO links are present using EV3 packets and $T_{ESCO}$=6 slots, a total scan window of at least 54 slots (33.75ms) 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.



LifeScan, Inc. Ex. 1019
Page 326 of 1420
IPR2025-00103

### 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 Section 2.4.4 on page 73 and Figure 2.7 on page 72. The TX and RX frequencies shall follow the inquiry hopping sequence and the inquiry response hopping sequence, and are determined by the general inquiry access code and the native clock of the discovering device. In between inquiry transmissions, the receiver shall scan for inquiry response messages. When a response is received, the entire packet (an FHS packet) shall be read. If the EIR bit in the FHS packet is set to one, the device should try to receive the extended inquiry response packet 1250μs after the start of the FHS packet. After this, 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 HCI [Part E] Section 7.3.50 on page 563), the RSSI value of the inquiry response message shall be measured. It shall keep probing at different hop channels and in 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. 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 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 (say 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 Baseband 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, Hold, or Park. 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 which 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 8.4 on page 158.



LifeScan, Inc. Ex. 1019
Page 327 of 1420
IPR2025-00103

Here it has been assumed that **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.

| | No synchronous links | One synchronous link (HV3) | Two synchronous links (HV3) |
|---|---|---|---|
| $N_{inquiry}$ | ≥ 256 | ≥ 512 | ≥ 768 |

*Table 8.4: Increase of train repetition when synchronous links are present.*

If an extended inquiry response packet could not be received because of higher priority traffic, the reception failed due to HEC or CRC failure, or because the packet type is not supported by the device, the inquiry response shall be reported to higher layers as an extended inquiry response with all-zero data.

### 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. If the recipient has non-zero extended inquiry response data to send it shall return an extended inquiry response packet after the FHS packet.

The following protocol in the slave's **inquiry response** shall be used. On the first inquiry message received in the inquiry scan substate the slave shall enter the inquiry response substate. If the slave has non-zero extended inquiry response data to send it shall return an FHS packet with the EIR bit set to one to the master 625μs after the inquiry message was received. It shall then return an extended inquiry response packet 1250μs after the start of the FHS packet. If the slave's extended inquiry response data is all zeroes the slave shall only return an FHS packet with the EIR bit set to zero. 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 ≥ 1.28s MAX_RAND shall be 1023, however, for scanning intervals < 1.28s 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 ≥ 1.28s. 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



LifeScan, Inc. Ex. 1019
Page 328 of 1420
IPR2025-00103

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 RAND. 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 different frequencies and at different times. Reserved synchronous slots should have priority over response packets; that is, if a response packet overlaps with a reserved synchronous slot, it shall not be sent but the next inquiry message is awaited. If a device has extended inquiry response data to send but the extended inquiry response packet overlaps with a reserved synchronous slot the FHS packet may be sent with the EIR bit set to zero.

The messaging during the inquiry routines is summarized in Table 8.5 on page 160. In step 1, the master transmits an inquiry message using the inquiry access code and its own clock. The slave responds with the **FHS** packet containing the slave's Bluetooth device address, native clock 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. If the slave has non-zero extended inquiry response data it sends an extended inquiry response packet to the master in step 3.

The extended inquiry response packet is an ACL packet with type DM1, DM3, DM5, DH1, DH3 or DH5. To minimize interference it is recommended to use the shortest packet that fits the data. The packet shall be sent on the same frequency as the FHS packet, 1250 µs after the start of the FHS packet. In the packet header, LT_ADDR shall be set to zero. TYPE shall be one of DM1, DM3, DM5, DH1, DH3 or DH5. FLOW, ARQN and SEQN shall all be set to zero and ignored during receipt. The HEC LFSR shall be initialized with the same DCI (default check initialization) as for the FHS packet. In the payload header, LLID shall contain the value 10 (start of an L2CAP message or no fragmentation). FLOW shall be set to zero and ignored upon receipt. The length of the payload body (LENGTH) shall be smaller than or equal to 240 bytes. The CRC LFSR shall be initialized with the same DCI as for the FHS packet. The data whitening LFSR shall be initialized with the same value as for the FHS packet.

The payload data has two parts, a significant part followed by a non-significant part. The significant part contains a sequence of data structures as defined in [Vol. 3, Part C] Section 8 on page 229. The non-significant part contains all-zero octets.

The baseband shall not change any octets in the significant part. When transmitting data, the non-significant part octets may be omitted from the payload.

A device shall store a single extended inquiry response packet. This packet shall be used with all IACs.



LifeScan, Inc. Ex. 1019
Page 329 of 1420
IPR2025-00103

| 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 |
| 3 | Extended Inquiry Response | DM1, DM3, DM5, DH1, DH3, DH5 | slave to master | inquiry response * | inquiry |

*Table 8.5: Messaging during inquiry routines.*

* The extended inquiry response packet is sent on the same frequency as the inquiry response (FHS) packet.

## 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 channel (master) access code, the master's Bluetooth clock, and the AFH_channel_map are used. **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 link managers 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 [Part E] Section 7.3.2 on page 502.

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 8.5 on page 161).



LifeScan, Inc. Ex. 1019
Page 330 of 1420
IPR2025-00103

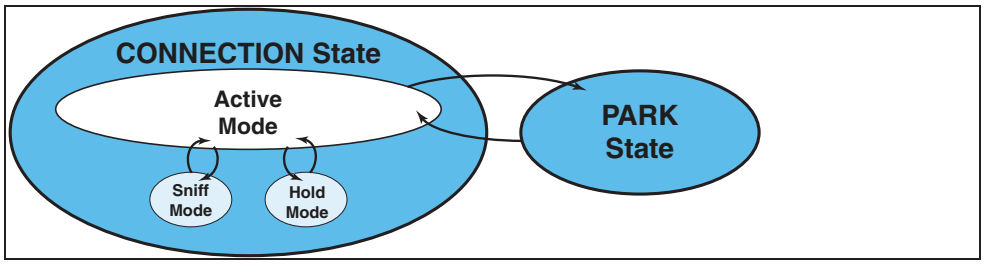



*Figure 8.5: Connection state.*

## 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 transmission 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 non-addressed 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 only need the channel access code 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 only respond 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 Section 8.9 on page 179).



LifeScan, Inc. Ex. 1019
Page 331 of 1420
IPR2025-00103

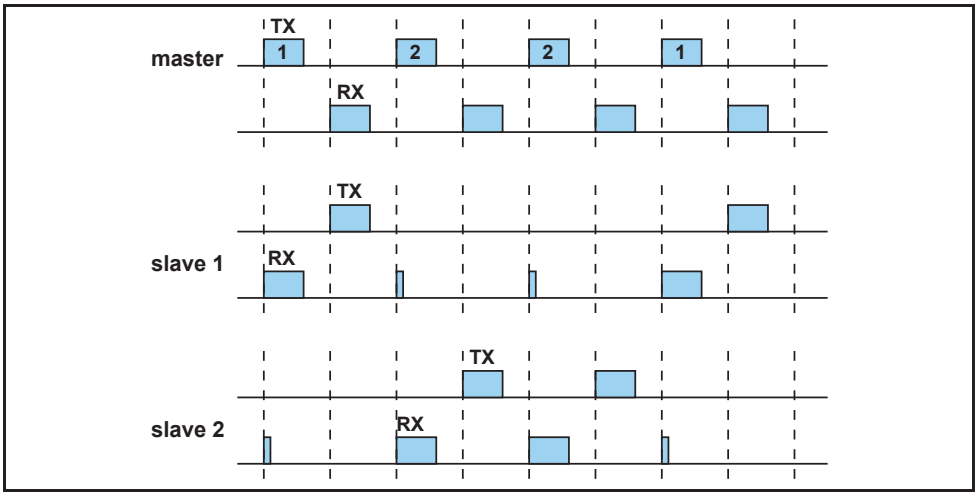



*Figure 8.6: RX/TX timing in multi-slave configuration*

For ACL logical transports the mode selection may be left to real time packet type selections. The packet type table (ptt) in section 6.5 allows the selection of Basic Rate or Enhanced Data Rate for each of the packet type codes, however; the DM1 packet is available in all packet type tables. ACL traffic over this given physical or logical link shall utilize the packet types in the given column of Packets defined for synchronous and asynchronous logical transport types.

### 8.6.1 Polling in the active mode

The master always has full control over the piconet. Due to the TDD scheme, slaves can only communicate with the master and not other slaves. In order to avoid collisions on the ACL logical transport, a slave is only allowed to transmit in the slave-to-master slot 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.

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 Link Manager (see [Part C] Section 4.1.8 on page 234).

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 Section 8.6.2 on page 162).

### 8.6.2 SCO

The SCO logical transport shall be established by the master sending an SCO setup message via the LM protocol. This message contains timing parameters



LifeScan, Inc. Ex. 1019
Page 332 of 1420
IPR2025-00103

including the SCO interval $T_{SCO}$ and the offset $D_{SCO}$ to specify the reserved slots.

In order to prevent clock wrap-around 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 1.25ms 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 2.5ms 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 carry 3.75ms of speech at a 64 Kobe/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 preceding slot. DM1 packets sent during SCO reserved slots shall only be used 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.

### 8.6.3 eSCO

The eSCO logical transport is established by the master sending an eSCO setup message via the LM protocol. 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 LM protocol. This message shall contain the eSCO interval $T_{ESCO}$ and an offset $D_{ESCO}$. In order to prevent clock wrap-around problems, an initialization flag in the LMP setup message indicates whether initialization procedure 1 or 2 shall be used. The initiating 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 responding device shall apply the initialization method as indicated by the initialization flag. 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:

$$CLK_{27-1} \bmod T_{ESCO} = D_{ESCO} \qquad \text{for initialization 1}$$

$$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{ESCO} = D_{ESCO} \qquad \text{for initialization 2}$$

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 to 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 preceding 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 only transmit on an eSCO LT_ADDR in the retransmission window 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 is 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 ACK'ed, the master shall use the POLL packet. When the master-to-slave packet has been ACK'ed, 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 is 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 ACK'ed, the slave shall use NULL packets. eSCO traffic should be given priority over ACL traffic in the retransmission window.

Figure 8.7 on page 165 shows the eSCO window when single slot packets are used.

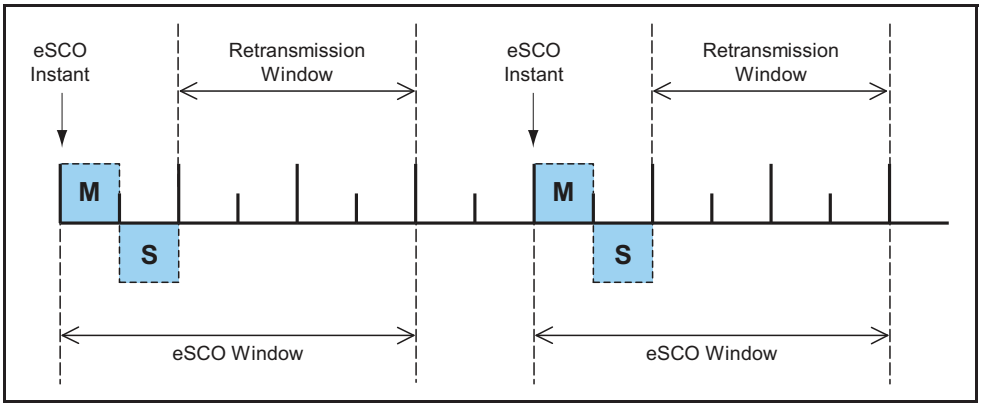



*Figure 8.7: eSCO Window Details for Single-Slot Packets*

When multi-slot 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 8.8 on page 166 shows the eSCO window when multi-slot packets are used in one direction and single-slot packets are used in the other direction.



LifeScan, Inc. Ex. 1019
Page 335 of 1420
IPR2025-00103

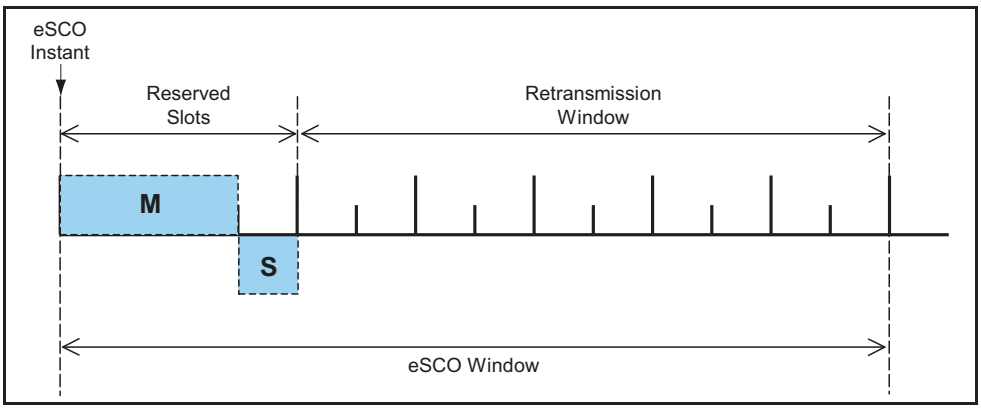



*Figure 8.8: eSCO Window Details for Asymmetric Traffic*

eSCO windows may overlap on the master, but shall not overlap on an individual slave.

In the reserved slots both master and slave shall only 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.

### 8.6.4 Broadcast scheme

The master of the piconet can broadcast messages to all slaves on the ASB-U, PSB-C, and PSB-U logical transports. 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).

The Broadcast LT_ADDR shall use a ptt=0.

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 Section 7.6.5 on page 143.



LifeScan, Inc. Ex. 1019
Page 336 of 1420
IPR2025-00103

In order to support the **PARK** state (as described in Section 8.9 on page 179), 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 Section 8.9 on page 179.

### 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 only involving 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 paused or disabled in the old piconet. A role switch shall not be performed if the physical link is in Sniff or Hold mode, in the **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 Bluetooth 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 Link Manager Protocol, [Part C] Section 4.4.2 on page 275. The baseband procedure is the same regardless of which alternative is used.

In step 1, the slave A and master B shall perform a TDD switch using the former hopping scheme (still using the Bluetooth device address and clock of device B), so there is no piconet switch yet. The slot offset information sent by slave 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 slave B.

At the moment of the TDD switch, both devices A and B shall start a timer with a time out of *newconnectionTO*. The timer shall be stopped in slave B as soon as it receives an FHS packet from master A on the TDD-switched channel. The timer shall be stopped in master A as soon as it receives an ID packet from slave B. 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 master A using the "old" piconet parameters.



LifeScan, Inc. Ex. 1019
Page 337 of 1420
IPR2025-00103

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 the slave on the old hopping sequence, both master A and slave B shall use the new channel parameters of the new piconet as indicated by the 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 beginning 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 channels. This timing information ranges from 0 to 1249 μs with a resolution of 1μs. 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, the new master A 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 time out 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 re-send 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), Channel Quality Driven Data Rate (CQDDR, Link Manager Protocol [Part C] Section 4.1.7 on page 233) and power control (Link Manager Protocol [Part C] Section 4.1.3 on page 225). 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 channel shall be set according to the rules in Section 7.6.2 on page 139.

A parked slave must be unparked before it can participate in a role switch.

### 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 channel access codes 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 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 only as a master in 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 this 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 Section 8.6.5 on page 167.

#### *8.6.6.1 Inter-piconet 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, it may choose not to listen in every master slot. In this case it should be recognized that the quality of service 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 it may choose not to 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 in 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 only be visited in the non-reserved slots in between reserved slots. This is only possible if there is a single SCO logical transport using HV3 packets or eSCO links where at least four slots remain in between the reserved slots. Since the multiple piconets are not synchronized, guard time must be left to account for misalignment. This means that only 2 slots can effectively be used to visit another piconet in between the HV3 packets.

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 native clock, create the two master clocks. Since the two master clocks drift independently, the slave must regularly update the offsets in order to keep synchronization to both masters.

### 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 *AFH_channel_map* input 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 *AFH_channel_map* input 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 baseband acknowledgement is received. When the baseband acknowledgement is received prior to the hop sequence switch instant, *AFH_Instant*, (See Link Manager Protocol [Part C] Section 4.1.4 on page 227) the hop sequence proceeds as shown in Figure 8.9 on page 170.

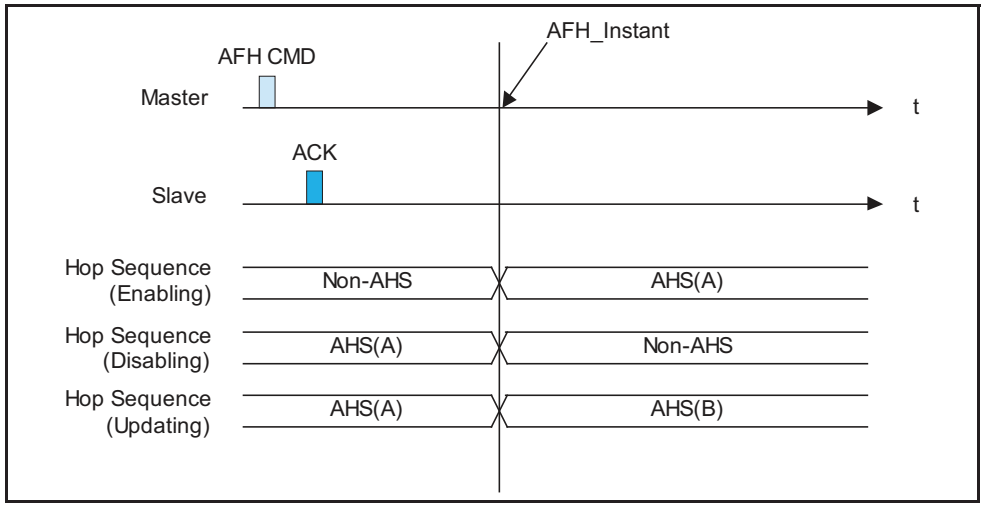



*Figure 8.9: Successful hop sequence switching*

When the baseband acknowledgement 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 acknowledgement (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 baseband acknowledgement 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 8.10 on page 171.

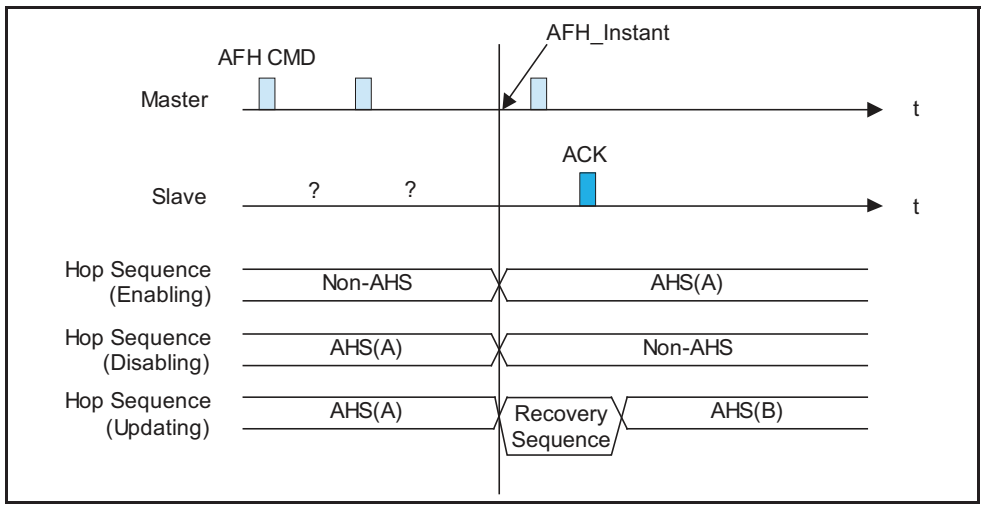



*Figure 8.10: Recovery hop sequences*

When the *AFH_Instant* occurs during a multi-slot packet transmitted by the master, the slave shall use the same hopping sequence parameters as the master used at the start of the multi-slot packet. An example of this is shown in Figure 8.11 on page 172. 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''.



LifeScan, Inc. Ex. 1019
Page 341 of 1420
IPR2025-00103

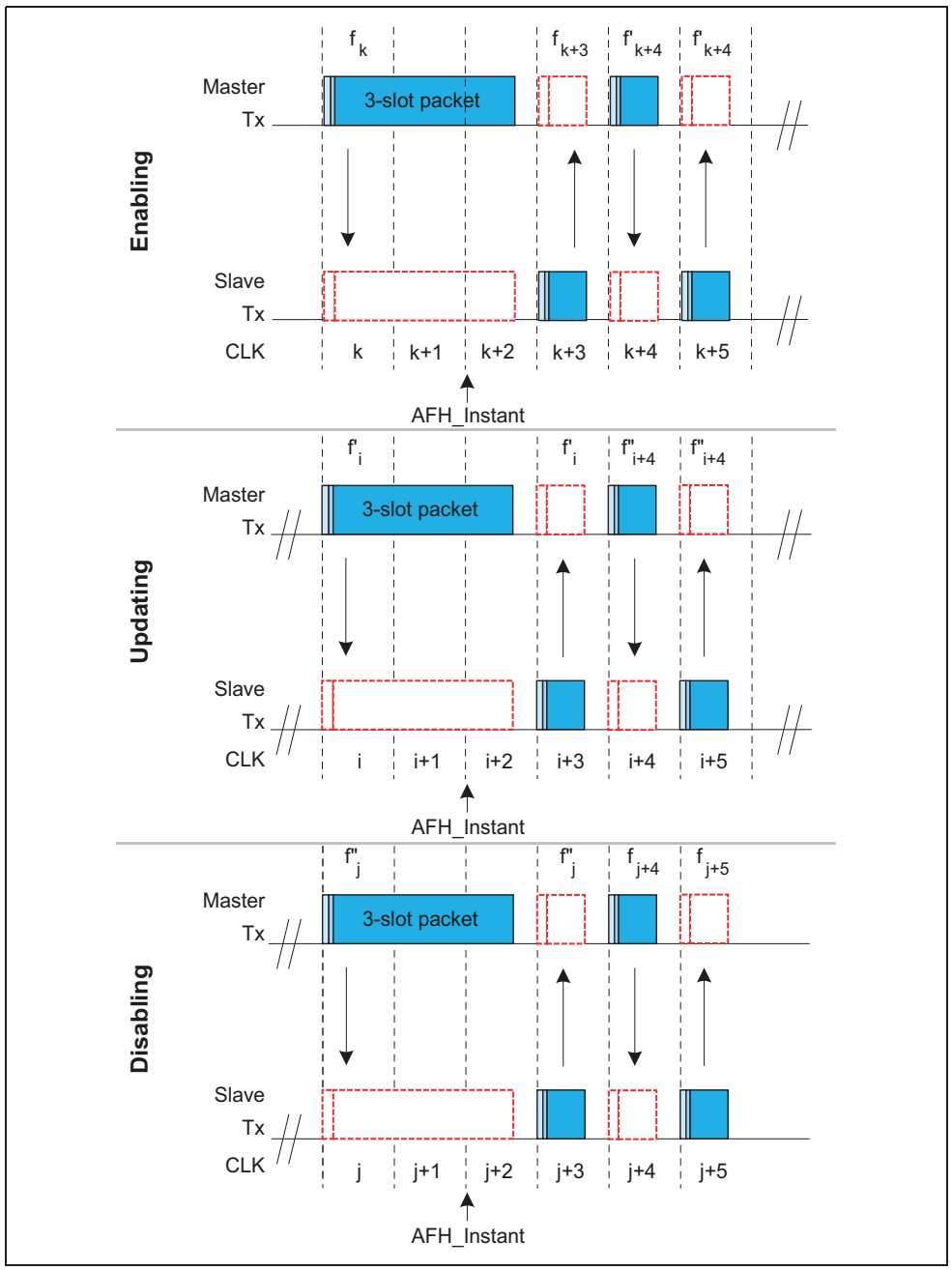



*Figure 8.11: AFH_Instant changes during multi-slot packets transmitted by the master.*



LifeScan, Inc. Ex. 1019
Page 342 of 1420
IPR2025-00103

### 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 (for example, 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 8.6 on page 173.

| Classification | Definition |
|---|---|
| *unknown* | A channel shall be classified as *unknown* if 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*. |
| *bad* | A channel may be classified as *bad* if an ACL or synchronous throughput failure measure associated with it has exceeded a threshold (defined by the particular channel assessment algorithm employed).<br>A channel may also be classified as *bad* if an interference-level measure associated with it, 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).<br>A channel shall be classified as *bad* when it is marked as *bad* in the most recent HCI *Set_AFH_Channel_Classification* |
| *good* | A channel shall be classified as *good* if it is not either *unknown* or *bad*. |

*Table 8.6: Channel classification descriptions*

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 the HCI Functional Specification [Part E] Section 7.3.54 on page 567 if HCI is present.
- Channel classification information from the Host using the HCI *Set_AFH_channel_classification* command, defined in the HCI Functional Specification [Part E] Section 7.3.54 on page 567 if HCI is present. Channels classified as *bad* in the most recent *AFH_Host_Channel_Classification* shall be marked as *unused* in the *AFH_channel_map*.
- Channel classification reports received from slaves in *LMP_channel_classification* PDUs, defined in the LMP Specification [Part C] Section 4.1.5 on page 230.



LifeScan, Inc. Ex. 1019
Page 343 of 1420
IPR2025-00103

The algorithm used by the master to combine these information sources and generate the *AFH_channel_map* is not defined in the specification and will be implementation specific. At no time shall the number of channels used be less than $N_{min}$, defined in Section 2.3.1 on page 70.

If a master that 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).

### 8.6.9 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.

#### *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 only sending 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 only involve a NAK (NAK is implicit on no reply). If there is 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.

#### *8.6.9.2 Slot occupancy*

As was described in Section 6.5 on page 111, 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.

#### *8.6.9.3 Recommendations for low-power operation*

The most common and flexible methods for reducing power consumption are the use of sniff and park. Hold can also be used by repeated negotiation of hold periods.



LifeScan, Inc. Ex. 1019
Page 344 of 1420
IPR2025-00103

#### *8.6.9.4 Enhanced Data Rate*

Enhanced Data Rate provides power saving because of the ability to send a given amount of data in either fewer packets or with the same (or similar) number of packets but with shorter payloads.

## 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 ACL slot to the master traffic, unless that link is being treated as a scatternet link or is absent due to hold mode. With sniff mode, the time slots when a slave is listening are reduced, so the master shall only transmit to a slave in specified time slots. The sniff anchor points are spaced regularly with an interval of $T_{sniff}$.

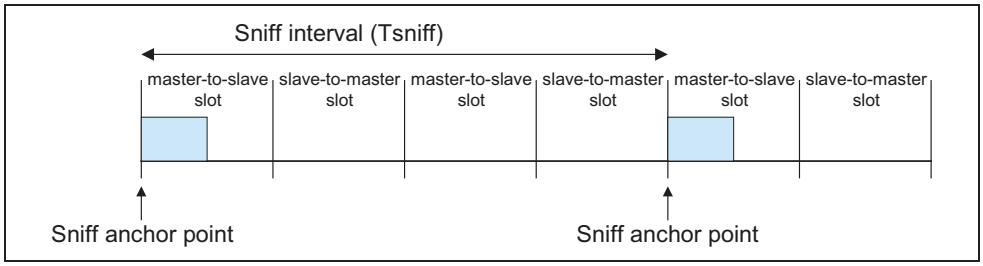



*Figure 8.12: Sniff anchor points*

The slave listens in master-to-slave transmission slots starting at the sniff anchor point. It shall use the following rules to determine whether to continue listening:

- If fewer than $N_{sniff\ attempt}$ master-to-slave transmission slots have elapsed since the sniff anchor point then the slave shall continue listening.
- 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 then it shall continue listening.
- 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 then it shall 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 then it may continue listening.
- 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 only applies 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 LM protocol. 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:

$$CLK(k+1) = CLK(k) + T_{sniff}$$

### 8.7.1 Sniff Transition Mode

Sniff transition mode is a special mode which is used during the transition between sniff and active mode. It is required because during this transition it is unclear which mode (Sniff or Active) the slave is in and it is necessary to ensure that the slave is polled correctly regardless of which mode it is in.

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.

The precise circumstances under which the master enters Sniff Transition Mode are defined in [Part C] Section 4.5.3.1 on page 287.

Only LMP data will ever be transferred in sniff transition mode. The ACL-C logical link shall carry control information exchanged between the link managers 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.

### 8.7.2 Sniff Subrating

When sniff subrating is enabled by the Link Manager a device alternates between sniff mode and sniff subrating mode. Sniff subrating mode allows a device to use a reduced number of sniff anchor points. A device shall transition from sniff mode to sniff subrating mode based on the sniff mode timeout value (see Section 8.7.2.1 on page 177). A device shall transition from sniff subrating mode to sniff mode whenever ACL-U or ACL-C data is received from the remote device.

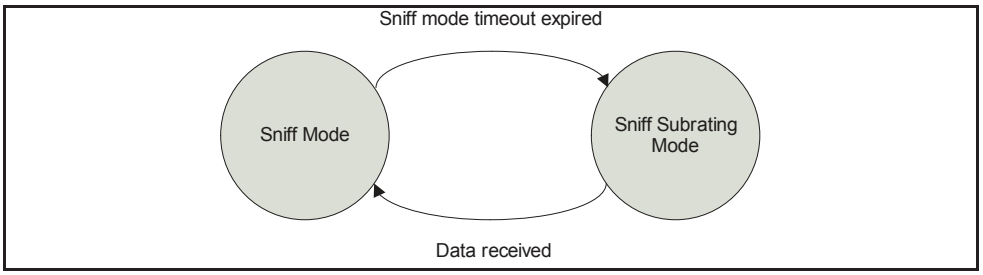



*Figure 8.13: Sniff subrating*

A slave device in sniff subrating mode shall also temporarily enter sniff mode after transmitting a packet requiring acknowledgement until the baseband acknowledgement is received.

Note: Once sniff subrating is enabled the master and slave devices may be in sniff mode or sniff subrating mode at different times. The rules defined in Section 8.7.2.2 on page 178 describe how the two devices maintain synchronization and can reliably exchange data.

#### *8.7.2.1 Sniff Mode Timeout*

Whenever a packet containing ACL-C or ACL-U data is received by a device in sniff subrating mode it shall transition to sniff mode, re-start the sniff mode timeout and all sniff anchor points shall be used until a minimum number of slots equal to the received min_sniff_mode_timeout have expired. If ACL-C or ACL-U data is received before min_sniff_mode_timeout slots have passed since the last ACL-C or ACL-U data was received, the timeout shall be restarted. NULL and POLL packets do not contain ACL data and shall not restart the sniff mode timeout.



LifeScan, Inc. Ex. 1019
Page 347 of 1420
IPR2025-00103

Note that there are two sniff mode timeout values: one for the local device and one for the remote device.

#### *8.7.2.2 Sniff Subrating Mode*

When the sniff mode timeout has expired a device shall enter sniff subrating mode. In sniff subrating mode the mandatory sniff subrate anchor points at which the master shall transmit to the slave and the slave shall listen for the master, are defined as follows (where M is the max subrate received by the master, N is the max subrate received by the slave, A is the sniff subrating instant, and k is any positive integer):

| | **Master** | **Slave** |
|---|---|---|
| **M=N** | $CLK_{M(k)} = A + [T_{sniff} \bullet M \bullet k]$ | $CLK_{N(k)} = A + [T_{sniff} \bullet N \bullet k]$ |
| **M>N** | At least once every j anchor points satisfying $CLK_{N(k)} = A + [T_{sniff} \bullet N \bullet k]$, where $j = \lfloor M / N \rfloor$ | $CLK_{N(k)} = A + [T_{sniff} \bullet N \bullet k]$ |
| **M<N** | $CLK_{M(k)} = A + [T_{sniff} \bullet M \bullet k]$ | At least once every j anchor points satisfying $CLK_{M(k)} = A + [T_{sniff} \bullet M \bullet k]$, where $j = \lfloor N / M \rfloor$ |

*Table 8.7: Sniff subrate anchor points*

1 Note: $\lfloor X \rfloor$ denotes the mathematical “floor” function, equivalent to rounding down to the nearest integer. Hence, $\lfloor X / Y \rfloor$ yields the highest integer j for which $Y \bullet j \leq X$.

2 Note: When sniff subrating is enabled, the rules specified in Section 8.7 for $N_{sniff\ attempt}$ and $N_{sniff\ Timeout}$ shall apply to sniff subrating anchor points.

## 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 synchronous 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 logical transport address(es) (LT_ADDR).

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 will wait for further master transmissions.



LifeScan, Inc. Ex. 1019
Page 348 of 1420
IPR2025-00103

## 8.9 PARK STATE

When a slave does not need to participate on the piconet 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 logical transport address 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 only be unparked 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 re-synchronize 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 the next section. 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 non-overlapping park beacon structures for each hop sequence. The beacon structures shall not overlap either their beacon slots or access windows.

In addition for using it for low power consumption, park 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 out respectively in 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).

### 8.9.1 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 which is transmitted periodically with a constant time interval. The beacon train is illustrated in Figure 8.14 on page 181. 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 $\Delta_B$. The start of the first beacon slot is referred to as the **beacon instant** and serves as the beacon timing reference. The beacon parameters $N_B$ and $T_B$ are chosen such that there are sufficient beacon slots for a parked slave to synchronize to during a certain time window in an error-prone environment.



LifeScan, Inc. Ex. 1019
Page 349 of 1420
IPR2025-00103

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 Section 8.6.2 on page 162), 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 DB 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:

1. Transmission of master-to-slave packets which the parked slaves can use for re-synchronization
2. Carrying messages to the parked slaves to change the beacon parameters
3. Carrying general broadcast messages to the parked slaves
4. Unparking of one or more parked slaves



LifeScan, Inc. Ex. 1019
Page 350 of 1420
IPR2025-00103

Since a slave can synchronize to any packet which is preceded by the proper channel access code, 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 indeed 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 synchronization 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.

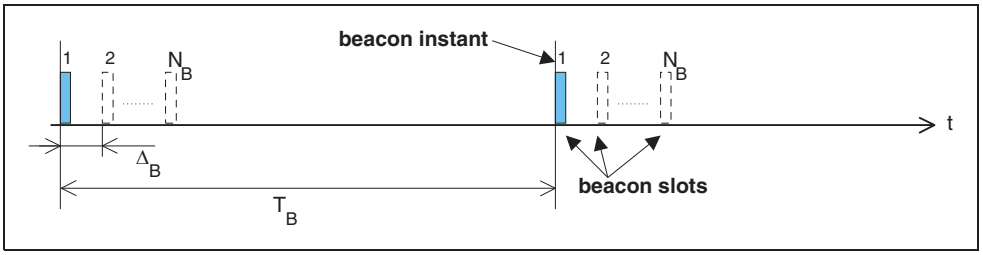



*Figure 8.14: General beacon train format*

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 Section 8.9.2).



LifeScan, Inc. Ex. 1019
Page 351 of 1420
IPR2025-00103

### 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 8.15 on page 182. The access window starts a fixed delay $D_{access}$ after the beacon instant. The width of the access window is $T_{access}$.

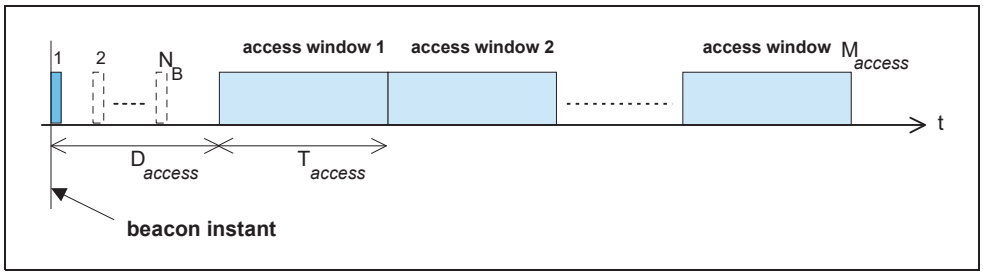



*Figure 8.15: Definition of access window*

The access window supports a polling slave access technique. The format of the polling technique is shown in Figure 8.16 on page 182. The same TDD structure is used as on the piconet 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 a parked slave is allowed to respond in corresponds to its access request address (AR_ADDR), see also Section 8.9.6 on page 185. For counting the half slots to determine the access request slot, the start of the access window is used, see Figure 8.16 on page 182. The slave shall only send an access request in the proper slave-to-master half slot if a broadcast packet has been received in the preceding master-to-slave slot. In this way, the master polls the parked slaves.

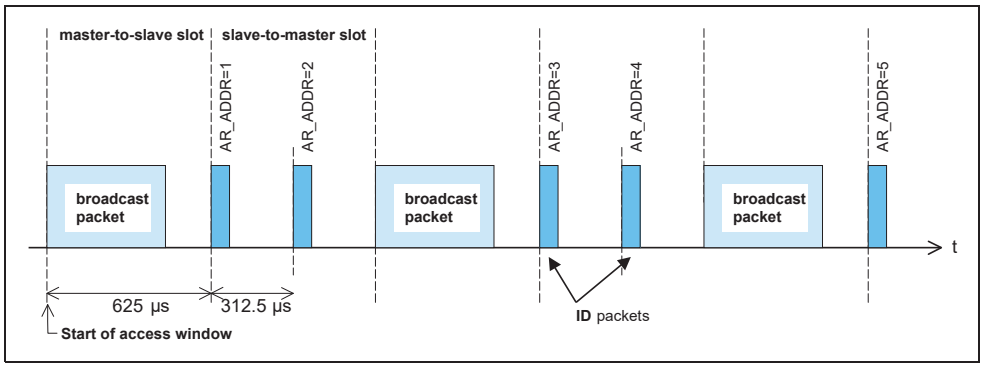



*Figure 8.16: Access procedure applying the polling technique.*

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 8.17 on page 183). 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.

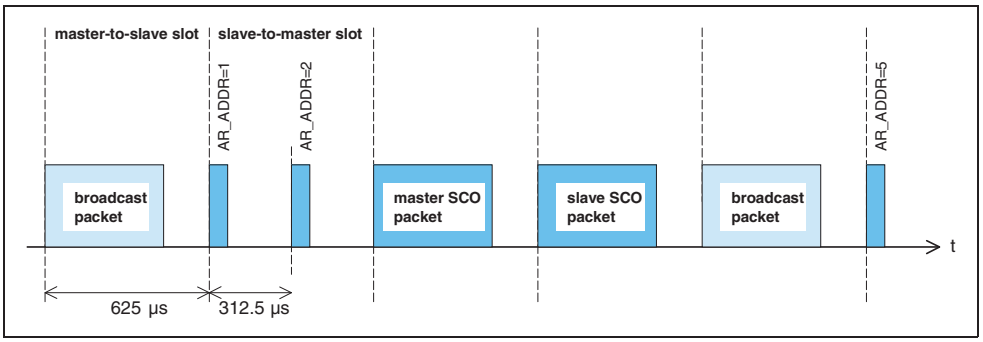



*Figure 8.17: Disturbance of access window by SCO traffic*

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.

### 8.9.3 Parked slave synchronization

Parked slaves wake up periodically to re-synchronize 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 re-synchronize. 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 re-synchronize. During the search, the slave may increase its search window, see also Section 2.2.5.2 on page 69. The separation between the beacon slots in the beacon train $\Delta_B$ shall be chosen such 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 which is longer than the beacon interval $T_B$, see Figure 8.18 on page 184. The slave sleep window shall be a multiple $N_{B\_sleep}$ of $T_B$.



LifeScan, Inc. Ex. 1019
Page 353 of 1420
IPR2025-00103

The precise beacon instant the slave may wake up on 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:

$CLK_{27-1} \bmod (N_{B\_sleep} \bullet T_B) = D_B + D_{B\_sleep} \bullet T_B$ for initialization 1

$(\overline{CLK}_{27}, CLK_{26-1}) \bmod (N_{B\_sleep} \bullet T_B) = D_B + D_{B\_sleep} \bullet T_B$ for initialization 2

where initialization 1 shall be chosen by the master if the MSB in the current master clock is 0 and initialization 2 shall be chosen by the master if the MSB in the current master clock is 1.

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) it wakes up in. If the parked slave wakes up, the minimum wake-up activity shall be to read the channel access code for re-synchronization and the packet header to check for broadcast messages.

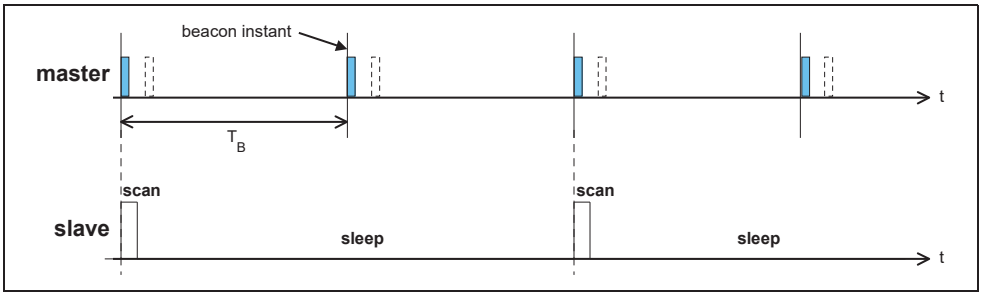



*Figure 8.18: Extended sleep interval of parked slaves.*

### 8.9.4 Parking

A master can park an active slave through the exchange of LMP commands. Before being put into park, 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 normal data packet and addresses the slave through its LT_ADDR.



LifeScan, Inc. Ex. 1019
Page 354 of 1420
IPR2025-00103

### 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 logical transport address LT_ADDR the slave shall use after it has re-entered 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 packet from the slave is 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 it has received a link supervision timeout for that slave 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, 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 Section 7.6.2.1 on page 140).

### 8.9.6 Slave-initiated unparking

A slave can request access to the channel through the access window defined in Section 8.9.2 on page 182. As shown in Figure 8.16 on page 182, the access window includes several slave-to-master half slots where the slave may send an access request message. The specific half slot the slave is allowed to respond in, corresponds to its access request address (AR_ADDR) which it received when it was parked. The order of the half slots (in Figure 8.16 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 device access code (DAC) of the master (which is the channel access code without the trailer). The parked slave shall only transmit an access request message in the half slot, 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 shall be sent.



LifeScan, Inc. Ex. 1019
Page 355 of 1420
IPR2025-00103

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 Section 8.9.2 on page 182), 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, 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 Section 7.6.2.1 on page 140).

### 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.

### 8.9.8 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 only allowed to send if addressed specifically.

# 9 AUDIO

On the air-interface, either a 64 kb/s log PCM (Pulse Code Modulation) format (A-law or μ-law) may be used, or a 64 kb/s CVSD (Continuous Variable Slope Delta 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 9.1 on page 187 summarizes the voice coding schemes supported on the air interface. The appropriate voice coding scheme is selected after negotiations between the Link Managers.

| **Voice Codecs** | |
|---|---|
| linear | CVSD |
| 8-bit logarithmic | A-law |
| | μ-law |

*Table 9.1: Voice coding schemes supported on the air interface.*

## 9.1 LOG PCM 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 shall be performed. The compression method shall follow ITU-T recommendations G. 711.

## 9.2 CVSD CODEC

A more robust format for voice over the air interface is delta modulation. This modulation scheme follows the waveform where the output bits indicate whether the prediction value is smaller or larger then 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 ksamples/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 9.1 on page 188, Figure 9.2 on page 188 and Figure 9.3 on page 188. The system shall be clocked at 64 kHz.



LifeScan, Inc. Ex. 1019
Page 357 of 1420
IPR2025-00103

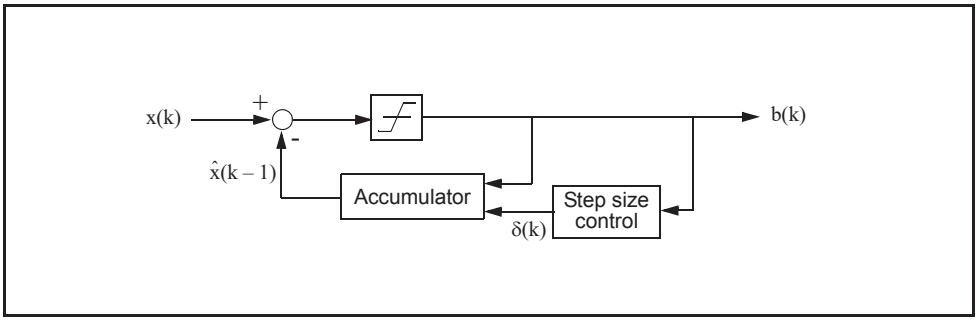



*Figure 9.1: Block diagram of CVSD encoder with syllabic companding.*

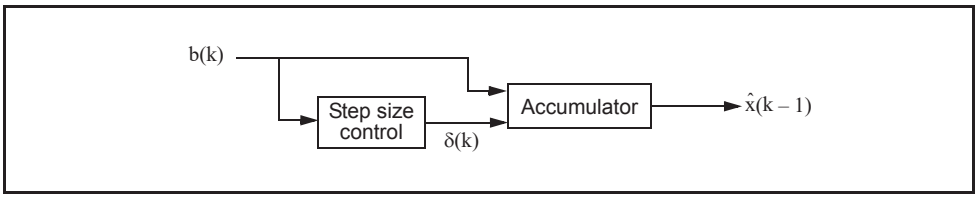



*Figure 9.2: Block diagram of CVSD decoder with syllabic companding.*

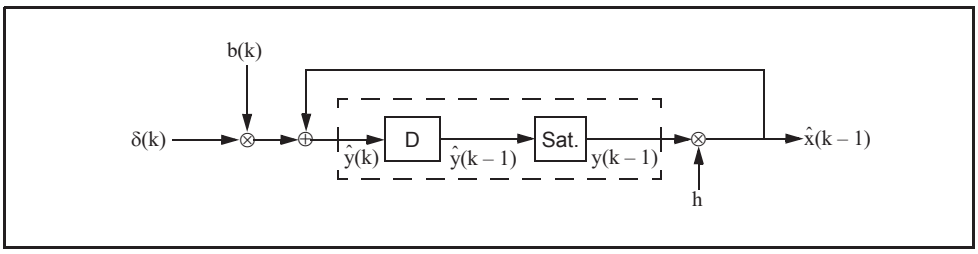



*Figure 9.3: Accumulator procedure.*

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 $\beta$ denote the decay factor for the step size, and, let $\alpha$ be the syllabic companding parameter. The latter parameter monitors the slope by considering the *K* most recent output bits

Let

$$\hat{x}(k) = hy(k). \qquad \text{(EQ 13)}$$

Then, the CVSD encoder internal state shall be updated according to the following set of equations:

$$b(k) = \mathrm{sgn}\{x(k) - \hat{x}(k-1)\}, \quad \text{(EQ 14)}$$

$$\alpha = \begin{cases} 1, & \text{if J bits in the last K output bits are equal,} \\ 0, & \text{otherwise,} \end{cases} \quad \text{(EQ 15)}$$

$$\delta(k) = \begin{cases} \min\{\delta(k-1) + \delta_{min}, \delta_{max}\}, & \alpha = 1, \\ \max\{\beta\delta(k-1), \delta_{min}\}, & \alpha = 0, \end{cases} \quad \text{(EQ 16)}$$

$$y(k) = \begin{cases} \min\{\hat{y}(k), y_{max}\}, & \hat{y}(k) \geq 0. \\ \max\{\hat{y}(k), y_{min}\}, & \hat{y}(k) < 0. \end{cases} \quad \text{(EQ 17)}$$

where

$$\hat{y}(k) = \hat{x}(k-1) + b(k)\delta(k). \quad \text{(EQ 18)}$$

In these equations, $\delta_{min}$ and $\delta_{max}$ 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 9.2 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

| Parameter | Value |
|---|---|
| $h$ | $1 - \frac{1}{32}$ |
| $\beta$ | $1 - \frac{1}{1024}$ |
| $J$ | 4 |
| $K$ | 4 |
| $\delta_{min}$ | 10 |
| $\delta_{max}$ | 1280 |
| $y_{min}$ | $-2^{15}$ or $-2^{15} + 1$ |
| $y_{max}$ | $2^{15} - 1$ |

*Table 9.2: CVSD parameter values. The values are based on a 16 bit signed number output from the accumulator.*



LifeScan, Inc. Ex. 1019
Page 359 of 1420
IPR2025-00103

## 9.3 ERROR HANDLING

In the DV, HV3, EV3, EV5, 2-EV3, 3-EV3, 2-EV5 and 3-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. However, when a packet is rejected because either the channel access code, 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 are 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 one-bit errors. Thus, there exist no detectable but uncorrectable error events for **HV1** packets.

## 9.4 GENERAL AUDIO REQUIREMENTS

### 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.

### 9.4.2 CVSD audio quality

For Bluetooth audio quality the requirements are put on the transmitter side. The 64 ksamples/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.

# 10 LIST OF FIGURES

Figure 1.1: Piconets with a single slave operation (a), a multi-slave operation (b) and a scatternet operation (c). 59
Figure 1.2: Standard Basic Rate packet format. ....60
Figure 1.3: Standard Enhanced Data Rate packet format ....60
Figure 1.4: Bluetooth clock. ....60
Figure 1.5: Format of BD_ADDR. ....62
Figure 2.1: Multi-slot packets ....66
Figure 2.2: Derivation of CLK in master (a) and in slave (b). ....67
Figure 2.3: RX/TX cycle of master transceiver in normal mode for single-slot packets. ....68
Figure 2.4: RX/TX cycle of slave transceiver in normal mode for single-slot packets. ....69
Figure 2.5: RX timing of slave returning from hold mode. ....70
Figure 2.6: Derivation of CLKE. ....71
Figure 2.7: RX/TX cycle of transceiver in PAGE mode. ....72
Figure 2.8: Timing of page response packets on successful page in first half slot ....73
Figure 2.9: Timing of page response packets on successful page in second half slot ....74
Figure 2.10: Timing of inquiry response packets on successful inquiry in first half slot ....76
Figure 2.11: Timing of inquiry response packets on successful inquiry in second half slot ....76
Figure 2.12: General block diagram of hop selection scheme. ....78
Figure 2.13: Hop selection scheme in CONNECTION state. ....79
Figure 2.14: Single- and multi-slot packets. ....80
Figure 2.15: Example of the same channel mechanism. ....80
Figure 2.16: Block diagram of the basic hop selection kernel for the hop system. ....81
Figure 2.17: XOR operation for the hop system. ....82
Figure 2.18: Permutation operation for the hop system. ....83
Figure 2.19: Butterfly implementation. ....83
Figure 2.20: Block diagram of adaptive hop selection mechanism ....85
Figure 4.1: Functional diagram of TX buffering. ....93
Figure 4.2: Functional diagram of RX buffering ....97
Figure 6.1: General Basic Rate packet format. ....102
Figure 6.2: General enhanced data rate packet format ....102
Figure 6.3: Access code format ....104
Figure 6.4: Preamble ....105
Figure 6.5: Construction of the sync word. ....106
Figure 6.6: LFSR and the starting state to generate . ....108



LifeScan, Inc. Ex. 1019
Page 361 of 1420
IPR2025-00103

Figure 6.7: Trailer in CAC when MSB of sync word is 0 (a), and when MSB of sync word is 1 (b). ..... 108
Figure 6.8: Header format. ..... 109
Figure 6.9: Format of the FHS payload. ..... 113
Figure 6.10: DV packet format ..... 116
Figure 6.11: **Synchronization sequence ..... 122**
Figure 6.12: Payload header format for Basic Rate single-slot ACL packets. ..... 123
Figure 6.13: Payload header format for multi-slot ACL packets and all EDR ACL packets. ..... 124
Figure 7.1: Header bit processes. ..... 129
Figure 7.2: Payload bit processes. ..... 129
Figure 7.3: The LFSR circuit generating the HEC. ..... 130
Figure 7.4: Initial state of the HEC generating circuit. ..... 131
Figure 7.5: HEC generation and checking. ..... 131
Figure 7.6: The LFSR circuit generating the CRC. ..... 132
Figure 7.7: Initial state of the CRC generating circuit. ..... 132
Figure 7.8: CRC generation and checking. ..... 132
Figure 7.9: Data whitening LFSR. ..... 133
Figure 7.10: Bit-repetition encoding scheme. ..... 134
Figure 7.11: LFSR generating the (15,10) shortened Hamming code. ..... 135
Figure 7.12: Stage 1 of the receive protocol for determining the ARQN bit. ..... 137
Figure 7.13: Stage 2 (ACL) of the receive protocol for determining the ARQN bit. 138
Figure 7.14: Stage 2 (eSCO) of the receive protocol for determining the ARQN bit. 139
Figure 7.15: Transmit filtering for packets with CRC. ..... 140
Figure 7.16: Broadcast repetition scheme ..... 144
Figure 8.1: State diagram of link controller. ..... 145
Figure 8.2: Conventional page (a), page while one synchronous link present (b), page while two synchronous links present (c). ..... 150
Figure 8.3: Messaging at initial connection when slave responds to first page message. ..... 152
Figure 8.4: Messaging at initial connection when slave responds to second page message. ..... 152
Figure 8.5: Connection state. ..... 161
Figure 8.6: RX/TX timing in multi-slave configuration ..... 162
Figure 8.7: eSCO Window Details for Single-Slot Packets ..... 165
Figure 8.8: eSCO Window Details for Asymmetric Traffic ..... 166
Figure 8.9: Successful hop sequence switching ..... 170
Figure 8.10: Recovery hop sequences ..... 171
Figure 8.11: AFH_Instant changes during multi-slot packets transmitted by the master. ..... 172



LifeScan, Inc. Ex. 1019
Page 362 of 1420
IPR2025-00103

Figure 8.12: Sniff anchor points .....175
Figure 8.13: Sniff subrating .....177
Figure 8.14: General beacon train format .....181
Figure 8.15: Definition of access window .....182
Figure 8.16: Access procedure applying the polling technique. .....182
Figure 8.17: Disturbance of access window by SCO traffic .....183
Figure 8.18: Extended sleep interval of parked slaves. .....184
Figure 9.1: Block diagram of CVSD encoder with syllabic companding. ...188
Figure 9.2: Block diagram of CVSD decoder with syllabic companding. ...188
Figure 9.3: Accumulator procedure. .....188
Figure 11.1: SLR measurement set-up. .....196
Figure 11.2: RLR measurement set-up. .....196
Figure 11.3: Plot of recommended frequency mask for Bluetooth. The GSM send frequency mask is given for comparison (dotted line) ...197
Figure 11.4: Timing constraint on AFH_Instant with slaves in park, hold and sniff 202



LifeScan, Inc. Ex. 1019
Page 363 of 1420
IPR2025-00103

# 11 LIST OF TABLES

Table 2.1: Control of the butterflies for the hop system ..... 82
Table 2.2: Control for hop system. ..... 86
Table 6.1: Summary of access code types. ..... 104
Table 6.2: Packets defined for synchronous and asynchronous logical transport types. ..... 112
Table 6.3: Description of the FHS payload ..... 114
Table 6.4: Contents of SR field ..... 115
Table 6.5: Contents of page scan mode field ..... 115
Table 6.6: Logical link LLID field contents ..... 124
Table 6.7: Use of payload header flow bit on the logical links. ..... 126
Table 6.8: Link control packets ..... 127
Table 6.9: ACL packets ..... 127
Table 6.10: Synchronous packets ..... 128
Table 8.1: Relationship between scan interval, and paging modes R0, R1, and R2. ..... 147
Table 8.2: Relationship between train repetition, and paging modes R0, R1 and R2 when synchronous links are present. ..... 150
Table 8.3: Initial messaging during start-up. ..... 151
Table 8.4: Increase of train repetition when synchronous links are present. ..... 158
Table 8.5: Messaging during inquiry routines. ..... 160
Table 8.6: Channel classification descriptions ..... 173
Table 8.7: Sniff subrate anchor points ..... 178
Table 9.1: Voice coding schemes supported on the air interface. ..... 187
Table 9.2: CVSD parameter values. The values are based on a 16 bit signed number output from the accumulator. ..... 189
Table 11.1: Recommended Frequency Mask for Bluetooth ..... 197



LifeScan, Inc. Ex. 1019
Page 364 of 1420
IPR2025-00103

# APPENDIX A: GENERAL AUDIO RECOMMENDATIONS

## MAXIMUM SOUND PRESSURE

It is the sole responsibility of each manufacturer to design their audio products in a safe way with regards to injury to the human ear. The Bluetooth Specification doesn't specify maximum sound pressure from an audio device.

## OTHER TELEPHONY NETWORK REQUIREMENTS

It is the sole responsibility of each manufacturer to design the Bluetooth audio product so that it meets the regulatory requirements of all telephony networks that it may be connected to.

## AUDIO LEVELS FOR BLUETOOTH

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.

The physical test set-up for Handsets and Headsets is described in ITU-T Recommendation P.51 and P.57

The physical test set-up for speakerphones and "Vehicle hands-free systems" is specified in ITU-T Recommendation P.34.

A general equation for computation of loudness rating (LR) for telephone sets is given by ITU-T recommendations P.79 and is given by

$$LR = -\frac{10}{m}\log_{10}\left(\sum_{i=N_1}^{N_2} 10^{m(s_i - w_i)/10}\right), \qquad \text{(EQ 19)}$$

where

$m$ is a constant (~ 0.2).

$w_i$ = weighting coefficient (different for the various LRs).

$S_i$ = the sensitivity at frequency $F_i$, of the electro-acoustic path

$N_1$,$N_2$, consecutive filter bank numbers (Art. Index: 200-4000 Hz)

(EQ 19) on page 195 is used for calculating the (SLR) according to Figure 11.1 on page 196, and (RLR) according to Figure 11.2 on page 196. When calculating LRs one must only include those 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 P.79 uses only the frequency band 200-4000 Hz in LR computations.



LifeScan, Inc. Ex. 1019
Page 365 of 1420
IPR2025-00103

## MICROPHONE PATH

### SLR measurement model

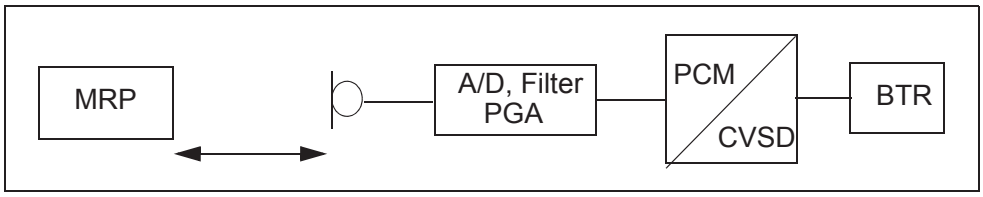



*Figure 11.1: SLR measurement set-up.*

## LOUDSPEAKER PATH

### RLR measurement model

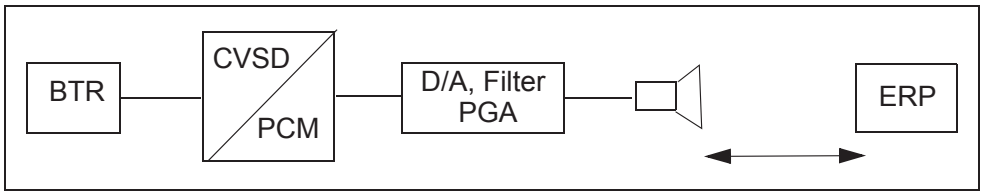



*Figure 11.2: RLR measurement set-up.*

## BLUETOOTH VOICE INTERFACE

The specification for the Bluetooth voice interface should follow in the first place the *ITU-T Recommendations P.79*, which specifies the loudness ratings for telephone sets. These recommendations give general guidelines and specific algorithms used for calculating the loudness ratings of the audio signal with respect to Ear Reference Point (ERP).

For Bluetooth voice interface to the different cellular system terminals, loudness and frequency recommendations based on the cellular standards should be used. For example, GSM 03.50 gives recommendation for both the loudness ratings and frequency mask for a GSM terminal interconnection with Bluetooth.

1- The output of the CVSD decoder are 16-bit linear PCM digital samples, at a sampling frequency of 8 ksample/second. Bluetooth also supports 8-bit log PCM samples of A-law and μ-law type. The sound pressure at the ear reference point for a given 16-bit CVSD sample, should follow the sound pressure level given in the cellular standard specification.

2- A maximum sound pressure which can be represented by a 16-bit linear PCM sample at the output of the CVSD decoder should be specified according to the loudness rating, in ITU P.79 and at PGA value of 0 dB. Programmable Gain Amplifiers (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 G.711, G.712, G.714 give guidelines and PCM value relationships. Zero-code suppression based on ITU-T G.711 is also recommended to avoid network mismatches.



LifeScan, Inc. Ex. 1019
Page 366 of 1420
IPR2025-00103

## FREQUENCY MASK

For interfacing a Bluetooth terminal 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 speech coders. A recommendation for a frequency mask is given in the Table 11.1 below. The Figure 11.3 below shows a plot of the frequency mask for Bluetooth (solid line). The GSM frequency mask (dotted line) is shown in Figure 11.3 for comparison.

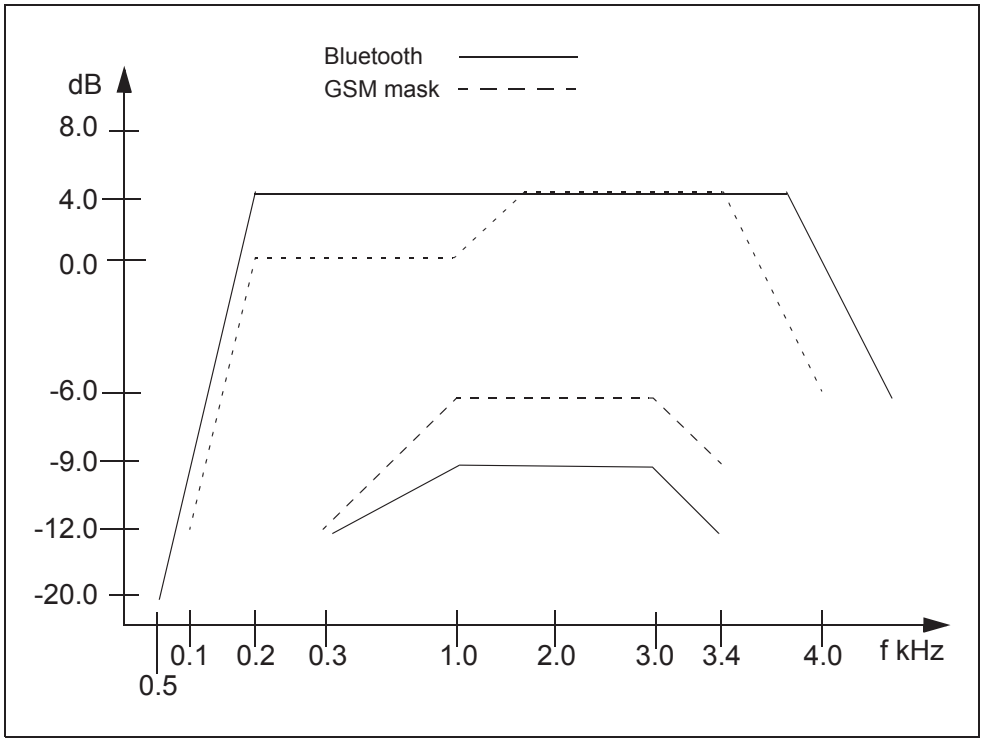



*Figure 11.3: Plot of recommended frequency mask for Bluetooth. The GSM send frequency mask is given for comparison (dotted line)*

| 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 | - |

*Table 11.1: Recommended Frequency Mask for Bluetooth*



LifeScan, Inc. Ex. 1019
Page 367 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 368 of 1420
IPR2025-00103

# APPENDIX B: TIMERS

This appendix contains a list of Baseband timers related to inactivity timeout defined in this specification. Definitions and default values of the timers are listed below

All timer values are given in slots.

## LIST OF TIMERS

### inquiryTO

The *inquiryTO* 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.

### pageTO

The *pageTO* 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.

### pagerespTO

In the slave, it defines the number of slots the slave awaits the master's response, FHS packet, after sending the page acknowledgment ID packet. In the master, *pagerespTO* defines the number of slots the master should wait for the FHS packet acknowledgment before returning to **page** substate. Both master and slave units should use the same value for this timeout, to ensure common page/scan intervals after reaching *pagerespTO*.

The *pagerespTO* value is 8 slots.

### newconnectionTO

Every time a new connection is started through paging, scanning, role switch or unparking, the master sends a POLL packet as the first packet in the new connection. Transmission and acknowledgment of this POLL packet is 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.

*newconnectionTO* value is 32 slots.



LifeScan, Inc. Ex. 1019
Page 369 of 1420
IPR2025-00103

### supervisionTO

The *supervisionTO* 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 baseband level a default value that is equivalent to 20 seconds will be used.



LifeScan, Inc. Ex. 1019
Page 370 of 1420
IPR2025-00103

# APPENDIX C: 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))
- *Disabled*

The master may place an AFH capable slave in any of the three AFH operating modes.

## Operation at the Master

A master that has one or more slaves in park, hold, or sniff 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 non-overlapping “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 the figure below. 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, its slave in sniff, and had time to un-park its parked slaves.

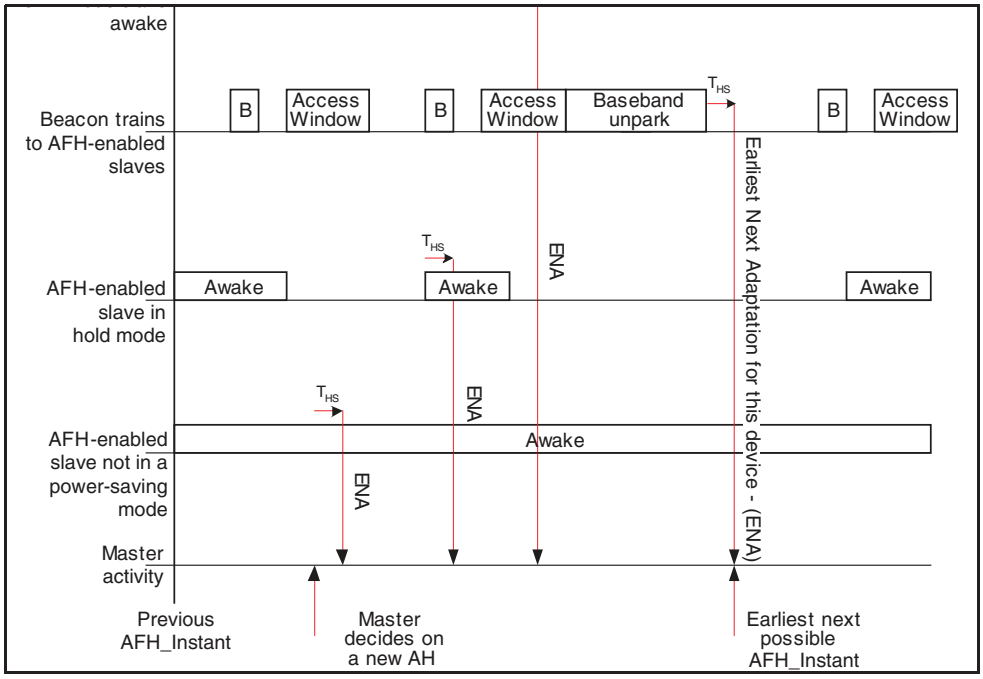



*Figure 11.4: Timing constraint on AFH_Instant with slaves in park, hold and sniff*

## Operation in park

A slave that is in the Park state cannot send or receive any AFH LMP messages. Once the slave has left the Park state the master may subsequently update the slave's adaptation.

## AFH Operation in Sniff

Once a slave has been placed in sniff mode, the master may periodically change its AHS without taking the slave out of sniff mode.

## AFH Operation in Hold

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.



LifeScan, Inc. Ex. 1019
Page 372 of 1420
IPR2025-00103

# LINK MANAGER PROTOCOL

***This specification describes the Link Manager Protocol (LMP) which is used for link set-up and control. The signals are interpreted and filtered out by the Link Manager on the receiving side and are not propagated to higher layers.***



LifeScan, Inc. Ex. 1019
Page 373 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 374 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 207**

**2 General Rules ..... 208**

2.1 Message Transport ..... 208
2.2 Synchronization ..... 208
2.3 Packet Format ..... 209
2.4 Transactions ..... 210
2.4.1 LMP Response Timeout ..... 211
2.5 Error Handling ..... 211
2.5.1 Transaction collision resolution ..... 212
2.6 Procedure Rules ..... 212
2.7 General Response Messages ..... 213
2.8 LMP Message Constraints ..... 213

**3 Device Features ..... 214**

3.1 General Description ..... 214
3.2 Feature Definitions ..... 214
3.3 Feature Mask Definition ..... 220
3.4 Link Manager Interoperability policy ..... 222

**4 Procedure Rules ..... 223**

4.1 Connection Control ..... 223
4.1.1 Connection establishment ..... 223
4.1.2 Detach ..... 224
4.1.3 Power control ..... 225
4.1.4 Adaptive frequency hopping ..... 227
4.1.5 Channel classification ..... 230
4.1.6 Link supervision ..... 232
4.1.7 Channel quality driven data rate change (CQDDR) ..... 233
4.1.8 Quality of service (QoS) ..... 234
4.1.9 Paging scheme parameters ..... 236
4.1.10 Control of multi-slot packets ..... 237
4.1.11 Enhanced Data Rate ..... 237
4.1.12 Encapsulated LMP PDUs ..... 239
4.2 Security ..... 240
4.2.1 Authentication ..... 240
4.2.2 Pairing ..... 242
4.2.3 Change link key ..... 245
4.2.4 Change current link key type ..... 246
4.2.5 Encryption ..... 248



LifeScan, Inc. Ex. 1019
Page 375 of 1420
IPR2025-00103

4.2.6 Request supported encryption key size ..... 255
4.2.7 Secure Simple Pairing ..... 255
4.3 Informational Requests ..... 268
4.3.1 Timing accuracy ..... 268
4.3.2 Clock offset ..... 270
4.3.3 LMP version ..... 270
4.3.4 Supported features ..... 271
4.3.5 Name request ..... 273
4.4 Role Switch ..... 274
4.4.1 Slot offset ..... 274
4.4.2 Role switch ..... 275
4.5 Modes of Operation ..... 277
4.5.1 Hold mode ..... 277
4.5.2 Park state ..... 279
4.5.3 Sniff mode ..... 286
4.6 Logical Transports ..... 290
4.6.1 SCO logical transport ..... 290
4.6.2 eSCO logical transport ..... 293
4.7 Test Mode ..... 298
4.7.1 Activation and deactivation of test mode ..... 298
4.7.2 Control of test mode ..... 299
4.7.3 Summary of test mode PDUs ..... 299

**5 Summary ..... 303**
5.1 PDU Summary ..... 303
5.2 Parameter Definitions ..... 313
5.3 LMP Encapsulated ..... 323
5.4 Default Values ..... 324

**6 List of Figures ..... 325**

**7 List of Tables ..... 329**

# 1 INTRODUCTION

The Link Manager Protocol (LMP) is used to control and negotiate all aspects of the operation of the Bluetooth connection between two devices. This includes the set-up and control of logical transports and logical links, and for control of physical links.

The Link Manager Protocol is used to communicate between the Link Managers (LM) on the two devices which 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 which 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.

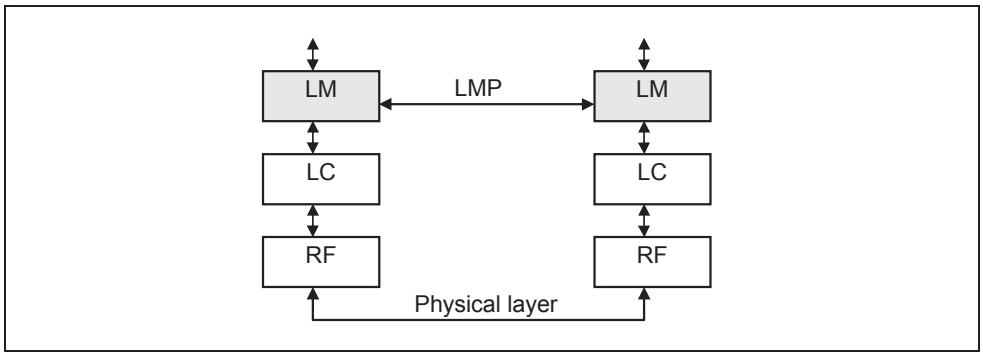



*Figure 1.1: Link Manager Protocol signaling layer.*



LifeScan, Inc. Ex. 1019
Page 377 of 1420
IPR2025-00103

# 2 GENERAL RULES

## 2.1 MESSAGE TRANSPORT

LMP messages shall be exchanged over the ACL-C logical link that is carried on the default ACL logical transport (Baseband Specification, Section 4.4, on page 92). The ACL-C logical link is distinguished from the ACL-U (which carries L2CAP and user data) by the Logical Link Identifier (LLID) field carried in the payload header of variable-length packets (Baseband Specification, Section 6.6.2, on page 123).

The ACL-C has a higher priority than other traffic - see Baseband Specification, Section 5.6, on page 101.

The error detection and correction capabilities of the baseband 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 is an implementation matter.

## 2.2 SYNCHRONIZATION

This section is informative and explains why many of the LMP message sequences are defined as they are.

LMP messages are carried on the ACL-C logical link, which does not guarantee a time to deliver or 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 logical transport address (LT_ADDR) may be re-used 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 Bluetooth clock 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 LC normally attempts to communicate with each slave no less often than every $T_{poll}$ slots (see Section 4.1.8 on page 234) based on the $T_{poll}$ for that slave.



LifeScan, Inc. Ex. 1019
Page 378 of 1420
IPR2025-00103

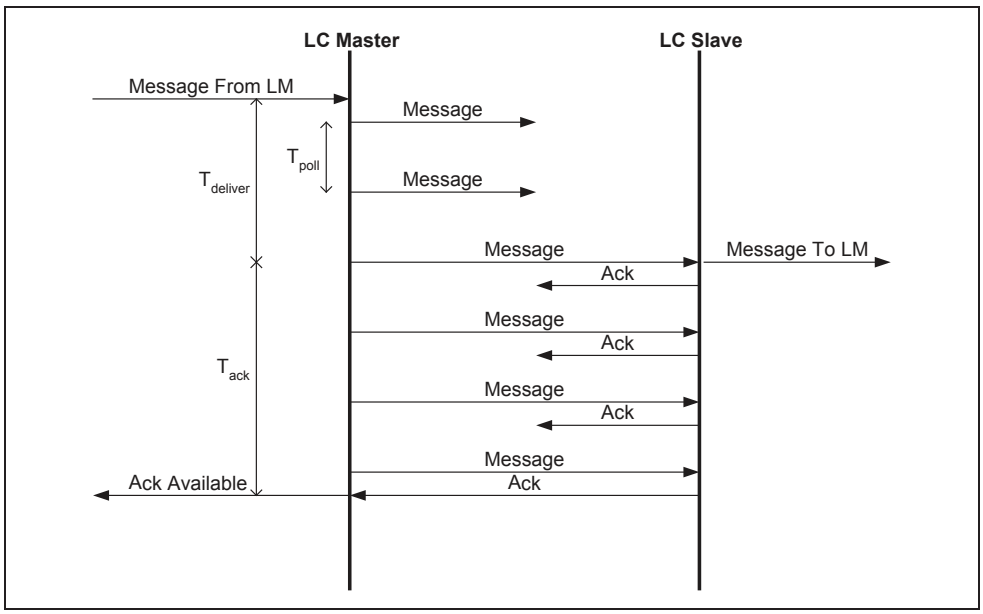



*Figure 2.1: Transmission of a message from master to slave[1].*

Figure 2.1 on page 209 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.

## 2.3 PACKET FORMAT

Each PDU is assigned either a 7 or a 15 bit opcode used to uniquely identify different types of PDUs, see Table 5.1 on page 303. 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 bytes used

1. Note 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.



LifeScan, Inc. Ex. 1019
Page 379 of 1420
IPR2025-00103

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.

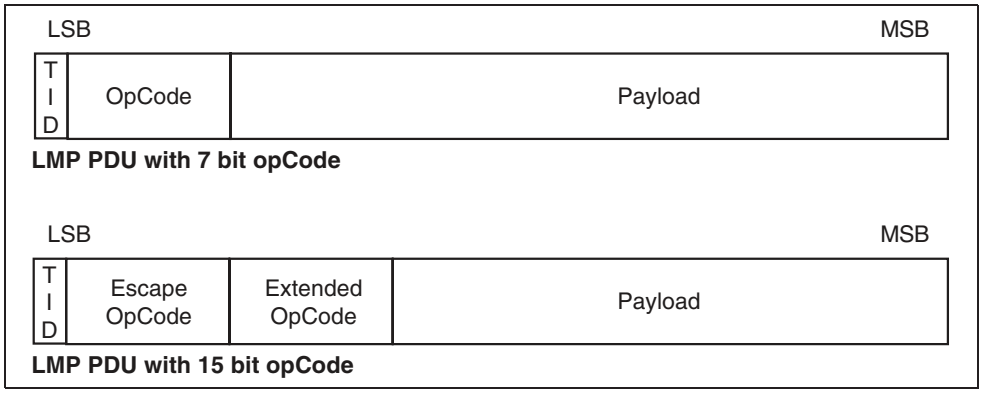



*Figure 2.2: Payload body when LMP PDUs are sent.*

## 2.4 TRANSACTIONS

The LMP operates in terms of transactions. A transaction is a connected set of message exchanges which achieve a particular purpose. All PDUs which form part of the same transaction shall have the same value for the transaction ID which is stored as part of the first byte of the opCode (see Section 2.3 on page 209).

The transaction ID is in the least significant bit. 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 Section 4 on page 223 shall be defined as a transaction. For pairing, see Section 4.2.2 on page 242, and encryption, see Section 4.2.5 on page 248, all sequences belonging to each section shall be counted as one transaction and shall use the same transaction ID. For connection establishment, see Section 4.1.1 on page 223, 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 Section 2.5 on page 211, the PDU to be rejected and LMP_not_accepted or LMP_not_accepted_ext shall form a single transaction.

### 2.4.1 LMP Response Timeout

The time between receiving a baseband packet carrying an LMP PDU and sending a baseband packet carrying a valid response PDU, according to the procedure rules in Section 4 on page 223, shall be less than the LMP Response Timeout. The value of this timeout is 30 seconds. Note that the LMP Response Timeout is applied not only to sequences described in Section 4 on page 223, but also to the series of the sequences defined as the transactions in Section 4 on page 223. 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, for example during the enabling of encryption.

The LMP Response Timeout shall restart each time an LMP PDU which requires a reply is queued for transmission by the baseband.

## 2.5 ERROR HANDLING

If the LM receives a PDU with unrecognized opcode, it shall respond with LMP_not_accepted or LMP_not_accepted_ext 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 LMP_not_accepted or LMP_not_accepted_ext with the error code *invalid LMP parameters*.

If the maximum response time, see Section 2.4 on page 210, is exceeded or if a link loss is detected (see Baseband Specification, Section 3.1, on page 90), 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 transmit 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, for example LMP_host_connection_req or LMP_unit_key, the LM shall return PDU LMP_not_accepted or LMP_not_accepted_ext with the error code *PDU not allowed*. If the PDU normally doesn't expect a reply, for example LMP_sres or LMP_temp_key, the PDU will be ignored.

The reception of an optional PDU which 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 the specification) it shall respond with LMP_not_accepted or LMP_not_accepted_ext with the error code *unknown LMP PDU*. If the LM recognizes the PDU as optional but unsupported then it shall reply with LMP_not_accepted or LMP_not_accepted_ext with the error code *unsupported LMP feature* if the PDU would normally generate a reply otherwise no reply is generated.



LifeScan, Inc. Ex. 1019
Page 381 of 1420
IPR2025-00103

### 2.5.1 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 cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending LMP_not_accepted or LMP_not_accepted_ext 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 cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending LMP_not_accepted or LMP_not_accepted_ext with the error code *LMP error different transaction collision*. The master- initiated procedure shall then be completed.

## 2.6 PROCEDURE RULES

Each procedure is described and depicted with a sequence diagram. The following symbols are used in the sequence diagrams:

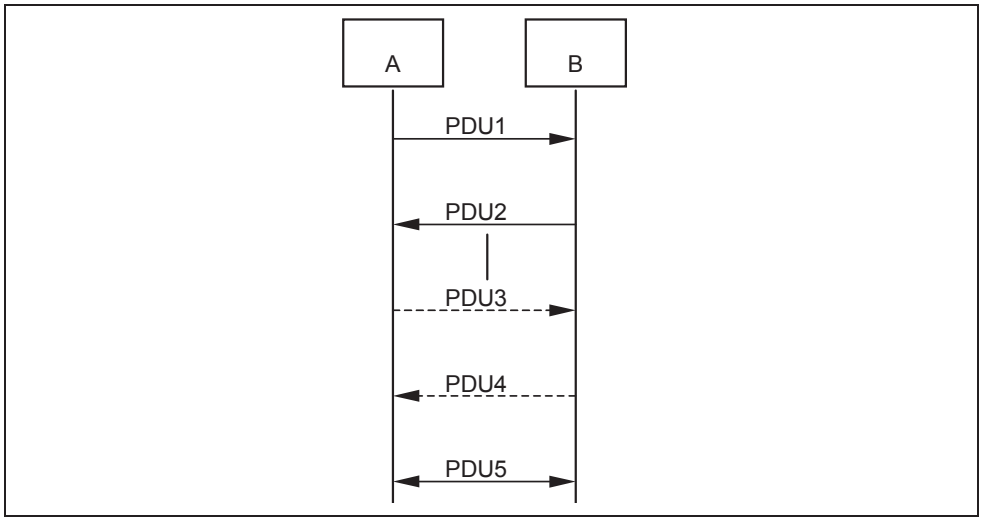



*Figure 2.3: Symbols used in sequence diagrams.*

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.



LifeScan, Inc. Ex. 1019
Page 382 of 1420
IPR2025-00103

## 2.7 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 which is accepted. LMP_not_accepted or LMP_not_accepted_ext includes the opcode of the message which is not accepted and the error code why it is not accepted.

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.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_accepted | op code |
| M | LMP_not_accepted | op code<br>error code |
| O | LMP_accepted_ext | escape op code<br>extended op code |
| O | LMP_not_accepted_ext | escape op code<br>extended op code<br>error code |

*Table 2.1: General response messages.*

## 2.8 LMP MESSAGE CONSTRAINTS

This section is informative.

- No LMP message shall exceed the maximum payload length of a single DM1 packet i.e. 17 bytes in length (Baseband Specification, Section 6.5.4.1, on page 119).
- All LM messages are of fixed length apart from those sent using broadcast in park state.
- The LMP version number shall not be used to indicate the presence or absence of functionality.



LifeScan, Inc. Ex. 1019
Page 383 of 1420
IPR2025-00103

# 3 DEVICE FEATURES

## 3.1 GENERAL DESCRIPTION

Each PDU is either Mandatory or Optional as defined by the M/O field in the tables of Section 4 on page 223. An M in this field shall indicate that support for the feature is mandatory. An O in this field shall indicate that support for the PDU is optional and it shall be followed by the number(s) of the feature(s) involved in brackets.

All features added after the 1.1 specification 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 which do not support them.

The LM does not need to be able to transmit a PDU which is optional. Support of optional PDUs is indicated by a device's features mask. The features mask can be read (see Section 4.3.4 on page 271). An LM shall not send or be sent any PDU which is incompatible with the features signalled in its or its peer's features mask, as detailed in Section 3.2.

## 3.2 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 Section 4.3.4 on page 271. |
| Timing accuracy | This feature indicates whether the LM supports requests for timing accuracy using the sequence defined in Section 4.3.1 on page 268. |
| Enhanced inquiry scan | This feature indicates whether the device is capable of supporting the enhanced inquiry scan mechanism as defined in Baseband Specification, Section 8.4.1, on page 156. 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 Baseband Specification, Section 8.4.1, on page 156. 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 Baseband Specification, Section 8.3.1, on page 146. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |

*Table 3.1: Feature Definitions*

| Feature | Definition |
|---|---|
| RSSI with inquiry results | This feature bit shall be set if the "Extended Inquiry Response" feature bit is set. This feature indicates whether the device is capable of reporting the RSSI with inquiry results as defined in Baseband Specification, Section 8.4.2, on page 157. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences. |
| Extended Inquiry Response | This feature indicates whether the device supports extended inquiry response as defined in the Baseband specification. If this bit is set, the "RSSI with inquiry results" feature bit shall also be set. |
| Paging parameter negotiation | This feature indicates whether the LM is capable of supporting the signaling of changes in the paging scheme as defined in Section 4.1.9 on page 236. |
| 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 in Baseband Specification, Section 6.6.2, on page 123. |
| AFH capable slave | This feature indicates whether the device is able to support the Adaptive Frequency Hopping mechanism as a slave as defined in Baseband Specification, Section 2.3, on page 70 using the LMP sequences defined in Section 4.1.4 on page 227. |
| AFH classification slave | This feature indicates whether the device is able to support the AFH classification mechanism as a slave as defined in Baseband Specification, Section 8.6.8, on page 173 using the LMP sequences defined Section 4.1.5 on page 230. |
| AFH capable master | This indicates whether the device is able to support the Adaptive Frequency Hopping mechanism as a master as defined in Baseband Specification, Section 2.3, on page 70 using the LMP sequences defined in Section 4.1.4 on page 227. |
| AFH classification master | This feature indicate whether the device is able to support the AFH classification mechanism as a master as defined in Baseband Specification, Section 8.6.8, on page 173 using the LMP sequences defined Section 4.1.5 on page 230. |
| 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 Section 4.1.3 on page 225. 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. |

*Table 3.1: Feature Definitions*

| Feature | Definition |
|---|---|
| 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 Section 4.1.3 on page 225. 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. |
| Channel Quality Driven Data Rate | 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 section Section 4.1.7 on page 233. |
| Broadcast encryption | This feature indicates whether the device is capable of supporting broadcast encryption as defined in [Part H] Section 4.2 on page 869 and also the sequences defined in Section 4.2.6 on page 255 and Section 4.2.4 on page 246.<br>Note: Devices compliant to versions of this specification 1.1 and earlier may support broadcast encryption even though this feature bit is not set. |
| Encryption | This feature shall be supported. This feature indicates whether the device supports the encryption of packet contents using the sequence defined in Section 4.2.5 on page 248. |
| Pause Encryption | When this feature bit is enabled on both sides, then the encryption pause/resume sequences shall be used. If either side does not support the Pause Encryption feature bit, then the encryption pause/resume sequences shall not be used. |
| Slot offset | This feature indicates whether the LM supports the transfer of the slot offset using the sequence defined in Section 4.4.1 on page 274. |
| Role switch | This feature indicates whether the device supports the change of master and slave roles as defined by baseband Section 8.6.5 on page 167 using the sequence defined in Section 4.4.2 on page 275. |
| Hold mode | This feature indicates whether the device is able to support Hold mode as defined in baseband Section 8.8 on page 178 using the LMP sequences defined in Section 4.5.1 on page 277. |
| Sniff mode | This feature indicates whether the device is able to support sniff mode as defined in baseband Section 8.7 on page 175 using the LMP sequences defined in Section 4.5.3 on page 286. |
| Sniff subrating | This feature indicates whether the device is able to support sniff subrating as defined in [PART B] Sniff Subrating on page 177 using the LMP sequences as defined in Sniff Subrating on page 288<br>.This feature bit shall be set if the "Sniff mode" feature bit is set. |
| Park state | This feature indicates whether the device is able to support Park state as defined in baseband Section 8.9 on page 179 using the LMP sequences defined in Section 4.5.2 on page 279. |

*Table 3.1: Feature Definitions*



LifeScan, Inc. Ex. 1019
Page 386 of 1420
IPR2025-00103

| Feature | Definition |
|---|---|
| SCO link | This feature shall indicate whether the device is able to support the SCO logical transport as defined in Baseband Specification, Section 4.3, on page 92, the HV1 packet defined in Baseband Specification, Section 6.5.2.1, on page 116 and receiving the DV packet defined in Baseband Specification, Section 6.5.2.4, on page 116 using the LMP sequence in Section 4.6.1 on page 290. |
| HV2 packets | This feature indicates whether the device is capable of supporting the HV2 packet type as defined in baseband Section 6.5.2.2 on page 116 on the SCO logical transport. Removed |
| HV3 packets | This feature indicates whether the device is capable of supporting the HV3 packet type as defined in baseband Section 6.5.2.3 on page 116 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 Baseband Specification, Section 9.1, on page 187 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 Baseband Specification, Section 9.1, on page 187 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 Baseband Specification, Section 9.2, on page 187 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 Baseband Specification, Section 6.4.3, on page 110 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 Baseband Specification, Section 5.5, on page 101 and the EV3 packet defined in Baseband Specification, Section 6.5.3.1, on page 117 using the LMP sequences defined in Section 4.6.2 on page 293. |
| EV4 packets | This feature indicates whether the device is capable of supporting the EV4 packet type defined in Baseband Specification, Section 6.5.3.2, on page 117 on the eSCO logical transport. |
| EV5 packets | This feature indicates whether the device is capable of supporting the EV5 packet type defined in Baseband Specification, Section 6.5.3.3, on page 117 on the eSCO logical transport. |
| Enhanced Data Rate ACL 2 Mbps mode | This feature indicates whether the device supports the transmission and reception of 2-DH1 packets for the transport of traffic on the ACL-U logical link. |
| Enhanced Data Rate ACL 3 Mbps mode | This feature indicates whether the device supports the transmission and reception of 3-DH1 packets for the transport of traffic on the ACL-U logical link.<br>This feature bit shall only be set if the “Enhanced Data Rate ACL 2 Mbps mode” feature bit is set. |

*Table 3.1: Feature Definitions*



LifeScan, Inc. Ex. 1019
Page 387 of 1420
IPR2025-00103

| Feature | Definition |
|---|---|
| 3-slot Enhanced Data Rate ACL packets | This feature indicates whether the device supports the transmission and reception of three-slot Enhanced Data Rate packets on the ACL-U logical link.<br>This feature bit shall only be set if the “Enhanced Data Rate ACL 2 Mbps mode” feature bit is set. |
| 5-slot Enhanced Data Rate ACL packets | This feature indicates whether the device supports the transmission and reception of 5-slot Enhanced Data Rate packets for the transport of traffic on the ACL-U logical link.<br>This feature bit shall only be set if the “Enhanced Data Rate ACL 2 Mbps mode” feature bit is set. |
| Enhanced Data Rate eSCO 2 Mbps mode | This feature indicates whether the device supports the transmission and reception of 2-EV3 packets for the transport of traffic on the eSCO logical transport. |
| Enhanced Data Rate eSCO 3 Mbps mode | This feature indicates whether the device supports the transmission and reception of 3-EV3 packets for the transport of traffic on the eSCO logical transport.<br>This feature bit shall only be set if the “Enhanced Data Rate eSCO 2 Mbps mode” feature bit is set. |
| 3-slot Enhanced Data Rate eSCO packets | This feature indicates whether the device supports the transmission and reception of 3-slot Enhanced Data Rate packets for the transport of traffic on the eSCO logical transport.<br>This feature bit shall only be set if the “Enhanced Data Rate eSCO 2 Mbps mode” feature bit is set. |
| Erroneous Data Reporting | This feature indicates whether the device is able to support the Packet_Status_Flag and the HCI commands "Write Default Erroneous Data Reporting" and "Read Default Erroneous Data Reporting".<br>This feature bit may be set if at least one of the "SCO link" or "Extended SCO link" feature bits are set. |
| Link Supervision Timeout Changed Event | This feature bit indicates whether the device supports the sending the HCI Link Supervision Timeout Changed Event to the Host. |
| Non-flushable Packet Boundary Flag | This feature indicates that the device supports the capability to correctly handle HCI ACL Data Packets with a Packet_Boundary_Flag value of 00 (Non-Automatically-Flushable packet). |
| Secure Simple Pairing (Controller Support) | This feature indicates whether the Link Manager is capable of supporting the Secure Simple Pairing, Section 4.2.7, “Secure Simple Pairing,” on page 255. |

*Table 3.1: Feature Definitions*



LifeScan, Inc. Ex. 1019
Page 388 of 1420
IPR2025-00103

| Feature | Definition |
|---|---|
| Secure Simple Pairing (Host Support) | This feature indicates whether the Host is capable of supporting the Secure Simple Pairing Section 4.2.7, “Secure Simple Pairing,” on page 255.<br>If HCI is supported, this bit shall be set if Write Simple Pairing Mode has been issued to enable the Secure Simple Pairing feature. When HCI is not supported, this bit may be set using a vendor specific mechanism,<br>This bit shall only be set if the Secure Simple Pairing (Controller Support) bit is set. |
| Encapsulated PDU | This feature indicates whether the device is capable of supporting the Encapsulated mechanism as defined in Section 4.1.12.1, “Sending an Encapsulated PDU,” on page 239.<br>This feature shall be set if Secure Simple Pairing is supported. |
| Variable Inquiry TX Power Response Level | This feature indicates whether the device is capable of setting the TX power level for Inquiry response. |

*Table 3.1: Feature Definitions*

## 3.3 FEATURE 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 least significant bit in byte 2 bit 4 and the most significant bit in byte 2 bit 6. All removed, unknown. or unassigned feature bits shall be set to 0 and ignored upon receipt.

| No. | Supported feature | Byte | Bit |
|---|---|---|---|
| 0 | 3 slot packets | 0 | 0 |
| 1 | 5 slot packets | 0 | 1 |
| 2 | Encryption | 0 | 2 |
| 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 | Channel quality driven data rate (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 (least significant bit) | 2 | 4 |
| 21 | Flow control lag (middle bit) | 2 | 5 |
| 22 | Flow control lag (most significant bit) | 2 | 6 |
| 23 | Broadcast Encryption | 2 | 7 |
| 24 | Reserved | 3 | 0 |
| 25 | Enhanced Data Rate ACL 2 Mbps mode | 3 | 1 |

*Table 3.2: Feature mask definitions (page 0)*



LifeScan, Inc. Ex. 1019
Page 390 of 1420
IPR2025-00103

| No. | Supported feature | Byte | Bit |
|---|---|---|---|
| 26 | Enhanced Data Rate ACL 3 Mbps mode | 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 |
| 36 | AFH classification slave | 4 | 4 |
| 37 | Reserved | 4 | 5 |
| 38 | Reserved | 4 | 6 |
| 39 | 3-slot Enhanced Data Rate ACL packets | 4 | 7 |
| 40 | 5-slot Enhanced Data Rate ACL packets | 5 | 0 |
| 41 | Sniff subrating | 5 | 1 |
| 42 | Pause encryption | 5 | 2 |
| 43 | AFH capable master | 5 | 3 |
| 44 | AFH classification master | 5 | 4 |
| 45 | Enhanced Data Rate eSCO 2 Mbps mode | 5 | 5 |
| 46 | Enhanced Data Rate eSCO 3 Mbps mode | 5 | 6 |
| 47 | 3-slot Enhanced Data Rate eSCO packets | 5 | 7 |
| 48 | Extended Inquiry Response | 6 | 0 |
| 51 | Secure Simple Pairing | 6 | 3 |
| 52 | Encapsulated PDU | 6 | 4 |
| 53 | Erroneous Data Reporting | 6 | 5 |
| 54 | Non-flushable Packet Boundary Flag | 6 | 6 |
| 56 | Link Supervision Timeout Changed Event | 7 | 0 |
| 57 | Inquiry Response TX Power Level | 7 | 1 |
| 63 | Extended features | 7 | 7 |

*Table 3.2: Feature mask definitions (page 0)*

| NO | Supported Feature | Byte | Bit |
|---|---|---|---|
| 64 | Secure Simple Pairing (Host Support) | 0 | 0 |

*Table 3.3: Extended feature mask definition (page 1)*

## 3.4 LINK MANAGER INTEROPERABILITY POLICY

Link managers of any version will interpreted using the lowest common subset of functionality by reading the LMP features mask (defined in Table 3.2 on page 220).

An optional LMP PDU shall only be sent to a device if the corresponding feature bit is set in its feature mask. The exception to this are certain PDUs (see Section 4.1.1 on page 223) which 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.



LifeScan, Inc. Ex. 1019
Page 392 of 1420
IPR2025-00103

# 4 PROCEDURE RULES

## 4.1 CONNECTION CONTROL

### 4.1.1 Connection establishment

After the paging procedure, LMP procedures with for clock offset request, LMP version, supported features, name request and detach may then be initiated.

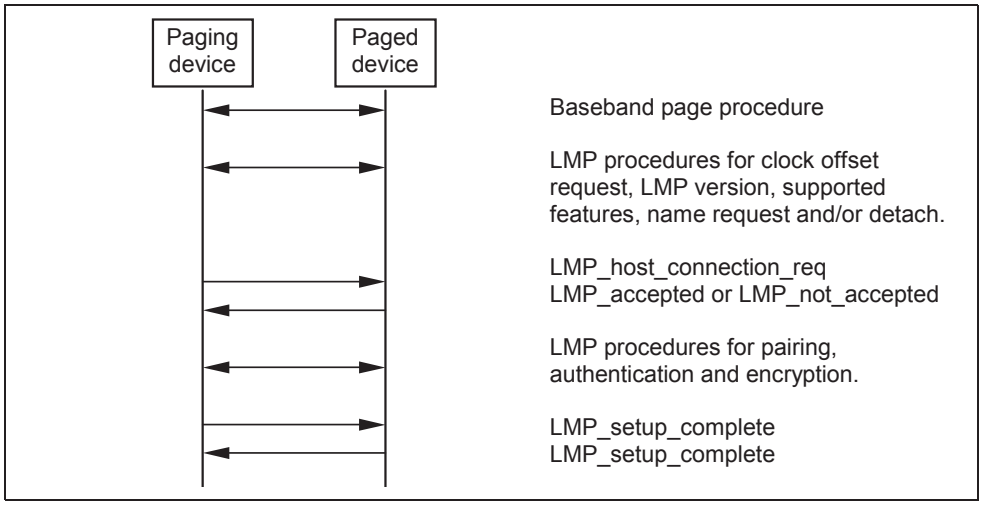



*Figure 4.1: Connection establishment.*

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 Section 4.4.2 on page 275, 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 which 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 LMP_host_connection_req it shall immediately disconnect the link using the mechanism described in Section 4.1.2 on page 224.



LifeScan, Inc. Ex. 1019
Page 393 of 1420
IPR2025-00103

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.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_host_connection_req | - |
| M | LMP_setup_complete | - |

*Table 4.1: Connection establishment PDU*

### 4.1.2 Detach

The connection between two Bluetooth 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.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_detach | error code |

*Table 4.2: Detach PDU*

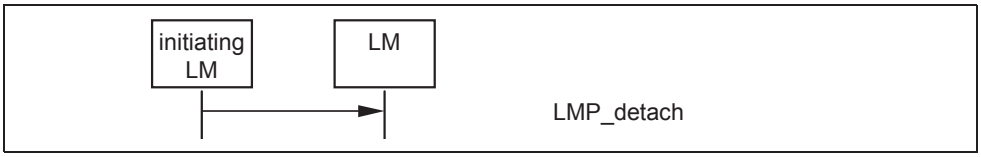
The initiating LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101). The initiating LM then queues the LMP_detach for transmission and it shall start a timer for $6*T_{poll}$ slots where $T_{poll}$ is the poll interval for the connection. If the initiating LM receives the baseband acknowledgement 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 re-used 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 re-used if the initiating LM is the master.

When the receiving LM receives the LMP_detach, 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 re-used immediately. If the receiver never receives the LMP_detach then a link supervision timeout will occur, the link will be detached, and the LT_ADDR may be re-used immediately.

If at any time during this or any other LMP sequence the Link supervision time-out expires then the link shall be terminated immediately and the LT_ADDR(S) may be re-used immediately.

If the connection is in hold mode, the initiating LM shall wait for hold mode to end before initiating the procedure defined above. 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 seconds) the procedure defined above shall be initiated anyway.




*Sequence 1: Connection closed by sending LMP_detach.*

### 4.1.3 Power control

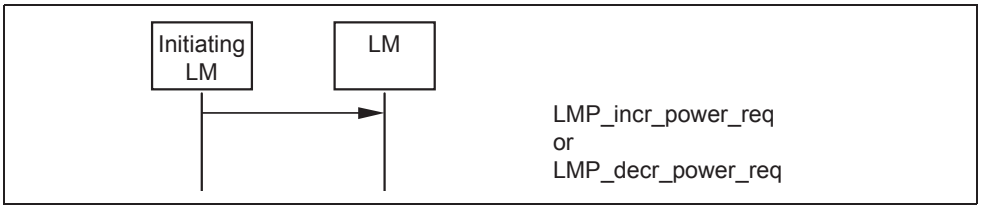
If the received signal characteristics differs too much from the preferred value of a Bluetooth device, it may request an increase or a decrease of the other device's TX power. The power adjustment requests may be made at anytime following a successful baseband paging procedure.

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 and if the recipient does not support power control it is allowed to send LMP_max_power in response to LMP_incr_power_req and LMP_min_power in response to LMP_decr_power_req. Another possibility is to send LMP_not_accepted 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. See Radio Specification Section 3, on page 32 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 only affect 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.

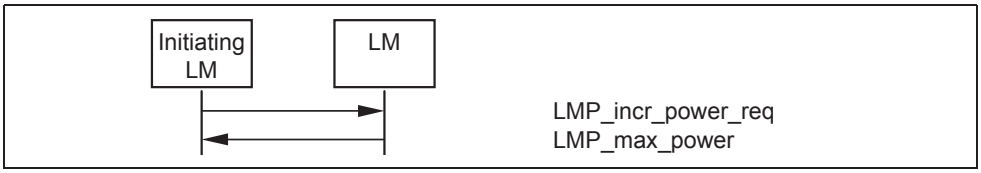
| M/O | PDU | Contents |
|---|---|---|
| O(9) | LMP_incr_power_req | for future use (1 Byte) |
| O(9) | LMP_decr_power_req | for future use (1 Byte) |
| O(18) | LMP_max_power | - |
| O(18) | LMP_min_power | - |

*Table 4.3: Power control PDU*

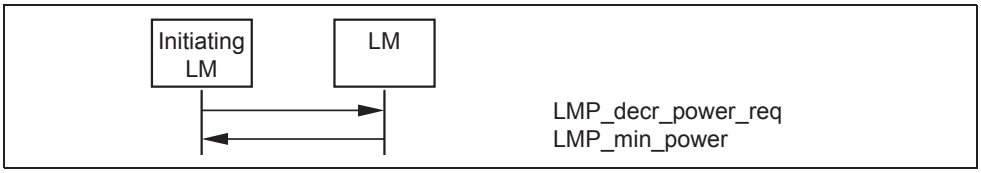



*Sequence 2: A device requests a change of the other device's TX power.*

If the receiver of LMP_incr_power_req is at maximum power LMP_max_power shall be returned. The device shall only request an increase again after having requested a decrease at least once. If the receiver of LMP_decr_power_req is at minimum power then LMP_min_power shall be returned and the device shall only request a decrease after having requested an increase at least once.




*Sequence 3: The TX power cannot be increased.*




*Sequence 4: The TX power cannot be decreased.*

One byte is reserved in LMP_incr/decr_power_req for future use. The parameter value shall be 0x00 and ignored upon receipt.



LifeScan, Inc. Ex. 1019
Page 396 of 1420
IPR2025-00103

### 4.1.4 Adaptive frequency hopping

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 only be used during the connection state.

| M/O | PDU | Contents |
|---|---|---|
| O(35) Rx<br>O(43) Tx | LMP_set_AFH | AFH_Instant,<br>AFH_Mode,<br>AFH_Channel_Map |

*Table 4.4: AFH PDU*

The LMP_set_AFH PDU contains three parameters: AFH_Instant, AFH_Mode, and AFH_Channel_Map. The parameter, AFH_Instant, specifies the instant at which the hopset switch will become effective. This is specified as a Bluetooth Clock value of the master's clock, that 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 hours of the current clock value. The parameter AFH_Mode, specifies whether AFH shall be enabled or disabled. The parameter AFH_Channel_Map, specifies the set of channels that shall be used if AFH is enabled.

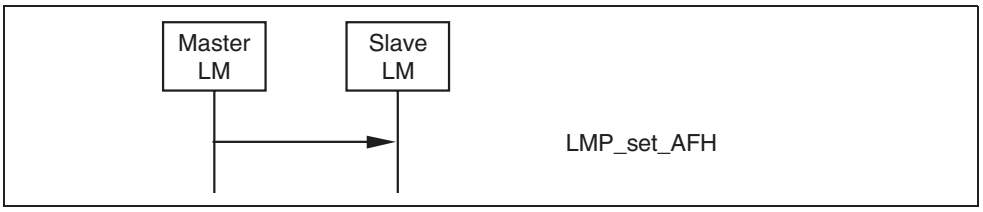
When the LMP_set_AFH PDU is received the AFH instant shall be compared with the current Bluetooth clock value. If it is in the past then the AFH_instant has passed and the slave shall immediately configure the hop selection kernel (see Baseband Specification, Section 2.6.3, on page 84) with the new AFH_mode and AFH_channel_map 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.

The master shall not send a new LMP_set_AFH PDU to a slave until it has received the baseband acknowledgement for any previous LMP_set_AFH addressed to that slave and the instant has passed.

Role switch while AFH is enabled shall follow the procedures define by Baseband Specification, Section 8.6.5, on page 167.

#### *4.1.4.1 Master enables AFH*

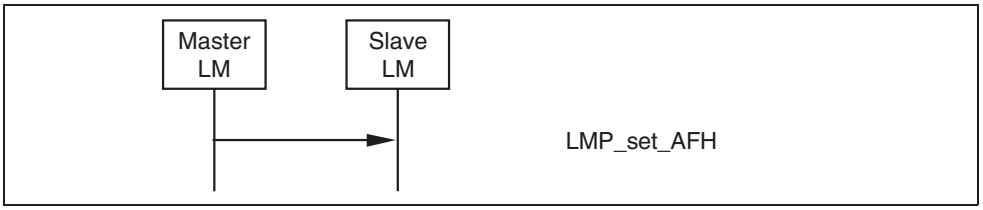
Prior to enabling AFH the master LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101). The master shall then enable AFH on a physical link by sending the LMP_set_AFH PDU with AFH_mode set to AFH_enabled, the AFH_channel_map parameter containing the set of used and unused channels, and an AFH_instant. 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 baseband acknowledgement has been received and the AFH_instant has passed. Once the baseband acknowledgement has been received the master shall restart transmission on the ACL-U logical link.




*Sequence 5: Master Enables AFH.*

#### *4.1.4.2 Master disables AFH*

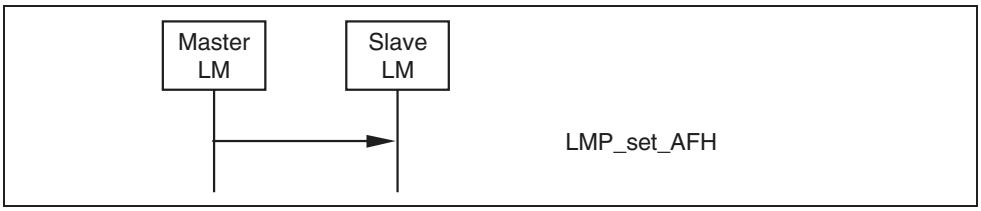
Prior to disabling AFH the master LM shall pause traffic on the ACL-U logical link (Baseband Specification, Section 5.3.1, on page 101). The master shall then disable AFH operation on a physical link by sending the LMP_set_AFH PDU with AFH_mode set to AFH_disabled and an AFH_instant. The AFH_channel_map 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 baseband acknowledgement has been received and the AFH_instant has passed. Once the baseband acknowledgement has been received the master shall restart transmission on the ACL-U logical link.




*Sequence 6: Master disables AFH.*

#### *4.1.4.3 Master updates AFH*

A master shall update the AFH parameters on a physical link by sending the LMP_set_AFH PDU with AFH_mode set to AFH_enabled, an AFH_instant and a new AFH_channel_map. The master shall consider the slave to have the updated AFH parameters once the baseband acknowledgement has been received and the AFH_instant has passed.




*Sequence 7: Master Updates AFH.*

#### *4.1.4.4 AFH operation in Park, Hold, and Sniff modes*

A slave in Park, Hold or Sniff shall retain the AFH_mode and AFH_channel_map prior to entering those modes. A master may change the AFH_mode while a slave is in sniff.

A master that receives a request from an AFH_enabled slave to enter Park, Hold, or Sniff 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 signaling, for Park, Hold or Sniff initiation, once the baseband acknowledgement for the LMP_set_AFH PDU has been received. Optionally, the master may delay the continuation of this LMP signaling 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, Hold or Sniff and decides not to 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 signaling.

### 4.1.5 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 5.2 on page 313 for the detailed format of the AFH_Channel_Classification parameter. When one channel in the $n^{th}$ channel pair is good and the other channel is unknown the $n^{th}$ channel pair shall be reported as good. When one channel in the $n^{th}$ channel pair is bad and the other is unknown the $n^{th}$ 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.

| M/O | PDU | Contents |
|---|---|---|
| O(36) Rx<br>O(44) Tx | LMP_channel_classification_req | AFH_Reporting_Mode,<br>AFH_Min_Interval,<br>AFH_Max_Interval |
| O(36) Tx<br>O(44) Rx | LMP_channel_classification | AFH_Channel_Classification |

*Table 4.5: Channel classification reporting PDU*

The LMP_channel_classification_req PDU contains three parameters: AFH_Reporting_Mode, AFH_Min_Interval, and AFH_Max_Interval. In the AFH_reporting_disabled state, the slave shall not generate any channel classification reports. The parameter AFH_min_interval, 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 parameter AFH_max_interval, 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_max_interval shall be equal to or larger than AFH_min_interval.

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.

AFH_reporting_mode is implicitly set to the AFH_reporting_disabled state when any of the following occur:

- Establishment of a connection at the baseband level
- Master-slave role switch
- Entry to park state operation

- Entry to hold mode operation

AFH_reporting_mode is implicitly restored to its former value when any of the following occur:

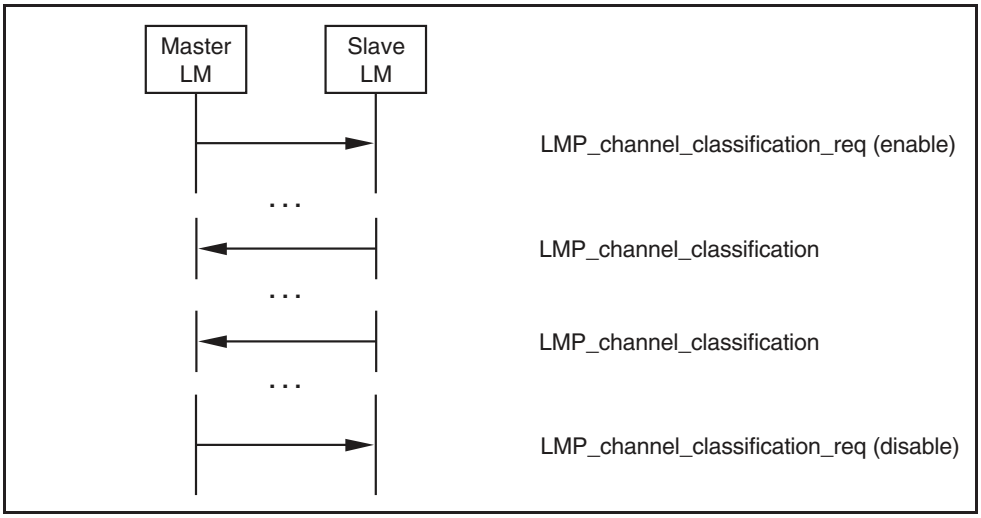
- Exit from park state operation
- Exit from hold mode
- Failure of Master-slave role switch

#### *4.1.5.1 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.



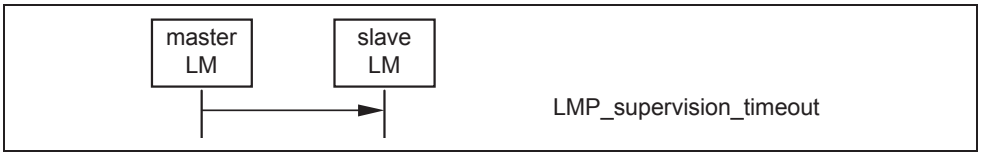

*Sequence 8: Channel classification reporting.*

### 4.1.6 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 Baseband Specification, Section 3.1, on page 90.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_supervision_timeout | supervision timeout |

*Table 4.6: Set supervision timeout PDU*




*Sequence 9: Setting the link supervision timeout.*



LifeScan, Inc. Ex. 1019
Page 402 of 1420
IPR2025-00103

### 4.1.7 Channel quality driven data rate change (CQDDR)

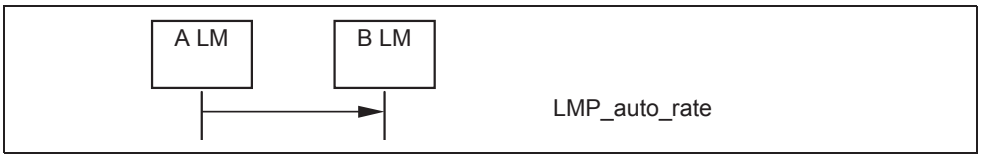
The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput. Device A sends the LMP_auto_rate PDU once to notify device B to enable this feature. Once enabled, device B may change the packet type(s) that A transmits by sending the LMP_preferred_rate PDU. This PDU has a parameter which 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 max slots (see Section 4.1.10 on page 237) even if the preferred size is greater than this value.

The data rate parameter includes the preferred rate for Basic Rate and Enhanced Data Rate modes. When operating in Basic Rate mode, the device shall use bits 0-2 to determine the preferred data rate. When operating in Enhanced Data Rate mode, the device shall use bits 3-6 to determine the preferred data rate. For devices that support Enhanced Data Rate, the preferred rates for both Basic Rate and Enhanced Data Rate modes shall be valid at all times.

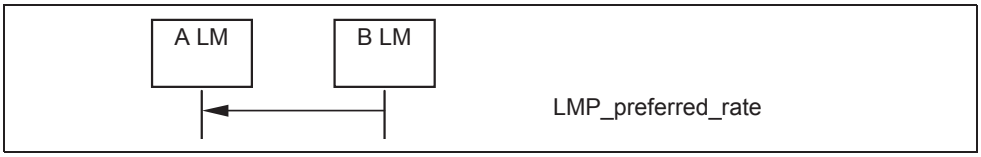
These PDUs may be sent at any time after connection setup is completed.

| M/O | PDU | Contents |
|---|---|---|
| O(10) | LMP_auto_rate | - |
| O(10) | LMP_preferred_rate | data rate |

*Table 4.7: Qualifty-driven change of the data rate PDU*




*Sequence 10: A notifies B to enable CQDDR*




*Sequence 11: B sends A a preferred packet type*

### 4.1.8 Quality of service (QoS)

The LM provides QoS capabilities. A poll interval, $T_{poll}$, that 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 Baseband Specification, Section 8.6.1, on page 162 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 Bluetooth device is a member, and during critical baseband 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.

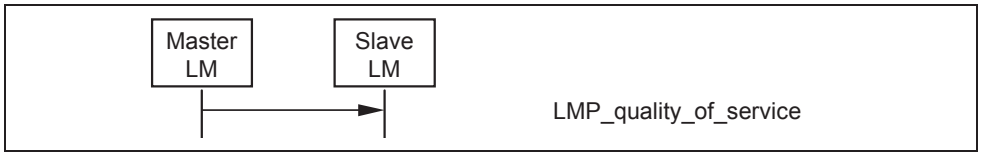
Master and slave negotiate the number of repetitions for broadcast packets ($N_{BC}$), see Baseband Specification, Section 7.6.5, on page 143.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_quality_of_service | poll interval<br>$N_{BC}$ |
| M | LMP_quality_of_service_req | poll interval<br>$N_{BC}$ |

*Table 4.8: Quality of Service PDU*

#### *4.1.8.1 Master notifies slave of the quality of service*

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.

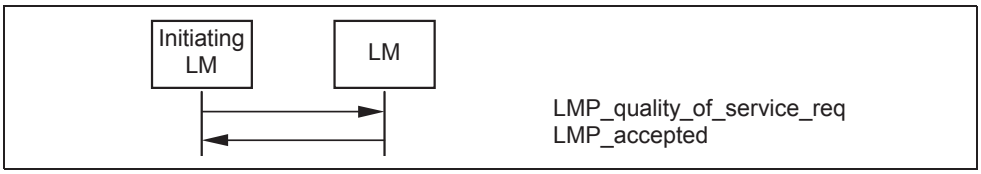



*Sequence 12: Master notifies slave of quality of service.*

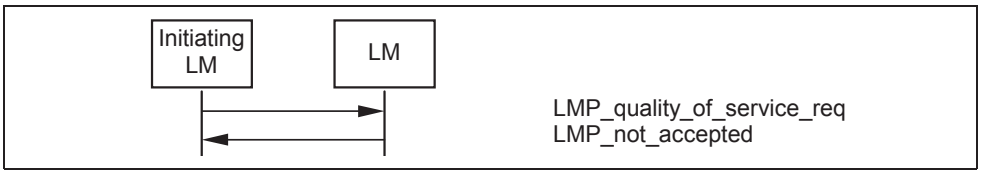
#### *4.1.8.2 Device requests new quality of service*

Either the master or the slave may request a new poll interval and $N_{BC}$ by sending an LMP_quality_of_service_req PDU to the slave. 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 quality of service 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 L2CAP "Quality of Service (QoS) Option" on page 58 in Vol. 3).




*Sequence 13: Device accepts new quality of service*




*Sequence 14: Device rejects new quality of service.*



LifeScan, Inc. Ex. 1019
Page 405 of 1420
IPR2025-00103

### 4.1.9 Paging scheme parameters

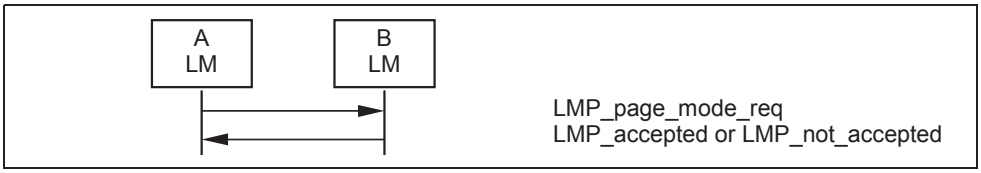
LMP provides a means to negotiate the paging scheme parameters that are used the next time a device is paged.

| M/O | PDU | Contents |
|---|---|---|
| O(17) | LMP_page_mode_req | paging scheme<br>paging scheme settings |
| O(17) | LMP_page_scan_mode_req | paging scheme<br>paging scheme settings |

*Table 4.9: Paging scheme request PDU*

#### *4.1.9.1 Page mode*

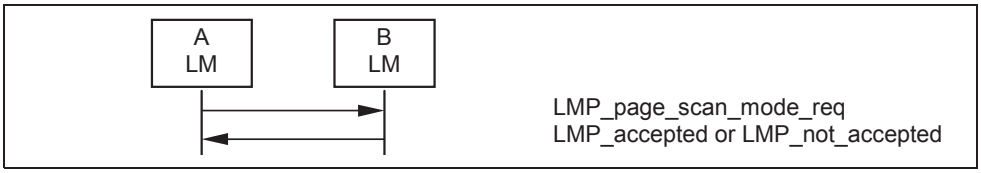
This procedure is initiated from device A and negotiates the paging scheme used when device A pages device B. Device A proposes a paging scheme including the parameters for this scheme and device B can accept or reject. On rejection the old setting will not be changed. A request to switch to a reserved paging scheme shall be rejected.




*Sequence 15: Negotiation for page mode.*

#### *4.1.9.2 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 reject the old setting is not changed. A request specifying the mandatory scheme shall be accepted. A request specifying a non-mandatory 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.



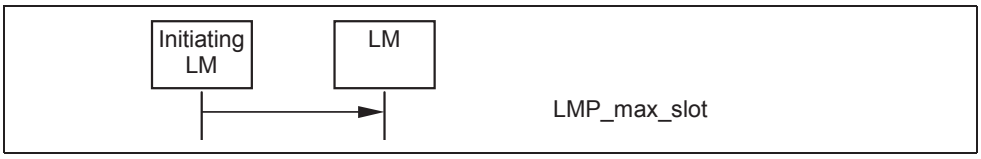

*Sequence 16: Negotiation for page scan mode*

### 4.1.10 Control of multi-slot packets

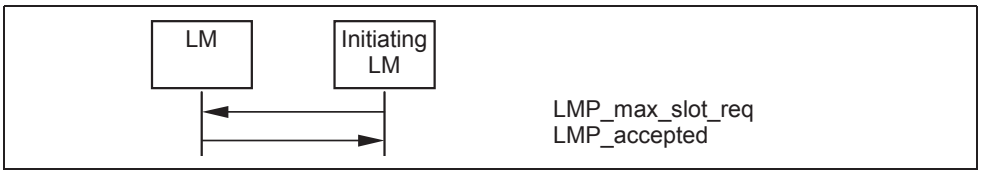
The number of consecutive slots used by a device on an ACL-U logical link can be limited. It does not affect traffic on the eSCO links where the packet sizes are defined as part of link setup. A device allows the remote device to use a maximum number of slots by sending the PDU LMP_max_slot providing max slots as parameter. Each device can request to use a maximal number of slots by sending the PDU LMP_max_slot_req providing max slots as 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 anytime after connection setup is completed.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_max_slot | max slots |
| M | LMP_max_slot_req | max slots |

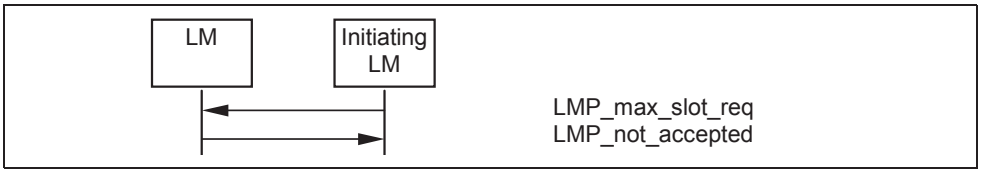
*Table 4.10: Multi-slot packet control PDU*




*Sequence 17: Device allows Remote Device to use a maximum number of slots.*




*Sequence 18: Device requests a maximum number of slots. Remote Device accepts.*




*Sequence 19: Device requests a maximum number of slots. Remote Device rejects.*

### 4.1.11 Enhanced Data Rate

A device may change the packet type table, ptt, to select which if any of the optional modulation schemes are to be used on an ACL logical transport.

Either the master or the slave may request a new packet type table and therefore the modulation scheme to be used on this ACL link. After a new baseband

connection, as a result of page or page scan, the default value for ptt shall be 0.

The change of the modulation mode for an ACL logical transport shall not affect the packet types used for an associated SCO logical transport on the same LT_ADDR.

Note: Enhanced Data Rate eSCO links are negotiated using the LMP eSCO link_req as described in section 4.6.2.

Before changing the packet type table, the initiator shall finalize the transmission of the current ACL packet with ACL-U information and shall stop ACL-U transmissions. It shall then send the LMP_packet_type_table_req PDU.

If the receiver rejects the change, then it shall respond with an LMP_not_accepted_ext PDU.

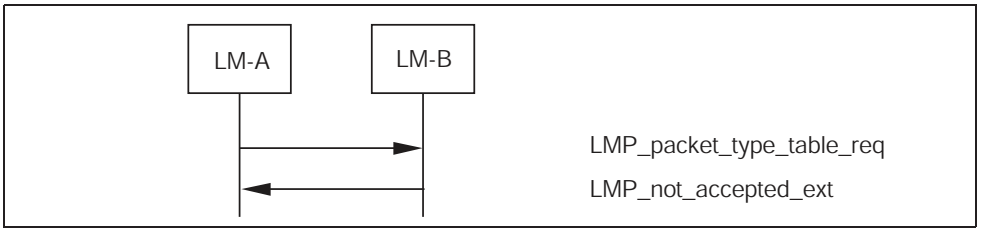
If the receiver accepts the change, then it shall finalize the transmission of the current ACL packet with ACL-U information and shall stop ACL-U transmissions, it shall change to the new packet type table and shall respond with an LMP_accepted_ext PDU. When it receives the baseband level acknowledgement for the LMP_accepted_ext PDU it shall restart ACL-U transmissions.

When the initiator receives an LMP_not_accepted_ext PDU the initiator shall restart ACL-U transmissions.

When the initiator receives an LMP_accepted_ext PDU it shall change the packet type table and restart ACL-U transmissions.

| M/O | PDU | Contents |
|---|---|---|
| O(25) | LMP_packet_type_table_req | packet type table |

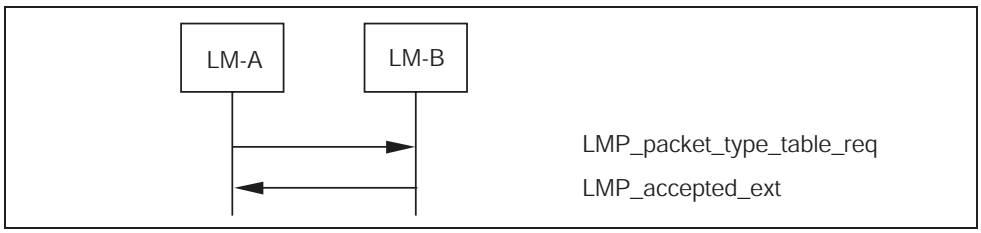
*Table 4.11: Enhanced Data Rate PDUs*




*Sequence 20: Packet type table change is rejected.*



LifeScan, Inc. Ex. 1019
Page 408 of 1420
IPR2025-00103




*Sequence 21: Packet type table change is accepted.*

### 4.1.12 Encapsulated LMP PDUs

Some transactions require sending LMP payload data that is longer than 16 octets. To enable a link manager to send a large PDU, an encapsulated LMP PDU is defined. An encapsulated LMP PDU is composed of a minimum of two LMP messages, a header PDU and one or more payload PDUs.

| M/O | PDU | Contents |
|---|---|---|
| O(52) | LMP_encapsulated_header | encapsulated major type<br>encapsulated minor type<br>encapsulated payload length |
| O(52) | LMP_encapsulated_payload | encapsulated data |

*Table 4.12: Encapsulated LMP PDUs*

#### *4.1.12.1 Sending an Encapsulated PDU*

The LMP_encapsulated_header PDU shall be sent by the initiating device when it needs to send an encapsulated PDU. This PDU shall be either accepted or rejected using the LMP_accepted or LMP_not_accepted PDUs. If the major and minor types are not supported the PDU shall be rejected with error code Unsupported LMP Parameter Value. If the LMP_encapsulated_header PDU is accepted, then one or more LMP_encapsulated_payload PDUs will be sent with the encapsulated data sent in sequence, 16 octets at a time, or if this is last packet, the correct number of octets and then zero padded.

Each LMP_encapsulated_payload PDU shall be accepted or rejected. If the LMP_encapsulated_header PDU is rejected, then the opcode in the LMP_not_accepted PDU shall be the opcode for the LMP_encapsulated_header and not the encapsulated major type or encapsulated minor type. If the LMP_encapsulated_payload PDU is rejected, then the opcode in the LMP_not_accepted PDU shall be the opcode for the LMP_encapsulated_payload.

A responding device may reject the final LMP_encapsulated_payload PDU after accepting the LMP_encapsulated_header PDU. This is so that the link

manager can still reject an encapsulated message after all the data has been received.

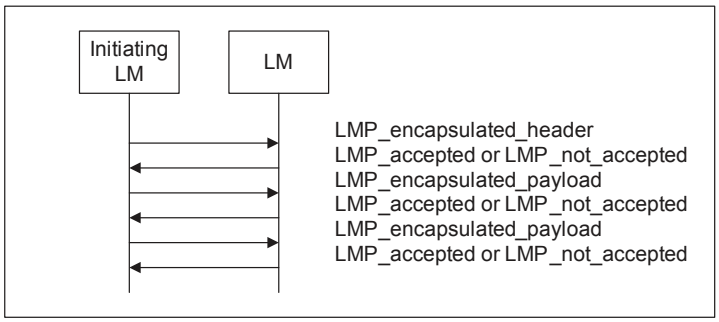



*Figure 4.2: Sequence 1: Sending an encapsulated PDU*

Between sending an LMP_encapsulated_header PDU and an LMP_encapsulated_payload PDU, or between each of the LMP_encapsulated_payload PDU, either device shall be able to send the following LMP PDUs without causing a "different transaction collision" error.

- LMP_channel_classification
- LMP_decr_power_req
- LMP_detach
- LMP_incr_power_req
- LMP_max_power
- LMP_max_slot
- LMP_min_power
- LMP_preferred_rate
- LMP_set_AFH

## 4.2 SECURITY

### 4.2.1 Authentication

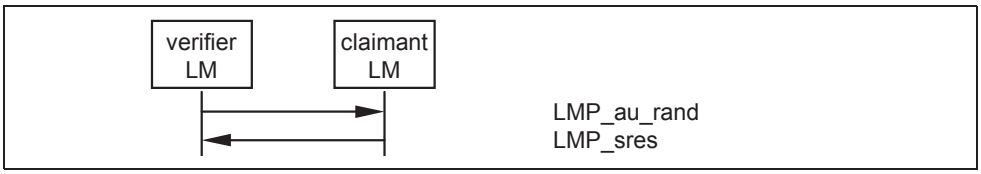
The authentication procedure is based on a challenge-response scheme as described in [Part H] Section 3.2.2 on page 862. The verifier sends an LMP_au_rand PDU that contains a random number (the challenge) to the claimant. The claimant calculates a response, that is a function of this challenge, the claimant's BD_ADDR and a secret key. The response is sent back to the verifier, that checks if the response was correct or not. The response shall be calculated as described in [Part H] Section 6.1 on page 881. A successful calculation of the authentication response requires that two devices share a secret key. This key is created as described in Section 4.2.2 on page 242. Both the master and the slave can be verifiers.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_au_rand | random number |
| M | LMP_sres | authentication response |

*Table 4.13: Authentication PDUs*

### *4.2.1.1 Claimant has link key*

If the claimant has a link key associated with the verifier, it shall calculate the response and sends it to the verifier with LMP_sres. 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 Section 4.1.2 on page 224.



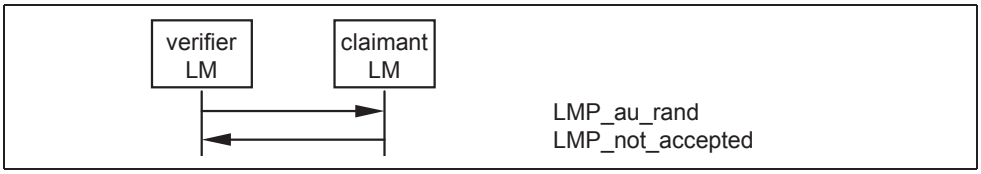

*Sequence 22: Authentication. Claimant has link key.*

Upon reception of an LMP_au_rand, an LM shall reply with LMP_sres before initiating its own authentication.

Note: There can be concurrent requests caused by the master and slave simultaneously initiating an authentication. The procedures in Section 2.5.1 on page 212 assures that devices will not have different Authenticated Ciphering Offset (ACO, see [Part H] Section 6.1 on page 881) when they calculate the encryption key.

### *4.2.1.2 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.




*Sequence 23: Authentication fails. Claimant has no link key.*

#### *4.2.1.3 Repeated attempts*

The scheme described in [Part H] Section 5.1 on page 880 shall be applied when an authentication fails. This will prevent an intruder from trying a large number of keys in a relatively short time.

### 4.2.2 Pairing

When two devices do not have a common link key an initialization key ($K_{init}$) shall be created using either the pairing or Secure Simple Pairing procedures. When pairing is used, $K_{init}$ shall be created based on a PIN, and a random number and a BD_ADDR. $K_{init}$ shall be created as specified in [Part H] Section 6.3 on page 885. 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 Section 4.2.2.1 on page 242 - Section 4.2.2.5 on page 245. The other device is referred to as the "responding LM" or "responder". The PDUs used in the pairing procedure are:

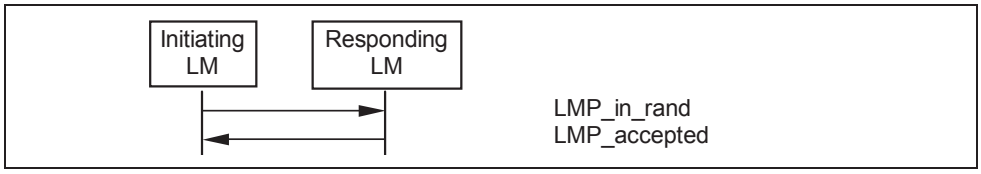
| M/O | PDU | Contents |
|---|---|---|
| M | LMP_in_rand | random number |
| M | LMP_au_rand | random number |
| M | LMP_sres | authentication response |
| M | LMP_comb_key | random number |
| M | LMP_unit_key | key |

*Table 4.14: Pairing PDUs*

All sequences described in Section 4 on page 223, 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 shall be used for all subsequent sequences.

#### *4.2.2.1 Responder accepts pairing*

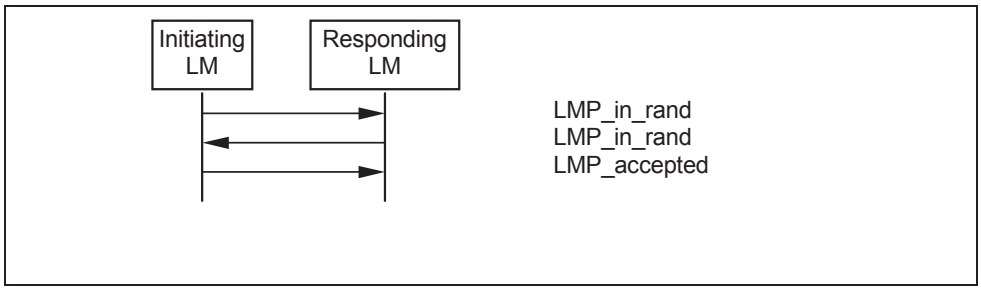
When the initiator sends an LMP_in_rand PDU and the responder shall reply with an LMP_accepted PDU. Both devices shall then calculate $K_{init}$ based on the BD_ADDR of the responder and the procedure continues with creation of the link key; see Section 4.2.2.4 on page 244.




*Sequence 24: Pairing accepted. Responder has a variable PIN. Initiator has a variable or fixed PIN.*

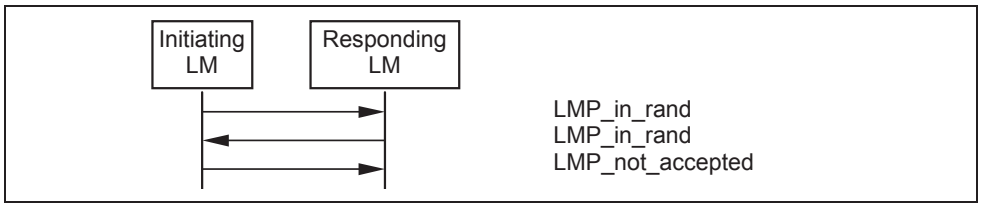
#### *4.2.2.2 Responder has a fixed PIN*

If the responder has a fixed PIN 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 Section 4.2.2.4 on page 244.




*Sequence 25: 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 shall be rejected by the initiator sending an LMP_not_accepted PDU with the error code *pairing not allowed*.



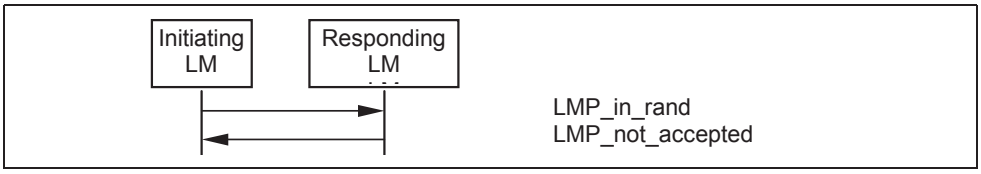

*Sequence 26: Both devices have a fixed PIN.*

#### *4.2.2.3 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.



LifeScan, Inc. Ex. 1019
Page 413 of 1420
IPR2025-00103




*Sequence 27: Responder rejects pairing.*

#### *4.2.2.4 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 Section 4.2.3 on page 245 and Section 4.2.4 on page 246. The link key created in the pairing procedure will either be 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, 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 [Part H] Section 3.2 on page 861.

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.

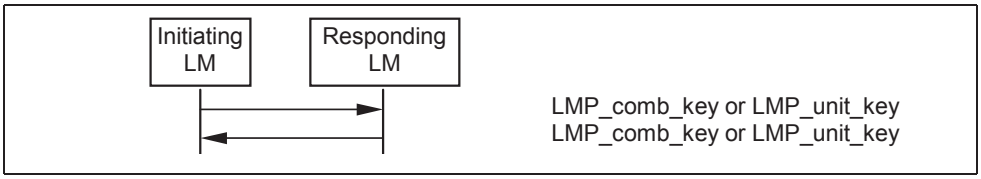
The use of unit keys is deprecated since it is implicitly insecure.

When the new link key has been created mutual authentication shall be performed to confirm that the same link key has been created in both devices. After mutual authentication, the initiating device shall pause and immediately resume encryption to produce a new encryption key. Note that this will cause a new encryption key to be generated from the ACO created during the mutual authentication process, and the random number sent in the LMP_start_encryption_req PDU which occurs in response to the resumption of encryption.

After mutual authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key. Note that this will cause a new encryption key to be generated from the ACO created during the mutual authentication process, and the random number sent in the LMP_start_encryption_req PDU which occurs in response to the resumption of encryption.



LifeScan, Inc. Ex. 1019
Page 414 of 1420
IPR2025-00103




*Sequence 28: Creation of the link key.*

#### *4.2.2.5 Repeated attempts*

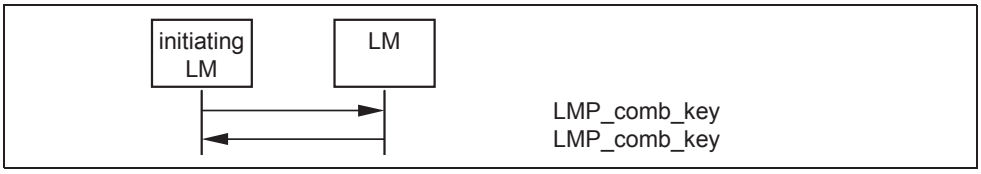
When the authentication after creation of the link key fails because of an incorrect authentication response, the same scheme as in Section 4.2.1.3 on page 242 shall be used. This prevents an intruder from trying a large number of different PINs in a relatively short time.

### 4.2.3 Change link key

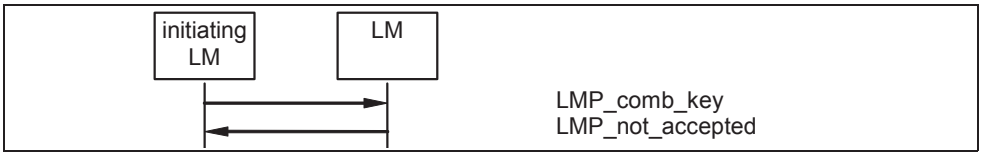
If the link key is derived from combination keys and the current link is the semi-permanent link key, the link key can be changed. If the link key is a unit key, the devices shall go through the pairing procedure in order to change the link key. The contents of the LMP_comb_key PDU is protected by a bitwise XOR with the current link key.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_comb_key | random number |

*Table 4.15: Change link key PDU*




*Sequence 29: Successful change of the link key*




*Sequence 30: 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 Section 4.2.4 on page 246.

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. After mutual authentication, if encryption is enabled, the initiating device shall pause and immediately resume encryption to produce a new encryption key. Note that this will cause a new encryption key to be generated from the ACO created during the mutual authentication process, and the random number sent in the LMP_start_encryption_req PDU which occurs in response to the resumption of encryption. The first authentication in the mutual authentication is performed with the device that initiated change link key as verifier. When finalized an authentication in the reverse direction is performed.

### 4.2.4 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 only be valid for the current connection, see [Part H] Section 3.1 on page 859. 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 only supported if broadcast encryption is supported as indicated by the LMP features mask.

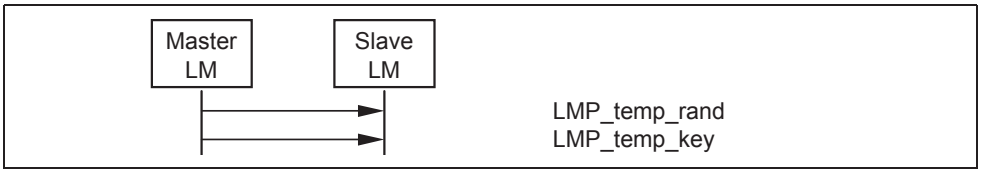
| M/O | PDU | Contents |
|---|---|---|
| O(23) | LMP_temp_rand | random number |
| O(23) | LMP_temp_key | key |
| O(23) | LMP_use_semi_permanent_key | - |

*Table 4.16: Change current link key PDU*

#### *4.2.4.1 Change to a temporary link key*

The master starts by creating the master key $K_{master}$ as specified in Security Specification (EQ 4), on page 866. 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 calculates $K_{master}$, based on OVL, that becomes the current link key. It shall be the current link key until the devices fall back to the semi-permanent link key, see Make the semi-permanent link key the current link key on page 247.

Note: the terminology in this section is the same as used in [Part H] Section 3.2.8 on page 866.




*Sequence 31: Change to a temporary link key*

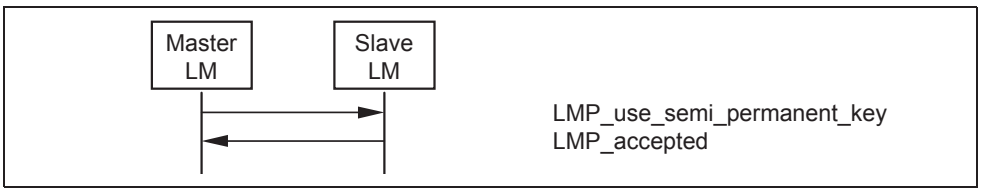
All sequences described in Section 4.2.4.1 on page 246, including the mutual authentication after $K_{master}$ has been created, shall form a single transaction. The transaction ID shall be set to 0.

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. This will allow the procedure to succeed even though one of the devices may be erroneous.

#### *4.2.4.2 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 master stopping encryption using the procedure described in Section 4.2.5.4 on page 251. Encryption may be restarted by the master according to the procedures in Encryption mode on page 248 subsection 3. 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.




*Sequence 32: Link key changed to the semi-permanent link key*

### 4.2.5 Encryption

If at least one authentication has been performed encryption may be used. In order for the master to use the same encryption parameters for all slaves in the piconet it shall issue a temporary key, $K_{master}$. The master shall make this key the current link key for all slaves in the piconet before encryption is started, see Section 4.2.4 on page 246. This is required if broadcast packets are to be encrypted.

| 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 | - |
| O (42) | LMP_pause_encryption_req | |
| O (42) | LMP_resume_encryption_req | |

*Table 4.17: Encryption handling PDU*

All sequences described in Section 4.2.5 shall form a single transaction. The transaction ID from the LMP_encryption_mode_req PDU shall be used for all start encryption and stop encryption sequences.

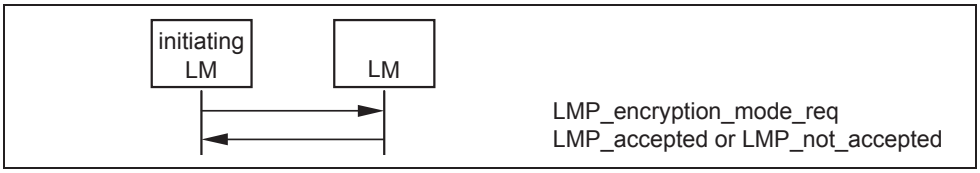
#### *4.2.5.1 Encryption mode*

The master and the slave must agree upon whether to use encryption (encryption mode=1 in LMP_encryption_mode_req) or not (encryption mode=0). If the semi-permanent key is used (Key_Flag=0x00) encryption shall only apply 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 LMP_encryption_mode_req with an encryption mode value of 2 however for backwards compatibility if the LMP_encryption_mode_req 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 Baseband Specification, Section 5.3.1, on page 101). 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 only be resumed after the attempt to encrypt or decrypt the logical transport is completed i.e. at the end of Sequence 33, 34 or 35.

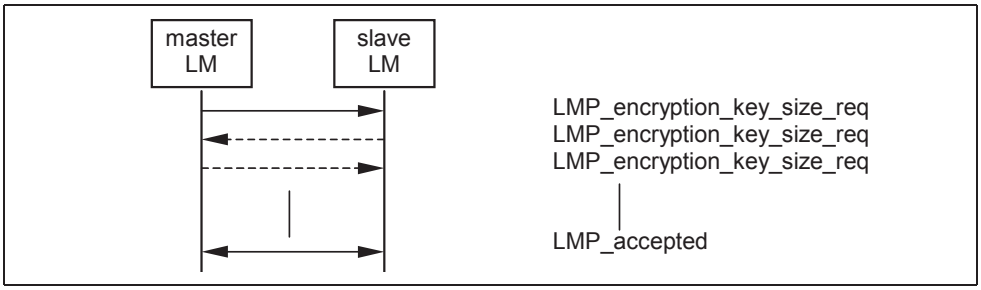



*Sequence 33: Negotiation for encryption mode*

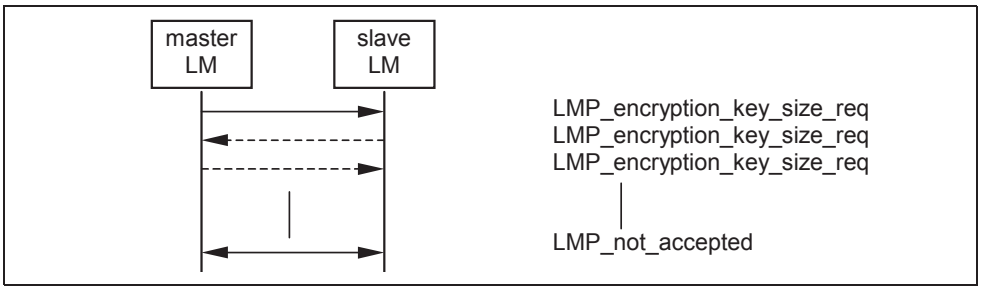
After a device has sent an LMP_encryption_mode_req PDU it shall not send an LMP_au_rand PDU before encryption is started. After a device has received an LMP_encryption_mode_req PDU and sent an LMP_accepted PDU it shall not send an LMP_au_rand PDU before encryption is started. If an LMP_au_rand PDU is 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.

#### *4.2.5.2 Encryption key size*

Note: this section uses the same terms as in [Part H] Section 4.1 on page 869. The master sends an LMP_encryption_key_size_req PDU including the suggested key size $L_{sug, m}$, 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 Section 4.2.5.3 on page 250. 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.




*Sequence 34: Encryption key size negotiation successful*




*Sequence 35: Encryption key size negotiation failed*

### *4.2.5.3 Start encryption*

To start encryption, the master issues the random number EN_RAND and calculates the encryption key. See [Part H] Section 3.2.5 on page 864. The random number shall be the same for all slaves in the piconet when broadcast encryption is used. The master then sends an LMP_start_encryption_req PDU, that includes EN_RAND. The slave shall calculate the encryption key when this message is received and shall acknowledge with an LMP_accepted PDU.

If encryption has been paused, then this sequence shall not be used.




*Sequence 36: Start of encryption*

Starting encryption shall be performed in three steps:

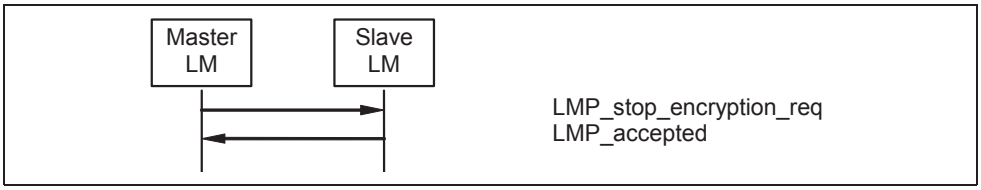
1. Master is configured to transmit unencrypted packets and to receive encrypted packets.
2. Slave is configured to transmit and receive encrypted packets.
3. Master is configured to transmit and receive encrypted packets.

Between step 1 and step 2, master-to-slave transmission is possible. This is when an LMP_start_encryption_req PDU is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3, slave-to-master
transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

#### *4.2.5.4 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 33 in Encryption mode on page 248). If accepted, encryption shall be stopped by the master sending an LMP_stop_encryption_req PDU and the slave shall respond with an LMP_accepted PDU according to Sequence 37.

If encryption has been paused, then this sequence shall not be used.




*Sequence 37: Stop of encryption*

Stopping encryption shall be performed in three steps, similar to the procedure for starting encryption.

1. Master is configured to transmit encrypted packets and to receive unencrypted packets.
2. Slave is configured to transmit and receive unencrypted packets.
3. Master is configured to transmit and receive unencrypted packets.

Between step 1 and step 2 master to slave transmission is possible. This is when an LMP_stop_encryption_req PDU is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3 slave to master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

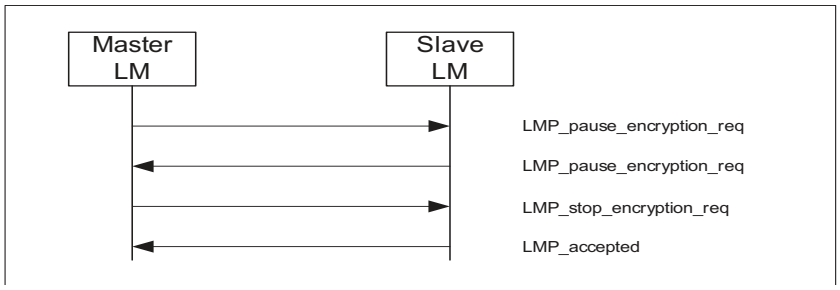
#### *4.2.5.5 Pause encryption*

To pause encryption without disabling encryption, a device shall finalize the transmission of the current ACL-U data packet and then send an LMP_pause_encryption_req PDU (with the transaction ID set to the role of the device at the time the LMP_pause_encryption_req PDU is sent).

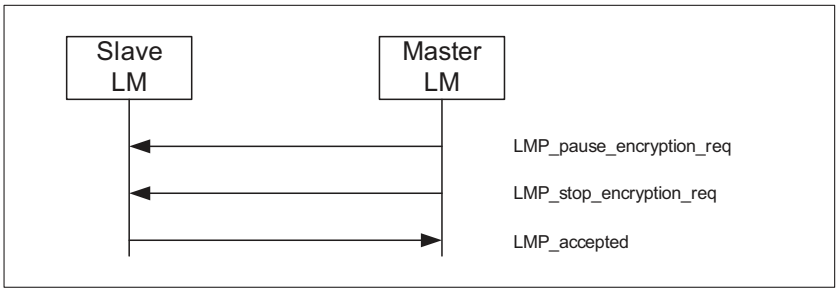
If the responding device is a master, then the master device shall finalize the transmission of the current ACL-U data packet and then respond with an LMP_stop_encryption_req PDU to the slave.

If the responding device is a slave, then the slave device shall finalize the transmission of the current ACL-U data packet and then respond with an LMP_pause_encryption_req PDU. The master device shall respond to the LMP_pause_encryption_req PDU with an LMP_stop_encryption_req PDU to the slave.

When the slave receives the LMP_stop_encryption_req PDU it shall respond with an LMP_accepted PDU.




*Sequence 38: Master-initiated pausing of encryption*




*Sequence 39: Slave-initiated pausing of encryption*

The LMP_pause_encryption_req PDU and LMP_stop_encryption_req PDU shall be rejected only when transaction collision needs to be resolved.

Pausing encryption shall be performed in three steps, similar to the procedure for stopping encryption.

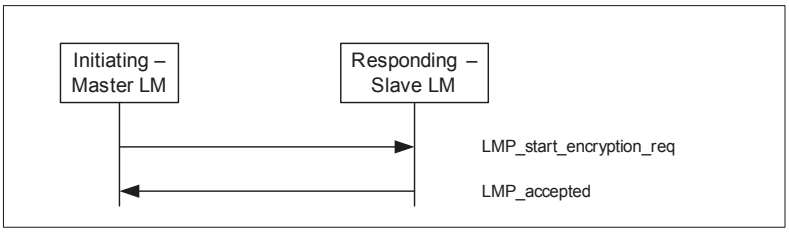
1. Master is configured to transmit encrypted packets and to receive unencrypted packets.
2. Slave is configured to transmit and receive unencrypted packets.
3. Master is configured to transmit and receive unencrypted packets.

Between step 1 and step 2 master-to-slave transmission is possible. This is when the LMP_stop_encryption_req PDU is transmitted from the master to the slave. Step 2 is triggered when the slave receives this message. Between step

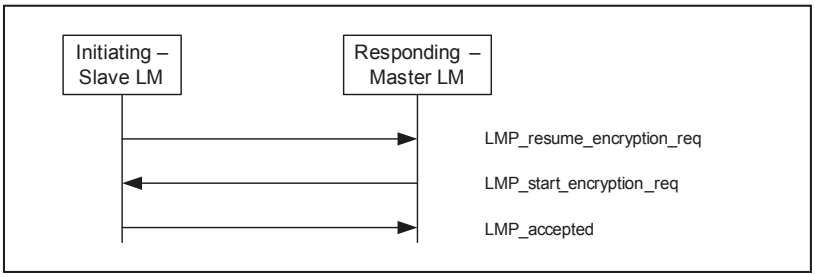
2 and step 3 slave-to-master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

Note: A device can only restart ACL-U data traffic by resuming encryption using the procedures in Section 4.2.5.6.

### *4.2.5.6 Resume Encryption*




*Sequence 40: Initiating Master LM resumes encryption*




*Sequence 41: Initiating Slave LM resume encryption*

If the responding device is a slave, then the slave shall calculate the encryption key and respond with an LMP_accepted PDU.

If the responding device is a master, then the master shall respond with an LMP_start_encryption_req PDU. The slave, upon receiving the LMP_start_encryption_req PDU from the master, shall calculate the encryption key and respond with an LMP_accepted PDU.

The LMP_resume_encryption_req PDU and the LMP_start_encryption_req PDU shall not be rejected.

Resuming encryption shall be performed in three steps, similar to the procedure for starting encryption:

1. Master is configured to transmit unencrypted packets and to receive encrypted packets.
2. Slave is configured to transmit and receive encrypted packets.
3. Master is configured to transmit and receive encrypted packets.

Between step 1 and step 2, master-to-slave transmission is possible. This is when the LMP_start_encryption_req PDU is transmitted from the master. Step 2 is triggered when the slave receives this message. Between step 2 and step 3, slave-to-master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

Note: For a slave initiated starting of encryption, step 1 is not started when the master has received the LMP_resume_encryption_req PDU from the slave, but when the master sends the LMP_start_encryption_req PDU.

#### *4.2.5.7 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 Section 4.2.4 on page 246, encryption shall be paused and resumed after completion, using the procedures in Section 4.2.5 on page 248, Section 4.2.5.1 and Section 4.2.5.2, and Section 4.2.3 on page 245 for the new parameters to be valid. If the Pause Encryption feature is not supported by both devices, encryption shall be stopped and re-started after completion, using the procedures in Section 4.2.5 on page 248, subsections 3-4 for the new parameters to be valid.

#### *4.2.5.8 Encryption key refresh*

The Link Manager shall refresh the encryption key within $2^{28}$ ticks of the Bluetooth clock from the previous start or resume of encryption. To refresh the encryption key, the Link Manager shall Pause encryption using the procedure in Section 4.2.5.5 on page 251 and immediately Resume encryption using the procedure in Section 4.2.5.6 on page 253.



LifeScan, Inc. Ex. 1019
Page 424 of 1420
IPR2025-00103

### 4.2.6 Request supported encryption key size

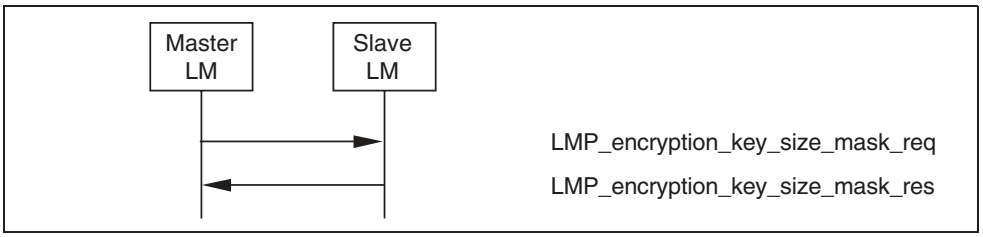
When broadcast encryption is supported via the LMP features mask, it is possible for the master to request a slave's supported encryption key sizes.

| M/O | PDU | Contents |
|---|---|---|
| O(23) | LMP_encryption_key_size_mask_req | |
| O(23) | LMP_encryption_key_size_mask_res | key size mask |

*Table 4.18: Encryption key size request PDU*

The master shall send an LMP_key_size_req PDU to the slave to obtain the slaves supported encryption key sizes.

The slave shall return a bit mask indicating all broadcast encryption key sizes supported. The least significant bit shall indicate support for a key size of 1, the next most significant bit 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.




*Sequence 42: Request for supported encryption key sizes.*

### 4.2.7 Secure Simple Pairing

There are four stages defined in the Secure Simple Pairing LM process:

- IO capabilities exchange
- Public key exchange
- Authentication stage 1
- Authentication stage 2

The devices shall first exchange the IO capabilities to determine the proper algorithm to be used. Three algorithms have been specified: Numeric comparison, Passkey entry, Out of band.

In following sections, the device requesting the IO capabilities is referred to as the "Initiating LM" or "Initiator". The other device is referred to as the "LM" or "Responder". This designation remains throughout the entire Secure Simple Pairing procedure.

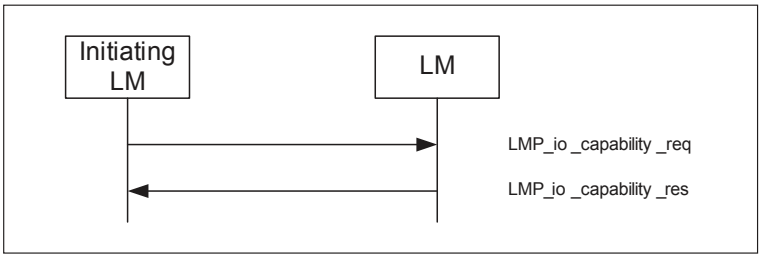
Note that the host shall enable Secure Simple Pairing before the LMP Secure Simple Pairing procedures start.

| M/O | PDU | Contents |
|---|---|---|
| O(51) | LMP_io_capability_req | IO_capabilities, OOB_Authentication_Data, Authentication_Requirements |
| O(51) | LMP_io_capability_res | IO_capabilities, OOB_Authentication_Data, Authentication_Requirements |
| O(51) | LMP_Simple_Pairing_Confirm | Commitment Value |
| O(51) | LMP_Simple_Pairing_Number | Nonce Value |
| O(51) | LMP_Dhkey_check | Confirmation value |
| O(51) | LMP_Numeric_Comparison_Failed | - |
| O(51) | LMP_OOB_Failed | - |
| O(51) | LMP_keypress_notification | Notification Type |
| O(51) | LMP_Passkey_Entry_Failed | - |

#### *4.2.7.1 IO Capability Exchange*

The Link Managers shall use the local and remote IO capabilities to determine which authentication method shall be used.

If Simple_Pairing_Mode is set to enabled, the Initiator shall request the IO capabilities from the Host. The initiator shall send an LMP_io_capability_req PDU to the Responder. If Simple_Pairing_Mode is set to enabled on the responding device, it shall reply with an LMP_io_capability_res PDU containing its IO capabilities description.



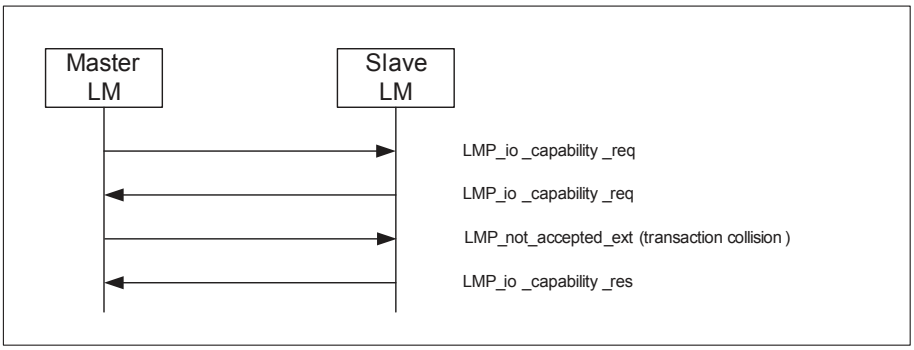

*Sequence 43: IO Capability Exchange*

##### *4.2.7.1.1 IO Capability Exchange Transaction Collision and Resolution*

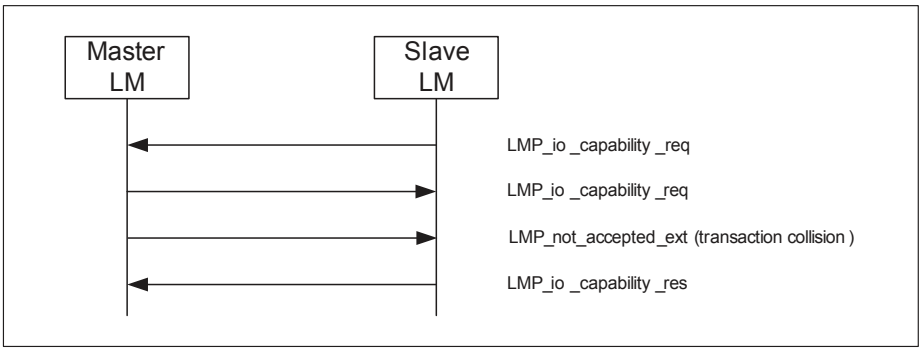
If both Link Managers attempt to become the initiator of the IO capability exchange, the master Link Manager shall send an LMP_not_accepted_ext

PDU with error code 'Transaction Collision'. The slave Link Manager shall respond with the LMP_io_capability_res PDU.

The master LM shall remain the initiating LM for the remainder of the Secure Simple Pairing sequences.




*Sequence 44: IO Capability exchange with transaction Collision (master transmitting LMP_io_capability_req first) and resolution*



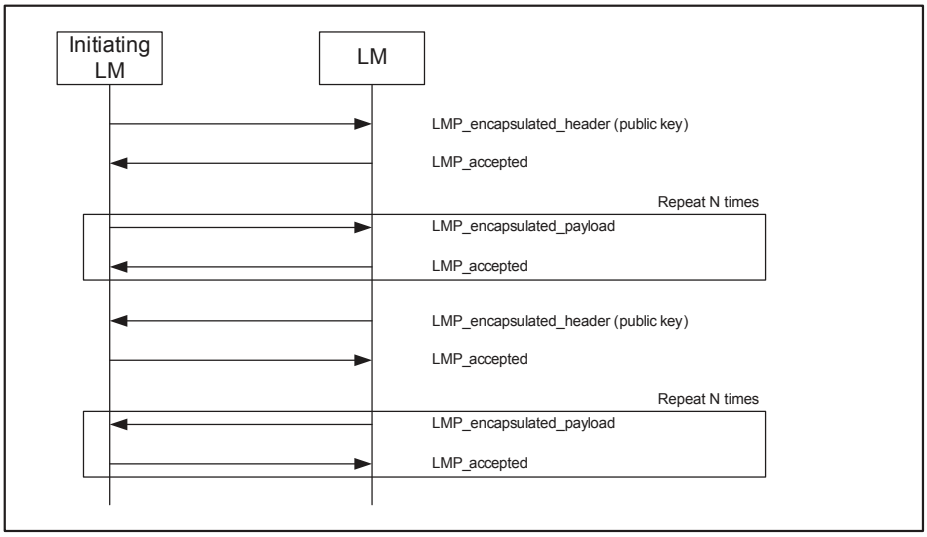

*Sequence 45: IO Capability exchange with transaction Collision (slave transmitting LMP_io_capability_req first) and resolution*

### *4.2.7.2 Public Key Exchange*

Once the IO capabilities are exchanged, public keys shall be exchanged between the two devices.

Since the public key size is 48 bytes long, the exchange shall be done using the LMP LMP_encapsulated_header and LMP_encapsulated_payload PDUs as defined in Section 4.1.12.

The Initiator shall first send its public key, and the Responder shall reply with its public key.




*Sequence 46: Public key exchange*

A public key shall be considered as received when the last LMP_encapsulated_payload has been received and the associated LMP_accepted PDU has been sent.

The device can then start computing its Diffie Hellman Key.

#### *4.2.7.3 Authentication stage 1*

The procedure used in Authentication Stage 1 shall be selected as follows:

- If one or both devices have the OOB Authentication Data parameter set to Received, the Out-of-Band procedure shall be used, otherwise.
- If both devices have the Authentication_Requirements parameter set to one of the man-in-the middle (MITM) Protection Not Required options, the Numeric Comparison procedure shall be used, otherwise.
- If one or both devices have the Authentication_Requirements parameter set to one of the MITM Protection Required options, the Passkey Entry procedure shall be used if the either the local or remote IO Capability is set to KeyboardOnly and the other IO capability is not set to NoInputNoOutput. Otherwise, the Numeric Comparison authentication procedure shall be used.

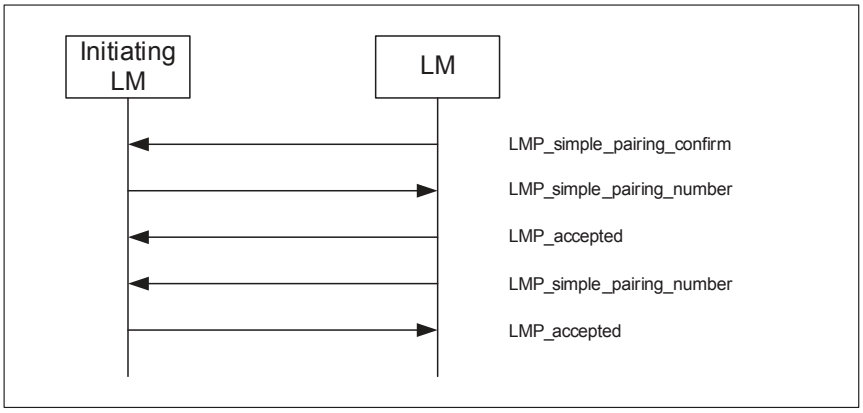
##### *4.2.7.3.1 Authentication Stage 1: Numeric Comparison*

Once public keys have been exchanged, both devices shall generate a random number.

The Responder shall compute its commitment as defined in Section 7.7.1 on page 898, and shall send this value to the Initiator by using LMP_simple_pairing_confirm PDU.

The Initiator shall then send an LMP_simple_pairing_number PDU with its generated random number. The Responder shall acknowledge by sending an LMP_accepted PDU.

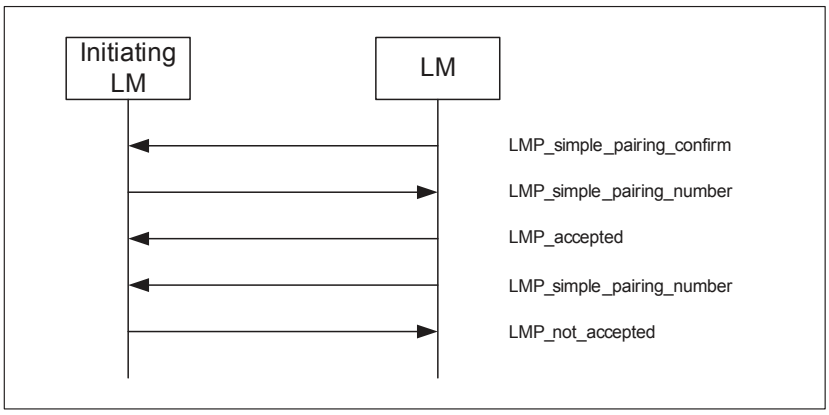
The Responder shall then send an LMP_simple_pairing_number containing its own generated random number. Upon reception, the Initiator shall calculate the commitment as defined in Section 7.7.1 on page 898 and compare it to the one received previously with the LMP_simple_pairing_confirm PDU. If both values are equal, the Initiator shall respond with an LMP_accepted PDU.




*Sequence 47: Numeric Comparison Authentication: Commitment check success*

#### *4.2.7.3.1.1 Commitment check failure*

If the calculated commitment by the Initiator is not equal to the received commitment, the Initiator shall abort the Secure Simple Pairing process by sending an LMP_not_accepted PDU with reason "Authentication Failure."

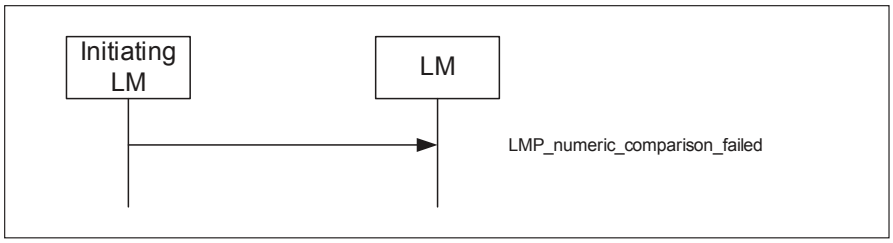



*Sequence 48: Numeric Comparison Authentication: Commitment check failure*

#### *4.2.7.3.1.2 Numeric Comparison failure on Initiator side*

If the user on the initiating side indicates that the confirm values do not match (as indicated by the HCI_User_Confirmation_Negative_Reply command) the initiating LM shall send an LMP_numeric_comparison_failure PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.




*Sequence 49: Authentication stage 1 Numeric Comparison Failure on Initiator Side*

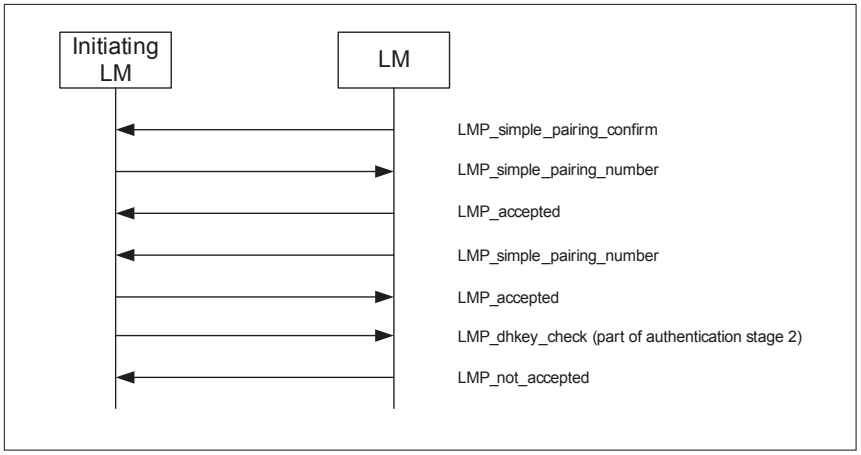
#### *4.2.7.3.1.3 Numeric Comparison failure on Responder Side*

If the user on the responding side indicates that the confirm values do not match (as indicated by the HCI_User_Confirmation_Negative_Reply command) the responding LM shall send an LMP_not_accepted PDU in response to the LMP_dhkey_check PDU.

Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.



LifeScan, Inc. Ex. 1019
Page 430 of 1420
IPR2025-00103

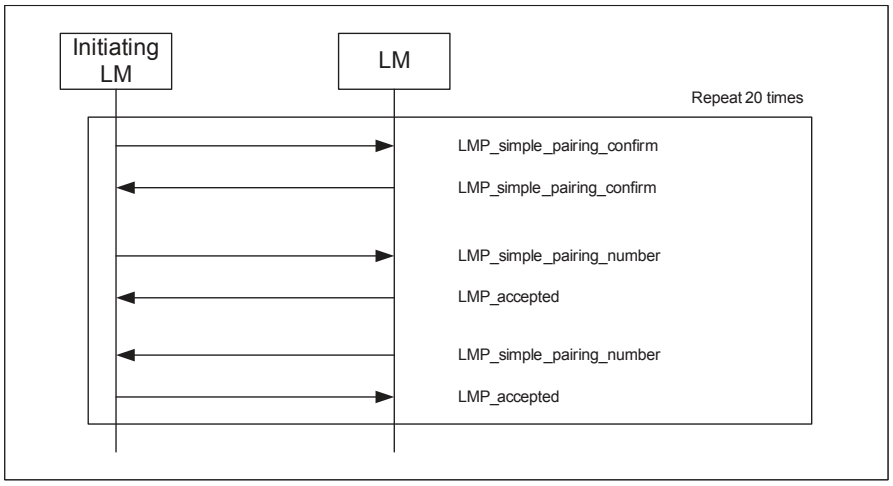



*Sequence 50: Authentication stage 1: Numeric Comparison Failure on Responder Side*

### *4.2.7.3.2 Authentication Stage 1: Passkey Entry Authentication*

The initiating LM shall start the following procedure after receiving the Passkey Reply from the host. This procedure shall be repeated 20 times:

1. The Initiator and the Responder shall generate a new random number.
2. The Initiator and the Responder shall calculate the local commitment value using the exchanged public keys, the local random number, and the passkey from the local Host, according to Section 7.7.1 on page 898.
3. The Initiator shall send an LMP_simple_pairing_confirm PDU with the commitment it calculated in step 2.
4. The Responder shall respond with an LMP_simple_pairing_confirm PDU with the commitment it calculated in step 2.
5. The Initiator shall then send an LMP_simple_pairing_number PDU with the random number it generated in step 1.
6. The Responder shall then calculate commitment from the exchanged public keys, the random number it received and the passkey from the local Host, according to Section 7.7.1 on page 898. If the calculated commitment and the received commitment are equal, the Responder shall reply with an LMP_accepted PDU.
7. The Responder shall then send an LMP_simple_pairing_number PDU with the random value it generated in step 1.
8. The Initiator shall calculate the commitment using exchanged public keys, the received the random number it received and the passkey from the local Host, according to Section 7.7.1 on page 898. If the calculated commitment is equal to the received commitment, the Initiator shall reply with an LMP_accepted PDU.



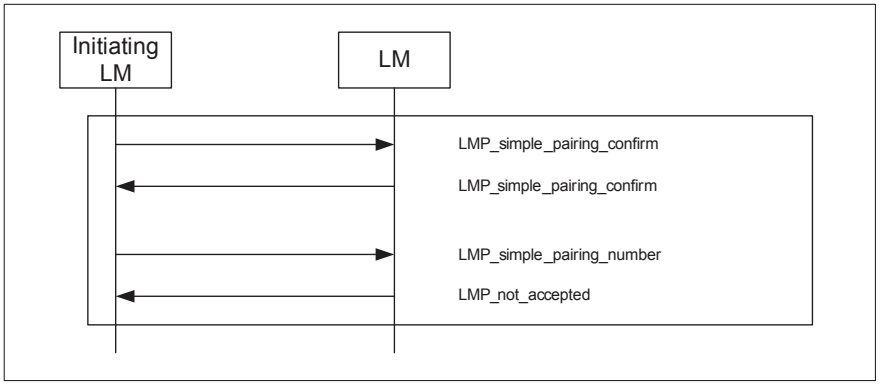

*Sequence 51: Authentication Passkey entry*

When this procedure has successfully passed 20 times, the Secure Simple Pairing process continues with the Authentication step 2 as described in Authentication stage 2: DHKey Check on page 267.

#### *4.2.7.3.2.1 Commitment check failure on the Responder side*

If during one of the 20 repetitions, the commitment calculated by the Responder is not equal to the one received from the Initiator (step 6), the Responder shall abort the Secure Simple Pairing process by sending an LMP_not_accepted PDU with reason "Authentication Failure."

Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.

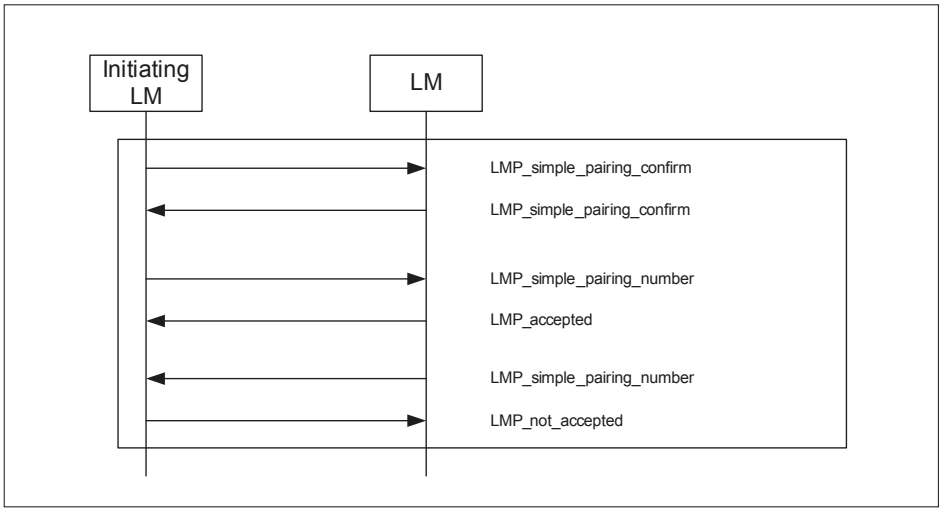



*Sequence 52: Authentication Passkey Entry: Commitment check failure on the Responder side*

#### *4.2.7.3.2.2 Commitment check failure on the Initiator side*

If during one of the 20 repetitions, the commitment calculated by the Initiator is not equal to the one received from the Responder (step 8), the Initiator shall abort the Secure Simple Pairing process by sending an LMP_not_accepted PDU with reason "Authentication Failure".

Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.

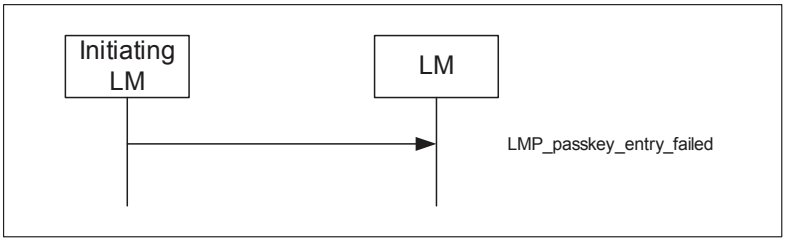



*Sequence 53: Authentication Passkey Entry: Commitment check failure on the Initiator side*

#### *4.2.7.3.2.3 Passkey Entry failure on Initiator side*

If the initiating side indicates that the passkey was not entered or canceled (as indicated by the HCI_Passkey_Request_Negative_Reply command) the initiating LM shall send an LMP_passkey_entry_failed PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

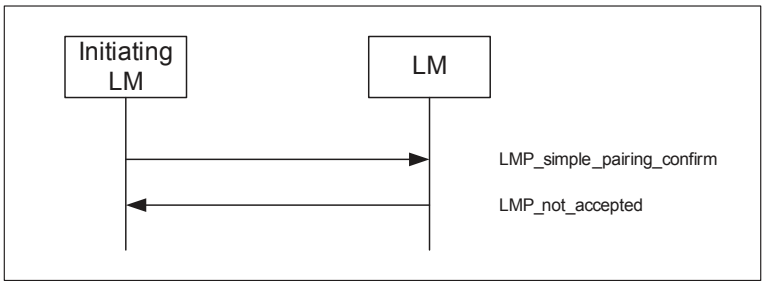



*Sequence 54: Authentication stage 1 Passkey Entry Failure on Initiator Side*

#### *4.2.7.3.3 Passkey Entry failure on Responding side*

If the responding side indicates that the passkey was not entered or canceled (as indicated by the HCI_Passkey_Request_Negative_Reply command), the responding LM shall send an LMP_not_accepted PDU in response to the LMP_simple_pairing_config PDU.

Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.

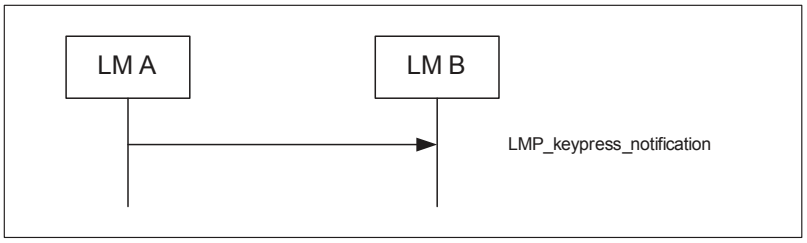



*Sequence 55: Authentication Stage 1 Passkey Entry Failure on Responding Side*

#### *4.2.7.3.4 Keypress Notifications*

A side with the KeyboardOnly IO capability may send notifications on key presses to the remote side using the LMP_keypress_notification PDU. This PDU may be sent by either the initiating LM, the responding LM or both LMs.

Note: a device with KeyboardOnly IO capabilities is not required to send any LMP_keypress_notification PDUs.




*Sequence 56: Keypress Notifications*

#### *4.2.7.3.5 Authentication Stage 1: OOB*

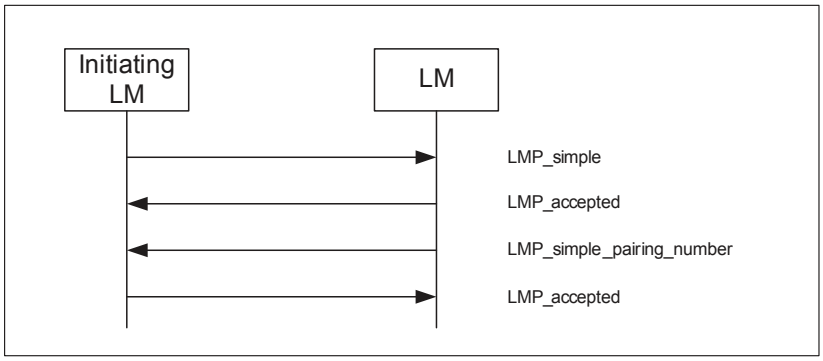
Upon reception of the OOB information (as defined in “Authentication Stage 1: Out of Band Protocol” on page 892 in Vol. 2) from the host, the devices shall compare the received commitment from its host, with the one calculated using the secret number received from the Host and the public key received from the remote device. If the local Host has not set the OOB Authentication Data

Present, the LM shall set the remote secret number to zero and base the subsequent calculations on this value.

If the commitment check on the initiator is valid, the Initiator shall then generate a random number (nonce) and send it to the Responder using an LMP_simple_pairing_number PDU. If the commitment succeeds, the Responder shall acknowledge by sending an LMP_accepted PDU otherwise it shall send an LMP_not_accepted PDU. The Responder shall then generate a random number (nonce) and send it to the Initiator using an LMP_simple_pairing_number PDU. If the commitment succeeds, the Initiator shall acknowledge by sending an LMP_accepted PDU otherwise it shall send an LMP_not_accepted PDU.

If the commitment values don't match in the Initiator, the procedure in Section 4.2.7.3.5.2, “Commitment check failure on the Initiator side,” on page 266 shall apply.

If the commitment values don't match in the Responder, the procedure in Section 4.2.7.3.5.1, “Commitment check failure on the Responder side,” on page 265 shall apply.




*Sequence 57: Authentication OOB: Only one device is OOB-r capable*

When these operations have been passed successfully, Secure Simple Pairing procedures continue with Authentication step 2 as described in Authentication stage 2: DHKey Check on page 267.

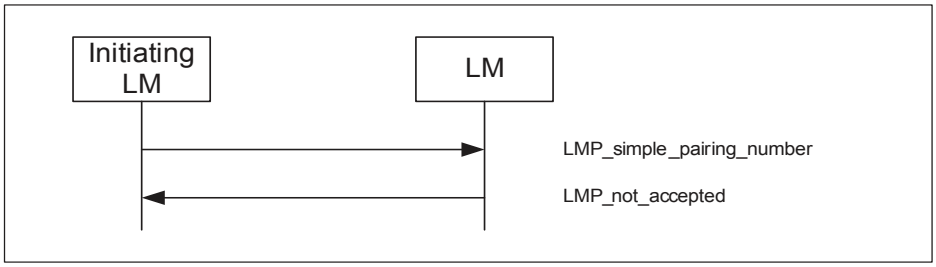
#### *4.2.7.3.5.1 Commitment check failure on the Responder side*

If the commitment received OOB from the host is not equal to the calculated commitment, the Responder shall send an LMP_not_accepted PDU with reason "Authentication Failure" in response to LMP_simple_pairing_number PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.



LifeScan, Inc. Ex. 1019
Page 435 of 1420
IPR2025-00103

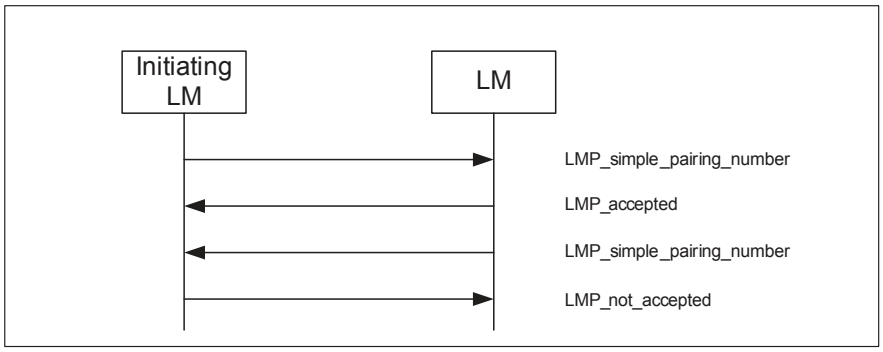



*Sequence 58: Authentication stage 1 OOB: Commitment check failure on the Responder side*

#### *4.2.7.3.5.2 Commitment check failure on the Initiator side*

If the commitment received OOB from the host is not equal to the calculated commitment, the Initiator shall send an LMP_not_accepted PDU with reason "Authentication Failure" in response to LMP_simple_pairing_number PDU.

Secure Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.




*Sequence 59: Authentication stage 1 OOB: Commitment check failure on the Initiator side*

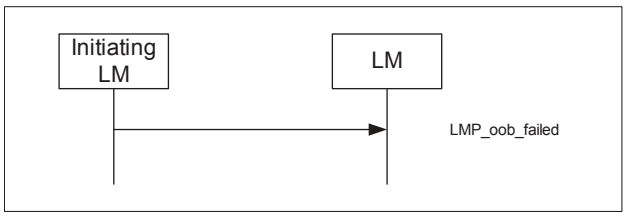
#### *4.2.7.3.6 Out-of-Band information not available on the Initiator side*

If the Host on the initiating side does not have out-of-band information the Initiator shall send an LMP_oob_failed PDU.

Simple Pairing process shall then be aborted. The Link Managers shall not disconnect the connection.



LifeScan, Inc. Ex. 1019
Page 436 of 1420
IPR2025-00103




*Sequence 60: Authentication stage 1 OOB: OOB information not available on the Initiator side*

### *4.2.7.4 Authentication stage 2: DHKey Check*

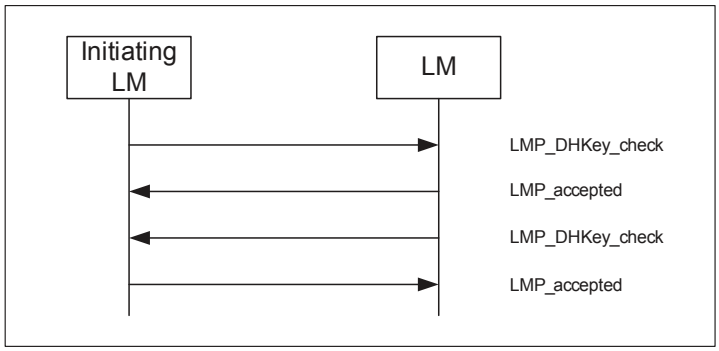
At this stage, both devices compute a new confirmation values based on Diffie-Hellman key and previously exchanged information according to Section 7.7.4, “The Simple Pairing Check Function f3,” on page 900.

The Initiator shall send an LMP_DHKey_check PDU to the Responder including the confirmation value it has computed. Upon reception, the Responder shall reply with an LMP_accepted PDU if the received value is equal to the one it has calculated according to Section 7.7.4 on page 900. If it fails, refer to Section 4.2.7.3.5.1 on page 265.

The Responder shall then send an LMP_DHKey_check PDU to the Initiator including its confirmation value it has computed. Upon reception, the Initiator shall reply with an LMP_accepted PDU if the received value is equal to the one it has calculated according to Section 7.7.4 on page 900. If it fails, refer to Section 4.2.7.4.1.1, “Check failure on the initiator side,” on page 268.

At this point, both devices shall compute the link key according to Section 7.7.3, “The Simple Pairing Key Derivation Function f2,” on page 899.

The Initiator shall then start standard mutual authentication as described in Section 4.2.1.1

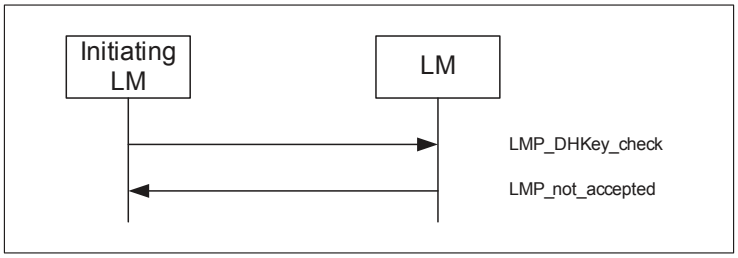



*Sequence 61: DHKey check*

#### *4.2.7.4.1 Check failure on the Responder side*

If the confirmation value received via LMP by the Responder is not equal to the one it has calculated according to Section 7.7.4 on page 900, the Responder shall send an LMP_not_accepted PDU with reason "Authentication Failure".

Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.

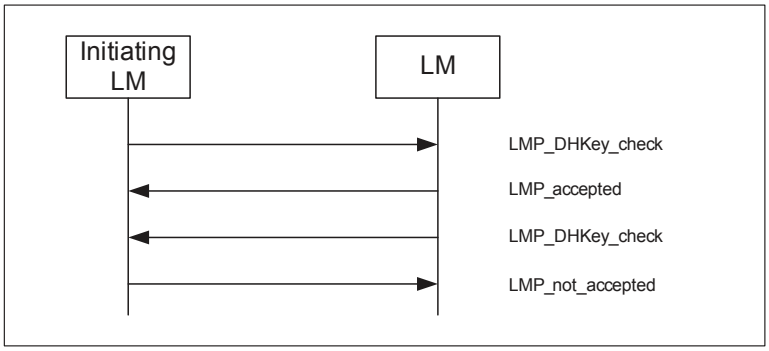



*Sequence 62: DHKey check: Check failure on the Responder side DHKey check: Check failure on the Responder*

#### *4.2.7.4.1.1 Check failure on the initiator side*

If the confirmation value received via LMP by the Initiator is not equal to the one it has calculated according to Section 7.7.4 on page 900, the Initiator shall send an LMP_not_accepted PDU with reason "Authentication Failure".

Secure Simple Pairing procedures shall then be aborted. The Link Managers shall not disconnect the connection.




*Sequence 63: DHKey check: check failure on the Initiator side*

## 4.3 INFORMATIONAL REQUESTS

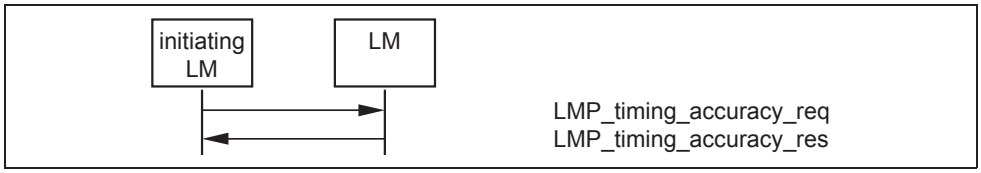
### 4.3.1 Timing accuracy

LMP supports requests for the timing accuracy. This information can be used to minimize the scan window during piconet physical channel re-synchronization

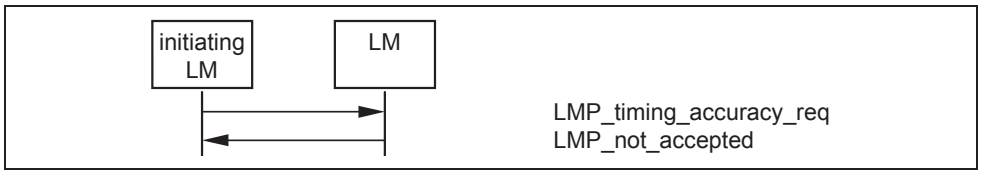
(see Baseband Specification, Section 2.2.5.2, on page 69). The timing accuracy parameters returned are the long term drift measured in ppm and the long term jitter measured in μs of the worst case clock used. These parameters are fixed for a certain device and shall be identical when requested several times. Otherwise, the requesting device shall assume worst case values (drift=250ppm and jitter=10μs).

| M/O | PDU | Contents |
|---|---|---|
| O(4) | LMP_timing_accuracy_req | - |
| O(4) | LMP_timing_accuracy_res | drift<br>jitter |

*Table 4.19: Request limited timing PDU*




*Sequence 64: The requested device supports timing accuracy information.*




*Sequence 65: The requested device does not support timing accuracy information.*



LifeScan, Inc. Ex. 1019
Page 439 of 1420
IPR2025-00103

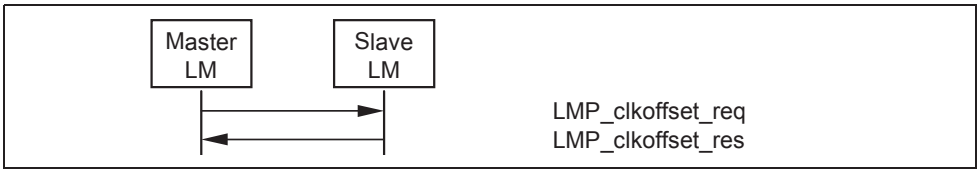
### 4.3.2 Clock offset

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 baseband paging procedure (i.e., before, during or after connection setup). The clock offset shall be defined by the following equation:

$(CLKN_{16\text{-}2\ slave} - CLKN_{16\text{-}2\ master})$ mod 2**15.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_clkoffset_req | - |
| M | LMP_clkoffset_res | clock offset |

*Table 4.20: PDUs used for clock offset request.*



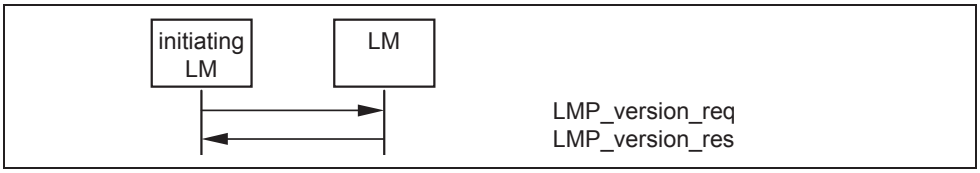

*Sequence 66: Clock offset requested.*

### 4.3.3 LMP version

LMP supports requests for the version of the LM protocol. The LMP_version_req and LMP_version_res PDUs contain three parameters: VersNr, CompId and SubVersNr. VersNr specifies the version of the Bluetooth LMP specification that the device supports. CompId is used to track possible problems with the lower Bluetooth layers. All companies that create a unique implementation of the LM shall have their own CompId. The same company is also responsible for the administration and maintenance of the SubVersNr. It is recommended that each company has 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. The sequence below is only used to exchange the parameters. LMP version can be requested at anytime following a successful baseband paging procedure.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_version_req | VersNr<br>CompId<br>SubVersNr |
| M | LMP_version_res | VersNr<br>CompId<br>SubVersNr |

*Table 4.21: PDUs used for LMP version request.*




*Sequence 67: Request for LMP version.*

### 4.3.4 Supported features

The supported features may be requested at anytime following a successful baseband 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.

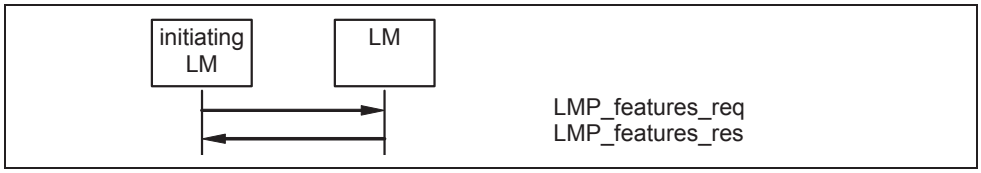
The number of features bits required will in the future exceed the size of a single page of features. An extended features mask is therefore provided to allow support for more than 64 features. Support for the extended features mask is indicated by the presence of the appropriate bit in the LMP features 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 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 does not support any page number it shall return a mask with every bit set to 0. It also contains the maximum features page number containing any non-zero 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.

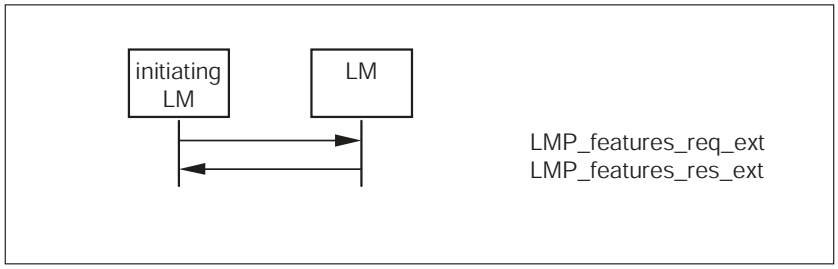
If the extended features request is not supported then all bits in all extended features pages for that device shall be assumed to be zero.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_features_req | features |
| M | LMP_features_res | features |
| O(63) | LMP_features_req_ext | features page<br>max supported page<br>extended features |
| O(63) | LMP_features_res_ext | features page<br>max supported page<br>extended features |

*Table 4.22: PDUs used for features request.*




*Sequence 68: Request for supported features.*




*Sequence 69: Request for extended features.*



LifeScan, Inc. Ex. 1019
Page 442 of 1420
IPR2025-00103

### 4.3.5 Name request

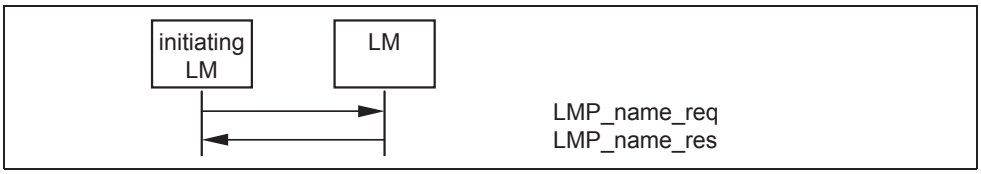
LMP supports name request to another device. The name is a user-friendly name associated with the device and consists of a maximum of 248 bytes coded according to the UTF-8 standard. The name is fragmented over one or more DM1 packets. When an LMP_name_req PDU is sent, a name offset indicates which fragment 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 $0 \leq N \leq 13$. In the first sent LMP_name_req PDU, name offset=0. Sequence 70 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 baseband paging procedure.

| M/O | PDU | Contents |
|---|---|---|
| M | LMP_name_req | name offset |
| M | LMP_name_res | name offset<br>name length<br>name fragment |

*Table 4.23: Name request PDUs*




*Sequence 70: Device's name requested and it responses.*

## 4.4 ROLE SWITCH

### 4.4.1 Slot offset

With LMP_slot_offset 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 baseband 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.

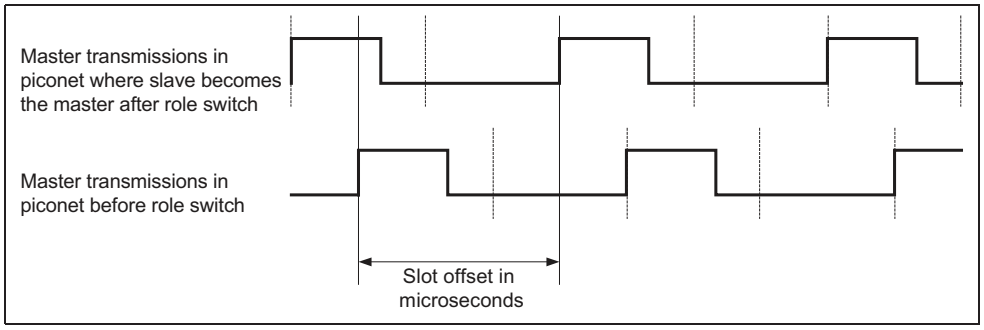
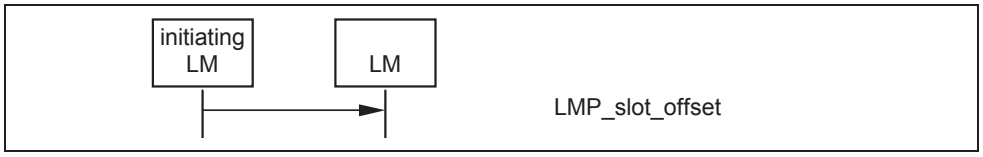



*Figure 4.3: Slot offset for role switch*

See Section 4.4 on page 274 for the use of LMP_slot_offset in the context of the role switch. In the case of role switch the BD_ADDR is that of the slave device.

| M/O | PDU | Contents |
|---|---|---|
| O(3) | LMP_slot_offset | slot offset<br>BD_ADDR |

*Table 4.24: Role switch PDU*




*Sequence 71: Slot offset information is sent*



LifeScan, Inc. Ex. 1019
Page 444 of 1420
IPR2025-00103

### 4.4.2 Role switch

Since the paging device always becomes the master of the piconet, a switch of the master slave role is sometimes needed, see Baseband Specification, Section 8.6.5, on page 167. A role switch may be performed anytime after the baseband paging procedure has completed.

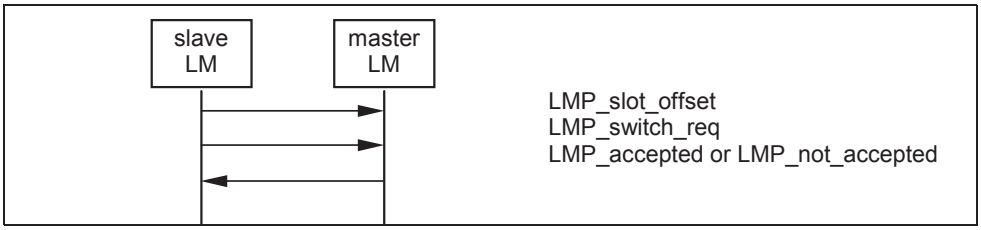
Support for LMP_slot_offset is mandatory if LMP_switch_req is supported.

The LMP_slot_offset shall be sent only if the ACL logical transport is in active mode. The LMP_switch_req shall be sent only if the ACL logical transport is in active mode, when encryption is disabled or paused, and all synchronous logical transports on the same physical link are disabled. Additionally, LMP_slot_offset or LMP_switch_req shall not be initiated or accepted while a synchronous logical transport is being negotiated by LM.

| M/O | PDU | Contents |
|---|---|---|
| O(5) | LMP_switch_req | switch instant |
| O(5) | LMP_slot_offset | slot offset<br>BD_ADDR |

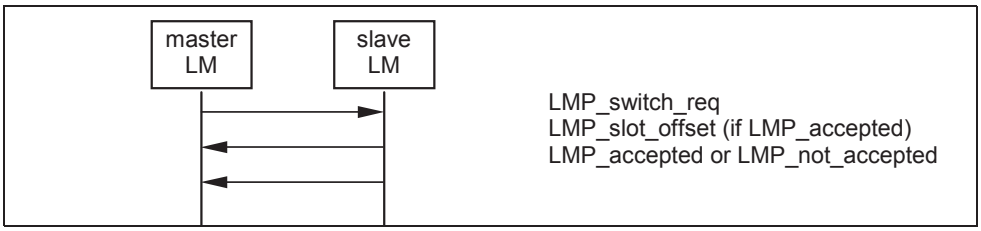
*Table 4.25: Role switch PDU*

The initiating LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101). If the encryption mode is set to "encryption" and both devices support pausing encryption, then the initiating device shall initiate the pause encryption sequence (see Section 4.2.5.5, “Pause encryption,” on page 251.) It shall then send an LMP_slot_offset PDU immediately followed by an LMP_switch_req PDU. If the master accepts the role switch and encryption is not already paused, it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101) and respond with an LMP_accepted PDU. When the role switch has been completed at the baseband level (successfully or not) if encryption was paused, the device that paused encryption shall initiate the resume encryption sequence (see Section 4.2.5.6, “Resume Encryption,” on page 253). If encryption was not paused, both devices shall re-enable 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 re-enables transmission on the ACL-U logical link. The transaction ID for the role switch PDUs in the sequence (LMP_slot_offset and LMP_switch_req along with the associated LMP_accepted or LMP_not_accepted) shall be set to 1.




*Sequence 72: Role switch (slave initiated).*

If the master initiates the role switch and if the encryption mode is set to "encryption" and both devices support pausing encryption, the master device shall initiate the pause encryption sequence (See [Part C] Section 4.2.5 on page 248 What section should be referenced?) otherwise it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101) and send an LMP_switch_req PDU. If the slave accepts the role switch and encryption is not already paused, it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101) and responds with an LMP_slot_offset PDU immediately followed by an LMP_accepted PDU. When the role switch has been completed at the baseband (successfully or not) if encryption was paused, the device that paused encryption shall initiate the resume encryption sequence (see Section 4.2.5.6, “Resume Encryption,” on page 253). If encryption was not paused, both devices re-enable 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 re-enables transmission on the ACL-U logical link. The transaction ID for the role switch PDUs in the sequence (LMP_slot_offset and LMP_switch_req along with the associated LMP_accepted or LMP_not_accepted) shall be set to 0.




*Sequence 73: Role switch (master initiated).*

The LMP_switch_req PDU contains a parameter, switch instant, which specifies the instant at which the TDD switch is performed. This is specified as a Bluetooth clock value of the master's clock, that 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 hours of the current clock value to avoid clock wrap.

The sender of the LMP_switch_req PDU selects the switch instant and queues the LMP_switch_req PDU to LC 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



LifeScan, Inc. Ex. 1019
Page 446 of 1420
IPR2025-00103

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 current master clock value. 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 then an LMP_accepted PDU shall be returned assuming the role switch is allowed 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. Adaptive Frequency Hopping shall follow the procedures described in Baseband Specification, Section 8.6.5, on page 167. The default value for max_slots shall be used.

## 4.5 MODES OF OPERATION

### 4.5.1 Hold mode

The ACL logical transport of a connection between two Bluetooth devices can be placed in hold mode for a specified hold time. See Baseband Specification, Section 8.8, on page 178 for details.

| M/O | PDU | Contents |
|---|---|---|
| O(6) | LMP_hold | hold time, hold instant |
| O(6) | LMP_hold_req | hold time, hold instant |

*Table 4.26: Hold mode PDUs*

The LMP_hold and LMP_hold_req PDUs both contain a parameter, hold instant, that specifies the instant at which the hold becomes effective. This is specified as a Bluetooth clock value of the master's clock, that 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 hours of the current clock value to avoid clock wrap.

#### *4.5.1.1 Master forces hold mode*

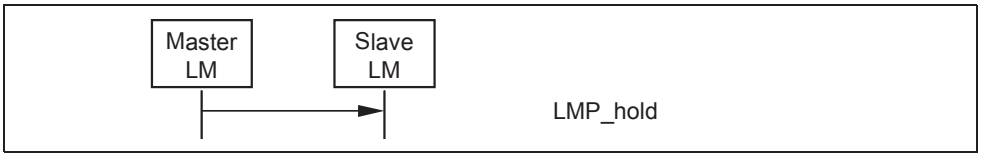
The master may force hold mode if there has previously been a request for hold mode that has been accepted. The hold time included in the PDU when the master forces hold mode shall not be longer than any hold time the slave has previously accepted when there was a request for hold mode.

The master LM shall first pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101). It shall select the hold instant and queue the LMP_hold PDU to its LC 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 baseband acknowledgement 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.

When the slave LM receives an LMP_hold PDU it compares the hold instant with the current master clock 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 re-enables transmission on the ACL-U logical link.




*Sequence 74: Master forces slave into hold mode*

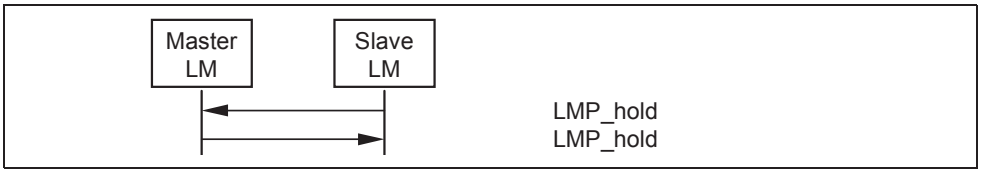
### *4.5.1.2 Slave forces hold mode*

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.

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 LC for transmission. It shall then wait for an LMP_hold PDU from the master acting according to the procedure described in Section 4.5.1.1.

When the master LM receives an LMP_hold PDU it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101). 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 Section 4.5.1.1.

When the master and slave LMs exit from Hold mode they shall re-enable transmission on the ACL-U logical link.




*Sequence 75: Slave forces master into hold mode*



LifeScan, Inc. Ex. 1019
Page 448 of 1420
IPR2025-00103

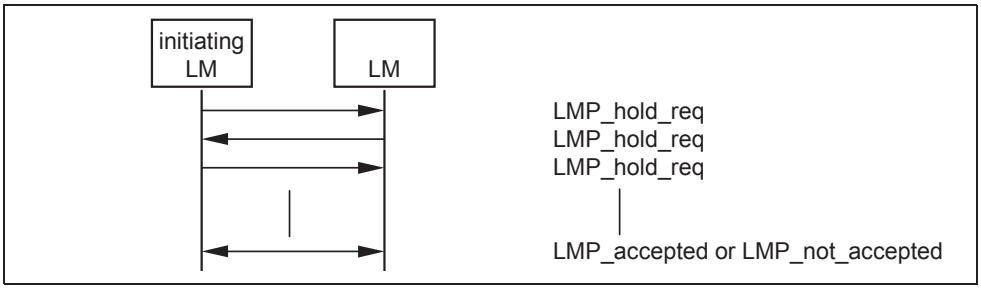
#### *4.5.1.3 Master or slave requests hold mode*

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 Baseband Specification, Section 5.3.1, on page 101). 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 contained hold instant is at least $9*T_{poll}$ slots in the future then this shall be used. The LMP_hold_req PDU shall then be queued to its LC for transmission and a timer shall be started to expire at this instant and the connection enters hold mode when it expires unless an LMP_not_accepted or LMP_hold_req PDU is received by its LM before that point. 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.

When each LM exits from Hold mode it shall re-enable transmission on the ACL-U logical link.




*Sequence 76: Negotiation for hold mode*

### 4.5.2 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 Baseband Specification, Section 8.9, on page 179 for details.

Note: 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 Baseband Specification, Section 3.1, on page 90).

All PDUs sent from the master to parked slaves are carried on the PSB-C logical link (LMP link of parked slave broadcast logical transport). These PDUs, 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 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 Baseband Specification, Section 8.9, on page 179. When a slave is placed in park state it is assigned a unique PM_ADDR, that 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.



LifeScan, Inc. Ex. 1019
Page 450 of 1420
IPR2025-00103

### 4.5.2.1 Master requests slave to enter park state

| M/O | PDU | Contents |
|---|---|---|
| O(8) | LMP_park_req | timing control flags<br>$D_B$<br>$T_B$<br>$N_B$<br>$\Delta_B$<br>PM_ADDR<br>AR_ADDR<br>$N_{Bsleep}$<br>$D_{Bsleep}$<br>$D_{access}$<br>$T_{access}$<br>$N_{acc\text{-}slots}$<br>$N_{poll}$<br>$M_{access}$<br>access scheme |
| O(8) | LMP_set_broadcast_scan_window | timing control flags<br>$D_B$ (optional)<br>broadcast scan window |
| O(8) | LMP_modify_beacon | timing control flags<br>$D_B$ (optional)<br>$T_B$<br>$N_B$<br>$\Delta_B$<br>$D_{access}$<br>$T_{access}$<br>$N_{acc\text{-}slots}$<br>$N_{poll}$<br>$M_{access}$<br>access scheme |

*Table 4.27: PDUs used for park state*



LifeScan, Inc. Ex. 1019
Page 451 of 1420
IPR2025-00103

| M/O | PDU | Contents |
|---|---|---|
| O(8) | LMP_unpark_PM_ADDR_req | timing control flags<br>$D_B$ (optional)<br>LT_ADDR 1st unpark<br>LT_ADDR 2nd unpark<br>PM_ADDR 1st unpark<br>PM_ADDR 2nd unpark<br>LT_ADDR 3rd unpark<br>LT_ADDR 4th unpark<br>PM_ADDR 3rd unpark<br>PM_ADDR 4th unpark<br>LT_ADDR 5th unpark<br>LT_ADDR 6th unpark<br>PM_ADDR 5th unpark<br>PM_ADDR 6th unpark<br>LT_ADDR 7th unpark<br>PM_ADDR 7th unpark |
| O(8) | LMP_unpark_BD_ADDR_req | timing control flags<br>$D_B$ (optional)<br>LT_ADDR<br>LT_ADDR (optional)<br>BD_ADDR<br>BD_ADDR (optional) |

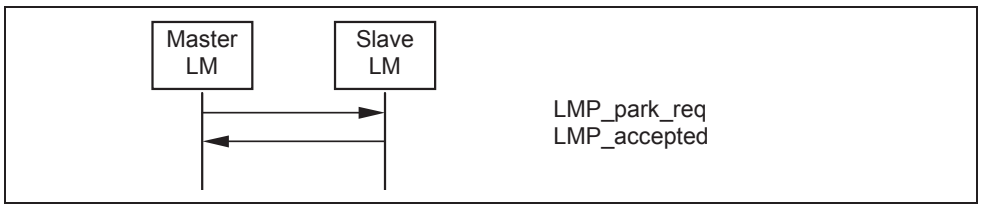
*Table 4.27: PDUs used for park state*

The master can request park state. The master LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101) 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 Baseband Specification, Section 5.3.1, on page 101). and then respond with an LMP_accepted PDU.

When the slave queues an LMP_accepted PDU it shall start a timer for 6*Tpoll slots. If the baseband acknowledgement is received before this timer expires it shall enter park state immediately otherwise it shall enter park state when the timer expires.

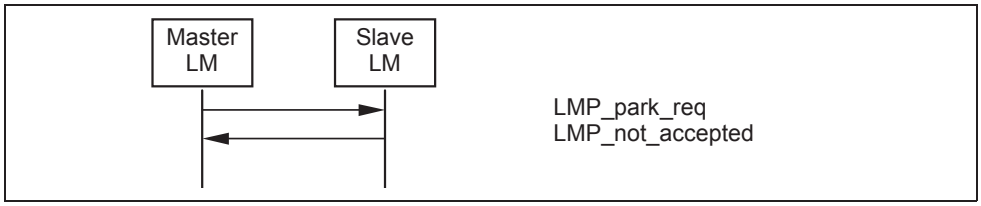
When the master receives an LMP_accepted PDU it shall start a timer for 6*Tpoll slots. When this timer expires the slave is in park state and the LT_ADDR may be re-used.

If the master never receives an LMP_accepted PDU then a link supervision timeout will occur.




*Sequence 77: 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 re-enable transmission on the ACL-U logical link.




*Sequence 78: Slave rejects to enter into park state*

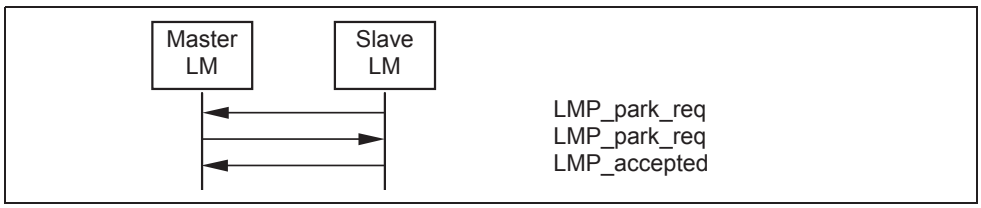
#### *4.5.2.2 Slave requests to enter park state*

The slave can request park state. The slave LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 101) 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 Baseband Specification, Section 5.3.1, on page 101) 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 parameter 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*Tpoll slots. If the baseband acknowledgement is received before this timer expires it shall enter park state immediately otherwise it shall enter park state when the timer expires.

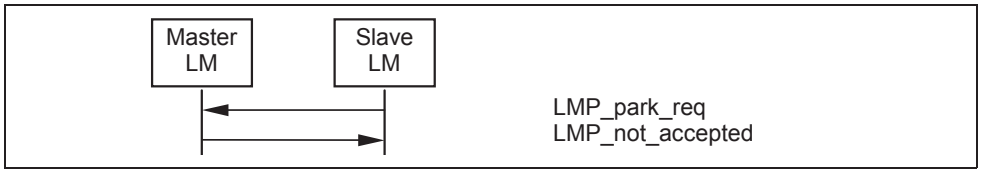
When the master receives an LMP_accepted PDU it shall start a timer for 6*Tpoll slots. When this timer expires the slave is in park state and the LT_ADDR may be re-used.

If the master never receives the LMP_accepted PDU then a link supervision timeout will occur.



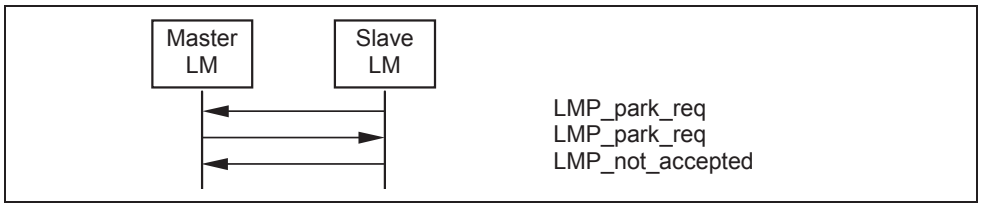

*Sequence 79: Slave requests to enter park state and accepts master's beacon parameters*

If the master does not agree that the slave enters park state it shall send an LMP_not_accepted PDU. The slave shall then re-enable transmission on the ACL-U logical link.




*Sequence 80: Master rejects slave's request to enter park state*

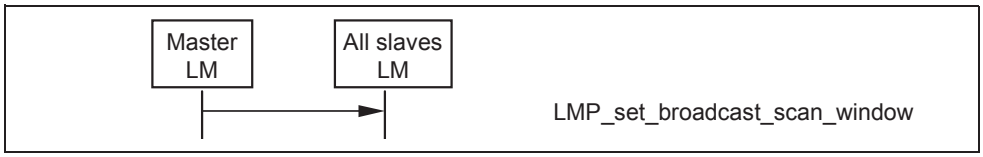
If the slave does not accept the parameters in the LMP_park_req PDU sent from the master it shall respond with an LMP_not_accepted PDU and both devices shall re-enable transmission on the ACL-U logical link.




*Sequence 81: Slave requests to enter park state, but rejects master's beacon parameters*

#### *4.5.2.3 Master sets up broadcast scan window*

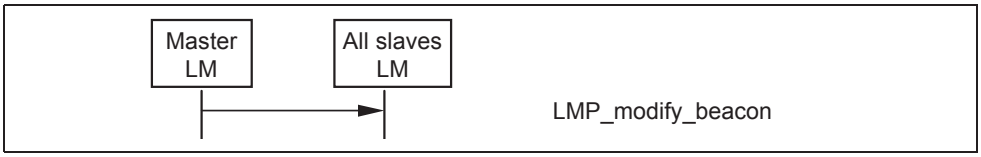
If more broadcast capacity is needed than the beacon train, the master may indicate to the slaves that more broadcast information will follow the beacon train by sending an LMP_set_broadcast_scan_window PDU. This message shall be sent in a broadcast packet at the beacon slot(s). The scan window shall start in the beacon instant and shall only be valid for the current beacon.




*Sequence 82: Master notifies all slaves of increase in broadcast capacity*

#### 4.5.2.4 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.



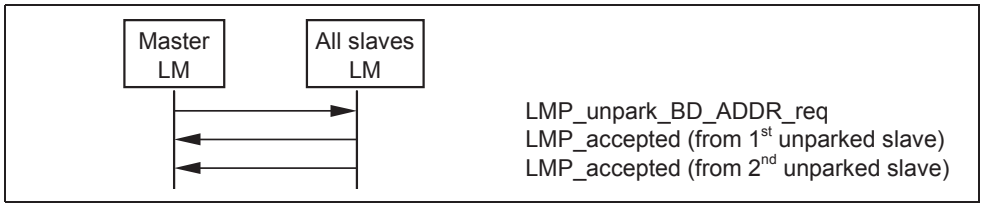

*Sequence 83: Master modifies beacon parameters*

#### 4.5.2.5 Unparking slaves

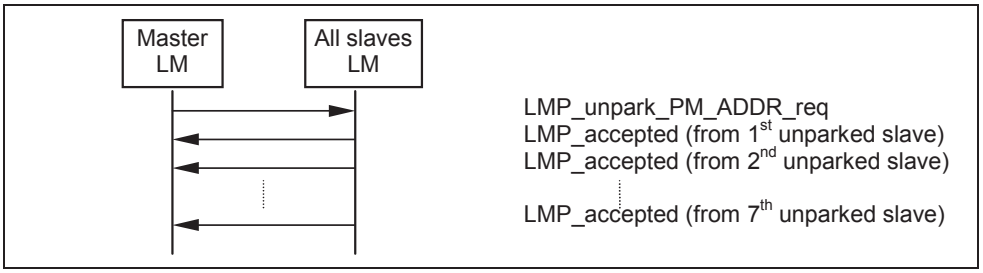
The master can unpark one or many slaves by sending a broadcast LMP message including the PM_ADDR or the BD_ADDR of the device(s) to be unparked. Broadcast LMP messages are carried on the PSB-C logical link. See Baseband Specification, Section 8.9.5, on page 185 for further details. This message also 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 certain time 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 7 slaves can be unparked with the same message. If they are identified with the BD_ADDR a maximum of 2 slaves can be unparked with the same message.

After a successful unparking, both devices re-enable transmission on the ACL-U logical link.




*Sequence 84: Master unparks slaves addressed with their BD_ADDR*




*Sequence 85: Master unparks slaves addressed with their PM_ADDR*

### 4.5.3 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 wrap-around 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 bit1 of the timing control flag is valid.

When the ACL logical transport is in sniff mode the master shall only start a transmission 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 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.

| M/O | PDU | Contents |
|---|---|---|
| O(7) | LMP_sniff_req | timing control flags<br>$D_{sniff}$<br>$T_{sniff}$<br>sniff attempt<br>sniff timeout |
| O(7) | LMP_unsniff_req | - |

*Table 4.28: Sniff mode PDUs*

#### *4.5.3.1 Master or slave requests sniff mode*

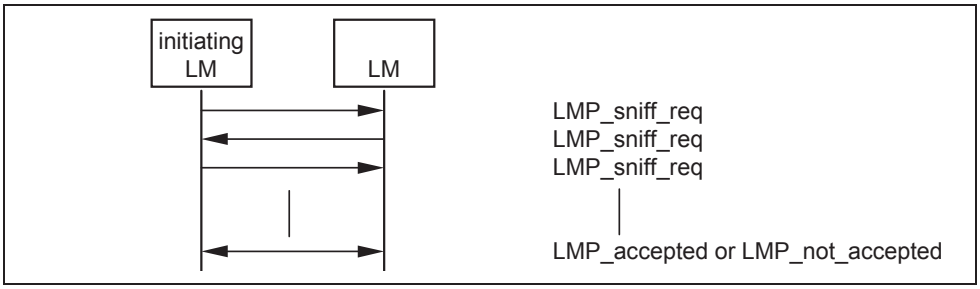
Either the master or the slave may request entry to sniff mode. The process is initiated by sending an LMP_sniff_req PDU containing a set of parameters. The receiving LM shall then decide whether to reject the attempt by sending an LMP_not_accepted PDU, to suggest different parameters by replying with an LMP_sniff_req PDU or to accept the request.

Before the first time that the master sends LMP_sniff_req 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 baseband acknowledgement 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 does not wish to enter sniff mode then it replies with an LMP_not_accepted PDU. If it is happy to enter 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.



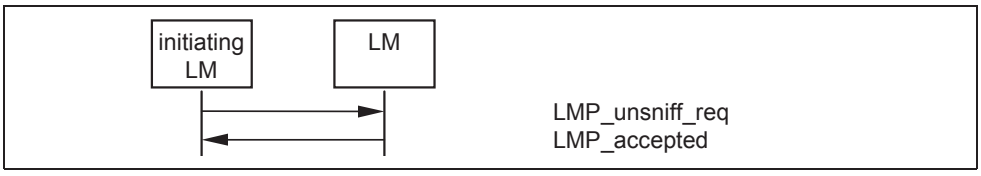

*Sequence 86: Negotiation for sniff mode*

#### *4.5.3.2 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 baseband acknowledgement for the LMP_accepted PDU it shall leave sniff transition mode and enter active mode.




*Sequence 87: Slave moved from sniff mode to active mode*

#### *4.5.3.3 Sniff Subrating*

Once sniff mode has been started, sniff subrating may be initiated by either Link Manager.

The LMP_sniff_subrating_req and LMP_sniff_subrating_res PDUs specify parameters that the peer and initiating device shall use respectively for sniff subrating.

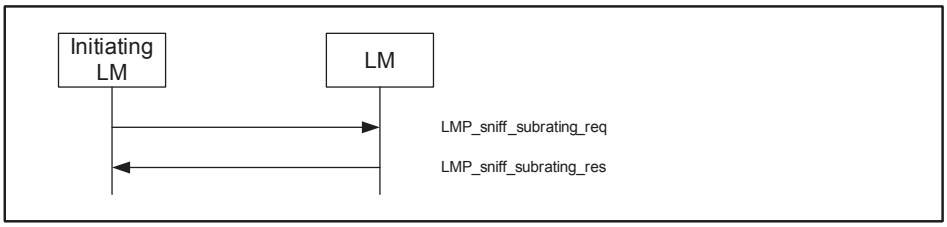
The sniff subrating instant value shall be used to calculate the first sniff subrating anchor point. The sniff subrating instant value shall be a maximum of $2^{16}$ time slots (40.9 seconds) from the current master clock and shall be a sniff anchor point. The sniff subrating instant value should indicate a clock value in the future with respect to the clock value when the LMP message is first sent.

If the LMP_sniff_subrating_req PDU is sent by the master, the sniff subrating instant value shall be used. The slave device shall reply with an LMP_sniff_subrating_res PDU using the same sniff subrating instant value given by the master.

When the LMP_sniff_subrating_req PDU is sent by the slave, the sniff subrating instant value shall be ignored. The master device shall reply with an

LMP_sniff_subrating_res PDU with the sniff subrating instant value that shall be used for sniff subrating.

The initiating device shall not transition to the new sniff subrating parameters until the sniff subrating instant has passed and the LMP_sniff_subrating_res PDU has been received. The non-initiating device shall remain in sniff mode and shall not transition to the new sniff subrating parameters until after the sniff subrating instant has passed and the baseband acknowledgement of the LMP_sniff_subrating_res PDU has been received.




*Sequence 88: LM accepts sniff subrating request*

A device shall not send a new LMP_sniff_subrating_req PDU until the previous sniff subrating transaction has completed and the sniff subrating instant has passed.

The maximum clock interval between two sniff subrating anchor points shall be less than the link supervision timeout. If the link supervision timeout needs to be updated to a shorter value than the clock interval between two sniff subrating anchor points, the master shall disable sniff subrating, shall send the LMP_Supervision_Timeout PDU with the new supervision timeout value, and shall start using the new supervision timeout value after receiving a baseband ACK for the LMP_Supervision_Timeout PDU. Upon reception of the LMP_Supervision_Timeout PDU the slave shall disable sniff subrating and shall start using the new supervision timeout value.

The master shall initiate sniff subrating with the max_sniff_subrate parameter less than the new supervision timeout. The slave shall respond with the LMP_sniff_subrating_res PDU with the max_sniff_subrate parameter less than the new supervision timeout.

Note: When changing the link supervision timeout while sniff subrating is enabled, refer to Section 4.5.3.3 on page 288.



LifeScan, Inc. Ex. 1019
Page 459 of 1420
IPR2025-00103

## 4.6 LOGICAL TRANSPORTS

When a connection is first established between two devices the connection consists of the default ACL logical links: ACL-C (for LMP messages) and ACL-U (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.

### 4.6.1 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 Baseband Specification, Section 8.6.2, on page 162 for details. A device shall initiate a request for HV2 or HV3 packet type only if the other device supports it (bits 12, 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 wrap-around 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 bit1 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.

| M/O | PDU | Contents |
|---|---|---|
| O(11) | LMP_SCO_link_req | SCO handle<br>timing control flags<br>$D_{sco}$<br>$T_{sco}$<br>SCO packet<br>air mode |
| O(11) | LMP_remove_SCO_link_req | SCO handle<br>error |

*Table 4.29: SCO link management PDUs*

#### *4.6.1.1 Master initiates an SCO link*

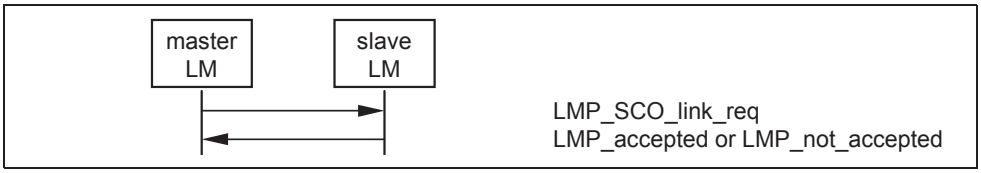
When establishing an SCO link the master sends a request, a LMP_SCO_link_req PDU, with parameters that specify the timing, packet type and coding that will be used on the SCO link. Each of the SCO packet types supports three different voice coding formats on the air-interface: μ-law log PCM, A-law log PCM and CVSD. The air coding by log PCM or CVSD may be deactivated to achieve a transparent synchronous data link at 64 kbits/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 shall be sent using the DM1 packet.

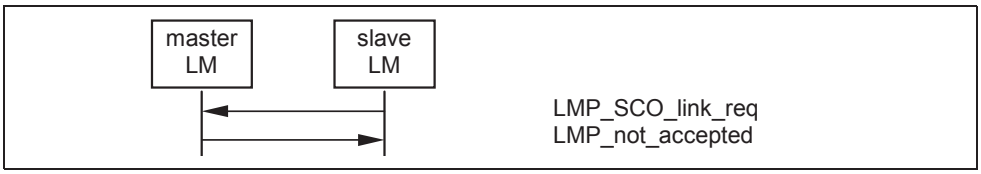



*Sequence 89: Master requests an SCO link*

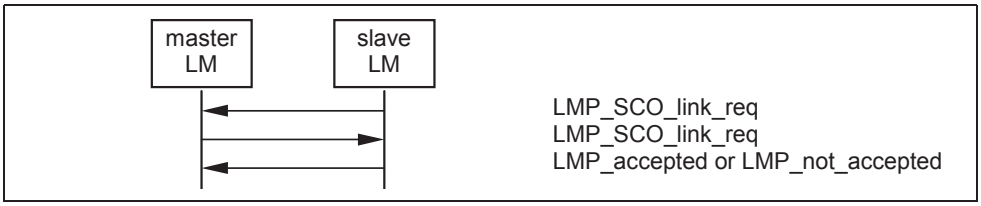
#### *4.6.1.2 Slave initiates an SCO link*

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, that 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.




*Sequence 90: Master rejects slave's request for an SCO link*




*Sequence 91: Master accepts slave's request for an SCO link*

#### *4.6.1.3 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 the master wishes to change parameters for. 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 Section 4.6.1.1 on page 290.

#### *4.6.1.4 Slave requests change of SCO parameters*

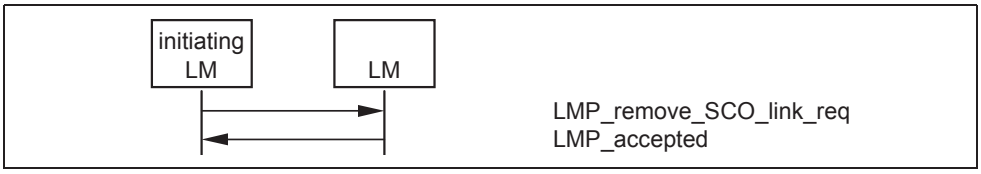
The slave sends an LMP_SCO_link_req PDU, where the SCO handle is the handle of the SCO link to be changed. The parameters timing control flags and $D_{sco}$ are not valid in this PDU. If the master does not accept the new parameters it shall reply with an LMP_not_accepted PDU and the SCO link is left 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 Section 4.6.1.2 on page 291.

#### *4.6.1.5 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.



LifeScan, Inc. Ex. 1019
Page 462 of 1420
IPR2025-00103




*Sequence 92: SCO link removed*

Note that the slave shall use the initialization flag appropriate to the master's BT clock. See section 8.6.3, “eSCO,” on page 164.

### 4.6.2 eSCO logical transport

After an ACL link has been established, one or more extended SCO (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 bit1 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.

| M/O | PDU | Contents |
|---|---|---|
| O(31) | LMP_eSCO_link_req | eSCO handle<br>eSCO LT_ADDR<br>timing control flags<br>$D_{eSCO}$<br>$T_{eSCO}$<br>$W_{eSCO}$<br>eSCO packet type M->S<br>eSCO packet type S->M<br>packet length M->S<br>packet length S->M<br>air mode<br>negotiation state |
| O(31) | LMP_remove_eSCO_link_req | eSCO handle<br>error |

*Table 4.30: PDUs used for managing the eSCO links*

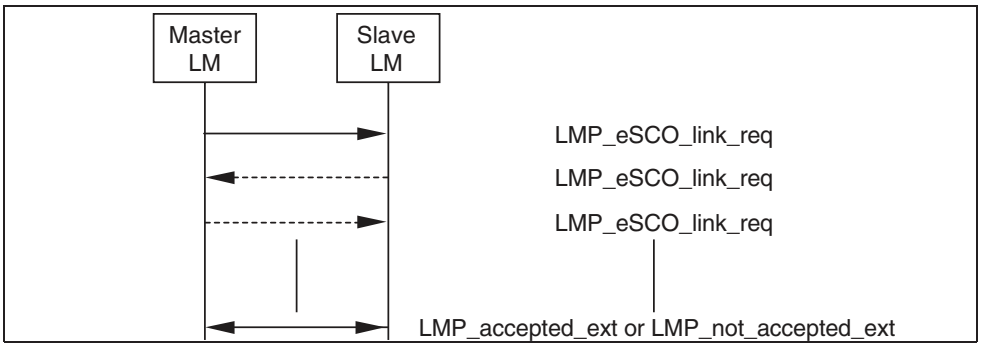
The parameters $D_{eSCO}$, $T_{eSCO}$, $W_{eSCO}$, eSCO packet type M->S, eSCO packet type S->M, packet length M->S, packet length S->M are henceforth referred to as the negotiable parameters.



LifeScan, Inc. Ex. 1019
Page 463 of 1420
IPR2025-00103

### *4.6.2.1 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 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 Section 4.6.2.5 on page 296.



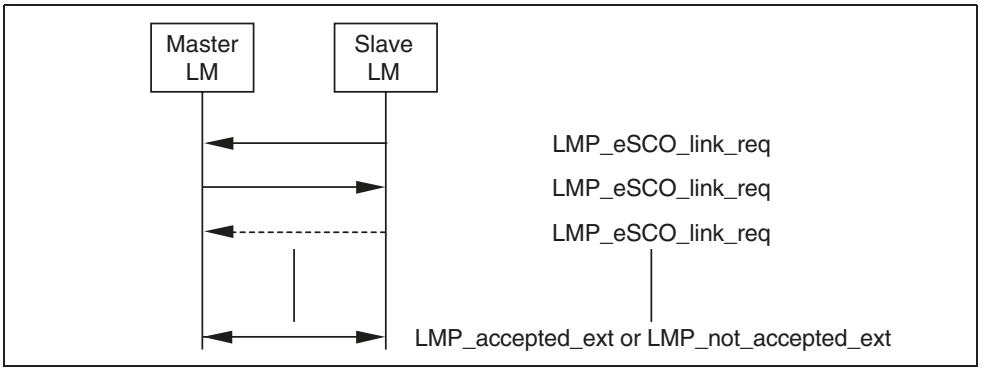

*Sequence 93: Master requests an eSCO link*

### *4.6.2.2 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 can accept this with an 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.



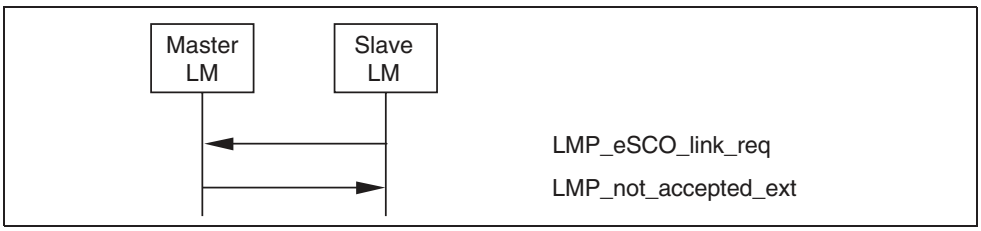
LifeScan, Inc. Ex. 1019
Page 464 of 1420
IPR2025-00103




*Sequence 94: Slave requests an eSCO link.*

Note that the slave should use the initialization flag appropriate to the master's Bluetooth clock. See Baseband section section 8.6.3.

The master may reject the request immediately with an LMP_not_accepted_ext PDU. The negotiation shall use the procedures defined in Section 4.6.2.5 on page 296.




*Sequence 95: Master rejects slave's request for an eSCO link.*

### *4.6.2.3 Master or slave requests change of eSCO parameters*

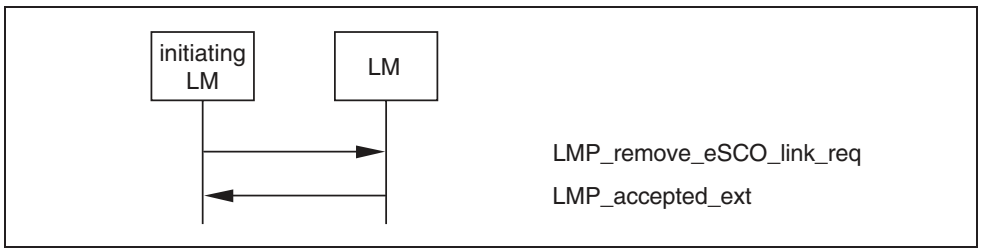
The master or slave may request a renegotiation of the eSCO parameters. The master or slave shall send an LMP_eSCO_link_req PDU with the eSCO handle of the eSCO link the device wishes to renegotiate. The remote device may accept the changed parameters immediately with LMP_accepted_ext PDU, or the negotiation 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 negotiation 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 Section 4.6.2.2 on page 294.

During re-negotiation, the eSCO LT_ADDR and eSCO handle shall not be re-negotiated and shall be set to the originally negotiated values. The negotiation shall use the procedures defined in Section 4.6.2.5 on page 296.

#### *4.6.2.4 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.




*Sequence 96: eSCO link removed*

#### *4.6.2.5 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 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 LMP_not_accepted_ext 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 LMP_not_accepted_ext) a new negotiation may be started with different bandwidth and air_mode parameters.

**Rule 4:** if the parameters will 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 LMP_not_accepted_ext.

**Rule 5:** once a device has received an LMP_eSCO_link_req 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 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 will 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 LMP_not_accepted_ext.

**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 LMP_not_accepted_ext.

**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.

#### *4.6.2.6 Negotiation state definitions*

Reserved Slot Violation: a reserved slot violation is when the receiving LM cannot setup 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).

Latency Violation: a latency violation is when the receiving LM cannot setup 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.

## 4.7 TEST MODE

LMP has PDUs to support different test modes used for certification and compliance testing of the Bluetooth radio and baseband. See "Test Methodology" on page 249 in Vol. 3 for a detailed description of these test modes.

### 4.7.1 Activation and deactivation of test mode

The activation may be carried out locally (via a HW or SW interface), or using the air interface.

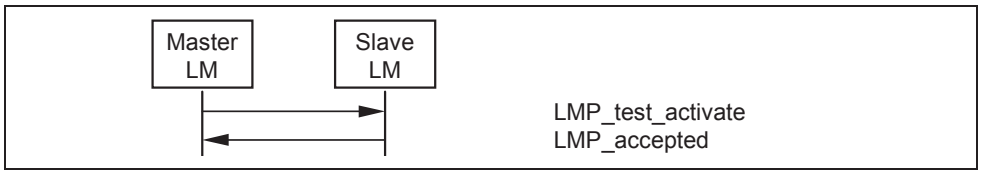
- For activation over the air interface, entering the test mode shall be locally enabled for security and type approval reasons. The implementation of this local enabling is not subject to standardization.

  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.

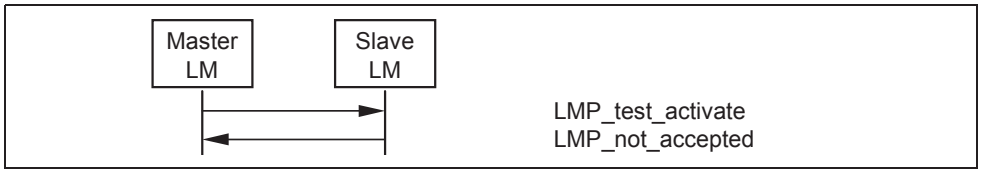
  The DUT shall return an LMP_Accepted on reception of an activation command. LMP_Not_Accepted shall be returned if the DUT is not locally enabled.
- If the activation is performed locally using a HW or SW interface, the DUT shall terminate all normal operation before entering the test mode.

  Until a connection to the tester exists, the device shall perform page scan and inquiry scan. Extended scan activity is recommended.

The test mode is activated by sending an LMP_test_activate PDU to the device under test (DUT). The DUT is always the slave. The lm shall be able to receive this message anytime. If entering test mode is locally enabled in the DUT it shall respond with an LMP_accepted PDU and test mode is entered. Otherwise the DUT responds with an LMP_not_accepted PDU and the DUT remains in normal operation. The error code in the LMP_not_accepted PDU shall be *PDU not allowed*.




*Sequence 97: Activation of test mode successful.*




*Sequence 98: Activation of test mode fails. Slave is not allowed to enter test mode.*



LifeScan, Inc. Ex. 1019
Page 468 of 1420
IPR2025-00103

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.

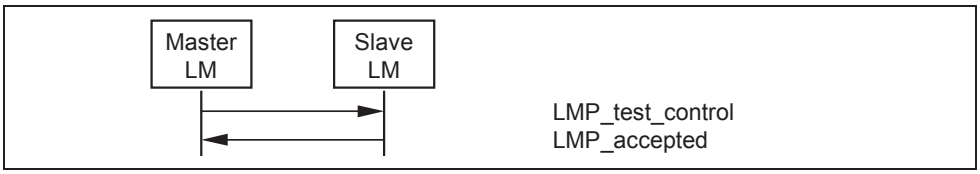
### 4.7.2 Control of test mode

Control and configuration is performed using special LMP commands (see Section 4.7.3 on page 299). These commands shall be rejected if the Bluetooth device is not in test mode. In this case, an LMP_not_accepted shall be returned. The DUT shall return an LMP_accepted on reception of a control command when in test mode.

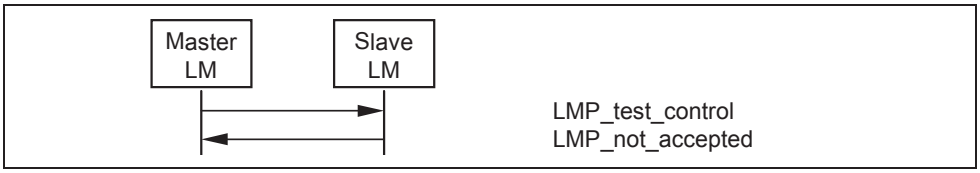
A Bluetooth device in test mode shall ignore all LMP commands not related to control of the test mode. LMP commands dealing with power control and the request for LMP features (LMP_features_req), and adaptive frequency hopping (LMP_set_AFH, LMP_channel_classification_req and LMP_channel_classification) 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 PDU 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*.




*Sequence 99: Control of test mode successful.*




*Sequence 100: Control of test mode rejected since slave is not in test mode.*

### 4.7.3 Summary of test mode PDUs

Table 4.14 lists all LMP messages used for test mode. To ensure that the contents of LMP_test_control PDU are suitably whitened (important when sent in

transmitter mode), all parameters listed in Table 4.31 on page 300 are XORed with 0x55 before being sent.

| LMP PDU | PDU number | Possible Direction | Contents | Position in Payload |
|---|---|---|---|---|
| LMP_test_activate | 56 | m → s | | |
| LMP_test_control | 57 | m → s | test scenario<br>hopping mode<br>TX frequency<br>RX frequency<br>power control mode<br>poll period<br>packet type<br>length of test data | 2<br>3<br>4<br>5<br>6<br>7<br>8<br>9-10 |
| LMP_detach | 7 | m → s | | |
| LMP_accepted | 3 | m ← s | | |
| LMP_not_accepted | 4 | m ← s | | |

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| Test scenario | 1 | u_int8 | | 0 Pause Test Mode<br>1 Transmitter test – 0 pattern<br>2 Transmitter test – 1 pattern<br>3 Transmitter test – 1010 pattern<br>4 Pseudorandom bit sequence<br>5 Closed Loop Back – ACL packets<br>6 Closed Loop Back – Synchronous packets<br>7 ACL Packets without whitening<br>8 Synchronous Packets without whitening<br>9 Transmitter test – 1111 0000 pattern<br>10–254 reserved<br>255 Exit Test Mode<br>The value is XORed with 0x55. |
| Hopping mode | 1 | u_int8 | | 0 RX/TX on single frequency<br>1 Normal hopping<br>2 Reserved<br>3 Reserved<br>4 Reserved<br>5–255 reserved<br>The value is XORed with 0x55. |
| TX frequency (for DUT) | 1 | u_int8 | | f = [2402 + k] MHz<br>The value is XORed with 0x55. |

*Table 4.31: Parameters used in LMP_Test_Control PDU*

| Name | Length (bytes) | Type | Unit | Detailed |
|---|---|---|---|---|
| RX frequency (for DUT) | 1 | u_int8 | | f = [2402 + k] MHz<br>The value is XORed with 0x55. |
| Power control mode | 1 | u_int8 | | 0 fixed TX output power<br>1 adaptive power control<br>The value is XORed with 0x55. |
| Poll period | 1 | u_int8 | 1.25 ms | The value is XORed with 0x55. |
| Packet type | 1 | u_int8 | | Bits 3-0<br>numbering as in packet header, see Baseband Specification<br>Bits 7-4<br>0: ACL/SCO<br>1: eSCO<br>2: Enhanced Data Rate ACL<br>3: Enhanced Data Rate eSCO<br>4-15: reserved<br>Other values are reserved<br>The value is XORed with 0x55. |
| length of test sequence (=length of user data in Baseband Specification) | 2 | u_int16 | 1 byte | unsigned binary number<br>The value is XORed with 0x5555. |

*Table 4.31: Parameters used in LMP_Test_Control PDU*

The control PDU is used for both transmitter and loop back tests. The following restrictions apply for the parameter settings:

| **Parameter** | **Restrictions Transmitter Test** | **Restrictions Loopback Test** |
|---|---|---|
| TX frequency | $0 \leq k \leq 93$ | $0 \leq k \leq 78$ |
| RX frequency | same as TX frequency | $0 \leq k \leq 78$ |
| Poll period | | not applicable (set to 0) |
| Length of test sequence | Depends on packet type. See table 6.9 and 6.10 in the Baseband specification | For ACL and SCO packets: not applicable (set to 0)<br><br>For eSCO packets: [see table 6.10 in the Baseband specification] |

# 5 SUMMARY

## 5.1 PDU SUMMARY

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| Escape 1 | variable | 124 | DM1 | m ↔ s | extended op code | 2 |
| | | | | | variable | 3-? |
| Escape 2 | variable | 125 | DM1 | m ↔ s | extended op code | 2 |
| | | | | | variable | 3-? |
| Escape 3 | variable | 126 | DM1 | m ↔ s | extended op code | 2 |
| | | | | | variable | 3-? |
| Escape 4 | variable | 127 | DM1 | m ↔ s | extended op code | 2 |
| | | | | | variable | 3-? |
| LMP_accepted | 2 | 3 | DM1/DV | m ↔ s | op code | 2 |
| LMP_accepted_ext | 4 | 127/01 | DM1 | m ↔ s | escape op code | 3 |
| | | | | | extended op code | 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_DHkey_ Check | 17 | 65 | DM1 | m ↔ s | Confirmation Value | 2-17 |

*Table 5.1: Coding of the different LM PDUs*



LifeScan, Inc. Ex. 1019
Page 473 of 1420
IPR2025-00103

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_encapsulated_header | 4 | 61 | DM1 | m ↔ s | encapsulated major type | 2 |
| | | | | | encapsulated minor type | 3 |
| | | | | | encapsulated pay-load length | 4 |
| LMP_encapsulated_payload | 17 | 62 | DM1 | m ↔ s | encapsulated data | 2-17 |
| 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 |
| 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 |
| | | | | | eSCO packet type M->S | 9 |
| | | | | | eSCO 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 |

*Table 5.1: Coding of the different LM PDUs*

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_features_req_ext | 12 | 127/03 | DM1 | m ↔ s | features page | 3 |
| | | | | | max 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 |
| | | | | | max 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_IO_Capability_req | 5 | 127/25 | DM1 | m ↔ s | IO_capabilities | 3 |
| | | | | | OOB Authentication Data | 4 |
| | | | | | Authentication_Requirement | 5 |
| LMP_IO_Capability_res | 5 | 127/26 | DM1 | m ↔ s | IO_capabilities | 3 |
| | | | | | OOB Authentication Data | 4 |
| | | | | | Authentication_Requirement | 5 |
| LMP_pause_encryption_req | 2 | 127/23 | DM1 | m ↔ s | – | |
| LMP_resume_encryption_req | 2 | 127/24 | DM1 | m ← s | – | |
| LMP_max_power | 1 | 33 | DM1/DV | m ↔ s | – | |
| LMP_max_slot | 2 | 45 | DM1/DV | m ↔ s | max slots | 2 |

*Table 5.1: Coding of the different LM PDUs*

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_max_slot_req | 2 | 46 | DM1/DV | m ↔ s | max 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 |
| | | | | | $\Delta_B$ | 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_numeric_comparison_failed | 2 | 127/27 | DM1 | m ↔ s | – | |
| LMP_not_accepted | 3 | 4 | DM1/DV | m ↔ s | op code | 2 |
| | | | | | error code | 3 |
| LMP_not_accepted_ext | 5 | 127/02 | DM1 | m ↔ s | escape op code | 3 |
| | | | | | extended op code | 4 |
| | | | | | error code | 5 |
| LMP_packet_type_table_req | 3 | 127/11 | DM1 | m ↔ s | packet type table | 3 |
| LMP_page_mode_req | 3 | 53 | DM1/DV | m ↔ s | paging scheme | 2 |
| | | | | | paging scheme settings | 3 |

*Table 5.1: Coding of the different LM PDUs*

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_page_scan_ mode_req | 3 | 54 | DM1/DV | m ↔ s | paging scheme | 2 |
| | | | | | paging scheme set-tings | 3 |
| LMP_park_req | 17 | 25 | DM1 | m ↔ s | timing control flags | 2 |
| | | | | | $D_B$ | 3-4 |
| | | | | | $T_B$ | 5-6 |
| | | | | | $N_B$ | 7 |
| | | | | | $\Delta_B$ | 8 |
| | | | | | PM_ADDR | 9 |
| | | | | | AR_ADDR | 10 |
| | | | | | $N_{Bsleep}$ | 11 |
| | | | | | $D_{Bsleep}$ | 12 |
| | | | | | $D_{access}$ | 13 |
| | | | | | $T_{access}$ | 14 |
| | | | | | $N_{acc-slots}$ | 15 |
| | | | | | $N_{poll}$ | 16 |
| | | | | | $M_{access}$ | 170-3 |
| | | | | | access scheme | v |
| LMP_passkey_ failed | 2 | 127/28 | DM1 | m ↔ s | – | |
| LMP_oob _failed | 2 | 127/29 | DM1 | m ↔ s | | |
| LMP_keypress _notification | 3 | 127/30 | DM1 | m ↔ s | Notification Type | 2 |
| 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_o f_service_req | 4 | 42 | DM1/DV | m ↔ s | poll interval | 2-3 |
| | | | | | $N_{BC}$ | 4 |

*Table 5.1: Coding of the different LM PDUs*



LifeScan, Inc. Ex. 1019
Page 477 of 1420
IPR2025-00103

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_remove_eSCO_link_req see Note 4 on page 312 | 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 |
| 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_Simple_Pairing_Confirm | 17 | 63 | DM1 | m ↔ s | Commitment Value | 2-17 |
| LMP_Simple_Pairing_Number | 17 | 64 | DM1 | m ↔ s | Nonce Value | 2-17 |
| 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 |

*Table 5.1: Coding of the different LM PDUs*

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| 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 | – | |
| 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 |

*Table 5.1: Coding of the different LM PDUs*

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| 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 |

*Table 5.1: Coding of the different LM PDUs*



LifeScan, Inc. Ex. 1019
Page 480 of 1420
IPR2025-00103

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| 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 6th unpark | 12 |
| | | | | | 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 |

*Table 5.1: Coding of the different LM PDUs*



LifeScan, Inc. Ex. 1019
Page 481 of 1420
IPR2025-00103

| LMP PDU | Length (bytes) | op code | Packet type | Possible direction | Contents | Position in payload |
|---|---|---|---|---|---|---|
| LMP_version_res | 6 | 38 | DM1/DV | m ↔ s | VersNr | 2 |
| | | | | | CompId | 3-4 |
| | | | | | SubVersNr | 5-6 |
| LMP_sniff_ subrating_req | 9 | 127/21 | DM1 | m ↔ s | max_sniff_subrate | 3 |
| | | | | | min_sniff_mode_ timeout | 4-5 |
| | | | | | sniff_subrating_inst ant | 6-9 |
| LMP_sniff_ subrating_res | 9 | 127/22 | DM1 | m ↔ s | max_sniff_subrate | 3 |
| | | | | | min_sniff_mode_ timeout | 4-5 |
| | | | | | sniff_subrating_ instant | 6-9 |

*Table 5.1: Coding of the different LM PDUs*

**Note1**: For LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_ADDR_req and LMP_unpark_PM_ADDR_req the parameter $D_B$ is optional. This parameter is only present if bit0 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.

**Note2:** For LMP_unpark_BD_ADDR 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.

**Note3:** For LMP_unpark_PM_ADDR 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 Note1).

**Note4:** Parameters coincide with their namesakes in LMP_<remove>_SCO_link_req apart from the following:

1. eSCO_LT_ADDR - the eSCO connection will be active on an additional LT_ADDR that needs to be defined. The master is allowed to re-assign an active eSCO link to a different LT_ADDR.



LifeScan, Inc. Ex. 1019
Page 482 of 1420
IPR2025-00103

2. $D_{eSCO}$, $T_{eSCO}$ - 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).
3. $W_{eSCO}$ - the eSCO retransmission window size (in slots)
4. packet type and packet length may be prescribed differently in Master-to-Slave or Slave-to-Master directions for asynchronous eSCO links
5. packet length (in bytes) - eSCO packet types no longer have fixed length
6. 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.

## 5.2 PARAMETER DEFINITIONS

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| access scheme | 1 | u_int4 | | 0: polling technique<br>1-15: Reserved | |
| AFH_channel_classification | 10 | multiple bytes | - | This parameter contains 40 2-bit fields.<br>The $n^{th}$ (numbering from 0) such field defines the classification of channels *2n* and *2n+1*, other than the $39^{th}$ field which just contains the classification of channel 78.<br>Each field interpreted as an integer whose values indicate:<br>0 = unknown<br>1 = good<br>2 = reserved<br>3 = bad | |

*Table 5.2: Parameters in LM PDUs.*

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| AFH_channel_map | 10 | multiple bytes | - | If *AFH_mode* is *AFH_enabled*, this parameter contains 79 1-bit fields, otherwise the contents are reserved.<br>The $n^{th}$ (numbering from 0) such field (in the range 0 to 78) contains the value for channel *n*.<br>Bit 79 is reserved (set to 0 when transmitted and ignored when received)<br>The 1-bit field is interpreted as follows:<br>0: channel *n* is unused<br>1: channel *n* is used | |
| AFH_instant | 4 | u_int32 | slots | Bits 27:1 of the Bluetooth master clock 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 30s) | |
| AFH_min_interval | 2 | u_int16 | slots | Range is 0x0640 to 0xBB80 slots (1 to 30s) | |
| AFH_mode | 1 | u_int8 | - | 0: AFH_disabled<br>1: AFH_enabled<br>2-255: Reserved | |
| AFH_reporting_mode | 1 | u_int8 | - | 0: AFH_reporting_disabled<br>1: AFH_reporting_enabled<br>2-255: reserved | |
| air mode | 1 | u_int8 | | 0: μ-law log<br>1: A-law log<br>2: CVSD<br>3: transparent data<br>4-255: Reserved | See Table 5.3 on page 323 |
| AR_ADDR | 1 | u_int8 | | | |

*Table 5.2: Parameters in LM PDUs.*

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| Authentication_ Requirements | 1 | u_int8 | | 0x00:MITM Protection Not Required – No Bonding<br>0x01: MITM Protection Required – No Bonding<br>0x02: MITM Protection Not Required – Dedicated Bonding<br>0x03: MITM Protection Required – Dedicated Bonding<br>0x04: MITM Protection Not Required – General Bonding<br>0x05: MITM Protection Required – General Bonding<br>0x06 to 0xFF– Reserved | |
| 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 | 1.25ms | ($CLKN_{16-2}$ slave - $CLKN_{16-2}$ master) mod $2^{15}$<br>MSbit of second byte not used. | |
| CompId | 2 | u_int16 | | see Bluetooth Assigned Numbers) | |
| Commitment value | 16 | Multiple bytes | | Little Endian format | |
| Confirmation value | 16 | Multiple bytes | | Little Endian format | |
| $D_{access}$ | 1 | u_int8 | slots | | |

*Table 5.2: Parameters in LM PDUs.*

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| data rate | 1 | u_int8 | | When in Basic Rate mode:<br>bit0 = 0: use FEC<br>bit0 = 1: do not use FEC<br>bit1-2=0: No packet-size preference available<br>bit1-2=1: use 1-slot packets<br>bit1-2=2: use 3-slot packets<br>bit1-2=3: use 5-slot packets<br><br>When in Enhanced Data Rate mode:<br>bit3-4=0: use DM1 packets<br>bit3-4=1: use 2 Mbps packets<br>bit3-4=2: use 3 Mbps packets<br>bit3-4=3: reserved<br>bit5-6=0: No packet-size preference available<br>bit5-6=1: use 1-slot packets<br>bit5-6=2: use 3-slot packets<br>bit5-6=3: use 5-slot packets<br><br>bit7: Reserved - shall be zero | |
| $D_B$ | 2 | u_int16 | slots | | |
| $\Delta_B$ | 1 | u_int8 | slots | | |
| $D_{Bsleep}$ | 1 | u_int8 | | | |
| $D_{eSCO}$ | 1 | u_int8 | slots | Valid range is 0 - 254 slots | See Table 5.3 on page 323 |
| drift | 1 | u_int8 | ppm | | |
| $D_{sco}$ | 1 | u_int8 | slots | Only even values are valid[1] | 0 to $T_{sco}$ -2 |
| $D_{sniff}$ | 2 | u_int16 | slots | Only even values are valid[1] | 0 to ($T_{sniff}$ - 2) |

*Table 5.2: Parameters in LM PDUs.*

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| encapsulated data | 16 | Multiple bytes | | MSBs zero padded when data is less than 16 bytes.<br>Little Endian format | |
| encapsulated major type | 1 | u_int8 | | See Table 5.3, "LMP encapsulated," on page 323 | |
| encapsulated minor type | 1 | u_int8 | | See Table 5.3, "LMP encapsulated," on page 323 | |
| encapsulated payload length | 1 | u_int8 | | See Table 5.3, "LMP encapsulated," on page 323 | |
| encryption mode | 1 | u_int8 | | 0: no encryption<br>1: encryption<br>2: encryption<br>3 -255: Reserved | |
| error code | 1 | u_int8 | | See Section Part D on page 331 | |
| escape op code | 1 | u_int8 | | Identifies which escape op code is being acknowledged: range 124-127 | |
| eSCO handle | 1 | u_int8 | | | |
| eSCO LT_ADDR | 1 | u_int8 | | Logical transport address 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<br>0x07: EV3<br>0x0C: EV4<br>0x0D: EV5<br>0x26: 2-EV3<br>0x2C: 2-EV5<br>0x37: 3-EV3<br>0x3D: 3-EV5<br>Other values are reserved | If the value is 0x00 the POLL packet shall be used by the master, the NULL packet shall be used by the slave. See Table 5.3 on page 323 |
| extended features | 8 | multiple bytes | | One page of extended features | |

*Table 5.2: Parameters in LM PDUs.*



LifeScan, Inc. Ex. 1019
Page 487 of 1420
IPR2025-00103

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| extended op code | 1 | u_int8 | | Which extended op code is being acknowledged | |
| features | 8 | multiple bytes | | See Table 3.2 on page 220 | |
| features page | 1 | u_int8 | | Identifies which page of extended features is being requested.<br>0 means standard features<br>1-255 other feature pages | |
| hold instant | 4 | u_int32 | slots | Bits 27:1 of the master Bluetooth clock value | |
| hold time | 2 | u_int16 | slots | Only even values are valid[1] | 0x0014 to 0x8000; shall not exceed (supervisi-onTO * 0.999) |
| IO_ Capabilities | 1 | u_int8 | | 0: Display only<br>1: Display YesNo<br>2: KeyboardOnly<br>3: NoInputNoOutput<br>4-255: reserved | |
| 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: least significant bit 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 win-dows | |
| max slots | 1 | u_int8 | slots | | |
| max_sniff_subr ate | 1 | u_int8 | sub-rate | | 1-255 |

*Table 5.2: Parameters in LM PDUs.*

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| min_sniff_ mode_timeout | 2 | u_int16 | slots | | 0x0000 to 0xFFFE (even) |
| max supported page | 1 | u_int8 | | Highest page of extended features which contains a non-zero bit for the originating device. Range 0-255 | |
| $N_{acc-slots}$ | 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<br>1: the latest received set of negotiable parameters were possible but these parameters are preferred.<br>2: the latest received set of negotiable parameters would cause a reserved slot violation.<br>3: the latest received set of negotiable parameters would cause a latency violation.<br>4: the latest received set of of negotiable parameters are not supported.<br>Other values are reserved | |
| $N_{poll}$ | 1 | u_int8 | | | |
| Nonce Value | 16 | Multiple bytes | | Little Endian Format | |

*Table 5.2: Parameters in LM PDUs.*



LifeScan, Inc. Ex. 1019
Page 489 of 1420
IPR2025-00103

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| Notification Value | 1 | U_int8 | | 0=passkey entry started<br>1=passkey digit entered<br>2=passkey digit erased<br>3=passkey cleared<br>4=passkey entry completed<br>5-255: reserved | |
| OOB Authentication Data | 1 | u_int8 | | 0: No OOB Authentication Data received<br>1: OOB Authentication Data received<br>2-255: reserved | |
| op code | 1 | u_int8 | | | |
| packet length | 2 | uint16 | bytes | Length of the eSCO payload<br>0 for POLL/NULL<br>1-30 for EV3<br>1-120 for EV4<br>1-180 for EV5<br>1-60 for 2-EV3<br>1-360 for 2-EV5<br>1-90 for 3-EV3<br>1-540 for 3-EV5<br>Other values are invalid | See Table 5.3 on page 323 |
| packet type table | 1 | u_int8 | | 0: 1Mbps only<br>1: 2/3 Mbps<br>2-255: reserved | 0-1 |
| paging scheme | 1 | u_int8 | | 0: mandatory scheme<br>1-255: Reserved | |
| paging scheme settings | 1 | u_int8 | | For mandatory scheme:<br>0: R0<br>1: R1<br>2: R2<br>3-255: Reserved | |
| PM_ADDR | 1 | u_int8 | | | |
| poll interval | 2 | u_int16 | slots | Only even values are valid[1] | 0x0006 to 0x1000 |
| random number | 16 | multiple bytes | | | |

*Table 5.2: Parameters in LM PDUs.*

| 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<br>1: HV2<br>2: HV3<br>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_subrating_instant | 4 | u_int32 | slots | Bits 27:1 of the master Bluetooth clock value | |
| sniff timeout | 2 | u_int16 | received slots | Number of receive slots | 0 to 0x0028 |
| SubVersNr | 2 | u_int16 | | Defined by each company | |
| supervision timeout | 2 | u_int16 | slots | 0 means an infinite time-out | 0 and 0x0190 to 0xFFFF |
| switch instant | 4 | u_int32 | slots | Bits 27:1 of the master Bluetooth clock 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 5.3 on page 323 |
| timing control flags | 1 | u_int8 | | bit0 = 0: o timing change<br>bit0 = 1: timing change<br>bit1 = 0: use initialization 1<br>bit1 = 1: use initialization 2<br>bit2 = 0: access window<br>bit2 = 1: no access window<br>bit3-7: Reserved | |
| $T_{sco}$ | 1 | u_int8 | slots | Only even values are valid[1] | 2 and 6 |

*Table 5.2: Parameters in LM PDUs.*

| Name | Length (bytes) | Type | Unit | Detailed | Mandatory range |
|---|---|---|---|---|---|
| $T_{sniff}$ | 2 | u_int16 | slots | Only even values are valid[1] | 0x0006 to 0x0540; shall not exceed (supervisionTO * 0.999) |
| VersNr | 1 | u_int8 | | See Bluetooth Assigned Numbers | |
| $W_{eSCO}$ | 1 | u_int8 | slots | Number of slots in the retransmission window<br>Valid range is 0 – 254 slots | See Table 5.3 on page 323 |

*Table 5.2: Parameters in LM PDUs.*

1. 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*.



LifeScan, Inc. Ex. 1019
Page 492 of 1420
IPR2025-00103

## 5.3 LMP ENCAPSULATED

| Name | Major Type | Minor Type | Payload Length | Detailed |
|---|---|---|---|---|
| P-192 Public Key | 1 | 1 | 48 | X, Y format<br>Bytes 23-0: X co-ordinate<br>Bytes 47-24: Y co-ordinate<br><br>Little Endian Format |

*Table 5.3: LMP encapsulated*

| | Single Slot Packets | 3-Slot Packets |
|---|---|---|
| **$D_{eSCO}$** | 0 to **$T_{eSCO}$**-2 (even) | 0 to **$T_{eSCO}$**-2 (even) |
| **$T_{eSCO}$** | EV3: 6<br>2-EV3: 6-12 (even)<br>3-EV3: 6-18 (even) | EV4: 16<br>EV5: 16<br>2-EV5: 16<br>3-EV5: 16 |
| **$W_{eSCO}$** | 0, 2, and 4 | 0 and 6 |
| **packet length M->S** | 10*$T_{eSCO}$ /2 | 10*$T_{eSCO}$ /2 |
| **packet length S->M** | 10*$T_{eSCO}$ /2 | 10*$T_{eSCO}$ /2 |
| **air mode** | At least one of A-law, mu-law, CVSD, transparent | transparent |

*Table 5.4: Mandatory parameter ranges for eSCO packet types*

## 5.4 DEFAULT VALUES

Devices shall use these values before anything else has been negotiated:

| Parameter | Value |
|---|---|
| AFH_mode | AFH_disabled |
| AFH_reporting_mode | AFH_reporting_disabled |
| drift | 250 |
| jitter | 10 |
| max slots | 1 |
| poll interval | 40 |

*Table 5.5: Device default values*



LifeScan, Inc. Ex. 1019
Page 494 of 1420
IPR2025-00103

# 6 LIST OF FIGURES

Figure 1.1: Link Manager Protocol signaling layer. ....207
Figure 2.1: Transmission of a message from master to slave. ....209
Figure 2.2: Payload body when LMP PDUs are sent. ....210
Figure 2.3: Symbols used in sequence diagrams. ....212
Figure 4.1: Connection establishment. ....223
Sequence 1: Connection closed by sending LMP_detach. ....225
Sequence 2: A device requests a change of the other device's TX power. ....226
Sequence 3: The TX power cannot be increased. ....226
Sequence 4: The TX power cannot be decreased. ....226
Sequence 5: Master Enables AFH. ....228
Sequence 6: Master disables AFH. ....228
Sequence 7: Master Updates AFH. ....229
Sequence 8: Channel classification reporting. ....231
Sequence 9: Setting the link supervision timeout. ....232
Sequence 10: A notifies B to enable CQDDR ....233
Sequence 11: B sends A a preferred packet type ....233
Sequence 12: Master notifies slave of quality of service. ....234
Sequence 13: Device accepts new quality of service ....235
Sequence 14: Device rejects new quality of service. ....235
Sequence 15: Negotiation for page mode. ....236
Sequence 16: Negotiation for page scan mode ....236
Sequence 17: Device allows Remote Device to use a maximum number of slots. ....237
Sequence 18: Device requests a maximum number of slots. Remote Device accepts. ....237
Sequence 19: Device requests a maximum number of slots. Remote Device rejects. ....237
Sequence 20: Packet type table change is rejected. ....238
Sequence 21: Packet type table change is accepted. ....239
Figure 4.2: Sequence 1: Sending an encapsulated PDU ....240
Sequence 22: Authentication. Claimant has link key. ....241
Sequence 23: Authentication fails. Claimant has no link key. ....241
Sequence 24: Pairing accepted. Responder has a variable PIN. Initiator has a variable or fixed PIN. ....243
Sequence 25: Responder has a fixed PIN and initiator has a variable PIN. ....243
Sequence 26: Both devices have a fixed PIN. ....243
Sequence 27: Responder rejects pairing. ....244
Sequence 28: Creation of the link key. ....245
Sequence 29: Successful change of the link key ....245

Sequence 30: Change of the link key not possible since the other device uses a unit key. 245
Sequence 31: Change to a temporary link key ..... 247
Sequence 32: Link key changed to the semi-permanent link key ..... 247
Sequence 33: Negotiation for encryption mode ..... 249
Sequence 34: Encryption key size negotiation successful ..... 250
Sequence 35: Encryption key size negotiation failed ..... 250
Sequence 36: Start of encryption ..... 250
Sequence 37: Stop of encryption ..... 251
Sequence 38: Master-initiated pausing of encryption ..... 252
Sequence 39: Slave-initiated pausing of encryption ..... 252
Sequence 40: Initiating Master LM resumes encryption ..... 253
Sequence 41: Initiating Slave LM resume encryption ..... 253
Sequence 42: Request for supported encryption key sizes. ..... 255
Sequence 43: IO Capability Exchange ..... 256
Sequence 44: IO Capability exchange with transaction Collision (master transmitting LMP_io_capability_req first) and resolution 257
Sequence 45: IO Capability exchange with transaction Collision (slave transmitting LMP_io_capability_req first) and resolution 257
Sequence 46: Public key exchange ..... 258
Sequence 47: Numeric Comparison Authentication: Commitment check success ..... 259
Sequence 48: Numeric Comparison Authentication: Commitment check failure ..... 260
Sequence 49: Authentication stage 1 Numeric Comparison Failure on Initiator Side ..... 260
Sequence 50: Authentication stage 1: Numeric Comparison Failure on Responder Side ..... 261
Sequence 51: Authentication Passkey entry ..... 262
Sequence 52: Authentication Passkey Entry: Commitment check failure on the Responder side ..... 262
Sequence 53: Authentication Passkey Entry: Commitment check failure on the Initiator side ..... 263
Sequence 54: Authentication stage 1 Passkey Entry Failure on Initiator Side ..... 263
Sequence 55: Authentication Stage 1 Passkey Entry Failure on Responding Side ..... 264
Sequence 56: Keypress Notifications ..... 264
Sequence 57: Authentication OOB: Only one device is OOB-r capable ... 265
Sequence 58: Authentication stage 1 OOB: Commitment check failure on the Responder side ..... 266
Sequence 59: Authentication stage 1 OOB: Commitment check failure on the Initiator side ..... 266
Sequence 60: Authentication stage 1 OOB: OOB information not available on



LifeScan, Inc. Ex. 1019
Page 496 of 1420
IPR2025-00103

the Initiator side ....266
Sequence 61: DHKey check ....267
Sequence 62: DHKey check: Check failure on the Responder side DHKey check: Check failure on the Responder ....268
Sequence 63: DHKey check: check failure on the Initiator side ....268
Sequence 64: The requested device supports timing accuracy information. ....269
Sequence 65: The requested device does not support timing accuracy information. ....269
Sequence 66: Clock offset requested. ....270
Sequence 67: Request for LMP version. ....271
Sequence 68: Request for supported features. ....272
Sequence 69: Request for extended features. ....272
Sequence 70: Device's name requested and it responses. ....273
Figure 4.3: Slot offset for role switch ....274
Sequence 71: Slot offset information is sent ....274
Sequence 72: Role switch (slave initiated). ....276
Sequence 73: Role switch (master initiated). ....276
Sequence 74: Master forces slave into hold mode ....278
Sequence 75: Slave forces master into hold mode ....278
Sequence 76: Negotiation for hold mode ....279
Sequence 77: Slave accepts to enter park state. ....283
Sequence 78: Slave rejects to enter into park state ....283
Sequence 79: Slave requests to enter park state and accepts master's beacon parameters ....284
Sequence 80: Master rejects slave's request to enter park state ....284
Sequence 81: Slave requests to enter park state, but rejects master's beacon parameters ....284
Sequence 82: Master notifies all slaves of increase in broadcast capacity ....284
Sequence 83: Master modifies beacon parameters ....285
Sequence 84: Master unparks slaves addressed with their BD_ADDR ....285
Sequence 85: Master unparks slaves addressed with their PM_ADDR ....286
Sequence 86: Negotiation for sniff mode ....287
Sequence 87: Slave moved from sniff mode to active mode ....288
Sequence 88: LM accepts sniff subrating request ....289
Sequence 89: Master requests an SCO link ....291
Sequence 90: Master rejects slave's request for an SCO link ....291
Sequence 91: Master accepts slave's request for an SCO link ....292
Sequence 92: SCO link removed ....293
Sequence 93: Master requests an eSCO link ....294
Sequence 94: Slave requests an eSCO link. ....295
Sequence 95: Master rejects slave's request for an eSCO link. ....295



LifeScan, Inc. Ex. 1019
Page 497 of 1420
IPR2025-00103

Sequence 96: eSCO link removed ..... 296
Sequence 97: Activation of test mode successful. ..... 298
Sequence 98: Activation of test mode fails. Slave is not allowed to enter test mode. ..... 298
Sequence 99: Control of test mode successful. ..... 299
Sequence 100: Control of test mode rejected since slave is not in test mode. ..... 299



LifeScan, Inc. Ex. 1019
Page 498 of 1420
IPR2025-00103

# 7 LIST OF TABLES

Table 2.1: General response messages. ....213
Table 3.1: Feature Definitions ....214
Table 3.2: Feature mask definitions (page 0) ....220
Table 3.3: Extended feature mask definition (page 1) ....222
Table 4.1: Connection establishment PDU ....224
Table 4.2: Detach PDU ....224
Table 4.3: Power control PDU ....225
Table 4.4: AFH PDU ....227
Table 4.5: Channel classification reporting PDU ....230
Table 4.6: Set supervision timeout PDU ....232
Table 4.7: Qualifty-driven change of the data rate PDU ....233
Table 4.8: Quality of Service PDU ....234
Table 4.9: Paging scheme request PDU ....236
Table 4.10: Multi-slot packet control PDU ....237
Table 4.11: Enhanced Data Rate PDUs ....238
Table 4.12: Encapsulated LMP PDUs ....239
Table 4.13: Authentication PDUs ....241
Table 4.14: Pairing PDUs ....242
Table 4.15: Change link key PDU ....245
Table 4.16: Change current link key PDU ....246
Table 4.17: Encryption handling PDU ....248
Table 4.18: Encryption key size request PDU ....255
Table 4.19: Request limited timing PDU ....269
Table 4.20: PDUs used for clock offset request. ....270
Table 4.21: PDUs used for LMP version request. ....271
Table 4.22: PDUs used for features request. ....272
Table 4.23: Name request PDUs ....273
Table 4.24: Role switch PDU ....274
Table 4.25: Role switch PDU ....275
Table 4.26: Hold mode PDUs ....277
Table 4.27: PDUs used for park state ....281
Table 4.28: Sniff mode PDUs ....286
Table 4.29: SCO link management PDUs ....290
Table 4.30: PDUs used for managing the eSCO links ....293
Table 4.31: Parameters used in LMP_Test_Control PDU ....300
Table 5.1: Coding of the different LM PDUs ....303
Table 5.2: Parameters in LM PDUs. ....313
Table 5.3: LMP encapsulated ....323
Table 5.4: Mandatory parameter ranges for eSCO packet types ....323
Table 5.5: Device default values ....324

LifeScan, Inc. Ex. 1019
Page 500 of 1420
IPR2025-00103

# ERROR CODES



LifeScan, Inc. Ex. 1019
Page 501 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 502 of 1420
IPR2025-00103

# CONTENTS

**1 Overview of Error Codes ..... 335**
1.1 Usage Descriptions ..... 335
1.2 HCI Command Errors ..... 335
1.3 List of Error Codes ..... 336
**2 Error Code Descriptions ..... 338**
2.1 Unknown HCI Command (0X01) ..... 338
2.2 Unknown Connection Identifier (0X02) ..... 338
2.3 Hardware Failure (0X03) ..... 338
2.4 Page Timeout (0X04) ..... 338
2.5 Authentication Failure (0X05) ..... 338
2.6 PIN or key Missing (0X06) ..... 338
2.7 Memory Capacity Exceeded (0X07) ..... 338
2.8 Connection Timeout (0X08) ..... 339
2.9 Connection Limit Exceeded (0X09) ..... 339
2.10 Synchronous Connection Limit to a Device Exceeded (0X0A) 339
2.11 ACL Connection Already Exists (0X0B) ..... 339
2.12 Command Disallowed (0X0C) ..... 339
2.13 Connection Rejected due to Limited Resources (0X0D) ..... 339
2.14 Connection Rejected due to Security Reasons (0X0E) ..... 339
2.15 Connection Rejected due to Unacceptable BD_ADDR (0X0F) 340
2.16 Connection Accept Timeout Exceeded (0X10) ..... 340
2.17 Unsupported Feature or Parameter Value (0X11) ..... 340
2.18 Invalid HCI Command Parameters (0X12) ..... 340
2.19 Remote User Terminated Connection (0X13) ..... 340
2.20 Remote Device Terminated Connection due to Low Resources (0X14) ..... 341
2.21 Remote Device Terminated Connection due to Power Off (0X15) ..... 341
2.22 Connection Terminated by Local Host (0X16) ..... 341
2.23 Repeated Attempts (0X17) ..... 341
2.24 Pairing not Allowed (0X18) ..... 341
2.25 Unknown LMP PDU (0X19) ..... 341
2.26 Unsupported Remote Feature / Unsupported LMP Feature (0X1A) ..... 341
2.27 SCO Offset Rejected (0X1B) ..... 341
2.28 SCO Interval Rejected (0X1C) ..... 342
2.29 SCO Air Mode Rejected (0X1D) ..... 342
2.30 Invalid LMP Parameters (0X1E) ..... 342
2.31 Unspecified Error (0X1F) ..... 342
2.32 Unsupported LMP Parameter Value (0X20) ..... 342

2.33 Role Change Not Allowed (0X21) ..... 342
2.34 LMP Response Timeout (0X22) ..... 342
2.35 LMP Error Transaction Collision (0X23) ..... 343
2.36 LMP PDU Not Allowed (0X24) ..... 343
2.37 Encryption Mode Not Acceptable (0X25) ..... 343
2.38 Link Key Can Not be Changed (0X26) ..... 343
2.39 Requested Qos Not Supported (0X27) ..... 343
2.40 Instant Passed (0X28) ..... 343
2.41 Pairing with Unit Key Not Supported (0X29) ..... 343
2.42 Different Transaction Collision (0x2a) ..... 343
2.43 QoS Unacceptable Parameter (0X2C) ..... 343
2.44 QoS Rejected (0X2D) ..... 344
2.45 Channel Classification Not Supported (0X2E) ..... 344
2.46 Insufficient Security (0X2F) ..... 344
2.47 Parameter out of Mandatory Range (0X30) ..... 344
2.48 Role Switch Pending (0X32) ..... 344
2.49 Reserved Slot Violation (0X34) ..... 344
2.50 Role Switch Failed (0X35) ..... 344
2.51 Extended Inquiry Response Too Large (0x36) ..... 344
2.52 Simple Pairing Not Supported By Host (0X37) ..... 345
2.53 Host Busy–Pairing(0X38) ..... 345



LifeScan, Inc. Ex. 1019
Page 504 of 1420
IPR2025-00103

# 1 OVERVIEW OF ERROR CODES

This document 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.

## 1.1 USAGE DESCRIPTIONS

The purpose of this section is to give descriptions of how the error codes should be used. It is beyond the scope of this document to give detailed descriptions of all situations where error codes can be used, especially as this may be implementation dependent.

## 1.2 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 may find 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 events. Which of the two events is used is implementation-dependent.

## 1.3 LIST OF ERROR CODES

The error code of 0x00 means Success. The possible range of failure error codes is 0x01-0xFF. Section 2 provides an error code usage description for each failure error code.

Values marked as "Reserved for Future Use", can be used in future versions of the specification. A host shall consider any error code that it does not explicitly understand equivalent to the "Unspecified Error (0x1F)."

| Error Code | Name |
|---|---|
| 0x00 | Success |
| 0x01 | Unknown HCI Command |
| 0x02 | Unknown Connection Identifier |
| 0x03 | Hardware Failure |
| 0x04 | Page Timeout |
| 0x05 | Authentication Failure |
| 0x06 | PIN or Key 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 |
| 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 |

*Table 1.1: List of Possible Error Codes*



LifeScan, Inc. Ex. 1019
Page 506 of 1420
IPR2025-00103

| Error Code | Name |
|---|---|
| 0x19 | Unknown LMP PDU |
| 0x1A | Unsupported Remote Feature / Unsupported LMP 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 Can Not 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 |
| 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 |
| 0x36 | Extended Inquiry Response Too Large |
| 0x37 | Secure Simple Pairing Not Supported By Host. |
| 0x38 | Host Busy - Pairing |

*Table 1.1: List of Possible Error Codes*



LifeScan, Inc. Ex. 1019
Page 507 of 1420
IPR2025-00103

# 2 ERROR CODE DESCRIPTIONS

## 2.1 UNKNOWN HCI COMMAND (0X01)

The Unknown HCI Command error code 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 document, or any vendor-specific OpCodes, or the command may not have been implemented.

## 2.2 UNKNOWN CONNECTION IDENTIFIER (0X02)

The Unknown Connection Identifier error code indicates that a command was sent from the host that should identify a connection, but that connection does not exist.

## 2.3 HARDWARE FAILURE (0X03)

The Hardware Failure error code 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.

## 2.4 PAGE TIMEOUT (0X04)

The Page Timeout error code 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.

## 2.5 AUTHENTICATION FAILURE (0X05)

The Authentication Failure error code 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.

## 2.6 PIN OR KEY MISSING (0X06)

The PIN or Key Missing error code is used when pairing failed because of a missing PIN, or authentication failed because of a missing Key.

## 2.7 MEMORY CAPACITY EXCEEDED (0X07)

The Memory Capacity Exceeded error code indicates to the host that the controller has run out of memory to store new parameters.



LifeScan, Inc. Ex. 1019
Page 508 of 1420
IPR2025-00103

## 2.8 CONNECTION TIMEOUT (0X08)

The Connection Timeout error code indicates that the link supervision timeout has expired for a given connection.

## 2.9 CONNECTION LIMIT EXCEEDED (0X09)

The Connection Limit Exceeded error code 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.

## 2.10 SYNCHRONOUS CONNECTION LIMIT TO A DEVICE EXCEEDED (0X0A)

The Synchronous Connection Limit to a Device Exceeded error code 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.

## 2.11 ACL CONNECTION ALREADY EXISTS (0X0B)

The ACL Connection Already Exists error code indicates that an attempt to create a new ACL Connection to a device when there is already a connection to this device.

## 2.12 COMMAND DISALLOWED (0X0C)

The Command Disallowed error code 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.

## 2.13 CONNECTION REJECTED DUE TO LIMITED RESOURCES (0X0D)

The Connection Rejected Due To Limited Resources error code indicates that an incoming connection was rejected due to limited resources.

## 2.14 CONNECTION REJECTED DUE TO SECURITY REASONS (0X0E)

The Connection Rejected Due To Security Reasons error code indicates that a connection was rejected due to security requirements not being fulfilled, like authentication or pairing.

## 2.15 CONNECTION REJECTED DUE TO UNACCEPTABLE BD_ADDR (0X0F)

The Connection Rejected due to Unacceptable BD_ADDR error code indicates that a connection was rejected because this device does not accept the BD_ADDR. This may be because the device will only accept connections from specific BD_ADDRs.

## 2.16 CONNECTION ACCEPT TIMEOUT EXCEEDED (0X10)

The Connection Accept Timeout Exceeded error code indicates that the Connection Accept Timeout has been exceeded for this connection attempt.

## 2.17 UNSUPPORTED FEATURE OR PARAMETER VALUE (0X11)

The Unsupported Feature Or Parameter Value error code indicates that a feature or parameter value in the HCI Command is not supported. This error code shall not be used in an LMP PDU.

## 2.18 INVALID HCI COMMAND PARAMETERS (0X12)

The Invalid HCI Command Parameters error code 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.

## 2.19 REMOTE USER TERMINATED CONNECTION (0X13)

The Remote User Terminated Connection error code indicates that the user on the remote device terminated the connection.



LifeScan, Inc. Ex. 1019
Page 510 of 1420
IPR2025-00103

## 2.20 REMOTE DEVICE TERMINATED CONNECTION DUE TO LOW RESOURCES (0X14)

The Remote Device Terminated Connection due to Low Resources error code indicates that the remote device terminated the connection because of low resources.

## 2.21 REMOTE DEVICE TERMINATED CONNECTION DUE TO POWER OFF (0X15)

The Remote Device Terminated Connection due to Power Off error code indicates that the remote device terminated the connection because the device is about to power off.

## 2.22 CONNECTION TERMINATED BY LOCAL HOST (0X16)

The Connection Terminated By Local Host error code indicates that the local device terminated the connection.

## 2.23 REPEATED ATTEMPTS (0X17)

The Repeated Attempts error code 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.

## 2.24 PAIRING NOT ALLOWED (0X18)

The Pairing Not Allowed error code indicates that the device does not allow pairing. For example, when a device only allows pairing during a certain time window after some user input allows pairing.

## 2.25 UNKNOWN LMP PDU (0X19)

The Unknown LMP PDU error code indicates that the controller has received an unknown LMP opcode.

## 2.26 Unsupported Remote Feature / Unsupported LMP Feature (0X1A)

The Unsupported Remote Feature error code indicates that the remote device does not support the feature associated with the issued command or LMP PDU.

## 2.27 SCO OFFSET REJECTED (0X1B)

The SCO Offset Rejected error code indicates that the offset requested in the LMP_SCO_link_req message has been rejected.

## 2.28 SCO INTERVAL REJECTED (0X1C)

The SCO Interval Rejected error code indicates that the interval requested in the LMP_SCO_link_req message has been rejected.

## 2.29 SCO AIR MODE REJECTED (0X1D)

The SCO Air Mode Rejected error code indicates that the air mode requested in the LMP_SCO_link_req message has been rejected.

## 2.30 INVALID LMP PARAMETERS (0X1E)

The Invalid LMP Parameters error code 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.

## 2.31 UNSPECIFIED ERROR (0X1F)

The Unspecified Error error code indicates that no other error code specified is appropriate to use.

## 2.32 UNSUPPORTED LMP PARAMETER VALUE (0X20)

The Unsupported LMP Parameter Value error code 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, for example during an LMP_hold_req, LMP_sniff_req and LMP_encryption_key_size_req message exchanges.

## 2.33 ROLE CHANGE NOT ALLOWED (0X21)

The Role Change Not Allowed error code indicates that a controller will not allow a role change at this time.

## 2.34 LMP RESPONSE TIMEOUT (0X22)

The LMP Response Timeout error code indicates that an LMP transaction failed to respond within the LMP response timeout.



LifeScan, Inc. Ex. 1019
Page 512 of 1420
IPR2025-00103

## 2.35 LMP ERROR TRANSACTION COLLISION (0X23)

The LMP Error Transaction Collision error code indicates that an LMP transaction has collided with the same transaction that is already in progress.

## 2.36 LMP PDU NOT ALLOWED (0X24)

The LMP PDU Not Allowed error code indicates that a controller sent an LMP message with an opcode that was not allowed.

## 2.37 ENCRYPTION MODE NOT ACCEPTABLE (0X25)

The Encryption Mode Not Acceptable error code indicates that the requested encryption mode is not acceptable at this time.

## 2.38 LINK KEY CAN NOT BE CHANGED (0X26)

The Link Key Can Not be Changed error code indicates that a link key can not be changed because a fixed unit key is being used.

## 2.39 REQUESTED QoS NOT SUPPORTED (0X27)

The Requested QoS Not Supported error code indicates that the requested Quality of Service is not supported.

## 2.40 INSTANT PASSED (0X28)

The Instant Passed error code indicates that an LMP PDU that includes an instant can not be performed because the instant when this would have occurred has passed.

## 2.41 PAIRING WITH UNIT KEY NOT SUPPORTED (0X29)

The Pairing With Unit Key Not Supported error code indicates that it was not possible to pair as a unit key was requested and it is not supported.

## 2.42 DIFFERENT TRANSACTION COLLISION (0X2A)

The Different Transaction Collision error code indicates that an LMP transaction was started that collides with an ongoing transaction.

## 2.43 QoS UNACCEPTABLE PARAMETER (0X2C)

The QoS Unacceptable Parameter error code indicates that the specified quality of service parameters could not be accepted at this time, but other parameters may be acceptable.

## 2.44 QoS REJECTED (0X2D)

The QoS Rejected error code indicates that the specified quality of service parameters can not be accepted and QoS negotiation should be terminated.

## 2.45 CHANNEL CLASSIFICATION NOT SUPPORTED (0X2E)

The Channel Classification Not Supported error code indicates that the controller can not perform channel classification because it is not supported.

## 2.46 INSUFFICIENT SECURITY (0X2F)

The Insufficient Security error code indicates that the HCI command or LMP message sent is only possible on an encrypted link.

## 2.47 PARAMETER OUT OF MANDATORY RANGE (0X30)

The Parameter Out Of Mandatory Range error code indicates that a parameter value requested is outside the mandatory range of parameters for the given HCI command or LMP message.

## 2.48 ROLE SWITCH PENDING (0X32)

The Role Switch Pending error code indicates that a Role Switch is pending. This can be used when an HCI command or LMP message can not be accepted because of a pending role switch. This can also be used to notify a peer device about a pending role switch.

## 2.49 RESERVED SLOT VIOLATION (0X34)

The Reserved Slot Violation error code indicates that the current Synchronous negotiation was terminated with the negotiation state set to Reserved Slot Violation.

## 2.50 ROLE SWITCH FAILED (0X35)

The Role Switch Failed error code 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.

## 2.51 EXTENDED INQUIRY RESPONSE TOO LARGE (0X36)

The Extended Inquiry Response Too Large error code indicates that the extended inquiry response, with the requested requirements for FEC, is too large to fit in any of the packet types supported by the controller.



LifeScan, Inc. Ex. 1019
Page 514 of 1420
IPR2025-00103

## 2.52 SIMPLE PAIRING NOT SUPPORTED BY HOST (0X37)

The Secure Simple Pairing Not Supported by Host error code indicates that the IO capabilities request or response was rejected because the sending Host does not support Secure Simple Pairing even though the receiving Link Manager does.

## 2.53 HOST BUSY–PAIRING(0X38)

The Host Busy - Pairing error code indicates that the Host is busy with another pairing operation and unable to support the requested pairing. The receiving device should retry pairing again later.



LifeScan, Inc. Ex. 1019
Page 515 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 516 of 1420
IPR2025-00103

# HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

***This document describes the functional specification for the Host Controller Interface (HCI).***
***The HCI provides a command interface to the baseband controller and link manager, and access to configuration parameters. This interface provides a uniform method of accessing the Bluetooth baseband capabilities.***



LifeScan, Inc. Ex. 1019
Page 517 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 518 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 357**
1.1 Lower Layers of the Bluetooth Software Stack ..... 357
**2 Overview of Host Controller Transport Layer ..... 359**
**3 Overview of Commands and Events ..... 360**
3.1 Generic Events ..... 361
3.2 Device Setup ..... 361
3.3 Controller Flow Control ..... 362
3.4 Controller Information ..... 362
3.5 Controller Configuration ..... 363
3.6 Device Discovery ..... 364
3.7 Connection Setup ..... 366
3.8 Remote Information ..... 368
3.9 Synchronous Connections ..... 369
3.10 Connection State ..... 370
3.11 Piconet Structure ..... 371
3.12 Quality of Service ..... 372
3.13 Physical Links ..... 374
3.14 Host Flow Control ..... 375
3.15 Link Information ..... 376
3.16 Authentication and Encryption ..... 377
3.17 Testing ..... 380
3.18 Alphabetical List of Commands and Events ..... 381
**4 HCI Flow Control ..... 386**
4.1 Host to Controller Data Flow Control ..... 386
4.2 Controller to Host Data Flow Control ..... 387
4.3 Disconnection Behavior ..... 388
4.4 Command Flow Control ..... 388
4.5 Command Error Handling ..... 389
**5 HCI Data Formats ..... 390**
5.1 Introduction ..... 390
5.2 Data and Parameter Formats ..... 390
5.3 Connection Handles ..... 391
5.4 Exchange of HCI-Specific Information ..... 392
5.4.1 HCI Command Packet ..... 392
5.4.2 HCI ACL Data Packets ..... 394
5.4.3 HCI Synchronous Data Packets ..... 396
5.4.4 HCI Event Packet ..... 398
**6 HCI Configuration Parameters ..... 399**



LifeScan, Inc. Ex. 1019
Page 519 of 1420
IPR2025-00103

6.1 Scan Enable ..... 399
6.2 Inquiry Scan Interval ..... 399
6.3 Inquiry Scan Window ..... 400
6.4 Inquiry Scan Type ..... 400
6.5 Inquiry Mode ..... 400
6.6 Page Timeout ..... 401
6.7 Connection Accept Timeout ..... 401
6.8 Page Scan Interval ..... 402
6.9 Page Scan Window ..... 402
6.10 Page Scan Period Mode (Deprecated) ..... 402
6.11 Page Scan Type ..... 403
6.12 Voice Setting ..... 403
6.13 PIN Type ..... 404
6.14 Link Key ..... 404
6.15 Failed Contact Counter ..... 405
6.16 Hold Mode Activity ..... 405
6.17 Link Policy Settings ..... 406
6.18 Flush Timeout ..... 407
6.19 Num Broadcast Retransmissions ..... 407
6.20 Link Supervision Timeout ..... 408
6.21 Synchronous Flow Control Enable ..... 408
6.22 Local Name ..... 409
6.23 Extended Inquiry Response ..... 409
6.24 Erroneous Data Reporting ..... 409
6.25 Class Of Device ..... 410
6.26 Supported Commands ..... 410

**7 HCI Commands and Events ..... 416**

7.1 Link Control Commands ..... 416
7.1.1 Inquiry Command ..... 416
7.1.2 Inquiry Cancel Command ..... 418
7.1.3 Periodic Inquiry Mode Command ..... 419
7.1.4 Exit Periodic Inquiry Mode Command ..... 422
7.1.5 Create Connection Command ..... 423
7.1.6 Disconnect Command ..... 426
7.1.7 Create Connection Cancel Command ..... 427
7.1.8 Accept Connection Request Command ..... 429
7.1.9 Reject Connection Request Command ..... 431
7.1.10 Link Key Request Reply Command ..... 432
7.1.11 Link Key Request Negative Reply Command ..... 434
7.1.12 PIN Code Request Reply Command ..... 435
7.1.13 PIN Code Request Negative Reply Command ..... 437



LifeScan, Inc. Ex. 1019
Page 520 of 1420
IPR2025-00103

7.1.14 Change Connection Packet Type Command .....438
7.1.15 Authentication Requested Command.....441
7.1.16 Set Connection Encryption Command .....443
7.1.17 Change Connection Link Key Command .....444
7.1.18 Master Link Key Command .....445
7.1.19 Remote Name Request Command .....446
7.1.20 Remote Name Request Cancel Command .....448
7.1.21 Read Remote Supported Features Command.....449
7.1.22 Read Remote Extended Features Command .....450
7.1.23 Read Remote Version Information Command.....451
7.1.24 Read Clock Offset Command.....452
7.1.25 Read LMP Handle Command .....453
7.1.26 Setup Synchronous Connection Command .....455
7.1.27 Accept Synchronous Connection Request Command 459
7.1.28 Reject Synchronous Connection Request Command .463
7.1.29 IO Capability Request Reply Command.....464
7.1.30 User Confirmation Request Reply Command .....467
7.1.31 User Confirmation Request Negative Reply Command .....468
7.1.32 User Passkey Request Reply Command .....469
7.1.33 User Passkey Request Negative Reply Command.....470
7.1.34 Remote OOB Data Request Reply Command .....471
7.1.35 Remote OOB Data Request Negative Reply Command .....473
7.1.36 IO Capability Request Negative Reply Command .....474
7.2 Link Policy Commands.....475
7.2.1 Hold Mode Command .....475
7.2.2 Sniff Mode Command.....477
7.2.3 Exit Sniff Mode Command.....480
7.2.4 Park State Command .....481
7.2.5 Exit Park State Command .....483
7.2.6 QoS Setup Command .....484
7.2.7 Role Discovery Command.....487
7.2.8 Switch Role Command.....488
7.2.9 Read Link Policy Settings Command .....490
7.2.10 Write Link Policy Settings Command .....491
7.2.11 Read Default Link Policy Settings Command.....493
7.2.12 Write Default Link Policy Settings Command.....494



LifeScan, Inc. Ex. 1019
Page 521 of 1420
IPR2025-00103

7.2.13 Flow Specification Command ..... 495
7.2.14 Sniff Subrating Command ..... 497
7.3 Controller & Baseband Commands ..... 498
7.3.1 Set Event Mask Command ..... 499
7.3.2 Reset Command ..... 502
7.3.3 Set Event Filter Command ..... 503
7.3.4 Flush Command ..... 508
7.3.5 Read PIN Type Command ..... 510
7.3.6 Write PIN Type Command ..... 511
7.3.7 Create New Unit Key Command ..... 512
7.3.8 Read Stored Link Key Command ..... 513
7.3.9 Write Stored Link Key Command ..... 514
7.3.10 Delete Stored Link Key Command ..... 516
7.3.11 Write Local Name Command ..... 517
7.3.12 Read Local Name Command ..... 518
7.3.13 Read Connection Accept Timeout Command ..... 519
7.3.14 Write Connection Accept Timeout Command ..... 520
7.3.15 Read Page Timeout Command ..... 521
7.3.16 Write Page Timeout Command ..... 522
7.3.17 Read Scan Enable Command ..... 523
7.3.18 Write Scan Enable Command ..... 524
7.3.19 Read Page Scan Activity Command ..... 525
7.3.20 Write Page Scan Activity Command ..... 527
7.3.21 Read Inquiry Scan Activity Command ..... 528
7.3.22 Write Inquiry Scan Activity Command ..... 529
7.3.23 Read Authentication Enable Command ..... 529
7.3.24 Write Authentication Enable Command ..... 530
7.3.25 Read Class of Device Command ..... 532
7.3.26 Write Class of Device Command ..... 532
7.3.27 Read Voice Setting Command ..... 534
7.3.28 Write Voice Setting Command ..... 535
7.3.29 Read Automatic Flush Timeout Command ..... 536
7.3.30 Write Automatic Flush Timeout Command ..... 537
7.3.31 Read Num Broadcast Retransmissions Command ..... 538
7.3.32 Write Num Broadcast Retransmissions Command ..... 539
7.3.33 Read Hold Mode Activity Command ..... 540
7.3.34 Write Hold Mode Activity Command ..... 541
7.3.35 Read Transmit Power Level Command ..... 542



LifeScan, Inc. Ex. 1019
Page 522 of 1420
IPR2025-00103

7.3.36 Read Synchronous Flow Control Enable Command...544
7.3.37 Write Synchronous Flow Control Enable Command...545
7.3.38 Set Controller To Host Flow Control Command...546
7.3.39 Host Buffer Size Command...547
7.3.40 Host Number Of Completed Packets Command...548
7.3.41 Read Link Supervision Timeout Command...551
7.3.42 Write Link Supervision Timeout Command...553
7.3.43 Read Number Of Supported IAC Command...555
7.3.44 Read Current IAC LAP Command...556
7.3.45 Write Current IAC LAP Command...557
7.3.46 Set AFH Host Channel Classification Command...559
7.3.47 Read Inquiry Scan Type Command...560
7.3.48 Write Inquiry Scan Type Command...561
7.3.49 Read Inquiry Mode Command...562
7.3.50 Write Inquiry Mode Command...563
7.3.51 Read Page Scan Type Command...564
7.3.52 Write Page Scan Type Command...565
7.3.53 Read AFH Channel Assessment Mode Command...566
7.3.54 Write AFH Channel Assessment Mode Command...567
7.3.55 Read Extended Inquiry Response Command...568
7.3.56 Write Extended Inquiry Response Command...569
7.3.57 Refresh Encryption Key...570
7.3.58 Read Simple Pairing Mode Command...571
7.3.59 Write Simple Pairing Mode Command...572
7.3.60 Read Local OOB Data Command...573
7.3.61 Read Inquiry Response Transmit Power Level Command 574
7.3.62 Write Inquiry Transmit Power Level Command...575
7.3.63 Send Keypress Notification Command...576
7.3.64 Read Default Erroneous Data Reporting...577
7.3.65 Write Default Erroneous Data Reporting...578
7.3.66 Enhanced Flush Command...579
7.4 Informational Parameters...580
7.4.1 Read Local Version Information Command...580
7.4.2 Read Local Supported Commands Command...583
7.4.3 Read Local Supported Features Command...584
7.4.4 Read Local Extended Features Command...585
7.4.5 Read Buffer Size Command...587



LifeScan, Inc. Ex. 1019
Page 523 of 1420
IPR2025-00103

7.4.6 Read BD_ADDR Command ..... 589
7.5 Status Parameters ..... 590
7.5.1 Read Failed Contact Counter Command ..... 590
7.5.2 Reset Failed Contact Counter Command ..... 592
7.5.3 Read Link Quality Command ..... 593
7.5.4 Read RSSI Command ..... 594
7.5.5 Read AFH Channel Map Command ..... 596
7.5.6 Read Clock Command ..... 598
7.6 Testing Commands ..... 600
7.6.1 Read Loopback Mode Command ..... 600
7.6.2 Write Loopback Mode Command ..... 601
7.6.3 Enable Device Under Test Mode Command ..... 604
7.6.4 Write Simple Pairing Debug Mode Command ..... 604
7.7 Events ..... 606
7.7.1 Inquiry Complete Event ..... 606
7.7.2 Inquiry Result Event ..... 607
7.7.3 Connection Complete Event ..... 609
7.7.4 Connection Request Event ..... 610
7.7.5 Disconnection Complete Event ..... 612
7.7.6 Authentication Complete Event ..... 613
7.7.7 Remote Name Request Complete Event ..... 614
7.7.8 Encryption Change Event ..... 615
7.7.9 Change Connection Link Key Complete Event ..... 616
7.7.10 Master Link Key Complete Event ..... 617
7.7.11 Read Remote Supported Features Complete Event ..... 618
7.7.12 Read Remote Version Information Complete Event ..... 619
7.7.13 QoS Setup Complete Event ..... 620
7.7.14 Command Complete Event ..... 622
7.7.15 Command Status Event ..... 623
7.7.16 Hardware Error Event ..... 624
7.7.17 Flush Occurred Event ..... 624
7.7.18 Role Change Event ..... 625
7.7.19 Number Of Completed Packets Event ..... 626
7.7.20 Mode Change Event ..... 627
7.7.21 Return Link Keys Event ..... 629
7.7.22 PIN Code Request Event ..... 630
7.7.23 Link Key Request Event ..... 631
7.7.24 Link Key Notification Event ..... 632



LifeScan, Inc. Ex. 1019
Page 524 of 1420
IPR2025-00103

7.7.25 Loopback Command Event ....634
7.7.26 Data Buffer Overflow Event ....634
7.7.27 Max Slots Change Event ....635
7.7.28 Read Clock Offset Complete Event ....636
7.7.29 Connection Packet Type Changed Event ....637
7.7.30 QoS Violation Event ....639
7.7.31 Page Scan Repetition Mode Change Event ....640
7.7.32 Flow Specification Complete Event ....641
7.7.33 Inquiry Result with RSSI Event ....643
7.7.34 Read Remote Extended Features Complete Event ....645
7.7.35 Synchronous Connection Complete Event ....646
7.7.36 Synchronous Connection Changed event ....649
7.7.37 Sniff Subrating Event ....650
7.7.38 Extended Inquiry Result Event ....652
7.7.39 Encryption Key Refresh Complete ....654
7.7.40 IO Capability Request Event ....655
7.7.41 IO Capability Response Event ....656
7.7.42 User Confirmation Request Event ....657
7.7.43 User Passkey Request Event ....658
7.7.44 Remote OOB Data Request Event ....658
7.7.45 Simple Pairing Complete Event ....659
7.7.46 Link Supervision Timeout Changed Event ....660
7.7.47 Enhanced Flush Complete Event ....661
7.7.48 User Passkey Notification Event ....661
7.7.49 Keypress Notification Event ....662
7.7.50 Remote Host Supported Features Notification Event ..662

**8 List of Figures ....664**

**9 List of Tables ....665**

**10 Appendix A: Deprecated Commands, Events and Configuration Parameters ....666**

10.1 Read Page Scan Mode Command ....667
10.2 Write Page Scan Mode Command ....668
10.3 Read Page Scan Period Mode Command ....669
10.4 Write Page Scan Period Mode Command ....670
10.5 Add SCO Connection Command ....671
10.6 Page Scan Mode Change Event ....673
10.7 Read Country Code Command ....674
10.8 Read Encryption Mode Command ....675
10.9 Write Encryption Mode Command ....675

10.10 Deprecated Parameters....676
10.10.1 Authentication Enable....676
10.10.2 Encryption Mode....676
10.10.3 Page Scan Mode....677



LifeScan, Inc. Ex. 1019
Page 526 of 1420
IPR2025-00103

# 1 INTRODUCTION

This document describes the functional specifications for the Host Controller Interface (HCI). The HCI provides a uniform interface method of accessing the Bluetooth controller capabilities. The next two sections provide a brief overview of the lower layers of the Bluetooth software stack and of the Bluetooth hardware. Section 2, provides an overview of the Lower HCI Device Driver Interface on the host device. Section 4, describes the flow control used between the Host and the Controller. Section 7, describes each of the HCI Commands in details, identifies parameters for each of the commands, and lists events associated with each command.

## 1.1 LOWER LAYERS OF THE BLUETOOTH SOFTWARE STACK

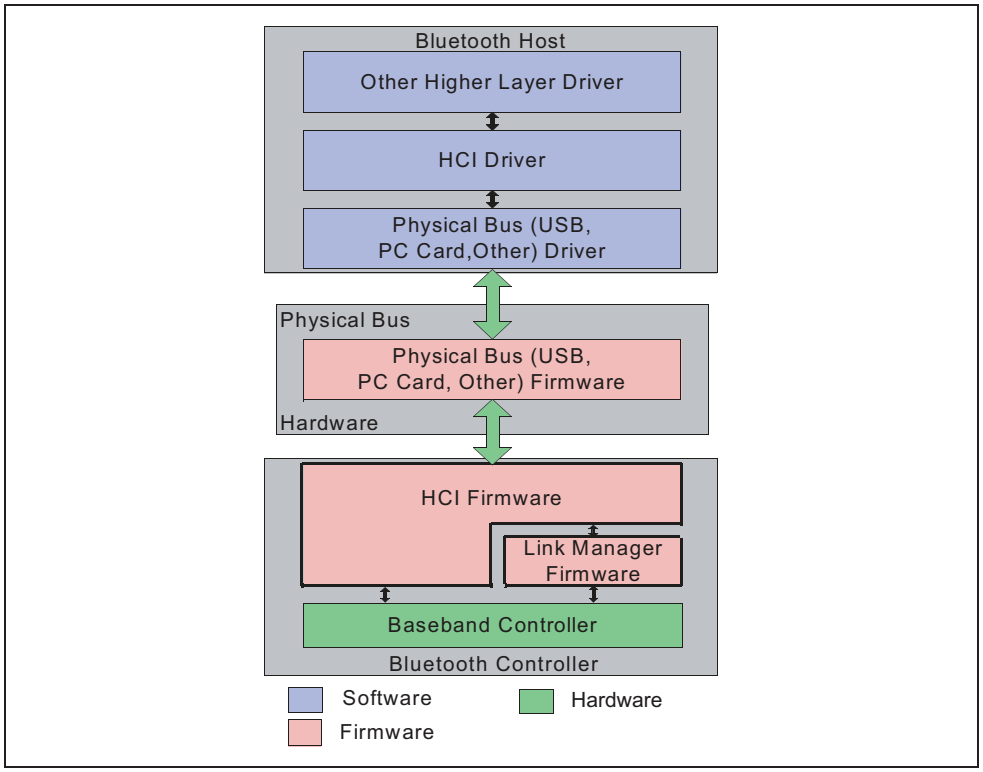



*Figure 1.1: Overview of the Lower Software Layers*

Figure 1.1 provides an overview of the lower software layers. The HCI firmware implements the HCI Commands for the Bluetooth hardware by accessing baseband commands, link manager commands, hardware status registers, control registers, and event registers.

Several layers may exist between the HCI driver on the host system and the HCI firmware in the Bluetooth hardware. These intermediate layers, the Host



LifeScan, Inc. Ex. 1019
Page 527 of 1420
IPR2025-00103

Controller Transport Layer, provide the ability to transfer data without intimate knowledge of the data.

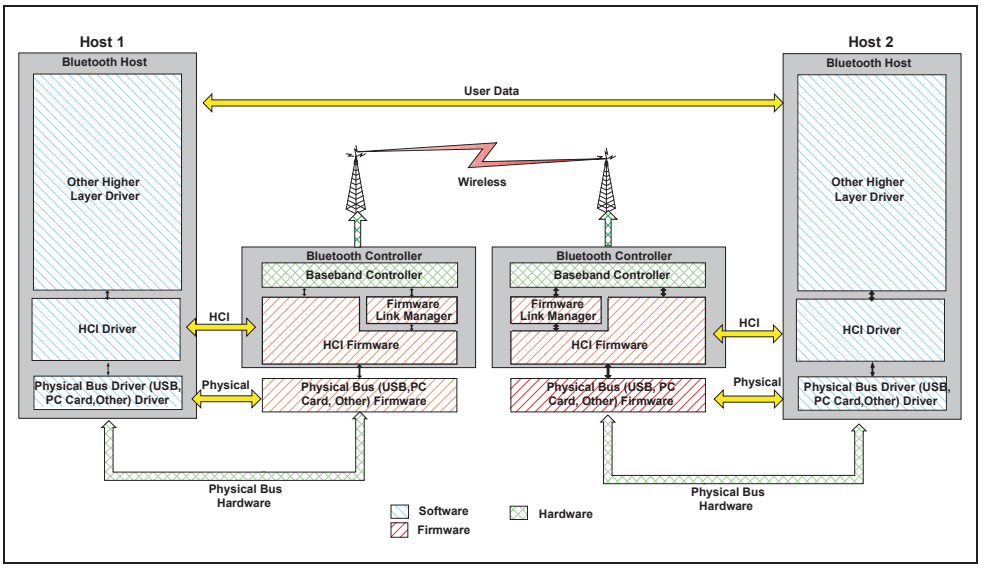



*Figure 1.2: End to End Overview of Lower Software Layers to Transfer Data*

Figure 1.2 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 Bluetooth hardware. The Host Control Transport Layer (i.e. physical bus) driver provides both HCI layers with the ability to exchange information with each other.

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.



LifeScan, Inc. Ex. 1019
Page 528 of 1420
IPR2025-00103

# 2 OVERVIEW OF HOST CONTROLLER TRANSPORT LAYER

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 specification volume 4)



LifeScan, Inc. Ex. 1019
Page 529 of 1420
IPR2025-00103

# 3 OVERVIEW OF COMMANDS AND EVENTS

The commands and events are sent between the Host and the Controller. These are grouped into logical groups by function.

| | |
|---|---|
| **Generic Events** | The generic events can occur due to multiple commands, or events that can occur at any time. |
| **Device Setup** | The device setup commands are used to place the Controller into a known state. |
| **Controller Flow Control** | The controller flow control commands and events are used to control data flow from the Host to the controller. |
| **Controller Information** | The controller information commands allow the Host to discover local information about the device. |
| **Controller Configuration** | The controller configuration commands and events allow the global configuration parameters to be configured. |
| **Device Discovery** | The device discovery commands and events allow a device to discover other devices in the surrounding area. |
| **Connection Setup** | The connection setup commands and events allow a device to make a connection to another device. |
| **Remote Information** | The remote information commands and events allow information about a remote device's configuration to be discovered. |
| **Synchronous Connections** | The synchronous connection commands and events allow synchronous connections to be created |
| **Connection State** | The connection state commands and events allow the configuration of a link, especially for low power operation. |
| **Piconet Structure** | The piconet structure commands and events allow the discovery and reconfiguration of piconet. |
| **Quality of Service** | The quality of service commands and events allow quality of service parameters to be specified. |
| **Physical Links** | The physical link commands and events allow the configuration of a physical link. |
| **Host Flow Control** | The Host flow control commands and events allow flow control to be used towards the Host. |
| **Link Information** | The link information commands and events allow information about a link to be read. |
| **Authentication and Encryption** | The authentication and encryption commands and events allow authentication of a remote device and then encryption of the link. |
| **Testing** | The testing commands and events allow a device to be placed into test mode. |

*Table 3.1: Overview of commands and events*

The version information in this section denotes the version number of the specification that this command or event was first specified



LifeScan, Inc. Ex. 1019
Page 530 of 1420
IPR2025-00103

## 3.1 GENERIC EVENTS

The generic events occur due to multiple commands, or events that can occur at any time.

| Name | Vers. | Summary description |
|---|---|---|
| Command Complete Event | 1.1 | The Command Complete event is 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 | The Command Status event is used to indicate 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 | The Hardware Error event is used to indicate some type of hardware failure for the Controller. |

*Table 3.2: Generic events*

## 3.2 DEVICE SETUP

The device setup group of commands are used to place the Controller into a known state.

| Name | Vers. | Summary description |
|---|---|---|
| Reset Command | 1.1 | The Reset command will reset the Controller, Link Manager, and the Bluetooth radio. |

*Device setup*



LifeScan, Inc. Ex. 1019
Page 531 of 1420
IPR2025-00103

## 3.3 CONTROLLER FLOW CONTROL

The controller flow control group of commands and events are used to control data flow from the Host to the Controller.

| Name | Vers. | Summary description |
|---|---|---|
| Read Buffer Size Command | 1.1 | The Read_Buffer_Size command returns the size of the HCI buffers. These buffers are used by the Controller to buffer data that is to be transmitted. |
| Number Of Completed Packets Event | 1.1 | The Number Of Completed Packets event is 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. |

*Table 3.3: Controller flow control*

## 3.4 CONTROLLER INFORMATION

The controller information group of commands allows the Host to discover local information about the device.

| Name | Vers. | Summary description |
|---|---|---|
| Read Local Version Information Command | 1.1 | The Read Local Version Information command will read the version information for the local Bluetooth device. |
| Read Local Supported Commands Command | 1.2 | The Read Local Supported Commands command requests a list of the supported HCI commands for the local device. |
| Read Local Supported Features Command | 1.1 | The Read Local Supported Features command requests a list of the supported features for the local device. |
| Read Local Extended Features Command | 1.2 | The Read Local Extended Features command requests a list of the supported extended features for the local device |
| Read BD_ADDR Command | 1.1 | The Read BD_ADDR command will read the value for the BD_ADDR parameter. |

*Table 3.4: Controller information*



LifeScan, Inc. Ex. 1019
Page 532 of 1420
IPR2025-00103

## 3.5 CONTROLLER CONFIGURATION

The controller configuration group of commands and events allows the global configuration parameters to be configured.

| Name | Vers. | Summary description |
|---|---|---|
| Read Local Name Command | 1.1 | The Read Local Name command provides the ability to read the stored user-friendly name for the Bluetooth device. |
| Write Local Name Command | 1.1 | The Write Local Name command provides the ability to modify the user-friendly name for the Bluetooth device. |
| Read Class of Device Command | 1.1 | The Read Class of Device command will read 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 | The Write Class of Device command will write 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 | The Read Number of Supported IAC command will read the value for the number of Inquiry Access Codes (IAC) that the local Bluetooth device can simultaneously listen for during an Inquiry Scan. |
| Read Current IAC LAP Command | 1.1 | The Read Current IAC LAP command will read the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. |
| Write Current IAC LAP Command | 1.1 | The Write Current IAC LAP command will write the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. |
| Read Scan Enable Command | 1.1 | The Read Scan Enable command will read the value for the Scan Enable configuration parameter, which controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices. |
| Write Scan Enable Command | 1.1 | The Write Scan Enable command will write the value for the Scan Enable configuration parameter, which controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices. |

*Table 3.5: Controller configuration*

| Name | Vers. | Summary description |
|---|---|---|
| Write Extended Inquiry Response Command | 2.1 + EDR | The Write Extended Inquiry Response command will write the data that the Bluetooth device sends in the extended inquiry response packet during inquiry response. |
| Read Extended Inquiry Response Command | 2.1 + EDR | The Read Extended Inquiry Response command will read the data that the Bluetooth device sends in the extended inquiry response packet during inquiry response. |
| Write Default Erroneous Data Reporting | 2.1 + EDR | The Write Default Erroneous Data Reporting command will write the value for the Erroneous Data Reporting configuration parameter, which controls whether the Bluetooth controller will provide data for every (e)SCO interval, with the Packet_Status_Flag in HCI Synchronous Data Packets set according to Section 5.4.3 on page 396. |
| Read Default Erroneous Data Reporting | 2.1 + EDR | The Read Default Erroneous Data Reporting command will read the value for the Erroneous Data Reporting configuration parameter, which controls whether the Bluetooth device will provide data for every (e)SCO interval, with the Packet_Status_Flag in HCI Synchronous Data Packets set according to Section 5.4.3 on page 396. |

*Table 3.5: Controller configuration*

## 3.6 DEVICE DISCOVERY

The device discovery group of commands and events allow a device to discovery other devices in the surrounding area.

| Name | Vers. | Summary description |
|---|---|---|
| Inquiry Command | 1.1 | The Inquiry command will cause the Bluetooth device to enter Inquiry Mode. Inquiry Mode is used to discovery other nearby Bluetooth devices. |
| Inquiry Result Event | 1.1 | The Inquiry Result event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process. |
| Inquiry Result with RSSI Event | 1.2 | The Inquiry Result with RSSI event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process. |
| Extended Inquiry Result Event | 2.1 + EDR | The Extended Inquiry Result event indicates that a Bluetooth device has responded with an extended inquiry response during the current Inquiry process. |

*Table 3.6: Device discovery*

| Name | Vers. | Summary description |
|---|---|---|
| Inquiry Cancel Command | 1.1 | The Inquiry Cancel command will cause the Bluetooth device to stop the current Inquiry if the Bluetooth device is in Inquiry Mode. |
| Inquiry Complete Event | 1.1 | The Inquiry Complete event indicates that the Inquiry is finished. |
| Periodic Inquiry Mode Command | 1.1 | The Periodic Inquiry Mode command is used to configure the Bluetooth device to perform an automatic Inquiry based on a specified period range. |
| Exit Periodic Inquiry Mode Command | 1.1 | The Exit Periodic Inquiry Mode command is used to end the Periodic Inquiry mode when the local device is in Periodic Inquiry Mode. |
| Read Inquiry Scan Activity Command | 1.1 | The Read Inquiry Scan Activity command will read 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 | The Write Inquiry Scan Activity command will write 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 | The Read Inquiry Scan Type command is used to read the Inquiry Scan Type configuration parameter of the local Bluetooth 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 | The Write Inquiry Scan Type command is used to write the Inquiry Scan Type configuration parameter of the local Bluetooth device. The Inquiry Scan Type configuration parameter can set the inquiry scan to either normal or interlaced scan. |
| Read Inquiry Mode Command | 1.2 | The Read Inquiry Mode command is used to read the Inquiry Mode configuration parameter of the local Bluetooth device. |
| Write Inquiry Mode Command | 1.2 | The Write Inquiry Mode command is used to write the Inquiry Mode configuration parameter of the local Bluetooth device. |
| Read Inquiry Response Transmit Power Level | 2.1 + EDR | This command will read the inquiry response Transmit Power level used to transmit the FHS and EIR data packets. This can be used directly in the Tx Power Level EIR data type. |
| Write Inquiry Transmit Power Level | 2.1 + EDR | This command is used to write the transmit power level used to transmit the inquiry (ID) data packets. |

*Table 3.6: Device discovery*

## 3.7 CONNECTION SETUP

The connection setup group of commands and events are used to allow a device to make a connection to another device.

| Name | Vers. | Summary description |
|---|---|---|
| Create Connection Command | 1.1 | The Create Connection command will cause the link manager to create an ACL connection to the Bluetooth device with the BD_ADDR specified by the command parameters. |
| Connection Request Event | 1.1 | The Connection Request event is used to indicate that a new incoming connection is trying to be established. |
| Accept Connection Request Command | 1.1 | The Accept Connection Request command is used to accept a new incoming connection request. |
| Reject Connection Request Command | 1.1 | The Reject Connection Request command is used to decline a new incoming connection request. |
| Create Connection Cancel Command | 1.2 | The Create Connection Cancel Command is used to cancel an ongoing Create Connection. |
| Connection Complete Event | 1.1 | The Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been established. |
| Disconnect Command | 1.1 | The Disconnect command is used to terminate an existing connection. |
| Disconnection Complete Event | 1.1 | The Disconnection Complete event occurs when a connection has been terminated. |
| Read Page Timeout Command | 1.1 | The Read Page Timeout command will read the value for the Page Reply Timeout configuration parameter, which determines the time the Bluetooth 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 | The Write Page Timeout command will write the value for the Page Reply Timeout configuration parameter, which allows the Bluetooth 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 | The Read Page Scan Activity command will read 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. |

*Table 3.7: Connection setup*



LifeScan, Inc. Ex. 1019
Page 536 of 1420
IPR2025-00103

| Name | Vers. | Summary description |
|---|---|---|
| Write Page Scan Activity Command | 1.1 | The Write Page Scan Activity command will write the value 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 | The Page Scan Repetition Mode Change event indicates that the connected remote Bluetooth device with the specified Connection_Handle has successfully changed the Page_Scan_Repetition_Mode (SR)." |
| Read Page Scan Type Command | 1.2 | The Read Page Scan Type command is used to read the page scan type of the local Bluetooth device. The Page Scan Type configuration parameter can set the page scan to either normal or interlaced scan. |
| Write Page Scan Type Command | 1.2 | The Write Page Scan Type command is used to write the page scan type of the local Bluetooth 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 | The Read Connection Accept Timeout command will read the value for the Connection Accept Timeout configuration parameter, which allows the Bluetooth hardware to automatically deny a connection request after a specified period has occurred, and to refuse a new connection. |
| Write Connection Accept Timeout Command | 1.1 | The Write Connection Accept Timeout command will write the value for the Connection Accept Timeout configuration parameter, which allows the Bluetooth hardware to automatically deny a connection request after a specified period has occurred, and to refuse a new connection. |

*Table 3.7: Connection setup*

## 3.8 REMOTE INFORMATION

The remote information group of commands and events allows information about a remote devices configuration to be discovered.

| Name | Vers. | Summary description |
|---|---|---|
| Remote Name Request Command | 1.1 | The Remote Name Request command is used to obtain the user-friendly name of another Bluetooth device. |
| Remote Name Request Cancel Command | 1.2 | The Remote Name Request Cancel Command is used to cancel an ongoing Remote Name Request. |
| Remote Name Request Complete Event | 1.1 | The Remote Name Request Complete event is used to indicate a remote name request has been completed. |
| Read Remote Supported Features Command | 1.1 | The Read Remote Supported Features command requests a list of the supported features of a remote device. |
| Read Remote Supported Features Complete Event | 1.1 | The Read Remote Supported Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote Bluetooth device specified by the Connection Handle event parameter. |
| Read Remote Extended Features Command | 1.2 | The Read Remote Extended Features command requests a list of the supported extended features of a remote device |
| Read Remote Extended Features Complete Event | 1.2 | The Read Remote Extended Features Complete Event is used to indicate the completion of the process of the Link Manager obtaining the supported Extended features of the remote Bluetooth device specified by the Connection Handle event parameter. |
| Read Remote Version Information Command | 1.1 | The Read Remote Version Information command will read the values for the version information for the remote Bluetooth device. |
| Read Remote Version Information Complete Event | 1.1 | The Read Remote Version Information Complete event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote Bluetooth device specified by the Connection Handle event parameter. |

*Table 3.8: Remote information*



LifeScan, Inc. Ex. 1019
Page 538 of 1420
IPR2025-00103

## 3.9 SYNCHRONOUS CONNECTIONS

The synchronous connections group of commands and events allows synchronous connections to be created.

| Name | Vers. | Summary description |
|---|---|---|
| Setup Synchronous Connection Command | 1.2 | 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. |
| Synchronous Connection Complete Event | 1.2 | The Synchronous Connection Complete event indicates to both the Hosts that a new Synchronous connection has been established. |
| Synchronous Connection Changed event | 1.2 | The Synchronous Connection Changed event indicates to the Host that an existing Synchronous connection has been reconfigured. |
| Accept Synchronous Connection Request Command | 1.2 | The Accept_Synchronous_Connection_Request command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection. |
| Reject Synchronous Connection Request Command | 1.2 | The Reject_Synchronous_Connection_Request is used to decline an incoming request for a synchronous link. |
| Read Voice Setting Command | 1.1 | The Read Voice Setting command will read the values for the Voice Setting configuration parameter, which controls all the various settings for the voice connections. |
| Write Voice Setting Command | 1.1 | The Write Voice Setting command will write the values for the Voice Setting configuration parameter, which controls all the various settings for the voice connections. |

*Table 3.9: Synchronous connections*



LifeScan, Inc. Ex. 1019
Page 539 of 1420
IPR2025-00103

## 3.10 CONNECTION STATE

The connection state group of commands and events allows the configuration of a link, especially for low power operation.

| Name | Vers. | Summary description |
|---|---|---|
| Mode Change Event | 1.1 | The Mode Change event is used to indicate that the current mode has changed. |
| Max Slots Change Event | 1.1 | The Max Slots Change event it used to indicate a change in the max slots by the LM. |
| Hold Mode Command | 1.1 | The Hold Mode command is used to initiate Hold Mode. |
| Sniff Mode Command | 1.1 | The Sniff Mode command is used to alter the behavior of the LM and have the LM place the local or remote device into the sniff mode. |
| Sniff Subrating Command | 2.1 + EDR | The Sniff Subrating command is used to configure the sniff subrating parameters in the local device |
| Sniff Subrating Event | 2.1 + EDR | The Sniff Subrating event is used to inform the Host of the local and remote transmit and receive latencies |
| Exit Sniff Mode Command | 1.1 | The Exit Sniff Mode command is used to end the sniff mode for a Connection Handle which is currently in sniff mode. |
| Park State Command | 1.1 | The Park State command is used to alter the behavior of the LM and have the LM place the local or remote device into the park state. |
| Exit Park State Command | 1.1 | The Exit Park State command is used to switch the Bluetooth device from park state back to active mode. |
| Read Link Policy Settings Command | 1.1 | The Read Link Policy Settings command will read 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. |
| Write Link Policy Settings Command | 1.1 | The Write Link Policy Settings command will write 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 | The Read Default Link Policy Settings command will read the Default Link Policy configuration parameter for all new connections. |

*Table 3.10: Connection state*



LifeScan, Inc. Ex. 1019
Page 540 of 1420
IPR2025-00103

| Name | Vers. | Summary description |
|---|---|---|
| Write Default Link Policy Settings Command | 1.2 | The Write Default Link Policy Settings command will write the Default Link Policy configuration parameter for all new connections. |

*Table 3.10: Connection state*

## 3.11 PICONET STRUCTURE

The piconet structure group of commands and events allows the discovery and reconfiguration of a piconet.

| Name | Vers. | Summary description |
|---|---|---|
| Role Discovery Command | 1.1 | The Role Discovery command is used for a Bluetooth device to determine which role the device is performing for a particular Connection Handle. |
| Switch Role Command | 1.1 | The Switch Role command is used to switch master and slave roles of the devices on either side of a connection. |
| Role Change Event | 1.1 | The Role Change event is used to indicate that the current Bluetooth role related to the particular connection has been changed. |
| Link Supervision Timeout Changed Event | 2.1 + EDR | The Link Supervision Timeout Changed event is used to notify the slave's Host when the link supervision timeout value has changed |

*Table 3.11: Piconet structure*

## 3.12 QUALITY OF SERVICE

The quality of service group of commands and events allows the configuration of links to allow for quality of service parameters to be specified.

| Name | Vers. | Summary description |
|---|---|---|
| Flow Specification Command | 1.2 | The Flow Specification command is used to specify the flow parameters for the traffic carried over the ACL connection identified by the Connection Handle. |
| Flow Specification Complete Event | 1.2 | The Flow Specification Complete event is used to inform the Host about the Quality of Service for the ACL connection the Controller is able to support. |
| QoS Setup Command | 1.1 | The QoS Setup command is used to specify Quality of Service parameters for a Connection Handle. |
| QoS Setup Complete Event | 1.1 | The QoS Setup Complete event is used to indicate that QoS is setup. |
| QoS Violation Event | 1.1 | The QoS Violation event is used to indicate the Link Manager is unable to provide the current QoS requirement for the Connection Handle. |
| Flush Command | 1.1 | The Flush command is used to discard all data that is currently pending for transmission in the Controller for the specified Connection Handle. |
| Flush Occurred Event | 1.1 | The Flush Occurred event is used to indicate that, for the specified Connection Handle, the data to be transmitted has been discarded. |
| Enhanced Flush Command | 2.1 + EDR | The Enhanced Flush Command is used to discard specific packets currently pending for transmission in the Controller for the specified Connection Handle. This command takes a parameter specifying the type of packets to be flushed. |
| Enhanced Flush Complete Event | 2.1 + EDR | The Enhanced Flush Complete event is used to indicate that an Enhanced Flush is complete. |
| Read Automatic Flush Timeout Command | 1.1 | The Read Automatic Flush Timeout will read the value for the Flush Timeout configuration parameter for the specified Connection Handle. The Flush Timeout parameter is only used for ACL connections. |
| Write Automatic Flush Timeout Command | 1.1 | The Write Automatic Flush Timeout will write the value for the Flush Timeout configuration parameter for the specified Connection Handle. The Flush Timeout parameter is only used for ACL connections. |

*Table 3.12: Quality of service*



LifeScan, Inc. Ex. 1019
Page 542 of 1420
IPR2025-00103

| Name | Vers. | Summary description |
|---|---|---|
| Read Failed Contact Counter Command | 1.1 | The Read Failed Contact Counter will read the value for the Failed Contact Counter configuration parameter for a particular connection to another device. |
| Reset Failed Contact Counter Command | 1.1 | The Reset Failed Contact Counter will reset the value for the Failed Contact Counter configuration parameter for a particular connection to another device. |
| Read Num Broadcast Retransmissions Command | 1.1 | The Read Num Broadcast Retransmissions command will read the parameter value for the Number of Broadcast Retransmissions for the device. |
| Write Num Broadcast Retransmissions Command | 1.1 | The Write Num Broadcast Retransmissions command will write the parameter value for the Number of Broadcast Retransmissions for the device. |

*Table 3.12: Quality of service*



LifeScan, Inc. Ex. 1019
Page 543 of 1420
IPR2025-00103

## 3.13 PHYSICAL LINKS

The physical links commands and events allows configuration of the physical link.

| Name | Vers. | Summary description |
|---|---|---|
| Read Link Supervision Timeout Command | 1.1 | The Read Link Supervision Timeout command will read 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 | The Write Link Supervision Timeout command will write the value for the Link Supervision Timeout configuration parameter for the device. This parameter is used by the device to determine link loss. |
| Link Supervision Timeout Event | 2.1 +EDR | A feature bit that indicates whether the device supports sending the HCI Link Supervision Timeout Changed Event to the Host. |
| Read AFH Channel Assessment Mode Command | 1.2 | The Read AFH Channel Assessment Mode command will read the value for the AFH Channel Classification Mode parameter. This value is used to enable or disable the Controller's channel assessment scheme. |
| Write AFH Channel Assessment Mode Command | 1.2 | The Write AFH Channel Assessment Mode command will write the value for the 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 | The Set AFH Host Channel Classification command allows the Host to specify a channel classification based on its "local information". |
| Change Connection Packet Type Command | 1.1 | The Change Connection Packet Type command is used to change which packet types can be used for a connection that is currently established. |
| Connection Packet Type Changed Event | 1.1 | The Connection Packet Type Changed event is used to indicate the completion of the process of the Link Manager changing the packet type mask used for the specified Connection Handle. |

*Table 3.13: Physical links*



LifeScan, Inc. Ex. 1019
Page 544 of 1420
IPR2025-00103

## 3.14 HOST FLOW CONTROL

The Host flow control group of commands and events allows flow control to be used towards the Host.

| Name | Vers. | Summary description |
|---|---|---|
| Host Buffer Size Command | 1.1 | The Host Buffer Size command is 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 | The Set Event Mask command is used to control which events are generated by the HCI for the Host. |
| Set Event Filter Command | 1.1 | 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. |
| Set Controller To Host Flow Control Command | 1.1 | The Set Controller To Host Flow Control command is 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 | The Host Number Of Completed Packets command is 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 | The Data Buffer Overflow event is used to indicate 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 | The Read Synchronous Flow Control Enable command provides the ability to read 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. |
| Write Synchronous Flow Control Enable Command | 1.1 | The Write Synchronous Flow Control Enable command 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. |

*Table 3.14: Controller flow control.*



LifeScan, Inc. Ex. 1019
Page 545 of 1420
IPR2025-00103

## 3.15 LINK INFORMATION

The link information group of commands and events allows information about a link to be read.

| Name | Vers. | Summary description |
|---|---|---|
| Read LMP Handle Command | 1.2 | The Read LMP Handle command will read the current LMP Handle associated with the Connection Handle. |
| Read Transmit Power Level Command | 1.1 | The Read Transmit Power Level command will read the values for the Transmit Power Level parameter for the specified Connection Handle. |
| Read Link Quality Command | 1.1 | The Read Link Quality command will read the value for the Link Quality for the specified Connection Handle. |
| Read RSSI Command | 1.1 | The Read RSSI command will read the value for the Received Signal Strength Indication (RSSI) for a Connection Handle to another Bluetooth device. |
| Read Clock Offset Command | 1.1 | The Read Clock Offset command allows the Host to read the clock offset of remote devices. |
| Read Clock Offset Complete Event | 1.1 | The Read Clock Offset Complete event is used to indicate the completion of the process of the LM obtaining the Clock offset information. |
| Read Clock Command | 1.2 | The Read Clock command will read an estimate of a piconet or the local Bluetooth Clock. |
| Read AFH Channel Map Command | 1.2 | The Read AFH Channel Map command will read the current state of the channel map for a connection. |

*Table 3.15: Link information*



LifeScan, Inc. Ex. 1019
Page 546 of 1420
IPR2025-00103

## 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.

| Name | Vers. | Summary description |
|---|---|---|
| Link Key Request Event | 1.1 | 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. |
| Link Key Request Reply Command | 1.1 | 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 Bluetooth device specified by BD_ADDR. |
| Link Key Request Negative Reply Command | 1.1 | 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 Bluetooth Device specified by BD_ADDR. |
| PIN Code Request Event | 1.1 | The PIN Code Request event is used to indicate that a PIN code is required to create a new link key for a connection. |
| PIN Code Request Reply Command | 1.1 | 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. |
| PIN Code Request Negative Reply Command | 1.1 | 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. |
| Link Key Notification Event | 1.1 | 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. |
| Authentication Requested Command | 1.1 | The Authentication Requested command is used to establish authentication between the two devices associated with the specified Connection Handle. |
| Authentication Complete Event | 1.1 | The Authentication Complete event occurs when authentication has been completed for the specified connection. |
| Set Connection Encryption Command | 1.1 | The Set Connection Encryption command is used to enable and disable the link level encryption. |

*Table 3.16: Authentication and encryption*

| Name | Vers. | Summary description |
|---|---|---|
| Encryption Change Event | 1.1 | 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. |
| Change Connection Link Key Command | 1.1 | The Change Connection Link Key command is used to force both devices of a connection associated to the Connection Handle, to generate a new link key. |
| Change Connection Link Key Complete Event | 1.1 | 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 had been completed. |
| Master Link Key Command | 1.1 | The Master Link Key command is used to 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 | The Master Link Key Complete event is used to indicate that the change in the temporary Link Key or in the semi-permanent link keys on the Bluetooth master side has been completed. |
| Read PIN Type Command | 1.1 | The Read PIN Type command is 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 | The Write PIN Type command is used for the Host to specify whether the Host supports variable PIN or only fixed PINs. |
| Read Stored Link Key Command | 1.1 | The Read Stored Link Key command provides the ability to read whether one or more link keys are stored in the Controller. |
| Return Link Keys Event | 1.1 | The Return Link Keys event is used to return stored link keys after a Read Stored Link Key command is used. |
| Write Stored Link Key Command | 1.1 | The Write Stored Link Key command provides the ability to write one or more link keys to be stored in the Controller. |
| Delete Stored Link Key Command | 1.1 | The Delete Stored Link Key command provides the ability to remove one or more of the link keys stored in the Controller. |
| Create New Unit Key Command | 1.1 | The Create New Unit Key command is used to create a new unit key. |
| Refresh Encryption Key | 2.1 + EDR | The Refresh Encryption Key is used by the Host to cause the Controller to refresh the encryption key by pausing and resuming encryption |

*Table 3.16: Authentication and encryption*



LifeScan, Inc. Ex. 1019
Page 548 of 1420
IPR2025-00103

| Name | Vers. | Summary description |
|---|---|---|
| Encryption Key Refresh Complete | 2.1 + EDR | The Encryption Key Refresh Complete event is used to indicate to the Host that the encryption key was refreshed on the given Connection_Handle any time encryption is paused and then resumed. T |
| IO Capability Request Reply | 2.1 + EDR | The IO_Capability_Request_Reply command is used to reply to an IO Capability Request event from the controller, and specifies the current I/O capabilities of the Host. |
| User Confirmation Request Reply | 2.1 + EDR | The User_Confirmation_Request_Reply command is used to reply to a User Confirmation Request event and indicates that the user selected "yes". It is also used when the host has no input and no output capabilities. |
| User Confirmation Request Negative Reply | 2.1 + EDR | The User_Confirmation_Request_Negative_Reply command is used to reply to a User Confirmation Request event and indicates that the user selected "no". This command will terminate Secure Simple Pairing. |
| User Passkey Request Reply | 2.1 + EDR | The User_Passkey_Request_Reply command is used to reply to a User Passkey Request event and specifies the Numeric_Value (passkey) entered by the user to be used in the Secure Simple Pairing process. |
| User Passkey Request Negative Reply | 2.1 + EDR | The User_Passkey_Request_Negative_Reply command is used to reply to a User Passkey Request event and indicates the host could not provide a passkey. This command will terminate Simple Pairing. |
| Remote OOB Data Request Reply | 2.1 + EDR | The Remote_OOB_Data_Request_Reply command is used to reply to a Remote OOB Data Request event with the C and R values received via an OOB transfer from a remote device identified by BD_ADDR. |
| Remote OOB Data Request Negative Reply | 2.1 + EDR | The Remote_OOB_Data_Request_Negative_Reply command is used to reply to a Remote OOB Data Request event that the Host does not have the C and R |
| Read Simple Pairing Mode | 2.1 + EDR | This command reads the Simple Pairing mode setting in the Controller. |
| Write Simple Pairing Mode | 2.1 + EDR | This command writes the Simple Pairing mode setting in the Controller. |
| Read Local OOB Data | 2.1 + EDR | This command is used to obtain a Simple Pairing Hash C and Simple Pairing Randomizer R which are intended to be transferred to a remote device using an OOB mechanism. |

*Table 3.16: Authentication and encryption*

| Name | Vers. | Summary description |
|---|---|---|
| Send Keypress Notification | 2.1 + EDR | This command is used during the Passkey Entry protocol by a device with KeyboardOnly IO capabilities. It is used by a Host to inform the remote device when keys have been entered or erased. |

*Table 3.16: Authentication and encryption*

## 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.

| Name | Vers. | Summary description |
|---|---|---|
| Read Loopback Mode Command | 1.1 | The Read Loopback Mode will read the value for the setting of the Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information. |
| Write Loopback Mode Command | 1.1 | The Write Loopback Mode will write the value for the setting of the Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information. |
| Loopback Command Event | 1.1 | The Loopback Command event is used to loop back all commands that the Host sends to the Controller with some exceptions. |
| Enable Device Under Test Mode Command | 1.1 | The Enable Device Under Test Mode command will allow the local Bluetooth 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 the Testing scenarios as described in the Bluetooth Test Mode document. |
| Write Simple Pairing Debug Mode | 2.1 + EDR | This command configures the Controller to use a predefined Diffie Hellman private key for Simple Pairing to enable debug equipment to monitor the encrypted connection. |

*Table 3.17: Testing*



LifeScan, Inc. Ex. 1019
Page 550 of 1420
IPR2025-00103

## 3.18 ALPHABETICAL LIST OF COMMANDS AND EVENTS

| Commands/Events | Group |
|---|---|
| Accept Connection Request Command | Connection Setup |
| Accept Synchronous 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 |
| Enable Device Under Test Mode Command | Testing |
| Encryption Change Event | Authentication and Encryption |
| Encryption Key Refresh Complete | Authentication and Encryption |
| Enhanced Flush Command | Quality of Service |
| Enhanced Flush Complete Event | Quality of Service |
| Exit Park State Command | Connection State |
| Exit Periodic Inquiry Mode Command | Device Discovery |
| Exit Sniff Mode Command | Connection State |
| Extended Inquiry Result Event | Device Discovery |
| Flow Specification Command | Quality of Service |
| Flow Specification Complete Event | Quality of Service |

*Table 3.18: Alphabetical list of commands and events.*

| Commands/Events | Group |
|---|---|
| Flush Command | Quality of Service |
| Flush Occurred Event | Quality of Service |
| Hardware Error Event | 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 |
| PIN Code Request Reply Command | Authentication and Encryption |
| QoS Setup Command | Quality of Service |
| QoS Setup Complete Event | Quality of Service |
| QoS Violation Event | Quality of Service |
| Read AFH Channel Assessment Mode Command | Physical Links |
| Read AFH Channel Map Command | Link Information |

*Table 3.18: Alphabetical list of commands and events.*

| Commands/Events | Group |
|---|---|
| Read Authentication Enable Command | Authentication and Encryption |
| Read Automatic Flush Timeout Command | Quality of Service |
| 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 Default Erroneous Data Reporting | Controller Configuration |
| Read Encryption Mode Command | Authentication and Encryption |
| Read Extended Inquiry Response Command | Controller Configuration |
| Read Failed Contact Counter Command | Quality of Service |
| 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 |
| Read Num Broadcast Retransmissions Command | Quality of Service |
| Read Number Of Supported IAC Command | Controller Information |
| Read Page Scan Activity Command | Connection Setup |

*Table 3.18: Alphabetical list of commands and events.*

| Commands/Events | Group |
|---|---|
| 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 |
| Refresh Encryption Key | Authentication and Encryption |
| Reject Connection Request Command | Connection Setup |
| Reject Synchronous 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 | Quality of Service |
| 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 |
| Setup Synchronous Connection Command | Synchronous Connections |

*Table 3.18: Alphabetical list of commands and events.*

| Commands/Events | Group |
|---|---|
| Sniff Mode Command | Connection State |
| Sniff Subrating Command | Connection State |
| Sniff Subrating Event | 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 | Quality of Service |
| 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 Default Erroneous Data Reporting | Controller Configuration |
| Write Extended Inquiry Response Command | Controller Configuration |
| 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 | Quality of Service |
| Write Page Scan Activity 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 |

*Table 3.18: Alphabetical list of commands and events.*



LifeScan, Inc. Ex. 1019
Page 555 of 1420
IPR2025-00103

# 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.

## 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 the following 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.

Note: The Number Of Completed Packets events will not report on synchronous Connection Handles if Synchronous Flow Control is disabled. (See Read/

Write Synchronous Flow Control Enable, Section 7.3.36 on page 544 and Section 7.3.37 on page 545)

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 is received from the Host for a given Connection Handle in the same order as it is received from the Host.

Data that is 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 it is received. This means the scheduling shall be decided separately for each Connection Handle basis. For each individual Connection Handle, the order of the data shall not be changed from the order in which the data has been created.

## 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 Host Controller To Host Flow Control command can be used to turn flow control on or off in that direction.

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 this section.

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.



LifeScan, Inc. Ex. 1019
Page 557 of 1420
IPR2025-00103

## 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 sent 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.

## 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 must start performing the commands in the order in which they are received. The Controller can 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 is set to 1 or more. Command Opcode 0x0000 is a NOP (No OPeration), and can 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 (for example, if there was a parameter error or the command is currently not allowed), 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.

## 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.

Note: The BD_ADDR return parameter of the command Read BD_ADDR 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.



LifeScan, Inc. Ex. 1019
Page 559 of 1420
IPR2025-00103

# 5 HCI DATA FORMATS

## 5.1 INTRODUCTION

The HCI provides a uniform command method of accessing the Bluetooth capabilities. The HCI Link commands provide the Host with the ability to control the link layer connections to other Bluetooth devices. These commands typically involve the Link Manager (LM) to exchange LMP commands with remote Bluetooth devices. For details see "Link Manager Protocol" on page 203 [Part C].

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 & Baseband, 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 HCI-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 HCI-Transport Layer used, it is recommended to use a default value of one second for this timer. This amount of time is also dependent on the number of commands unprocessed in the command queue.

## 5.2 DATA AND PARAMETER FORMATS

- All values are in Binary and Hexadecimal Little Endian formats unless otherwise noted
- In addition, all parameters which can have negative values must use 2's 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 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 multi-octet 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'.
- Values or parameters marked as Reserved for Future Use, shall be set to 0 unless explicitly stated otherwise on transmission, and shall be ignored on reception. Parameter values or opcodes that an implementation does not know how to interpret shall be ignored, and the operation that is being attempted shall be completed with the correct signaling. The host or controller shall not stop functioning because of receiving a reserved value.

## 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 events. Broadcast Connection Handles are handled differently, and are described below.

The first time the Host sends an HCI Data Packet with Broadcast_Flag set to 01b (active slave broadcast) or 10b (parked slave broadcast) after a power-on or a reset, the value of the Connection Handle parameter must be a value which is not currently assigned by the Host Controller. The Host must use different Connection Handles for active broadcast and piconet broadcast.

The Controller must then continue to use the same connection Handles for each type of broadcast until a reset is made. Note: The Host Controller must not send a Connection Complete event containing a new Connection_Handle that it knows is used for broadcast.

Note: 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 which 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 above conflict case, 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.

## 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 next sections specify the HCI packet formats.

### 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 5.1, and the definition of each field is explained below.

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, called the 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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 562 of 1420
IPR2025-00103

Note: the OGF composed of all ‘zeros’ and an OCF or all ‘zeros’ is the NOP command. This command Opcode may be used in Command Flow Control. (See Section 4.4 on page 388)

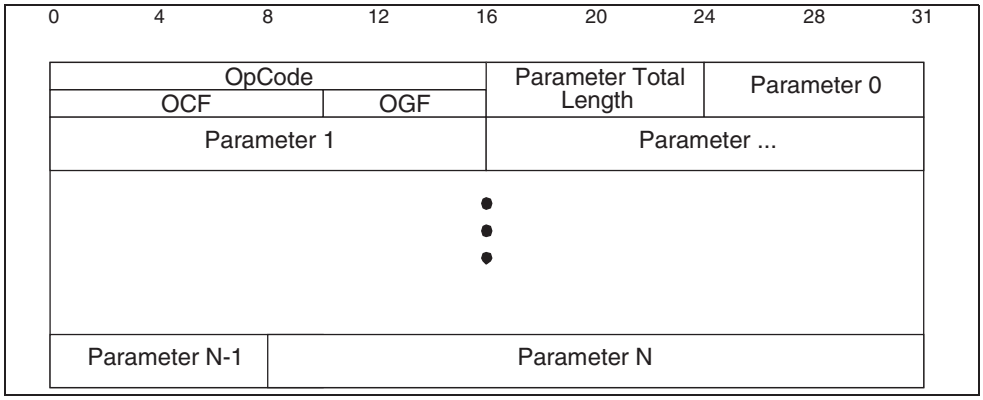



*Figure 5.1: HCI Command Packet*

*Op_Code:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | OGFRange (6 bits): 0x00-0x3F (0x3E reserved for Bluetooth logo testing and 0x3F reserved for vendor-specific debug commands)<br>OCF Range (10 bits): 0x0000-0x03FF |

*Parameter_Total_Length:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Lengths of all of the parameters contained in this packet measured in octets. (N.B.: total length of parameters, not number of parameters) |

*Parameter 0 - N:* *Size: Parameter Total Length*

| Value | Parameter Description |
|---|---|
| 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. |

### 5.4.2 HCI ACL Data Packets

HCI ACL Data Packets are used to exchange data between the Host and Controller. There are two types of HCI ACL Data Packets: Automatically-Flushable and Non-Automatically-Flushable. Automatically-Flushable HCI Data Packets are flushed based on the setting of an automatic flush timer (see Section 7.3.29 on page 536). Non-Automatically-Flushable HCI Data Packets are not controlled by the automatic flush timeout and shall not be automatically flushed. The format of the HCI ACL Data Packet is shown in Figure 5.2. The definition for each of the fields in the data packets is explained below.

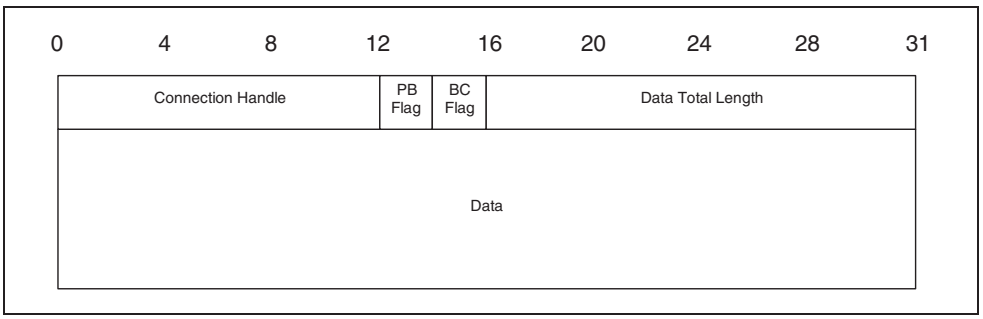



*Figure 5.2: HCI ACL Data Packet*

*Connection_Handle:* *Size: 12 Bits*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection Handle to be used for transmitting a data packet or segment.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Flags:* *Size: 2 Bits*

*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 7 in the second octet of the HCI ACL Data packet.*

*Packet_Boundary_Flag:* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| 00 | First non-automatically-flushable packet of Higher Layer Message (start of a non-automatically-flushable L2CAP packet) |
| 01 | Continuing fragment packet of Higher Layer Message |
| 10 | First automatically-flushable packet of Higher Layer Message (start of an automatically-flushable L2CAP packet) |
| 11 | Reserved for future use |



LifeScan, Inc. Ex. 1019
Page 564 of 1420
IPR2025-00103

*Broadcast_Flag (in packet from Host to Controller):* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| 00 | No broadcast. Only point-to-point. |
| 01 | Active Slave Broadcast: 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 | Parked Slave Broadcast: packet is sent to all 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 (in packet from Controller to Host):* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| 00 | Point-to-point |
| 01 | Packet received as a slave not in park state (either Active Slave Broadcast or Parked Slave Broadcast) |
| 10 | .Packet received as a slave in park state (Parked Slave Broadcast) |
| 11 | Reserved for future use. |

Note: active slave broadcast 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.

*Data_Total_Length:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length of data measured in octets. |



LifeScan, Inc. Ex. 1019
Page 565 of 1420
IPR2025-00103

### 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 5.3. The definition for each of the fields in the data packets is explained below.

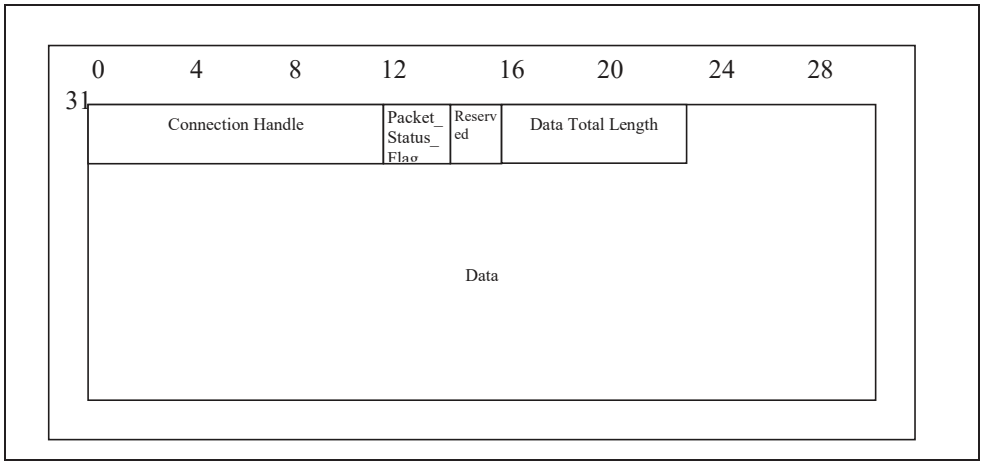



*Figure 5.3: HCI Synchronous Data Packet*

*Connection_Handle:* *Size: 12 Bits*

| Value | Parameter Description |
|---|---|
| 0xXXX | Connection handle to be used to for transmitting a synchronous data packet or segment.<br>Range: 0x0000-0x0EFF (0x0F00- 0x0FFF Reserved for future use) |

The Packet_Status_Flag bits consist of two bits, which are located from bit 4 to 5 in the second octet of the HCI Synchronous Data packet.

The Host shall set the Packet_Status_Flag bits to 00.

If the Erroneous_Data_Reporting parameter was set to disabled when the synchronous connection was created, the Controller shall set the Packet_Status_Flag bits to 00 and whether or not data is provided for cases when a valid (e)SCO packet was not received is unspecified.

If the Erroneous_Data_Reporting parameter was set to enabled when the synchronous connection was created, the controller shall set the Packet_Status_Flag according to the following table.

*Packet_Status_Flag (in packets sent by the Controller)* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| 00 | Correctly received data. The payload data belongs to received eSCO or SCO packets that the baseband marked as "good data". |
| 01 | Possibly invalid data. At least one eSCO packet has been marked by the baseband as "data with possible errors" and all others have been marked as "good data" in the eSCO interval(s) corresponding to the HCI Synchronous Data Packet. |
| 10 | No data received. All data from the baseband received during the (e)SCO interval(s) corresponding to the HCI Synchronous Data Packet have been marked as "lost data" by the baseband. The Payload data octets shall be set to 0. |
| 11 | Data partially lost. Not all, but at least one (e)SCO packet has been marked as "lost data" by the baseband in the (e)SCO intervals corresponding to the HCI Synchronous Data Packet. The payload data octets corresponding to the missing (e)SCO packets shall be set to 0. |

**Note**: some HCI transports and/or controller implementations will align the HCI Synchronous Data Packets with the (e)SCO baseband packets such that data integrity can be explicitly marked in the Packet_Status_Flag. For HCI transports or Controller implementations that do not preserve this alignment, information in the Packet_Status_Flag may be ambiguous.

The Reserved Bits consist of two bits which are located from bit 6 to bit 7 in the second octet of the HCI Synchronous Data packet.

*Reserved:* *Size: 2 Bits*

| Value | Parameter Description |
|---|---|
| XXXX | Reserved for future use. |

*Data_Total_Length:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of synchronous data measured in octets |



LifeScan, Inc. Ex. 1019
Page 567 of 1420
IPR2025-00103

### 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 5.4, and the definition of each field is explained below.

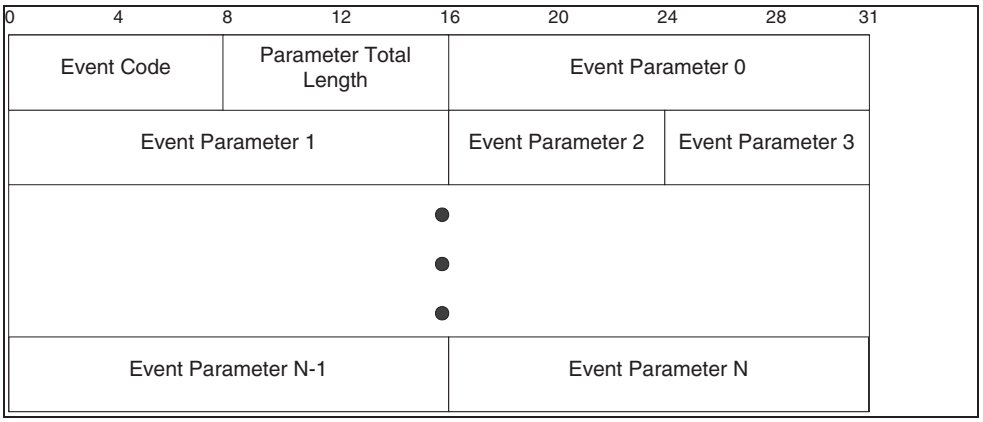



*Figure 5.4: HCI Event Packet*

*Event_Code:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Each event is assigned a 1-Octet event code used to uniquely identify different types of events.<br>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:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Length of all of the parameters contained in this packet, measured in octets |

*Event_Parameter 0 - N:* *Size: Parameter Total Length*

| Value | Parameter Description |
|---|---|
| 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. |

# 6 HCI CONFIGURATION PARAMETERS

## 6.1 SCAN ENABLE

The Scan_Enable parameter controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth 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.

| Value | Parameter Description |
|---|---|
| 0x00 | No Scans enabled. |
| 0x01 | Inquiry Scan enabled.<br>Page Scan always disabled. |
| 0x02 | Inquiry Scan disabled.<br>Page Scan enabled. |
| 0x03 | Inquiry Scan enabled.<br>Page Scan enabled. |
| 0x04-0xFF | Reserved [0x04 limited page scan removed] |

## 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.

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0012 to 0x1000; only even values are valid<br>Default: 0x1000<br>Mandatory Range: 0x0012 to 0x1000<br>Time = N * 0.625 msec<br>Time Range: 11.25 to 2560 msec<br>Time Default: 2.56 sec |

## 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 only be less than or equal to the Inquiry_Scan_Interval.

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0011 to 0x1000<br>Default: 0x0012<br>Mandatory Range: 0x0011 to Inquiry Scan Interval<br>Time = N * 0.625 msec<br>Time Range: 10.625 msec to 2560 msec<br>Time Default: 11.25 msec |

## 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. For details, see the Baseband Specification, “Inquiry scan substate” on page 156 [Part B].

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| 0x02-0xFF | Reserved |

## 6.5 INQUIRY MODE

The Inquiry_Mode configuration parameter indicates whether inquiry returns Inquiry Result events in the standard format, with RSSI, or with RSSI and extended inquiry response information.

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Inquiry Result event format |
| 0x01 | Inquiry Result format with RSSI |
| 0x02 | Inquiry Result with RSSI format or Extended Inquiry Result format |
| 0x03-0xFF | Reserved |

## 6.6 PAGE TIMEOUT

The Page_Timeout configuration parameter defines the maximum time the local Link Manager will wait for a baseband 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 baseband level, the connection attempt will be considered to have failed.

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0001 to 0xFFFF<br>Default: 0x2000<br>Mandatory Range: 0x0016 to 0xFFFF<br>Time = N * 0.625 msec<br>Time Range: 0.625 msec to 40.9 sec<br>Time Default: 5.12 sec |

## 6.7 CONNECTION ACCEPT TIMEOUT

The Connection_Accept_Timeout configuration parameter allows the Bluetooth 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.

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0001 to 0xB540<br>Default: 0x1F40<br>Mandatory Range: 0x00A0 to 0xB540<br>Time = N * 0.625 msec<br>Time Range: 0.625 msec to 29 sec<br>Time Default: 5 sec |

## 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.

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0012 to 0x1000; only even values are valid<br>Default: 0x0800<br>Mandatory Range: 0x0012 to 0x1000<br>Time = N * 0.625 msec<br>Time Range: 11.25 msec to 2560 msec<br>Time Default: 1.28 sec |

## 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 only be less than or equal to the Page_Scan_Interval.

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0011 to 0x1000<br>Default: 0x0012<br>Mandatory Range: 0x0011 to Page Scan Interval<br>Time = N * 0.625 msec<br>Time Range: 10.625 msec to Page Scan Interval<br>Time Default: 11.25 msec |

## 6.10 PAGE SCAN PERIOD MODE (DEPRECATED)

Every time an inquiry response message is sent, the Bluetooth 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 Bluetooth device will use the mandatory page scan mode for all following page scans.

Note: the timer T_mandatory_pscan will be reset at each new inquiry response. For details see the “Baseband Specification” on page 53 [Part B]. (Keyword: SP-Mode, FHS-Packet, T_mandatory_pscan, Inquiry-Response).

| Value | Parameter Description |
|---|---|
| 0x00 | P0 |
| 0x01 | P1 |
| 0x02 | P2 |

| | |
|---|---|
| 0x03-0xFF | Reserved. |

## 6.11 PAGE SCAN TYPE

The Page_Scan_Type parameter indicates whether inquiry scanning will be done using non-interlaced scan or interlaced scan. For details, see the Baseband Specification, “Page scan substate” on page 146 [Part B].

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| 0x02-0xFF | Reserved |

## 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.

| 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: 1’s complement |
| XX01XXXXXX | Input Data Format: 2’s 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 Linear PCM). |
| XXXXXXXX00 | Air Coding Format: CVSD |
| XXXXXXXX01 | Air Coding Format: μ-law |

| Value | Parameter Description |
|---|---|
| XXXXXXXX10 | Air Coding Format: A-law |
| XXXXXXXX11 | Air Coding Format: Transparent Data |

## 6.13 PIN TYPE

The PIN Type configuration parameter determines whether the Link Manager assumes that the Host supports variable PIN codes or a fixed PIN code. The host controller uses the PIN-type information during pairing.

| Value | Parameter Description |
|---|---|
| 0x00 | Variable PIN. |
| 0x01 | Fixed PIN. |

## 6.14 LINK KEY

The Controller can store a limited number of link keys for other Bluetooth devices. Link keys are shared between two Bluetooth 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 Bluetooth Controller when needed. A Link Key is associated with a BD_ADDR.

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX | Link Key for an associated BD_ADDR. |

## 6.15 FAILED CONTACT COUNTER

The Failed_Contact_Counter records the number of consecutive incidents in which either the slave or master didn't 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. The Failed_Contact_Counter for a connection is reset to zero on the following conditions:

1. When a new connection is established
2. When the Failed_Contact_Counter is > zero and an L2CAP packet is acknowledged for that connection
3. When the Reset_Failed_Contact_Counter command has been issued

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of consecutive failed contacts for a connection corresponding to the Connection Handle. |

## 6.16 HOLD MODE ACTIVITY

The Hold_Mode_Activity 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 Mode 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 is only valid if all connections are in Hold Mode.

| 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. |



LifeScan, Inc. Ex. 1019
Page 575 of 1420
IPR2025-00103

## 6.17 LINK POLICY SETTINGS

The Link_Policy_Settings parameter determines the behavior of the local Link Manager 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, or sniff mode. The local Link Manager 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 only be used for requests from a remote device or from the local Link Manager 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.

Note: 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 only be done 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.

Note that the previous description 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.

| 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. |
| 0x0008<br>–<br>0x8000 | Reserved for Future Use. |

## 6.18 FLUSH TIMEOUT

The Flush_Timeout configuration parameter is used for ACL connections only. The Flush Timeout is defined in the Baseband specification section 7.6.3, “Flushing Payloads,” on page 142. This parameter allows automatically-flushable ACL packets to be automatically flushed without the Host device issuing a Flush command. This provides support for isochronous data, such as audio. Non-automatically-flushable ACL packets shall not be affected by this parameter (see Section 5.4.2 on page 394) providing support for both asynchronous and isochronous data on the same ACL connection. When the L2CAP packet that is currently being transmitted is automatically ‘flushed’, the Failed Contact Counter is incremented by one.

| Value | Parameter Description |
|---|---|
| 0 | Timeout = ∞; No Automatic Flush |
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0001 to 0x07FF<br>Mandatory Range: 0x0002 to 0x07FF<br>Time = N * 0.625 msec<br>Time Range: 0.625 msec to 1279.375 msec |

## 6.19 NUM 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 sets the value $N_{BC}$ in the baseband to one greater than the Num Broadcast Retransmissions value. (See Baseband Specification, Section 7.6.5, on page 143) This parameter should be adjusted as the link quality measurement changes.

| Value | Parameter Description |
|---|---|
| N = 0xXX | $N_{BC}$ = N + 1<br>Range 0x00-0xFE |

## 6.20 LINK SUPERVISION TIMEOUT

The Link_Supervision_Timeout parameter is used by the master or slave Bluetooth device to monitor link loss. If, for any reason, no Baseband 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.

Note: Setting the Link_Supervision_Timeout 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 Bluetooth Device every ~40 seconds. By using the No Link_Supervision_Timeout setting, the scalability of the Park state is not limited.

| Value | Parameter Description |
|---|---|
| 0x0000 | No Link_Supervision_Timeout. |
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0001 to 0xFFFF<br>Default: 0x7D00<br>Mandatory Range: 0x0190 to 0xFFFF<br>Time = N * 0.625 msec<br>Time Range: 0.625 msec to 40.9 sec<br>Time Default: 20 sec |

## 6.21 SYNCHRONOUS FLOW CONTROL ENABLE

The Synchronous Flow Control Enable configuration 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.

| 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. |



LifeScan, Inc. Ex. 1019
Page 578 of 1420
IPR2025-00103

## 6.22 LOCAL NAME

The user-friendly Local Name provides the user the ability to distinguish one Bluetooth 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.

Note: The Local Name configuration parameter is a string parameter. Endianess does therefore not apply to the Local Name configuration parameter. The first octet of the name is received first.

| Value | Parameter Description |
|---|---|
| | A UTF-8 encoded User Friendly Descriptive Name for the device.<br>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. |

## 6.23 EXTENDED INQUIRY RESPONSE

The Extended Inquiry Response provides information about the local device in response to inquiry from remote devices. The configuration parameter has two parts, a significant part followed by a non-significant part. The non-significant part contains only zero octets. The length of the extended inquiry response configuration parameter is 240 octets. The data format of the significant part is defined in [Vol 3 Part C] Section 8 on page 229]

| Value | Parameter Description |
|---|---|
| | Information about the local device that will be sent in an extended inquiry response packet to remote devices during inquiry response. |

## 6.24 ERRONEOUS DATA REPORTING

The Erroneous_Data_Reporting configuration parameter shall be used for SCO and eSCO connections only. This parameter determines if the Controller is required to provide data to the Host for every (e)SCO interval, with the Packet_Status_Flag in HCI Synchronous Data Packets set according to section 5.4.3, “HCI Synchronous Data Packets,” on page 396.

| Value | Parameter Description |
|---|---|
| 0x00 | Erroneous data reporting disabled |
| 0x01 | Erroneous data reporting enabled. |
| 0x02 - 0xFF | Reserved |

## 6.25 CLASS OF DEVICE

The Class_of_Device parameter is used to indicate the capabilities of the local device to other devices.

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device. |

## 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.

| 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 |
| | 7 | Cancel Create Connection |
| 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 Request |

| Octet | Bit | Command Supported |
|---|---|---|
| 2 | 0 | Set Connection Encryption |
| | 1 | Change Connection Link Key |
| | 2 | Master Link Key |
| | 3 | Remote Name Request |
| | 4 | Cancel Remote Name Request |
| | 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 Mark |
| | 7 | Reset |



LifeScan, Inc. Ex. 1019
Page 581 of 1420
IPR2025-00103

| 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 | Reserved |
| | 5 | Reserved |
| | 6 | Reserved |
| | 7 | Reserved |
| 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 |

| 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 Host 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 | Reserved |
| | 6 | Reserved |
| | 7 | Read Page Scan Mode |
| 12 | 0 | Write Page Scan Mode |
| | 1 | Set AFH 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 |

| 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 Count |
| | 3 | Reset Failed Contact Count |
| | 4 | Get Link Quality |
| | 5 | Read RSSI |
| | 6 | Read AFH Channel Map |
| | 7 | Read BD 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 |
| 17 | 0 | Read Extended Inquiry Response |
| | 1 | Write Extended Inquiry Response |
| | 2 | Refresh Encryption Key |
| | 3 | Reserved |
| | 4 | Sniff Subrating |
| | 5 | Read Simple Pairing Mode |
| | 6 | Write Simple Pairing Mode |
| | 7 | Read Local OOB Data |

| Octet | Bit | Command Supported |
|---|---|---|
| 18 | 0 | Read Inquiry Response Transmit Power |
| | 1 | Write Inquiry Transmit Power Level |
| | 2 | Read Default Erroneous Data Reporting |
| | 3 | Write Default Erroneous Data Reporting |
| | 4 | Reserved |
| | 5 | Reserved |
| | 6 | Reserved |
| | 7 | IO Capability Request Reply |
| 19 | 0 | User Confirmation Request Reply |
| | 1 | User Confirmation Request Negative Reply |
| | 2 | User Passkey Request Reply |
| | 3 | User Passkey Request Negative Reply |
| | 4 | Remote OOB Data Request Reply |
| | 5 | Write Simple Pairing Debug Mode |
| | 6 | Enhanced Flush |
| | 7 | Remote OOB Data Request Negative Reply |
| 20 | 0 | Reserved |
| | 1 | Reserved |
| | 2 | Send Keypress Notification |
| | 3 | IO Capabilities Response Negative Reply |
| | 4 | Reserved |
| | 5 | Reserved |
| | 6 | Reserved |
| | 7 | Reserved |

# 7 HCI COMMANDS AND EVENTS

## 7.1 LINK CONTROL COMMANDS

The Link Control commands allow the Controller to control connections to other Bluetooth devices. When the Link Control commands are used, the Link Manager (LM) controls how the Bluetooth piconets and scatternets are established and maintained. These commands instruct the LM to create and modify link layer connections with Bluetooth remote devices, perform Inquiries of other Bluetooth devices in range, and other LMP commands. For the Link Control commands, the OGF is defined as 0x01.

### 7.1.1 Inquiry Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Inquiry | 0x0001 | LAP, Inquiry_Length, Num_Responses | |

**Description:**

This command will cause the Bluetooth device to enter Inquiry Mode. Inquiry Mode is used to discover other nearby Bluetooth devices. The LAP input parameter contains the LAP from which the inquiry access code 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 Bluetooth 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 Inquiry Complete event will have a summary of the result from the Inquiry process, which reports the number of nearby Bluetooth devices that responded. When a Bluetooth device responds to the Inquiry message, an Inquiry Result event will occur to notify the Host of the discovery.

A device which 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 command Set_Event_Filter. 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 tries to report a particular device only once during an inquiry or inquiry period.

**Command Parameters:**



LifeScan, Inc. Ex. 1019
Page 586 of 1420
IPR2025-00103

*LAP:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0x9E8B00–<br>0X9E8B3F | This is the LAP from which the inquiry access code should be derived when the inquiry procedure is made;<br>see “Bluetooth Assigned Numbers. |

*Inquiry_Length:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Maximum amount of time specified before the Inquiry is halted.<br>Size: 1 octet<br>Range: 0x01 – 0x30<br>Time = N * 1.28 sec<br>Range: 1.28 – 61.44 Sec |

*Num_Responses:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unlimited number of responses. |
| 0xXX | Maximum number of responses from the Inquiry before the Inquiry is halted.<br>Range: 0x01 – 0xFF |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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 Bluetooth device which responds to the Inquiry message. In addition, multiple Bluetooth devices which respond to the Inquire message may be combined into the same event. An Inquiry Complete event is generated when the Inquiry process has completed.

Note: 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.

### 7.1.2 Inquiry Cancel Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Inquiry_Cancel | 0x0002 | | Status |

**Description:**

This command will cause the Bluetooth device to stop the current Inquiry if the Bluetooth device is in Inquiry Mode. This command allows the Host to interrupt the Bluetooth device and request the Bluetooth device to perform a different task. The command should only be issued after the Inquiry command has been issued, a Command Status event has been received for the Inquiry command, and before the Inquiry Complete event occurs.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Inquiry_Cancel command succeeded. |
| 0x01-0xFF | Inquiry_Cancel command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away)**:

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.



LifeScan, Inc. Ex. 1019
Page 588 of 1420
IPR2025-00103

### 7.1.3 Periodic Inquiry Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Periodic_ Inquiry_Mode | 0x0003 | Max_Period_Length,<br>Min_Period_Length,<br>LAP,<br>Inquiry_Length,<br>Num_Responses | Status |

**Description:**

The Periodic_Inquiry_Mode command is used to configure the Bluetooth device to enter the Periodic Inquiry Mode that performs an automatic Inquiry. Max_Period_Length and Min_Period_Length 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 inquiry access code shall be derived when the inquiry procedure is made. The Inquiry_Length parameter specifies the total duration of the InquiryMode 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 Bluetooth 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 Bluetooth device responds to the Inquiry message an Inquiry Result event will occur to notify the Host of the discovery.

Note: Max_Period_Length > Min_Period_Length > Inquiry_Length

A device which 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 command Set_Event_Filter. 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 tries to report a particular device only once during an inquiry or inquiry period.

**Command Parameters**:

*Max_Period_Length:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum amount of time specified between consecutive inquiries.<br>Size: 2 octets<br>Range: 0x03 – 0xFFFF<br>Time = N * 1.28 sec<br>Range: 3.84 – 83884.8 Sec<br>0.0 – 23.3 hours |

*Min_Period_Length:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum amount of time specified between consecutive inquiries.<br>Size: 2 octets<br>Range: 0x02 – 0xFFFE<br>Time = N * 1.28 sec<br>Range: 2.56 – 83883.52 Sec<br>0.0 – 23.3 hours |

*LAP:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0x9E8B00–<br>0X9E8B3F | This is the LAP from which the inquiry access code should be derived when the inquiry procedure is made;<br>see "Bluetooth Assigned Numbers" |

*Inquiry_Length:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Maximum amount of time specified before the Inquiry is halted.<br>Size: 1 octet<br>Range: 0x01 – 0x30<br>Time = N * 1.28 sec<br>Range: 1.28 – 61.44 Sec |

*Num_Responses:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Unlimited number of responses. |
| 0xXX | Maximum number of responses from the Inquiry before the Inquiry is halted.<br>Range: 0x01 – 0xFF |



LifeScan, Inc. Ex. 1019
Page 590 of 1420
IPR2025-00103

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Periodic Inquiry Mode command succeeded. |
| 0x01-0xFF | Periodic Inquiry Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

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 Bluetooth device which responds to the Inquiry message. In addition, multiple Bluetooth devices which response 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.



LifeScan, Inc. Ex. 1019
Page 591 of 1420
IPR2025-00103

### 7.1.4 Exit Periodic Inquiry Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Exit_Periodic_Inquiry_Mode | 0x0004 | | Status |

**Description**:

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.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Exit Periodic Inquiry Mode command succeeded. |
| 0x01-0xFF | Exit Periodic Inquiry Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

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.

### 7.1.5 Create Connection Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Create_Connection | 0x0005 | BD_ADDR,<br>Packet_Type,<br>Page_Scan_Repetition_Mode,<br>Reserved,<br>Clock_Offset,<br>Allow_Role_Switch | |

**Description:**

This command will cause the Link Manager to create a connection to the Bluetooth device with the BD_ADDR specified by the command parameters. This command causes the local Bluetooth device to begin the Page process to create a link level connection. The Link Manager 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 Link Manager shall use for the ACL connection. When sending HCI ACL Data Packets the Link Manager shall only use the packet type(s) specified by the 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 bit-wise OR operation of the different packet types. The Link Manager 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 below). 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). For a definition of the different packet types see the “Baseband Specification” on page 53 [Part B].

Note: The Host should enable as many packet types as possible for the Link Manager to perform efficiently. However, the Host must not enable packet types that the local device does not support.

### Command Parameters:

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to be connected. |

*Packet_Type:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0x0001 | Reserved for future use. |
| 0x0002 | 2-DH1 may not be used. |
| 0x0004 | 3-DH1 may not be used. |
| 0x0008[1] | DM1 may be used. |
| 0x0010 | DH1 may be used. |
| 0x0020 | Reserved for future use. |
| 0x0040 | Reserved for future use. |
| 0x0080 | Reserved for future use. |
| 0x0100 | 2-DH3 may not be used. |
| 0x0200 | 3-DH3 may not be used. |
| 0x0400 | DM3 may be used. |
| 0x0800 | DH3 may be used. |
| 0x1000 | 2-DH5 may not be used. |
| 0x2000 | 3-DH5 may not be used. |
| 0x4000 | DM5 may be used. |
| 0x8000 | DH5 may be used. |

1. This bit will be interpreted as set to 1 by Bluetooth V1.2 or later controllers.

*Page_Scan_Repetition_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| 0x03 – 0xFF | Reserved. |

*Reserved:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved, must be set to 0x00.<br>See "Page Scan Mode" on page 677. |

*Clock_Offset:* *Size: 2 Octets*

| Bit format | Parameter Description |
|---|---|
| Bit 14-0 | Bit 16-2 of CLKslave-CLKmaster. |
| Bit 15 | Clock_Offset_Valid_Flag<br>Invalid Clock Offset = 0<br>Valid Clock Offset = 1 |

*Allow_Role_Switch:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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 Bluetooth 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.

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.

### 7.1.6 Disconnect Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Disconnect | 0x0006 | Connection_Handle, Reason | |

**Description:**

The Disconnection 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 Bluetooth 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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for the connection being disconnected.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Reason:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x05, 0x13-0x15, 0x1A, 0x29 | Authentication Failure error code (0x05), Other End Terminated Connection error codes (0x13-0x15), Unsupported Remote Feature error code (0x1A) and Pairing with Unit Key Not Supported error code (0x29), see "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 596 of 1420
IPR2025-00103

### 7.1.7 Create Connection Cancel Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Create_Connection _Cancel | 0x0008 | BD_ADDR | Status, BD_ADDR |

**Description:**

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.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Create Connection command request that was issued before and is subject of this cancellation request |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Create Connection Cancel command succeeded |
| 0x01-0xff | Create Connection Cancel command failed. See “Error Codes” on page 331 [Part D] for list of error codes |

*BD_ADDR:* *Size: 6 Octet*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Create Connection command that was issued before and is the subject of this cancellation request. |



LifeScan, Inc. Ex. 1019
Page 597 of 1420
IPR2025-00103

**Event(s) generated (unless masked away):**

When the Create Connection Cancel command has completed, a Command Complete event shall be generated.

If the connection is already established by the baseband, 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 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).



LifeScan, Inc. Ex. 1019
Page 598 of 1420
IPR2025-00103

### 7.1.8 Accept Connection Request Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Accept_Connection Request | 0x0009 | BD_ADDR,<br>Role | |

**Description:**

The Accept_Connection_Request command is used to accept a new incoming connection request. The Accept_Connection_Request command shall only be issued after a Connection Request event has occurred. The Connection Request event will return the BD_ADDR of the device which is requesting the connection. This command will cause the Link Manager to create a connection to the Bluetooth device, with the BD_ADDR specified by the command parameters. The Link Manager 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 Link Manager 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.

Note: The Link Manager 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 Bluetooth Module.

Note: When accepting synchronous connection request, the Role parameter is not used and will be ignored by the Controller.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to be connected |

*Role:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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. |

**Return Parameters**:

None.

**Event(s) generated (unless masked away):**

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 Link Manager 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.



LifeScan, Inc. Ex. 1019
Page 600 of 1420
IPR2025-00103

### 7.1.9 Reject Connection Request Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reject_Connection_Request | 0x000A | BD_ADDR, Reason | |

**Description:**

The Reject_Connection_Request command is used to decline a new incoming connection request. The Reject_Connection_Request command shall only be called 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.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device to reject the connection from. |

*Reason:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x0D-0x0F | Host Reject Error Code. See “Error Codes” on page 331 [Part D] for list of Error Codes and descriptions. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.

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.

### 7.1.10 Link Key Request Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Link_Key_Request_Reply | 0x000B | BD_ADDR, Link_Key | Status, BD_ADDR |

**Description:**

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 Bluetooth 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 Link Manager to respond to the request from the remote Link Manager (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 Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 203 [Part C].)

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of which the Link Key is for. |

*Link_Key:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX | Link Key for the associated BD_ADDR. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Link_Key_Request_Reply command succeeded. |
| 0x01-0xFF | Link_Key_Request_Reply command failed. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of which the Link Key request reply has completed. |

**Event(s) generated (unless masked away):**

The Link_Key_Request_Reply command will cause a Command Complete event to be generated.



LifeScan, Inc. Ex. 1019
Page 603 of 1420
IPR2025-00103

### 7.1.11 Link Key Request Negative Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Link_Key_Request_Negative_Reply | 0x000C | BD_ADDR | Status, BD_ADDR |

**Description:**

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 Bluetooth 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 Link Manager to respond to the request from the remote Link Manager (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 Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 203 [Part C].)

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which the Link Key is for. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Link_Key_Request_Negative_Reply command succeeded. |
| 0x01-0xFF | Link_Key_Request_Negative_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which the Link Key request negative reply has completed. |

**Event(s) generated (unless masked away):**

The Link_Key_Request_Negative_Reply command will cause a Command Complete event to be generated.

### 7.1.12 PIN Code Request Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_PIN_Code_Request_Reply | 0x000D | BD_ADDR,<br>PIN_Code_Length,<br>PIN_Code | Status, BD_ADDR |

**Description:**

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 Link Manager to respond to the request from the remote Link Manager (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 Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 203 [Part C].)

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which the PIN code is for. |

*PIN_Code_Length:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The PIN code length specifics the length, in octets, of the PIN code to be used.<br>Range: 0x01-0x10 |

*PIN_Code:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | 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 therefore not apply to the PIN_Code Parameter. The first octet of the PIN code should be transmitted first. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | PIN_Code_Request_Reply command succeeded. |
| 0x01-0xFF | PIN_Code_Request_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which the PIN Code request reply has completed. |

**Event(s) generated (unless masked away):**

The PIN_Code_Request_Reply command will cause a Command Complete event to be generated.



LifeScan, Inc. Ex. 1019
Page 606 of 1420
IPR2025-00103

### 7.1.13 PIN Code Request Negative Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_PIN_Code_ Request_Negative Reply | 0x000E | BD_ADDR | Status, BD_ADDR |

**Description:**

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 Link Manager to respond to the request from the remote Link Manager (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 Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 203 [Part C].)

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which this command is responding to. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | PIN_Code_Request_Negative_Reply command succeeded. |
| 0x01-0xFF | PIN_Code_Request_Negative_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX XXXX | BD_ADDR of the Device which the PIN Code request negative reply has completed. |

**Event(s) generated (unless masked away):**

The PIN_Code_Request_Negative_Reply command will cause a Command Complete event to be generated.



LifeScan, Inc. Ex. 1019
Page 607 of 1420
IPR2025-00103

### 7.1.14 Change Connection Packet Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Change_Connection_ Packet_Type | 0x000F | Connection_Handle, Packet_Type | |

**Description:**

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 Link Manager can use for the connection. When sending HCI ACL Data Packets the Link Manager shall only use 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. For a definition of the different packet types see the "Baseband Specification" on page 53 [Part B].

Note: The Host should enable as many packet types as possible for the Link Manager to perform efficiently. However, the Host must not enable packet types that the local device does not support.

Note: to change an eSCO connection, use the Setup Synchronous Connection command.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to for transmitting and receiving voice or data. Returned from creating a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 608 of 1420
IPR2025-00103

*Packet_Type:* *Size: 2 Octets*

*For ACL Link_Type*

| Value | Parameter Description |
|---|---|
| 0x0001 | Reserved for future use. |
| 0x0002 | 2-DH1 may not be used. |
| 0x0004 | 3-DH1 may not be used. |
| 0x0008[1] | DM1 may be used. |
| 0x0010 | DH1 may be used. |
| 0x0020 | Reserved for future use. |
| 0x0040 | Reserved for future use. |
| 0x0080 | Reserved for future use. |
| 0x0100 | 2-DH3 may not be used. |
| 0x0200 | 3-DH3 may not be used. |
| 0x0400 | DM3 may be used. |
| 0x0800 | DH3 may be used. |
| 0x1000 | 2-DH5 may not be used. |
| 0x2000 | 3-DH5 may not be used. |
| 0x4000 | DM5 may be used. |
| 0x8000 | DH5 may be used. |

1. This bit will be interpreted as set to 1 by Bluetooth V1.2 or later controllers.



LifeScan, Inc. Ex. 1019
Page 609 of 1420
IPR2025-00103

*For SCO Link Type*

| Value | Parameter Description |
|---|---|
| 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. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Change Connection Packet Type command, the Controller sends the Command Status event to the Host. In addition, when the Link Manager 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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 610 of 1420
IPR2025-00103

### 7.1.15 Authentication Requested Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Authentication_ Requested | 0x0011 | Connection_Handle | |

**Description:**

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 Link Manager shall not automatically detach the link. The Host is responsible for issuing a Disconnect command to terminate the link if the action is appropriate.

Note: the Connection_Handle command parameter is used to identify the other Bluetooth device, which forms the connection. The Connection Handle should be a Connection Handle for an ACL connection.

**Command Parameters:**

*Connection_Handle:* *Size 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to set up authentication for all Connection Handles with the same Bluetooth device end-point as the specified Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Authentication_Requested command, it sends the Command Status event to the Host. The Link Key Request event shall be generated when Simple Pairing Mode is enabled. If authentication fails and Simple Pairing Mode is not enabled, the PIN Code Request event may be generated. The Authentication Complete event will occur when the authentication has been completed for the connection and is indication that this command has been completed.

Note: 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.

On receipt of an Authentication Requested Command, a local Controller that is not in Simple Pairing Mode may use an existing stored link key and respond immediately to the Host with an Authentication Complete event.

This behavior is not recommended for controllers as it offers the Host no method for enhancing the security of an existing link (e.g., in the case where a profile mandating a minimum passkey length is started over a link that is already authenticated with shorter passkey than the new service requires).



LifeScan, Inc. Ex. 1019
Page 612 of 1420
IPR2025-00103

### 7.1.16 Set Connection Encryption Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Connection_ Encryption | 0x0013 | Connection_Handle, Encryption_Enable | |

**Description:**

The Set_Connection_Encryption command is used to enable and disable the link level encryption. Note**:** the Connection_Handle command parameter is used to identify the other Bluetooth device which 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.

**Command Parameters:**

*Connection_Handle:* *Size 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to enable/disable the link layer encryption for all Connection Handles with the same Bluetooth device end-point as the specified Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Encryption_Enable:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Turn Link Level Encryption OFF. |
| 0x01 | Turn Link Level Encryption ON. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Set_Connection_Encryption command, the Controller sends the Command Status event to the Host. When the Link Manager has completed enabling/disabling encryption for the connection, the Controller on the local Bluetooth device will send an Encryption Change event to the Host, and the Controller on the remote device will also generate an Encryption Change event.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 613 of 1420
IPR2025-00103

### 7.1.17 Change Connection Link Key Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Change_Connection_ Link_Key | 0x0015 | Connection_Handle | |

**Description:**

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.

Note: the Connection_Handle command parameter is used to identify the other Bluetooth device forming the connection. The Connection Handle should be a Connection Handle for an ACL connection.

**Command Parameters:**

*Connection_Handle:* *Size 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Change_Connection_Link_Key command, the Controller sends the Command Status event to the Host. When the Link Manager has changed the Link Key for the connection, the Controller on the local Bluetooth 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.

Note: no 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.



LifeScan, Inc. Ex. 1019
Page 614 of 1420
IPR2025-00103

### 7.1.18 Master Link Key Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Master_Link_Key | 0x0017 | Key_Flag | |

**Description:**

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.

**Command Parameters:**

*Key_Flag:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Use semi-permanent Link Keys. |
| 0x01 | Use Temporary Link Key. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Master_Link_Key command, the Controller sends the Command Status event to the Host. When the Link Manager 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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 615 of 1420
IPR2025-00103

### 7.1.19 Remote Name Request Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_Name_Request | 0x0019 | BD_ADDR,<br>Page_Scan_Repetition_Mode,<br>Reserved,<br>Clock_Offset | |

**Description:**

The Remote_Name_Request command is used to obtain the user-friendly name of another Bluetooth device. The user-friendly name is used to enable the user to distinguish one Bluetooth 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.

When the Remote Supported Host Features Notification event is unmasked and when the Remote_Name_Request command initiates a connection, the Link Manager shall read the remote LMP features mask pages 0 and 1.

Note: 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 LMP features and name of the remote device.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the device whose name is requested. |

*Page_Scan_Repetition_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |



LifeScan, Inc. Ex. 1019
Page 616 of 1420
IPR2025-00103

| | |
|---|---|
| 0x03 – 0xFF | Reserved. |

*Reserved:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reserved, must be set to 0x00.<br>See "Page Scan Mode" on page 677. |

*Clock_Offset:* *Size: 2 Octets*

| Bit format | Parameter Description |
|---|---|
| Bit 14.0 | Bit 16.2 of CLKslave-CLKmaster. |
| Bit 15 | Clock_Offset_Valid_Flag<br>Invalid Clock Offset = 0<br>Valid Clock Offset = 1 |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Remote_Name_Request command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to obtain the remote host supported features, if present, the Controller on the local Bluetooth device will send a Remote Host Supported Features Notification event. When the Link Manager has completed the LMP messages to obtain the remote name, the Controller on the local Bluetooth device will send a Remote Name Request Complete event to the Host. If the remote host supported features page is present, the Remote Host Supported Features Notification event will be sent before the Remote Name Request Complete event. If not, only the Remote Name Request Complete event will be sent.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 617 of 1420
IPR2025-00103

### 7.1.20 Remote Name Request Cancel Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_Name_Request_Cancel | 0x001A | BD_ADDR | Status, BD_ADDR |

**Description:**

This command is used to request cancellation of the ongoing remote name request process, which was started by the Remote Name Request command.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Remote Name Request command that was issued before and that is subject of this cancellation request |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Remote Name Request Cancel command succeeded |
| 0x01-0xff | Remote Name Request Cancel command failed. See “Error Codes” on page 331 [Part D] for list of error codes |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Remote Name Request Cancel command that was issued before and that was subject of this cancellation request |

**Event(s) generated (unless masked away):**

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 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)*.

### 7.1.21 Read Remote Supported Features Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Remote_Supported_Features | 0x001B | Connection_Handle | |

**Description:**

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. For details see "Link Manager Protocol" on page 203 [Part C].

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's LMP-supported features list to get.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Read_Remote_Supported_Features command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to determine the remote features, the Controller on the local Bluetooth 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. Note: 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.

### 7.1.22 Read Remote Extended Features Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Remote_Extended_Features | 0x001C | Connection_Handle, Page Number | |

**Description:**

The HCI_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 only available if the extended features feature is implemented by the remote device. The Read Remote Extended Features Complete event will return the requested information. For details see "Link Manager Protocol" on page 203 [Part C].

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle identifying the remote device for which extended feature information is required. Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use) |

*Page Number:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Requests the normal LMP features as returned by HCI_Read_Remote_Supported_Features |
| 0x01-0xFF | Return the corresponding page of features |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Read_Remote_Extended_Features command the Controller sends the Command Status command to the Host. When the Link Manager has completed the LMP sequence to determine the remote extended features the controller on the local device will generate a Read Remote Extended Features 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.



LifeScan, Inc. Ex. 1019
Page 620 of 1420
IPR2025-00103

Note: no Command Complete event will ever 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.

### 7.1.23 Read Remote Version Information Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Remote_Version_ Information | 0x001D | Connection_Handle | |

**Description:**

This command will obtain the values for the version information for the remote Bluetooth device identified by the Connection_Handle parameter. The Connection_Handle must be a Connection_Handle for an ACL connection.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's version information to get.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Read_Remote_Version_Information command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to determine the remote version information, the Controller on the local Bluetooth 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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 621 of 1420
IPR2025-00103

### 7.1.24 Read Clock Offset Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Clock_Offset | 0x001F | Connection_Handle | |

**Description**:

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, for example, when the local Host has issued Create_Connection or Remote_Name_Request. The Connection_Handle must be a Connection_Handle for an ACL connection.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Clock Offset parameter is returned.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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 Link Manager has completed the LMP messages to obtain the Clock Offset information, the Controller on the local Bluetooth device will send a Read Clock Offset Complete event to the Host.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 622 of 1420
IPR2025-00103

### 7.1.25 Read LMP Handle Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_LMP_Handle | 0x0020 | Connection_Handle | Status, Connection_Handle, LMP_Handle, Reserved |

**Description:**

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.

**Command Parameters:**

*Connection_Handle:* *Size 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify which connection to be used for reading the LMP Handle. This must be a synchronous handle.<br>Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_LMP_Handle command succeeded. |
| 0x01 – 0xFF | Read_LMP_Handle command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the LMP_Handle has been read.<br>Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 623 of 1420
IPR2025-00103

*LMP_Handle:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The LMP Handle is the LMP Handle that is associated with this Connection Handle.<br>For a synchronous handle, this would be the LMP Synchronous Handle used when negotiating the synchronous connection in the link manager. |

*Reserved:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | This parameter is reserved, must to set to zero. |

**Events(s) generated (unless masked away):**

When the Read_LMP_Handle command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 624 of 1420
IPR2025-00103

### 7.1.26 Setup Synchronous Connection Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Setup_Synchronous_ Connection | 0x0028 | Connection_Handle<br>Transmit_Bandwidth<br>Receive_Bandwidth<br>Max_Latency<br>Voice_Setting<br>Retransmission_Effort<br>Packet Type | |

**Description:**

The HCI 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 64kbps or transparent synchronous data.

When used to setup 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 transmit and receive bandwidth specify how much bandwidth shall be available for transmitting and for receiving data. While in many cases the receive and transmit 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 don't care.

When used to modify an existing synchronous logical transport, the Transmit_Bandwidth, Receive_Bandwidth and Voice_Settings 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 field is a bitmap 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.

Note: The link manager 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 link manager may choose to set up either a SCO or an eSCO connection, if the parameters allow, using the corresponding LMP sequences.

Note: To modify a SCO connection, use the Change Connection Packet Type command.

Note: If the lower layers cannot achieve the exact transmit and receive bandwidth requested subject to the other parameters, then the link shall be rejected.

A synchronous connection may only be created when an ACL connection already exists and when it is not in Park state.

**Command Parameters:**

*Connection_Handle:* *2 octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for the ACL connection being used to create a synchronous Connection or for the existing Connection that shall be modified.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Transmit_Bandwidth:* *4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Transmit bandwidth in octets per second. |

*Receive_Bandwidth:* *4 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Receive bandwidth in octets per second. |



LifeScan, Inc. Ex. 1019
Page 626 of 1420
IPR2025-00103

*Max_Latency:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000-0x0003 | Reserved |
| 0x0004-0xFFFE | This is a value in milliseconds representing the upper limit of the sum of the synchronous interval, the size of the eSCO window. (See Figure 8.7 in the Baseband specification) |
| 0xFFFF | Don't care. |

*Voice_Setting:* *2 octets (10 bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12 on page 403. | |

*Retransmission_Effort:* *1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No retransmissions |
| 0x01 | At least one retransmission, optimize for power consumption. |
| 0x02 | At least one retransmission, optimize for link quality |
| 0xFF | Don’t care |
| 0x03 – 0xFE | Reserved |



LifeScan, Inc. Ex. 1019
Page 627 of 1420
IPR2025-00103

*Packet_Type:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 | HV1 may be used. |
| 0x0002 | HV2 may be used. |
| 0x0004 | HV3 may be used. |
| 0x0008 | EV3 may be used. |
| 0x0010 | EV4 may be used. |
| 0x0020 | EV5 may be used. |
| 0x0040 | 2-EV3 may not be used. |
| 0x0080 | 3-EV3 may not be used. |
| 0x0100 | 2-EV5 may not be used. |
| 0x0200 | 3-EV5 may not be used. |
| 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 |

**Return Parameters:**

None

**Event(s) generated (unless masked away)**

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 Setup Complete Event or Connection Request Event will be sent to either host. This command cannot be used to change the parameters of an SCO link.

Note: 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.

### 7.1.27 Accept Synchronous Connection Request Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Accept_ Synchronous_ Connection_Request | 0x0029 | BD_ADDR<br>Transmit_Bandwidth<br>Receive_Bandwidth<br>Max_Latency<br>Content_Format<br>Retransmission_Effort<br>Packet_Type | |

**Description:**

The Accept_Synchronous_Connection_Request command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection. The Command shall only be issued after a Connection_Request event with link type SCO or eSCO has occurred. 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 values are required values for the new link and shall be met. The Max_Latency is an upper bound to the acceptable latency for the Link, as defined in Section 7.1.26 on page 455 Setup_Synchronous_Connection and shall not be exceeded. Content_Format 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 don't 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 field shall be set to one. If all bits are set in the packet type field then all packets types shall be allowed.

If the Link Type of the incoming request is SCO, then only the Transmit_Bandwidth, Max_Latency, Content_Format, and Packet_Type fields are valid.

If the Connection_Request event is masked away, and the Controller is not set to auto-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.

**Command Parameters:**

*BD_ADDR:* *6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device requesting the connection |

*Transmit_Bandwidth:* *4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000-0xFFFFFFFE | Maximum possible transmit bandwidth in octets per second. |
| 0xFFFFFFFF | Don't care |

Default: Don't care

*Receive_Bandwidth:* *4 octets*

| Value | Parameter Description |
|---|---|
| 0x00000000-0xFFFFFFFE | Maximum possible receive bandwidth in octets per second. |
| 0xFFFFFFFF | Don't care |

Default: Don't care

*Max_Latency:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0x0000-0x0003 | Reserved |
| 0x0004-0xFFFE | This is a value in milliseconds representing the upper limit of the sum of the synchronous interval and the size of the eSCO window. |
| 0xFFFF | Don't care. |

Default: Don't care



LifeScan, Inc. Ex. 1019
Page 630 of 1420
IPR2025-00103

*Content_Format:* *2 octets (10 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 00XXXXXXXX | Input Coding: Linear |
| 01XXXXXXXX | Input Coding: u-law |
| 10XXXXXXXX | Input Coding: A-law |
| 11XXXXXXXX | Reserved for future use. |
| XX00XXXXXX | Input Data Format: 1's complement |
| XX01XXXXXX | Input Data Format: 2's 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: u-law |
| XXXXXXXX10 | Air Coding Format: A-law |
| XXXXXXXX11 | Air Coding Format: Transparent Data |

Default: When links are auto-accepted, the values written by the HCI_Write_Voice_Settings are used.

*Retransmission_Effort:* *1 octet*

| Value | Parameter Description |
|---|---|
| 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 | Don't care |

Default: Don't care



LifeScan, Inc. Ex. 1019
Page 631 of 1420
IPR2025-00103

*Packet_Type:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0x0001 | HV1 may be used. |
| 0x0002 | HV2 may be used. |
| 0x0004 | HV3 may be used. |
| 0x0008 | EV3 may be used. |
| 0x0010 | EV4 may be used. |
| 0x0020 | EV5 may be used. |
| 0x0040 | 2-EV3 may not be used. |
| 0x0080 | 3-EV3 may not be used |
| 0x0100 | 2-EV5 may not be used. |
| 0x0200 | 3-EV5 may not be used. |
| 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 - means all packet types may be used.

**Return Parameters:**

None

**Event(s) generated (unless masked away):**

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.



LifeScan, Inc. Ex. 1019
Page 632 of 1420
IPR2025-00103

### 7.1.28 Reject Synchronous Connection Request Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reject_Synchronous_ Connection_Request | 0x002A | BD_ADDR<br>Reason | |

**Description:**

The Reject_Synchronous_Connection_Request is used to decline an incoming request for a synchronous link. It shall only be issued 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.

**Command Parameters:**

*BD_ADDR:* *6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device requesting the connection |

*Reason:* *1 octet*

| Value | Parameter Description |
|---|---|
| 0x0D-0x0F | Host Reject Error Code. See “Error Codes” on page 331 [Part D] for error codes and descriptions. |

**Return Parameters:**

None.

**Event(s) Generated (unless masked away):**

When the Host Controller receives the Reject_Synchronous_Connection_Request, 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.

### 7.1.29 IO Capability Request Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_IO_Capability_ Request_Reply | 0x002B | BD_ADDR, IO_Capability, OOB_Data_Present, Authentication_ Requirements | Status, BD_ADDR |

**Description:**

The IO_Capability_Request_Reply command is used to reply to an IO Capability Request event from the controller, and specifies the current I/O capabilities of the host. This includes the host input, output and out-of-band (OOB) capabilities.

The Authentication_Requirements parameter shall be set to man-in-the middle (MITM) Protection Required Single Profile, MITM Protection Required – General Bonding, or MITM Protection Required – Dedicated Bonding if an authenticated link key is required by the Host. The Authentication_Requirements parameter may be set to MITM Protection Not Required - Single Profile, MITM Protection Not Required – General Bonding, or MITM Protection Not Required – Dedicated Bonding if an authenticated link key is not required. If one or both Hosts set the Authentication_Requirements parameter to MITM Protection Required - Single Profile, MITM Protection Required – General Bonding, or MITM Protection Required – Dedicated Bonding, the Link Managers shall use the IO_Capability parameter to determine the authentication procedure. A Host that sets the Authentication_Requirements parameter to MITM Protection Required - Single Profile, MITM Protection Required – General Bonding, or MITM Protection Required – All Profiles shall verify that the resulting Link Key type meets the security requirements.

If both Hosts set the Authentication_Requirements parameter to MITM Protection Not Required - Single Profile, MITM Protection Not Required – General Bonding, or MITM Protection Not Required – Dedicated Bonding, the Link Managers shall use the numeric comparison authentication procedure and the Hosts shall use the Just Works Association Model.

The OOB_Data_Present parameter shall be set to "OOB authentication data from remote device present" if the host has received OOB data from a device with the same BD_ADDR sent in the IO Capability Request event. Otherwise OOB_Data_Present shall be set to "OOB authentication data not present".

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*



LifeScan, Inc. Ex. 1019
Page 634 of 1420
IPR2025-00103

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |



LifeScan, Inc. Ex. 1019
Page 635 of 1420
IPR2025-00103

*IO_Capability:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | DisplayOnly |
| 0x01 | DisplayYesNo |
| 0x02 | KeyboardOnly |
| 0x03 | NoInputNoOutput |
| 0x04 - 0xFF | Reserved for future use |

*OOB_Data_Present:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | OOB authentication data not present |
| 0x01 | OOB authentication data from remote device present |
| 0x02 - 0xFF | Reserved for future use |

*Authentication_Requirements:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | MITM Protection Not Required – No Bonding. Numeric comparison with automatic accept allowed. |
| 0x01 | MITM Protection Required – No Bonding. Use IO Capabilities to determine authentication procedure |
| 0x02 | MITM Protection Not Required – Dedicated Bonding. Numeric comparison with automatic accept allowed. |
| 0x03 | MITM Protection Required – Dedicated Bonding. Use IO Capabilities to determine authentication procedure |
| 0x04 | MITM Protection Not Required – General Bonding. Numeric Comparison with automatic accept allowed. |
| 0x05 | MITM Protection Required – General Bonding. Use IO capabilities to determine authentication procedure. |
| 0x06 - 0xFF | Reserved for future use |

**Return Parameters***:*

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | IO_Capability_Request_Reply command succeeded |
| 0x01 - 0xFF | IO_Capability_Request_Reply command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*



LifeScan, Inc. Ex. 1019
Page 636 of 1420
IPR2025-00103

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**

When the IO_Capability_Request_Reply command has completed, a Command Complete event shall be generated. When the device is the initiator of Secure Simple Pairing, an IO Capability Response event will be generated. Additionally, when the OOB_Data_Present parameter indicates that OOB authentication data from the remote device is present, the Remote OOB Data Request event will be generated.

### 7.1.30 User Confirmation Request Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Confirmation_Request_Reply | 0x002C | BD_ADDR | Status, BD_ADDR |

**Description:**

The User_Confirmation_Request_Reply command is used to reply to a User Confirmation Request event and indicates that the user selected "yes". It is also used when the host has no input and no output capabilities.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | User_Confirmation_Request_Reply command succeeded |
| 0x01 - 0xFF | User_Confirmation_Request_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |



LifeScan, Inc. Ex. 1019
Page 637 of 1420
IPR2025-00103

**Event(s) generated (unless masked away):**

When the User_Confirmation_Request_Reply command has completed, a Command Complete event shall be generated.

### 7.1.31 User Confirmation Request Negative Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Confirmation_ Request_Negative_Reply | 0x002D | BD_ADDR | Status<br>BD_ADDR |

**Description:**

The User_Confirmation_Request_Negative_Reply command is used to reply to a User Confirmation Request event and indicates that the user selected "no". This command shall cause the initiating Link Manager to transmit an LMP_Numeric_Comparison_Failed PDU and terminate Secure Simple Pairing.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | User_Confirmation_Request_Negative_Reply command succeeded |
| 0x01 - 0xFF | User_Confirmation_Request_Negative_Reply command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**

When the User_Confirmation_Request_Negative_Reply command has completed, a Command Complete event shall be generated.

### 7.1.32 User Passkey Request Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Passkey_Request_Reply | 0x002E | BD_ADDR, Numeric_Value | Status, BD_ADDR |

**Description:**

The User_Passkey_Request_Reply command is used to reply to a User Passkey Request event and specifies the Numeric_Value (passkey) entered by the user to be used in the Secure Simple Pairing process.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

*Numeric_Value:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 - 0x000F423F | Numeric value (passkey) entered by user. Valid values are decimal 000000 - 999999. |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | User_Passkey_Request_Reply command succeeded |
| 0x01 - 0xFF | User_Passkey_Request_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX X | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**

When the User_Passkey_Request_Reply command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 639 of 1420
IPR2025-00103

### 7.1.33 User Passkey Request Negative Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_User_Passkey_Request Negative_Reply | 0x002F | BD_ADDR | Status, BD_ADDR |

**Description:**

The User_Passkey_Request_Negative_Reply command is used to reply to a User Passkey Request event and indicates the host could not provide a pass-key. This command shall cause the initiating Link Manager to transmit an LMP_Passkey_Entry_Failed PDU and terminate Simple Pairing.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | User_Passkey_Request_Negative_Reply command succeeded |
| 0x01 - 0xFF | User_Passkey_Request_Negative_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**

When the User_Passkey_Negative_Request_Reply command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 640 of 1420
IPR2025-00103

### 7.1.34 Remote OOB Data Request Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_OOB_Data_Request_Reply | 0x0030 | BD_ADDR, C, R | Status, BD_ADDR |

**Description:**

The Remote_OOB_Data_Request_Reply command is used to reply to a Remote OOB Data Request event with the C and R values received via an OOB transfer from a remote device identified by BD_ADDR. If the R value is not present in the received OOB data from the remote device, the Host shall set R to zeros.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device from which the C and R values were received |

*C:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Simple Pairing Hash C |

*R:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX | Simple Pairing Randomizer R |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Remote_OOB_Data_Request_Reply command succeeded |
| 0x01 - 0xFF | Remote_OOB_Data_Request_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXX<br>X | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**

When the Remote_OOB_Data_Request_Reply command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 642 of 1420
IPR2025-00103

### 7.1.35 Remote OOB Data Request Negative Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Remote_OOB_Data_ Request_Negative_Reply | 0x0033 | BD_ADDR | Status, BD_ADDR |

**Description:**

The Remote_OOB_Data_Request_Negative_Reply command is used to reply to a Remote OOB Data Request event that the Host does not have the C and R values associated with the remote device identified by BD_ADDR.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Remote_OOB_Data_Request_Negative_Reply command succeeded |
| 0x01 - 0xFF | Remote_OOB_Data_Request_Negative_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**

When the Remote_OOB_Data_Request_Negative_Reply command has completed, a Command Complete event shall be generated.

### 7.1.36 IO Capability Request Negative Reply Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_IO_Capability Request_Negative_Reply | 0x0034 | BD_ADDR<br>Reason | Status,<br>BD_ADDR |

**Description:**

The IO_Capability_Request_Negative_Reply command shall be used to reject a pairing attempt after an HCI IO Capability Request event has been received by the Host. The reason for the rejection is given in the Reason parameter. Error code 0x37 (Simple Pairing not Supported by Host) shall not be used in the Reason parameter.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in the Simple Pairing process |

*Reason:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason that Simple Pairing rejected. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | IO_Capability_Request_Negative_Reply command succeeded. |
| 0x01 - 0xFF | IO_Capability_Request_Negative_Reply command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in Simple Pairing process |

**Event(s) generated (unless masked away):**

When the IO_Capability_Request_Negative_Reply command has completed, a Command Complete event shall be generated.

## 7.2 LINK POLICY COMMANDS

The Link Policy Commands provide methods for the Host to affect how the Link Manager manages the piconet. When Link Policy Commands are used, the LM still controls how Bluetooth piconets and scatternets are established and maintained, depending on adjustable policy parameters. These policy commands modify the Link Manager behavior that can result in changes to the link layer connections with Bluetooth remote devices.

Note: only one ACL connection can exist between two Bluetooth Devices, and therefore there can only be one ACL HCI Connection Handle for each physical link layer Connection. The Bluetooth Controller provides policy adjustment mechanisms to provide support for a number of different policies. This capability allows one Bluetooth module to be used to support many different usage models, and the same Bluetooth module can be incorporated in many different types of Bluetooth devices. For the Link Policy Commands, the OGF is defined as 0x02.

### 7.2.1 Hold Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Hold_Mode | 0x0001 | Connection_Handle,<br>Hold_Mode_Max_Interval,<br>Hold_Mode_Min_Interval | |

**Description:**

The Hold_Mode command is used to alter the behavior of the Link Manager, and have it place the ACL baseband connection associated by the specified Connection Handle into the 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 the hold mode. The local and remote devices will negotiate the length in the 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 the 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 the hold mode after the negotiation with the remote device. Therefore the Hold_Mode_Min_Interval cannot be greater than the Hold_Mode_Max_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 Bluetooth devices, and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.

Note: 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.

Note: 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 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.

The Hold_Mode_Max_Interval shall be less than the Link Supervision Timeout configuration parameter.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Hold_Mode_Max_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum acceptable number of Baseband slots to wait in Hold Mode.<br>Time Length of the Hold = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0002-0xFFFE; only even values are valid.<br>Time Range: 1.25ms - 40.9 sec<br>Mandatory Range: 0x0014 to 0x8000 |

*Hold_Mode_Min_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum acceptable number of Baseband slots to wait in Hold Mode.<br>Time Length of the Hold = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0002-0xFF00; only even values are valid<br>Time Range: 1.25 msec - 40.9 sec<br>Mandatory Range: 0x0014 to 0x8000 |

**Return Parameters:**

None.



LifeScan, Inc. Ex. 1019
Page 646 of 1420
IPR2025-00103

**Event(s) generated (unless masked away):**

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 Bluetooth device.

Note: 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.

### 7.2.2 Sniff Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Sniff_Mode | 0x0003 | Connection_Handle,<br>Sniff_Max_Interval,<br>Sniff_Min_Interval,<br>Sniff_Attempt,<br>Sniff_Timeout | |

**Description:**

The Sniff Mode command is used to alter the behavior of the Link Manager and have it place the ACL baseband connection associated with the specified Connection Handle into the 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 the Sniff Mode. The Sniff_Min_Interval shall not be greater than the Sniff_Max_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 parameters, see Baseband Specification, Section 8.7, on page 175. 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 Bluetooth devices, and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.

Note: 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



LifeScan, Inc. Ex. 1019
Page 647 of 1420
IPR2025-00103

the sniff mode through a Mode Change event when it sends the data. Note that it is possible for the master to transmit data to a slave without exiting sniff mode (see description in Baseband Specification, Section 8.7, on page 175).

Note: 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 broadcast and the Broadcast_Flag is set to Active Broadcast or Piconet Broadcast. In that case, the broadcast data will only be received by a slave in sniff mode if that slave happens to listen to the master when the broadcast is made.

The Sniff_Max_Interval shall be less than the Link Supervision Timeout configuration parameter.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Sniff_Max_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0002 to 0xFFFE; only even values are valid<br>Mandatory Range: 0x0006 to 0x0540<br>Time = N * 0.625 msec<br>Time Range: 1.25 msec to 40.9 sec |

*Sniff_Min_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x0002 to 0xFFFE; only even values are valid<br>Mandatory Range: 0x0006 to 0x0540<br>Time = N * 0.625 msec<br>Time Range: 1.25 msec to 40.9 sec) |

*Sniff_Attempt:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Number of Baseband receive slots for sniff attempt.<br>Length = N* 1.25 msec<br>Range for N: 0x0001 – 0x7FFF<br>Time Range: 0.625msec - 40.9 Seconds<br>Mandatory Range for Controller: 1 to $T_{sniff}/2$ |

*Sniff_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Number of Baseband receive slots for sniff timeout.<br>Length = N * 1.25 msec<br>Range for N: 0x0000 – 0x7FFF<br>Time Range: 0 msec - 40.9 Seconds<br>Mandatory Range for Controller: 0 to 0x0028 |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 649 of 1420
IPR2025-00103

### 7.2.3 Exit Sniff Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Exit_Sniff_Mode | 0x0004 | Connection_Handle | |

**Description:**

The Exit_Sniff_Mode command is used to end the sniff mode for a Connection Handle, which is currently in sniff mode. The Link Manager will determine and issue the appropriate LMP commands to remove the sniff mode for the associated Connection Handle.

Note: in addition, the Connection Handle cannot be one of the synchronous link types.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

A Command Status event for this command will occur when 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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 650 of 1420
IPR2025-00103

### 7.2.4 Park State Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Park_State | 0x0005 | Connection_Handle,<br>Beacon_Max_Interval,<br>Beacon_Min_Interval | |

**Description:**

The Park State command is used to alter the behavior of the Link Manager, and have the LM place the baseband 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 Min Interval cannot be greater than the Beacon Max 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 Bluetooth devices, allows the devices to enter Inquiry Scan, Page Scan, provides support for large number of Bluetooth Devices in a single piconet, and a number of other possible activities.

Note: 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 0x0C *Command Disallowed*.

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.

Note: 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) and depending on whether the implementation supports this or not (optional feature). The optional feature of automatic unpark/park can also be used for link supervision. Whether Mode Change events are returned or not at automatic unpark/park if this is implemented, is vendor specific. This could be controlled by a vendor specific HCI command.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Beacon_Max_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x000E to 0xFFFE; only even values are valid<br>Mandatory Range: 0x000E to 0x1000<br>Time = N * 0.625 msec<br>Time Range: 8.75 msec to 40.9 sec |

*Beacon_Min_Interval* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Range: 0x000E to 0xFFFE; only even values are valid<br>Mandatory Range: 0x000E to 0x1000<br>Time = N * 0.625 msec<br>Time Range: 8.75 msec to 40.9 sec |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 652 of 1420
IPR2025-00103

### 7.2.5 Exit Park State Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Exit_Park_State | 0x0006 | Connection_Handle | |

**Description:**

The Exit_Park_State command is used to switch the Bluetooth device from park state back to the active mode. This command may only be issued 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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.

Note: 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.

### 7.2.6 QoS Setup Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_QoS_Setup | 0x0007 | Connection_Handle,<br>Flags,<br>Service_Type,<br>Token_Rate,<br>Peak_Bandwidth,<br>Latency,<br>Delay_Variation | |

**Description:**

The QoS_Setup command is used to specify Quality of Service 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 "Logical Link Control and Adaptation Protocol Specification" on page 13[vol. 3]. This allows the Link Manager 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. Bluetooth 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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify which connection for the QoS Setup.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Flags:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 – 0xFF | Reserved for Future Use. |

*Service_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic. |
| 0x01 | Best Effort. |
| 0x02 | Guaranteed. |
| 0x03-0xFF | Reserved for Future Use. |

*Token_Rate:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Rate in octets per second. |

*Peak_Bandwidth:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Peak Bandwidth in octets per second. |

*Latency:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Latency in microseconds. |

*Delay_Variation:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Delay Variation in microseconds. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the QoS_Setup command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to establish the requested QoS parameters, the Controller on the local Bluetooth 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.



LifeScan, Inc. Ex. 1019
Page 655 of 1420
IPR2025-00103

Note: 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.



LifeScan, Inc. Ex. 1019
Page 656 of 1420
IPR2025-00103

### 7.2.7 Role Discovery Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Role_Discovery | 0x0009 | Connection_Handle | Status, Connection_Handle, Current_Role |

**Description:**

The Role_Discovery command is used for a Bluetooth 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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Role_Discovery command succeeded |
| 0x01-0xFF | Role_Discovery command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify which connection the Role_Discovery command was issued on.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Current_Role:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Current Role is Master for this Connection Handle. |
| 0x01 | Current Role is Slave for this Connection Handle. |

**Event(s) generated (unless masked away):**

When the Role_Discovery command has completed, a Command Complete event will be generated.

### 7.2.8 Switch Role Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Switch_Role | 0x000B | BD_ADDR, Role | |

**Description:**

The Switch_Role command is used for a Bluetooth device to switch the current role the device is performing for a particular connection with another specified Bluetooth device. The BD_ADDR command parameter indicates for which connection the role switch is to be performed. The Role indicates the requested new role that the local device performs.

Note: the BD_ADDR command parameter must specify a Bluetooth device for which a connection already exists.

Note: 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.

Note: 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.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the connected device with which a role switch is to be performed. |

*Role:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Change own Role to Master for this BD_ADDR. |
| 0x01 | Change own Role to Slave for this BD_ADDR. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.



LifeScan, Inc. Ex. 1019
Page 658 of 1420
IPR2025-00103

Note: 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.



LifeScan, Inc. Ex. 1019
Page 659 of 1420
IPR2025-00103

### 7.2.9 Read Link Policy Settings Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Link_Policy_Settings | 0x000C | Connection_Handle | Status,<br>Connection_Handle<br>Link_Policy_Settings |

**Description:**

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. Section 6.17 on page 406.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Link_Policy_Settings command succeeded. |
| 0x01-0xFF | Read_Link_Policy_Settings command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Link_Policy_Settings* *Size: 2 Octets*

| 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. |
| 0x0008 – 0x8000 | Reserved for Future Use. |

**Event(s) generated (unless masked away):**

When the Read_Link_Policy_Settings command has completed, a Command Complete event will be generated.

### 7.2.10 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 |

**Description:**

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 Section 6.17 on page 406.

The default value is the value set by the Write Default Link Policy Settings Command.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 661 of 1420
IPR2025-00103

*Link_Policy_Settings* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 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. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Link_Policy_Settings command succeeded. |
| 0x01-0xFF | Write_Link_Policy_Settings command failed. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Event(s) generated (unless masked away):**

When the Write_Link_Policy_Settings command has completed, a Command Complete event will be generated.

### 7.2.11 Read Default Link Policy Settings Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Default_Link_Policy_Settings | 0x000E | | Status, Default_Link_Policy_Settings |

**Description:**

This command will read the Default Link Policy setting for all new connections.

Note: Please refer to the Link Policy Settings configuration parameter for more information. See Section 6.17 on page 406.

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Link_Policy_Settings command succeeded |
| 0x01-0xFF | Read_Link_Policy_Settings command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Default_Link_Policy_Settings* *Size: 2 Octets*

| 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. |

**Event(s) generated (unless masked away):**

When the Read_Default_Link_Policy_Settings command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 663 of 1420
IPR2025-00103

### 7.2.12 Write Default Link Policy Settings Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Default_Link_Policy_Settings | 0x000F | Default_Link_Policy_Settings | Status |

**Description:**

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.

Note: Please refer to the Link Policy Settings configuration parameter for more information. See Section 6.17 on page 406.

**Command Parameters:**

*Default_Link_Policy_Settings* *Size: 2 Octets*

| 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. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Link_Policy_Settings command succeeded |
| 0x01-0xFF | Write_Link_Policy_Settings command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Default_Link_Policy_Settings command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 664 of 1420
IPR2025-00103

### 7.2.13 Flow Specification Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Flow_Specification | 0x0010 | Connection_Handle,<br>Flags,<br>Flow_direction,<br>Service_Type,<br>Token_Rate,<br>Token_Bucket_Size,<br>Peak_Bandwidth,<br>Access Latency | |

**Description:**

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 Link Manager to have the parameters of the outgoing as well as the incoming flow for the ACL connection. The flow parameters are defined in the L2CAP specification "Quality of Service (QoS) Option" on page 58[vol. 3]. The Link Manager will determine if the flow parameters can be supported. Bluetooth devices that are both master and slave can use this command.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle used to identify for which ACL connection the Flow is specified.<br>Range: 0x0000 - 0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Flags:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 – 0xFF | Reserved for Future Use. |

*Flow_direction:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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. |



LifeScan, Inc. Ex. 1019
Page 665 of 1420
IPR2025-00103

*Service_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic |
| 0x01 | Best Effort |
| 0x02 | Guaranteed |
| 0x03 – 0xFF | Reserved for Future Use |

*Token Rate:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Rate in octets per second |

*Token Bucket Size:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Bucket Size in octets |

*Peak_Bandwidth:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Peak Bandwidth in octets per second |

*Access Latency:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Latency in microseconds |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

When the Controller receives the Flow Specification command, the Controller sends the Command Status event to the Host. When the Link Manager has determined if the Flow specification can be supported, the Controller on the local Bluetooth 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.

Note: 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.



LifeScan, Inc. Ex. 1019
Page 666 of 1420
IPR2025-00103

### 7.2.14 Sniff Subrating Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Sniff_Subrating | 0x0011 | Connection Handle<br>Maximum_Latency<br>Minimum_Remote_Timeout<br>Minimum_Local_Timeout | Status<br>Connection_Handle |

**Description:**

The Sniff_Subrating command shall specify the parameters for sniff subrating for a given link. The interval shall be determined from the sniff interval and the maximum subrate latency parameters from the command. The link may have smaller subrates and therefore lower latencies and longer timeouts than those specified. When the sniff subrate has been exchanged a Sniff Subrating event shall be generated. If this command is used on a link in sniff mode this shall cause sniff subrating to be negotiated at the Link Manager, otherwise sniff subrating shall be negotiated only after the device has entered the sniff mode.

The Connection Handle shall be the primary Connection Handle between the two devices.

The Maximum Latency parameter shall define the maximum allowed sniff subrate of the remote device.

Note: If the Host does not write the sniff subrating parameters prior to sniff subrating being initiated by the Link Manager the default values shall be used.

Note: Setting both subrate values to zero is equivalent to sniff mode without subrating enabled.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000 – 0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Maximum_Latency:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The Maximum Latency parameter shall be used to calculate the maximum_sniff subrate that the remote device may use.<br>Default: 0x0000<br>Latency = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0xFFFE<br>Time Range: 0 sec – 40.9 sec |



LifeScan, Inc. Ex. 1019
Page 667 of 1420
IPR2025-00103

*Minimum_Remote_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum base sniff subrate timeout that the remote device may use<br>Default: 0x0000<br>Latency = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0xFFFE<br>Time Range: 0 sec - 40.9 sec |

*Minimum_Local_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Minimum base sniff subrate timeout that the local device may use.<br>Default: 0x0000<br>Latency = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0xFFFE<br>Time Range: 0 sec - 40.9 sec |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The Sniff Subrating command succeeded. |
| 0x01 - 0xFF | Sniff Subrating command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000 - 0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Event(s) generated (unless masked away)**

The controller sends the Command Complete event for this command to the Host when it has received the Sniff_Subrating command.

Furthermore, an _Sniff_Subrating_event shall occur when the sniff subrating has been negotiated for the specified Connection Handle.

## 7.3 CONTROLLER & BASEBAND COMMANDS

The Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware. These parameters provide control of Bluetooth devices and of the capabilities of the Controller, Link Manager, and

Baseband. The host device can use these commands to modify the behavior of the local device. For the HCI Control and Baseband Commands, the OGF is defined as 0x03.

### 7.3.1 Set Event Mask Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Event_Mask | 0x0001 | Event_Mask | Status |

**Description:**

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 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 Bluetooth devices. The event mask allows the Host to control how much it is interrupted.

**Command Parameters:**

*Event_Mask:* *Size: 8 Octets*

| Value | Parameter Description |
|---|---|
| 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 |



LifeScan, Inc. Ex. 1019
Page 669 of 1420
IPR2025-00103

| Value | Parameter Description |
|---|---|
| 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 |
| 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 |
| 0x0000200000000000 | Sniff Subrating Event |
| 0x0000400000000000 | Extended Inquiry Result Event |
| 0x0000800000000000 | Encryption Key Refresh Complete Event |
| 0x0001000000000000 | IO Capability Request Event |

| Value | Parameter Description |
|---|---|
| 0x0002000000000000 | IO Capability Request Reply Event |
| 0x0004000000000000 | User Confirmation Request Event |
| 0x0008000000000000 | User Passkey Request Event |
| 0x0010000000000000 | Remote OOB Data Request Event |
| 0x0020000000000000 | Simple Pairing Complete Event |
| 0x0040000000000000 | Reserved |
| 0x00800000000000 00 | Link Supervision Timeout Changed Event |
| 0x0100000000000000 | Enhanced Flush Complete Event |
| 0x0400000000000000 | User Passkey Notification Event |
| 0x0800000000000000 | Keypress Notification Event |
| 0x1000000000000000 | Remote Host Supported Features Notification Event |
| 0xE000000000000000 | Reserved for future use |
| 0x00 00 1F FF FF FF FF FF | **Default** |

### Return Parameters:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Set_Event_Mask command succeeded. |
| 0x01-0xFF | Set_Event_Mask command failed. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |

### Event(s) generated (unless masked away):

When the Set_Event_Mask command has completed, a Command Complete event will be generated.

### 7.3.2 Reset Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reset | 0x0003 | | Status |

**Description:**

The Reset command will reset the Controller and the Link Manager. The reset command shall not affect the used HCI transport layer since the HCI transport layers may have reset mechanisms of their own. After the reset is completed, the current operational state will be lost, the Bluetooth device will enter standby mode and the Controller will automatically revert to the default values for the parameters for which default values are defined in the specification.

Note: the Host is not allowed to send additional HCI commands before the Command Complete event related to the Reset command has been received.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reset command succeeded, was received and will be executed. |
| 0x01-0xFF | Reset command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the reset has been performed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 672 of 1420
IPR2025-00103

### 7.3.3 Set Event Filter Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Set_Event_Filter | 0x0005 | Filter_Type, Filter_Condition_Type, Condition | Status |

**Description:**

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 which 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 is not equal to 0x00. (The old event filters will not be overwritten). To clear all event filters, the Filter_Type = 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 = 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 only sends Inquiry Results to the Host 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:

1. Return responses from all devices during the Inquiry process
2. A device with a specific Class of Device responded to the Inquiry process
3. 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 command.

The Connection Setup filter allows the Host to specify that the Controller only sends a Connection Complete or Connection Request event to the Host 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:

1. Allow Connections from all devices
2. Allow Connections from a device with a specific Class of Device
3. Allow Connections from a device with a specific BD_ADDR



LifeScan, Inc. Ex. 1019
Page 673 of 1420
IPR2025-00103

For each of these conditions, an Auto_Accept_Flag parameter 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 auto 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).

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, and is paged by another device which meets one on 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 *Memory Full* error code and the filter will not be installed.

Note: the Clear All Filters has no Filter Condition Types or Conditions.

Note: In the condition that a connection is auto 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.

**Command Parameters:**

*Filter_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Clear All Filters (Note: In this case, the Filter_Condition_type and Condition parameters should not be given, they should have a length of 0 octets. Filter_Type should be the only parameter.) |
| 0x01 | Inquiry Result. |
| 0x02 | Connection Setup. |
| 0x03-0xFF | Reserved for Future Use. |

*Filter Condition Types: For each Filter Type one or more Filter Condition types exists.*

*Inquiry_Result_Filter_Condition_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Return responses from all devices during the Inquiry process.<br>(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 Section 7.1.1 on page 416 and Section 7.1.3 on page 419) |
| 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 |

*Connection_Setup_Filter_Condition_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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: For each Filter Condition Type defined for the Inquiry Result Filter and the Connection Setup Filter, zero or more Condition parameters are required – depending on the filter condition type and filter type.*

*Condition for Inquiry_Result_Filter_Condition_Type = 0x00*

*Condition:* *Size: 0 Octet*

| Value | Parameter Description |
|---|---|
| | The Condition parameter is not used. |

*Condition for Inquiry_Result_Filter_Condition_Type = 0x01*

**Condition:**

Size: 6 Octets

*Class_of_Device:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0x000000 | Default, Return All Devices. |
| 0xXXXXXX | *Class of Device* of Interest. |

*Class_of_Device_Mask:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Bit Mask used to determine which bits of the Class of Device parameter are 'don't care'. Zero-value bits in the mask indicate the 'don't care' bits of the Class of Device. |

*Condition for Inquiry_Result_Filter_Condition_Type = 0x02*

**Condition:**

Size: 6 Octets

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of Interest |

*Condition for Connection_Setup_Filter_Condition_Type = 0x00*

**Condition:**

Size: 1 Octet

*Auto_Accept_Flag:* *Size:1 Octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Do NOT Auto accept the connection. (Auto accept is off) |
| 0x02 | Do Auto accept the connection with role switch disabled. (Auto accept is on). |
| 0x03 | Do Auto accept the connection with role switch enabled. (Auto accept is on).<br>Note: When auto 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. |



LifeScan, Inc. Ex. 1019
Page 676 of 1420
IPR2025-00103

*Condition for Connection_Setup_Filter_Condition_Type = 0x01*

**Condition:**

Size: 7 Octets

*Class_of_Device:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0x000000 | Default, Return All Devices. |
| 0xXXXXXX | *Class of Device* of Interest. |

*Class_of_Device_Mask:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Bit Mask used to determine which bits of the Class of Device parameter are 'don't care'. Zero-value bits in the mask indicate the 'don't care' bits of the Class of Device.<br>Note: For an incoming SCO connection, if the class of device is unknown then the connection will be accepted. |

*Auto_Accept_Flag:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Do NOT Auto accept the connection. (Auto accept is off) |
| 0x02 | Do Auto accept the connection with role switch disabled. (Auto accept is on). |
| 0x03 | Do Auto accept the connection with role switch enabled. (Auto accept is on).<br>Note: When auto 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. |

*Condition for Connection_Setup_Filter_Condition_Type* = 0x02

**Condition:**

Size: 7 Octets

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device of Interest. |



LifeScan, Inc. Ex. 1019
Page 677 of 1420
IPR2025-00103

*Auto_Accept_Flag:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Do NOT Auto accept the connection. (Auto accept is off) |
| 0x02 | Do Auto accept the connection with role switch disabled. (Auto accept is on). |
| 0x03 | Do Auto accept the connection with role switch enabled. (Auto accept is on). **Note: When auto 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. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Set_Event_Filter command succeeded. |
| 0x01-0xFF | Set_Event_Filter command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

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.

### 7.3.4 Flush Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Flush | 0x0008 | Connection_Handle | Status, Connection_Handle |

**Description:**

The Flush command is used to discard all data that is 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. Both automatically-flushable and non-automatically-flushable packets shall be discarded (see Section 5.4.2 on page 394). After this, all data that is sent to the Controller for the same Connection Handle will be discarded by the Controller until an HCI Data Packet with one of the start Packet_Boundary_Flag values (0x00 or 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 baseband should try to retransmit a baseband packet for a Connection Handle before all data that is 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 Section 7.3.29 on page 536) can be used to automatically flush an automatically-flushable L2CAP packet that is currently being transmitted after the specified flush timer has expired.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify which connection to flush.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Flush command succeeded. |
| 0x01-0xFF | Flush command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify which connection the flush command was issued on.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Event(s) generated (unless masked away):**

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.

### 7.3.5 Read PIN Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_PIN_Type | 0x0009 | | Status, PIN_Type |

**Description:**

The Read PIN Type command is used to read the PIN_Type configuration parameter. See Section 6.13 on page 404.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_PIN_Type command succeeded. |
| 0x01-0xFF | Read_PIN_Type command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*PIN_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Variable PIN. |
| 0x01 | Fixed PIN. |

**Event(s) generated (unless masked away):**

When the Read_PIN_Type command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 680 of 1420
IPR2025-00103

### 7.3.6 Write PIN Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_PIN_Type | 0x000A | PIN_Type | Status |

**Description:**

The Write_PIN_Type command is used to write the PIN Type configuration parameter. See Section 6.13 on page 404.

**Command Parameters:**

*PIN_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Variable PIN. |
| 0x01 | Fixed PIN. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write PIN Type command succeeded. |
| 0x01-0xFF | Write PIN Type command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_PIN_Type command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 681 of 1420
IPR2025-00103

### 7.3.7 Create New Unit Key Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Create_New_Unit_Key | 0x000B | | Status |

**Description:**

The Create_New_Unit_Key command is used to create a new unit key. The Bluetooth 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.

Note: this command will not have any effect for a device which doesn't use unit keys (i.e. a device which uses only combination keys).

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Create New Unit Key command succeeded. |
| 0x01-0xFF | Create New Unit Key command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Create_New_Unit_Key command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 682 of 1420
IPR2025-00103

### 7.3.8 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 |

**Description:**

The Read_Stored_Link_Key command provides the ability to read whether one or more link keys are stored in the Bluetooth Controller. The Bluetooth Controller can store a limited number of link keys for other Bluetooth devices. Link keys are shared between two Bluetooth devices, and are used for all security transactions between the two devices. The read link key command shall not return the link keys value. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the Bluetooth 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 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 Section 6.14 on page 404.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the stored link key to be read. |

*Read_All_Flag:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Return Link Key for specified BD_ADDR. |
| 0x01 | Return all stored Link Keys. |
| 0x02-0xFF | Reserved for future use. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Stored_Link_Key command succeeded. |
| 0x01-0xFF | Read_Stored_Link_Key command failed. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |



LifeScan, Inc. Ex. 1019
Page 683 of 1420
IPR2025-00103

*Max_Num_Keys:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total Number of Link Keys that the Controller can store.<br>Range: 0x0000 – 0xFFFF |

*Num_Keys_Read:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of Link Keys Read.<br>Range: 0x0000 – 0xFFFF |

**Event(s) generated (unless masked away):**

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.

### 7.3.9 Write Stored Link Key Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Stored_Link_Key | 0x0011 | Num_Keys_To_Write,<br>*BD_ADDR*[i],<br>Link_Key[i] | Status,<br>Num_Keys_Written |

**Description:**

The Write_Stored_Link_Key command provides the ability to write one or more link keys to be stored in the Bluetooth Controller. The Bluetooth Controller can store a limited number of link keys for other Bluetooth devices. If no additional space is available in the Bluetooth 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 Bluetooth 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 Bluetooth Controller when needed. See Section 6.14 on page 404.

Note: Link Keys are only stored by issuing this command.

**Command Parameters:**

*Num_Keys_To_Write:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Link Keys to Write.<br>Range: 0x01 - 0x0B |

*BD_ADDR [i]:* *Size: 6 Octets * Num_Keys_To_Write*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the associated Link Key. |

*Link_Key:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX<br>XXXXXXXXXX<br>XXXXXXXXXX | Link Key for an associated BD_ADDR. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Stored_Link_Key command succeeded. |
| 0x01-0xFF | Write_Stored_Link_Key command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Num_Keys_Written:* *Size: 1 Octets*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Link Keys successfully written.<br>Range: 0x00 – 0x0B |

**Event(s) generated (unless masked away):**

When the Write_Stored_Link_Key command has completed, a Command Complete event will be generated.

### 7.3.10 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 |

**Description:**

The Delete_Stored_Link_Key command provides the ability to remove one or more of the link keys stored in the Bluetooth Controller. The Bluetooth Controller can store a limited number of link keys for other Bluetooth devices. Link keys are shared between two Bluetooth 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 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 Section 6.14 on page 404.

**Command Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the link key to be deleted. |

*Delete_All_Flag:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Delete only the Link Key for specified BD_ADDR. |
| 0x01 | Delete all stored Link Keys. |
| 0x02-0xFF | Reserved for future use. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Delete_Stored_Link_Key command succeeded. |
| 0x01-0xFF | Delete_Stored_Link_Key command failed. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |



LifeScan, Inc. Ex. 1019
Page 686 of 1420
IPR2025-00103

*Num_Keys_Deleted:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of Link Keys Deleted |

**Event(s) generated (unless masked away):**

When the Delete_Stored_Link_Key command has completed, a Command Complete event will be generated.

### 7.3.11 Write Local Name Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Local_Name | 0x0013 | Local Name | Status |

**Description:**

The Write_Local_Name command provides the ability to modify the user-friendly name for the Bluetooth device. See Section 6.22 on page 409.

**Command Parameters:**

*Local Name:* *Size: 248 Octets*

| Value | Parameter Description |
|---|---|
| | A UTF-8 encoded User-Friendly Descriptive Name for the device.<br>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.** |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Local_Name command succeeded. |
| 0x01-0xFF | Write_Local_Name command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Local_Name command has completed, a Command Complete event will be generated.

### 7.3.12 Read Local Name Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Name | 0x0014 | | Status, Local_Name |

**Description:**

The Read_Local_Name command provides the ability to read the stored user-friendly name for the Bluetooth device. See Section 6.22 on page 409.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Local_Name command succeeded |
| 0x01-0xFF | Read_Local_Name command failed see "Error Codes" on page 331 [Part D] for list of Error Codes |

*Local_Name:* *Size: 248 Octets*

| Value | Parameter Description |
|---|---|
| | A UTF-8 encoded User Friendly Descriptive Name for the device.<br>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. |

**Event(s) generated (unless masked away):**

When the Read_Local_Name command has completed a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 688 of 1420
IPR2025-00103

### 7.3.13 Read Connection Accept Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Connection_Accept_Timeout | 0x0015 | | Status,<br>Conn_Accept_Timeout |

**Description:**

This command will read the value for the Connection_Accept_Timeout configuration parameter. See Section 6.7 on page 401.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Connection_Accept_Timeout command succeeded. |
| 0x01-0xFF | Read_Connection_Accept_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Conn_Accept_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Connection Accept Timeout measured in Number of Baseband slots.<br>Interval Length = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 – 0xB540<br>Time Range: 0.625 msec -29 seconds |

**Event(s) generated (unless masked away):**

When the Read_Connection_Timeout command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 689 of 1420
IPR2025-00103

### 7.3.14 Write Connection Accept Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Connection_Accept_Timeout | 0x0016 | Conn_Accept_Timeout | Status |

**Description:**

This command will write the value for the Connection_Accept_Timeout configuration parameter. See Section 6.7 on page 401.

**Command Parameters:**

*Conn_Accept_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Connection Accept Timeout measured in Number of Baseband slots.<br>Interval Length = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 – 0xB540<br>Time Range: 0.625 msec - 29 seconds<br>Default: N = 0x1FA0 Time = 5.06 Sec<br>Mandatory Range for Controller: 0x00A0 to 0xB540 |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Connection_Accept_Timeout command succeeded. |
| 0x01-0xFF | Write_Connection_Accept_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Connection_Accept_Timeout command has completed, a Command Complete event will be generated.

### 7.3.15 Read Page Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Timeout | 0x0017 | | Status,<br>Page_Timeout |

**Description:**

This command will read the value for the Page_Timeout configuration parameter. See Section 6.6 on page 401.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Page_Timeout command succeeded. |
| 0x01-0xFF | Read_Page_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Page_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Page Timeout measured in Number of Baseband slots.<br>Interval Length = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 – 0xFFFF<br>Time Range: 0.625 msec -40.9 Seconds |

**Event(s) generated (unless masked away):**

When the Read_Page_Timeout command has completed, a Command Complete event will be generated.

### 7.3.16 Write Page Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Timeout | 0x0018 | Page_Timeout | Status |

**Description:**

This command will write the value for the Page_Timeout configuration parameter. The Page_Timeout configuration parameter defines the maximum time the local Link Manager will wait for a baseband 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 baseband level, the connection attempt will be considered to have failed.

**Command Parameters:**

*Page_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0 | Illegal Page Timeout. Must be larger than 0. |
| N = 0xXXXX | Page Timeout measured in Number of Baseband slots.<br>Interval Length = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 – 0xFFFF<br>Time Range: 0.625 msec -40.9 Seconds<br>**Default: N = 0x2000 Time = 5.12 Sec**<br>Mandatory Range for Controller: 0x0016 to 0xFFFF |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Page_Timeout command succeeded. |
| 0x01-0xFF | Write_Page_Timeout command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Page_Timeout command has completed, a Command Complete event will be generated.

### 7.3.17 Read Scan Enable Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Scan_Enable | 0x0019 | | Status, Scan_Enable |

**Description:**

This command will read the value for the Scan_Enable parameter configuration parameter. See Section 6.1 on page 399.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Scan_Enable command succeeded. |
| 0x01-0xFF | Read_Scan_Enable command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Scan_Enable:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Scans enabled. |
| 0x01 | Inquiry Scan enabled.<br>Page Scan disabled. |
| 0x02 | Inquiry Scan disabled.<br>Page Scan enabled. |
| 0x03 | Inquiry Scan enabled.<br>Page Scan enabled. |
| 0x04-0xFF | Reserved |

**Event(s) generated (unless masked away):**

When the Read_Scan_Enable command has completed, a Command Complete event will be generated.

### 7.3.18 Write Scan Enable Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Scan_Enable | 0x001A | Scan_Enable | Status |

**Description:**

This command will write the value for the Scan_Enable configuration parameter. See Section 6.1 on page 399.

**Command Parameters:**

*Scan_Enable:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Scans enabled. **Default.** |
| 0x01 | Inquiry Scan enabled.<br>Page Scan disabled. |
| 0x02 | Inquiry Scan disabled.<br>Page Scan enabled. |
| 0x03 | Inquiry Scan enabled.<br>Page Scan enabled. |
| 0x04-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Scan_Enable command succeeded. |
| 0x01-0xFF | Write_Scan_Enable command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Scan_Enable command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 694 of 1420
IPR2025-00103

### 7.3.19 Read Page Scan Activity Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Scan_ Activity | 0x001B | | Status, Page_Scan_Interval, Page_Scan_Window |

**Description:**

This command will read the value for Page_Scan_Interval and Page_Scan_Window configuration parameters. See Section 6.8 on page 402 and Section 6.9 on page 402.

Note: Page Scan is only performed when Page_Scan is enabled (see 6.1, 7.3.17 and 7.3.18).

A changed Page_Scan_Interval could change the local Page_Scan_ Repetition_Mode (see "Baseband Specification" on page 53 [Part B], Keyword: SR-Mode).

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Page_Scan_Activity command succeeded. |
| 0x01-0xFF | Read_Page_Scan_Activity command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Page_Scan_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0012 – 0x1000<br>Time = N * 0.625 msec<br>Range: 11.25 msec - 2560 msec; only even values are valid |



LifeScan, Inc. Ex. 1019
Page 695 of 1420
IPR2025-00103

*Page_Scan_Window:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0011 - 0x1000<br>Time = N * 0.625 msec<br>Range: 10.625 msec - 2560 msec |

**Event(s) generated (unless masked away):**

When the Read_Page_Scan_Activity command has completed, a Command Complete event will be generated.

### 7.3.20 Write Page Scan Activity Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Scan_Activity | 0x001C | Page_Scan_Interval, Page_Scan_Window | Status |

**Description:**

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 Section 6.8 on page 402 and Section 6.9 on page 402.

Note: Page Scan is only performed when Page_Scan is enabled (see 6.1, 7.3.17 and 7.3.18). A changed Page_Scan_Interval could change the local Page_Scan_Repetition_Mode (see Baseband Specification, Section 8.3.1, on page 146).

**Command Parameters:**

*Page_Scan_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| See Section 6.8 on page 402 | |

*Page_Scan_Window:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| See Section 6.9 on page 402 | |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Page_Scan_Activity command succeeded. |
| 0x01-0xFF | Write_Page_Scan_Activity command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Page_Scan_Activity command has completed, a Command Complete event will be generated.

### 7.3.21 Read Inquiry Scan Activity Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_ Inquiry_Scan_Activity | 0x001D | | Status, Inquiry_Scan_Interval, Inquiry_Scan_Window |

**Description:**

This command will read the value for Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameter. See Section 6.2 on page 399 and Section 6.3 on page 400.

Note: Inquiry Scan is only performed when Inquiry_Scan is enabled see 6.1, 7.3.17 and 7.3.18).

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Inquiry_Scan_Activity command succeeded. |
| 0x01-0xFF | Read_Inquiry_Scan_Activity command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Inquiry_Scan_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0012 – 0x1000<br>Time = N * 0.625 msec<br>Range: 11.25 - 2560 msec; only even values are valid |

*Inquiry_Scan_Window:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Size: 2 Octets<br>Range: 0x0011 - 0x1000<br>Time = N * 0.625 msec<br>Range: 10.625 msec - 2560 msec |

**Event(s) generated (unless masked away):**

When the Read_Inquiry_Scan_Activity command has completed, a Command Complete event will be generated.

### 7.3.22 Write Inquiry Scan Activity Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Scan_Activity | 0x001E | Inquiry_Scan_Interval, Inquiry_Scan_Window | Status |

**Description:**

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 Section 6.2 on page 399 and Section 6.3 on page 400.

Note: Inquiry Scan is only performed when Inquiry_Scan is enabled (see 6.1, 7.3.17 and 7.3.18).

**Command Parameters:**

*Inquiry_Scan_Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| See Section 6.2 on page 399. | |

*Inquiry_Scan_Window:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| See Section 6.3 on page 400. | |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Inquiry_Scan_Activity command succeeded. |
| 0x01-0xFF | Write_Inquiry_Scan_Activity command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Inquiry_Scan_Activity command has completed, a Command Complete event will be generated.

### 7.3.23 Read Authentication Enable Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|

| | | | |
|---|---|---|---|
| HCI_Read_ Authentication_Enable | 0x001F | | Status, Authentication_Enable |

**Description:**

This command will read the value for the Authentication_Enable configuration parameter. See Section 10.10.1 on page 676.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Authentication_Enable command succeeded. |
| 0x01-0xFF | Read_Authentication_Enable command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Authentication_Enable:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication not required. |
| 0x01 | Authentication required for all connections. |
| 0x02-0xFF | Reserved |

**Event(s) generated (unless masked away):**

When the Read_Authentication_Enable command has completed, a Command Complete event will be generated.

### 7.3.24 Write Authentication Enable Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_ Authentication_Enable | 0x0020 | Authentication_Enable | Status |

**Description:**

This command will write the value for the Authentication_Enable configuration parameter. See Authentication Enable on page 676.

The Authentication_Enable configuration parameter shall only apply to connections (e.g. send an LMP_in_rand or LMP_au_rand) when the remote device's



LifeScan, Inc. Ex. 1019
Page 700 of 1420
IPR2025-00103

Host or Controller does not support Secure Simple Pairing or when the local Host does not support Secure Simple Pairing.

Note: Requires LM to read host features during connection setup.

**Command Parameters:**

*Authentication_Enable:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication not required. **Default.** |
| 0x01 | Authentication required for all connections. |
| 0x02-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write Authentication_Enable command succeeded. |
| 0x01-0xFF | Write Authentication_Enable command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Authentication_Enable command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 701 of 1420
IPR2025-00103

### 7.3.25 Read Class of Device Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Class_of_Device | 0x0023 | | Status,<br>Class_of_Device |

**Description:**

This command will read the value for the Class_of_Device parameter. See Section 6.25 on page 410.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Class_of_Device command succeeded. |
| 0x01-0xFF | Read_Class_of_Device command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Class_of_Device:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device. |

**Event(s) generated (unless masked away):**

When the Read_Class_of_Device command has completed, a Command Complete event will be generated.

### 7.3.26 Write Class of Device Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Class_of_Device | 0x0024 | Class_of_Device | Status |

**Description:**

This command will write the value for the Class_of_Device parameter. See Section 6.25 on page 410.

**Command Parameters:**

*Class_of_Device:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Class_of_Device command succeeded. |
| 0x01-0xFF | Write_Class_of_Device command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Class_of_Device command has completed, a Command Complete event will be generated.

### 7.3.27 Read Voice Setting Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Voice_Setting | 0x0025 | | Status,<br>Voice_Setting |

**Description:**

This command will read the values for the Voice_Setting configuration parameter. See Section 6.12 on page 403.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Voice_Setting command succeeded. |
| 0x01-0xFF | Read_Voice_Setting command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Voice_Setting:* *Size: 2 Octets (10 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12 on page 403. | |

**Event(s) generated (unless masked away):**

When the Read_Voice_Setting command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 704 of 1420
IPR2025-00103

### 7.3.28 Write Voice Setting Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Voice_Setting | 0x0026 | Voice_Setting | Status |

**Description:**

This command will write the values for the Voice_Setting configuration parameter. See Section 6.12 on page 403.

**Command Parameters:**

*Voice_Setting:* *Size: 2 Octets (10 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| See Section 6.12 on page 403. | |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Voice_Setting command succeeded. |
| 0x01-0xFF | Write_Voice_Setting command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Voice_Setting command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 705 of 1420
IPR2025-00103

### 7.3.29 Read Automatic Flush Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Automatic_Flush_Timeout | 0x0027 | Connection_Handle | Status, Connection_Handle, Flush_Timeout |

**Description:**

This command will read the value for the Flush_Timeout parameter for the specified Connection Handle. See Section 6.18 on page 407.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Flush Timeout to read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Automatic_Flush_Timeout command succeeded. |
| 0x01-0xFF | Read_Automatic_Flush_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Flush Timeout has been read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Flush_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0 | Timeout = ∞; No Automatic Flush |
| N = 0xXXXX | Flush Timeout = N * 0.625 msec<br>Size: 11 bits<br>Range: 0x0001 – 0x07FF |

**Event(s) generated (unless masked away):**

When the Read_Automatic_Flush_Timeout command has completed, a Command Complete event will be generated.

### 7.3.30 Write Automatic Flush Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Automatic_Flush_Timeout | 0x0028 | Connection_Handle, Flush_Timeout | Status, Connection_Handle |

**Description:**

This command will write the value for the Flush_Timeout parameter for the specified Connection Handle. See Section 6.18 on page 407.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Flush Timeout to write to.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Flush_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0 | Timeout = ∞; No Automatic Flush. **Default.** |
| N = 0xXXXX | Flush Timeout = N * 0.625 msec<br>Size: 11 bits<br>Range: 0x0001 – 0x07FF<br>Mandatory Range for Controller: 0x0002 to 0x07FF |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Automatic_Flush_Timeout command succeeded. |
| 0x01-0xFF | Write_Automatic_Flush_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Flush Timeout has been written.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Event(s) generated (unless masked away):**

When the Write_Automatic_Flush_Timeout command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 707 of 1420
IPR2025-00103

### 7.3.31 Read Num Broadcast Retransmissions Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Num_Broadcast_ Retransmissions | 0x0029 | | Status, Num_Broadcast_Retransmissions |

**Description:**

This command will read the device's parameter value for the Number of Broadcast Retransmissions. See Section 6.19 on page 407

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Num_Broadcast_Retransmissions command succeeded. |
| 0x01-0xFF | Read_Num_Broadcast_Retransmissions command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Num_Broadcast_Retransmissions:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Section 6.19 on page 407. | |

**Event(s) generated (unless masked away):**

When the Read_Num_Broadcast_Retransmission command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 708 of 1420
IPR2025-00103

### 7.3.32 Write Num Broadcast Retransmissions Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Num_Broadcast_ Retransmissions | 0x002A | Num_Broadcast_ Retransmissions | Status |

**Description:**

This command will write the device's parameter value for the Number of Broadcast Retransmissions. See Section 6.19 on page 407.

**Command Parameters:**

*Num_Broadcast_Retransmissions:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Section 6.19 on page 407. | |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Num_Broadcast_Retransmissions command succeeded. |
| 0x01-0xFF | Write_Num_Broadcast_Retransmissions command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Num_Broadcast_Retransmissions command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 709 of 1420
IPR2025-00103

### 7.3.33 Read Hold Mode Activity Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Hold_Mode_Activity | 0x002B | | Status, Hold_Mode_Activity |

**Description:**

This command will read the value for the Hold_Mode_Activity parameter. See Section 6.16 on page 405.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Hold_Mode_Activity command succeeded. |
| 0x01-0xFF | Read_Hold_Mode_Activity command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Hold_Mode_Activity:* *Size: 1 Octet*

| 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. |

**Event(s) generated (unless masked away):**

When the Read_Hold_Mode_Activity command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 710 of 1420
IPR2025-00103

### 7.3.34 Write Hold Mode Activity Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Hold_Mode_Activity | 0x002C | Hold_Mode_Activity | Status |

**Description:**

This command will write the value for the Hold_Mode_Activity parameter. See Section 6.16 on page 405.

**Command Parameters:**

*Hold_Mode_Activity:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Maintain current Power State. **Default.** |
| 0x01 | Suspend Page Scan. |
| 0x02 | Suspend Inquiry Scan. |
| 0x04 | Suspend Periodic Inquiries. |
| 0x08-0xFF | Reserved for Future Use. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Hold_Mode_Activity command succeeded. |
| 0x01-0xFF | Write_Hold_Mode_Activity command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Hold_Mode_Activity command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 711 of 1420
IPR2025-00103

### 7.3.35 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 |

**Description:**

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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Transmit Power Level setting to read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read Current Transmit Power Level. |
| 0x01 | Read Maximum Transmit Power Level. |
| 0x02-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Transmit_Power_Level command succeeded. |
| 0x01-0xFF | Read_Transmit_Power_Level command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Transmit Power Level setting is returned.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Transmit_Power_Level:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Size: 1 Octet (signed integer)<br>Range: -30 ≤ N ≤ 20<br>Units: dBm |

**Event(s) generated (unless masked away):**

When the Read_Transmit_Power_Level command has completed, a Command Complete event will be generated.

### 7.3.36 Read Synchronous Flow Control Enable Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_ Synchronous_Flow_ Control_Enable | 0x002E | | Status, Synchronous_Flow_ Control_Enable |

**Description:**

The Read_Synchronous_Flow_Control_Enable command provides the ability to read the Synchronous_Flow_Control_Enable setting. See Section 6.21 on page 408.

Note: the Synchronous_Flow_Control_Enable setting can only be changed if no connections exist.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Synchronous_Flow_Control_Enable command succeeded |
| 0x01-0xFF | Read_Synchronous_Flow_Control_Enable command failed see "Error Codes" on page 331 [Part D] for list of Error Codes |

*Synchronous_Flow_Control_Enable:* *Size: 1 Octet*

| 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. |

**Event(s) generated (unless masked away):**

When the Read_Synchronous_Flow_Control_Enable command has completed a Command Complete event will be generated.

### 7.3.37 Write Synchronous Flow Control Enable Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Synchronous_ Flow_Control_Enable | 0x002F | Synchronous_Flow_ Control_Enable | Status |

**Description:**

The Write_Synchronous_Flow_Control_Enable command provides the ability to write the Synchronous_Flow_Control_Enable setting. See Section 6.21 on page 408.

Note: the Synchronous_Flow_Control_Enable setting can only be changed if no connections exist.

**Command Parameters:**

*Synchronous_Flow_Control_Enable:* *Size: 1 Octet*

| 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.<br>**Default** |
| 0x01 | Synchronous Flow Control is enabled. Number of Completed Packets events will be sent from the Controller for synchronous Connection Handles. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Synchronous_Flow_Control_Enable command succeeded |
| 0x01-0xFF | Write_Synchronous_Flow_Control_Enable command failed see "Error Codes" on page 331 [Part D] for list of Error Codes |

**Event(s) generated (unless masked away):**

When the Write_Synchronous_Flow_Control_Enable command has completed a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 715 of 1420
IPR2025-00103

### 7.3.38 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 |

**Description:**

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 only contain 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 only contain 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 setting shall only be changed if no connections exist.

**Command Parameters:**

*Flow_Control_Enable:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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** both for HCI ACL Data Packets and HCI synchronous Data Packets in direction from Controller to Host. |
| 0x04-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Set_Controller_To_Host_Flow_Control command succeeded. |
| 0x01-0xFF | Set_Controller_To_Host_Flow_Control command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Set_Controller_To_Host_Flow_Control command has completed, a Command Complete event will be generated.

### 7.3.39 Host Buffer Size Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Host_Buffer_Size | 0x0033 | Host_ACL_Data_Packet_Length,<br>Host_Synchronous_Data_Packet_Length,<br>Host_Total_Num_ACL_Data_Packets,<br>Host_Total_Num_Synchronous_Data_Packets | Status |

**Description:**

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 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.

Note: 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.

**Command Parameters:**

*Host_ACL_Data_Packet_Length:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total number of HCI synchronous Data Packets that can be stored in the data<br>buffers of the Host. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Host_Buffer_Size command succeeded. |
| 0x01-0xFF | Host_Buffer_Size command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Host_Buffer_Size command has completed, a Command Complete event will be generated.

### 7.3.40 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] | |

**Description:**

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_Packetscommand 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 only be issued by the Host 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.

Note: 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.

**Command Parameters:**

*Number_Of_Handles:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of Connection Handles and Host_Num_Of_Completed_Packets parameters pairs contained in this command.<br>Range: 0-255 |

*Connection_Handle[i]: Size: Number_Of_Handles*2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Host_Num_Of_Completed_Packets [i]: Size: Number_Of_Handles * 2 Octets*

| Value | Parameter Description |
|---|---|
| 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.<br>Range for N: 0x0000-0xFFFF |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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 *Invalid HCI Command Parameters error code*. 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.

### 7.3.41 Read Link Supervision Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Link_Supervision_ Timeout | 0x0036 | Connection_Handle | Status,<br>Connection_Handle,<br>Link_Supervision_ Timeout |

**Description:**

This command will read the value for the Link_Supervision_Timeout parameter for the device.

Note: the Connection_Handle used for this command must be the ACL connection to the appropriate device. See Section 6.20 on page 408.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Link Supervision Timeout value is to be read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Link_Supervision_Timeout command succeeded. |
| 0x01-0xFF | Read_Link_Supervision_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Link Supervision Timeout value was read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 721 of 1420
IPR2025-00103

*Link_Supervision_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No Link_Supervision_Timeout. |
| N = 0xXXXX | Measured in Number of Baseband slots<br>Link_Supervision_Timeout = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 – 0xFFFF<br>Time Range: 0.625ms - 40.9 sec |

**Event(s) generated (unless masked away):**

When the Read_Link_Supervision_Timeout command has completed, a Command Complete event will be generated.

### 7.3.42 Write Link Supervision Timeout Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Link_Supervision_Timeout | 0x0037 | Connection_Handle,<br>Link_Supervision_Timeout | Status,<br>Connection_Handle |

**Description:**

This command will write the value for the Link_Supervision_Timeout parameter for the device. This command shall only be issued on the master for the given Connection Handle. If this command is issued on a slave, the command shall be rejected with the Command Disallowed.

Note: 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 Section 6.20 on page 408.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Link Supervision Timeout value is to be written.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Link_Supervision_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0x0000 | No Link_Supervision_Timeout. |
| N = 0xXXXX | Measured in Number of Baseband slots<br>Link_Supervision_Timeout = N*0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 – 0xFFFF<br>Time Range: 0.625ms – 40.9 sec<br>**Default**: N = 0x7D00<br>Link_Supervision_Timeout = 20 sec<br>Mandatory Range for Controller: 0x0190 to 0xFFFF; plus 0 for infinite timeout |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|

| | |
|---|---|
| 0x00 | Write_Link_Supervision_Timeout command succeeded. |
| 0x01-0xFF | Write_Link_Supervision_Timeout command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Link Supervision Timeout value was written.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Event(s) generated (unless masked away):**

When the Write_Link_Supervision_Timeout command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 724 of 1420
IPR2025-00103

### 7.3.43 Read Number Of Supported IAC Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Number_Of_Supported_IAC | 0x0038 | | Status,<br>Num_Support_IAC |

**Description:**

This command will read the value for the number of Inquiry Access Codes (IAC) that the local Bluetooth device can simultaneous listen for during an Inquiry Scan. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some Bluetooth devices support additional IACs.

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Number_Of_Supported_IAC command succeeded. |
| 0x01-0xFF | Read_Number_Of_Supported_IAC command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Num_Support_IAC* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Specifies the number of Supported IAC that the local Bluetooth device can simultaneous listen for during an Inquiry Scan.<br>Range: 0x01-0x40 |

**Event(s) generated (unless masked away):**

When the Read_Number_Of_Supported_IAC command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 725 of 1420
IPR2025-00103

### 7.3.44 Read Current IAC LAP Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Current_IAC_LAP | 0x0039 | | Status, Num_Current_IAC, IAC_LAP[i] |

**Description:**

This command reads the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some Bluetooth devices support additional IACs.

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Current_IAC_LAP command succeeded. |
| 0x01-0xFF | Read_Current_IAC_LAP command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Num_Current_IAC* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Specifies the number of IACs which are currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan.<br>Range: 0x01-0x40 |

*IAC_LAP[i]* *Size: 3 Octets * Num_Current_IAC*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | LAPs used to create the IAC which is currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan.<br>Range: 0x9E8B00-0x9E8B3F |

**Event(s) generated (unless masked away):**

When the Read_Current_IAC_LAP command has completed, a Command Complete event will be generated.

### 7.3.45 Write Current IAC LAP Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Current_IAC_LAP | 0x003A | Num_Current_IAC,<br>IAC_LAP[i] | Status |

**Description:**

This command writes the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some Bluetooth devices support additional IACs.

This command shall clear any existing IACs and stores Num_Current_IAC and the IAC_LAPs in to the controller. If Num_Current_IAC is greater than Num_Support_IAC then only the first Num_Support_IACs shall be stored in the controller, and a Command Complete event with error code *Success (0x00)* shall be generated.

**Command Parameters:**

*Num_Current_IAC* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Specifies the number of IACs which are currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan.<br>Range: 0x01-0x40 |

*IAC_LAP[i]* *Size: 3 Octets * Num_Current_IAC*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | LAP(s) used to create IAC which is currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan.<br>Range: 0x9E8B00-0x9E8B3F.<br>The GIAC is the default IAC to be used. If additional IACs are supported, additional default IAC will be determined by the manufacturer. |



LifeScan, Inc. Ex. 1019
Page 727 of 1420
IPR2025-00103

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Current_IAC_LAP command succeeded. |
| 0x01-0xFF | Write_Current_IAC_LAP command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Current_IAC_LAP command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 728 of 1420
IPR2025-00103

### 7.3.46 Set AFH Host Channel Classification Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| Set_AFH_Host_Channel_ Classification | 0x003F | AFH_Host_Channel_ Classification | Status |

**Description:**

The Set_AFH_Host_Channel_Classification command allows the Bluetooth 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 features.

If this command is used, then updates should be sent within 10 seconds, of the host knowing that the channel classification has changed. The interval between two successive commands sent shall be at least 1 second.

**Command Parameters:**

*AFH_Host_Channel_Classification:* *Size: 10 Octets (79 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX XXXXXXXXXX XX | This parameter contains 79 1-bit field.<br>The $n^{th}$ such field (in the range 0 to 78) contains the value for channel *n*:<br>Channel *n* is bad = 0<br>Channel *n* is unknown = 1<br>The most significant bit is reserved and shall be set to 0<br>At least $N_{min}$ channels shall be marked as unknown. (See Baseband Specification, Section 2.3.1, on page 70) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Set_AFH_Host_Channel_Classification command succeeded. |
| 0x01-0xFF | Set_AFH_Host_Channel_Classification command failed. “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Set_AFH_Host_Channel_Classification command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 729 of 1420
IPR2025-00103

### 7.3.47 Read Inquiry Scan Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry_Scan_Type | 0x0042 | | Status, Inquiry_Scan_Type |

**Description:**

This command is used to read the Inquiry_Scan_Type configuration parameter of the local Bluetooth device. See Section 6.4 on page 400.
For details, see the Baseband Specification, "Inquiry scan substate" on page 156 [Part B].

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Inquiry_Scan_Type command succeeded |
| 0x01-0xFF | Read_Inquiry_Scan_Type command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Inquiry_Scan_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| 0x02-0xFF | Reserved |

**Event(s) generated (unless masked away):**

When the Read_Inquiry_Scan_Type command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 730 of 1420
IPR2025-00103

### 7.3.48 Write Inquiry Scan Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Scan_Type | 0x0043 | Scan_Type | Status |

**Description:**

This command is used to write the Inquiry Scan Type configuration parameter of the local Bluetooth device. See Section 6.4 on page 400. For details, see the Baseband Specification, "Inquiry scan substate" on page 156 [Part B].

**Command Parameters:**

*Scan_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| 0x02-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Inquiry_Scan_Type command succeeded |
| 0x01-0xFF | Write_Inquiry_Scan_Type command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Inquiry_Scan_Type command has completed, a Command Complete event will be generated.

### 7.3.49 Read Inquiry Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry_Mode | 0x0044 | | Status, Inquiry_Mode |

**Description:**

This command is used to read the Inquiry_Mode configuration parameter of the local Bluetooth device. See Section 6.5 on page 400.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Inquiry_Mode command succeeded. |
| 0x01-0xFF | Read_Inquiry_Mode command failed. See "Error Codes" on page 331 [Part D] for list of Error Codes. |

*Inquiry_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Inquiry Result event format |
| 0x01 | Inquiry Result format with RSSI |
| 0x02 | Inquiry Result with RSSI format or Extended Inquiry Result format |
| 0x03-0xFF | Reserved |

**Event(s) generated (unless masked away):**

When the Read_Inquiry_Mode command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 732 of 1420
IPR2025-00103

### 7.3.50 Write Inquiry Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Mode | 0x0045 | Inquiry_Mode | Status |

**Description:**

This command is used to write the Inquiry_Mode configuration parameter of the local Bluetooth device. See Section 6.5 on page 400.

**Command Parameters:**

*Inquiry_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Standard Inquiry Result event format (default) |
| 0x01 | Inquiry Result format with RSSI |
| [0x02 | Inquiry Result with RSSI format or Extended Inquiry Result format |
| 0x03-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Inquiry_Mode command succeeded. |
| 0x01-0xFF | Write_Inquiry_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Inquiry_Mode command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 733 of 1420
IPR2025-00103

### 7.3.51 Read Page Scan Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Scan_Type | 0x0046 | | Status, Page_Scan_Type |

**Description:**

This command is used to read the Page Scan Type configuration parameter of the local Bluetooth device. See Section 6.11 on page 403. For details, see the Baseband Specification, "Page scan substate" on page 146 [Part B].

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Page_Scan_Type command succeeded. |
| 0x01-0xFF | Read_Page_Scan_Type command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Page_Scan_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| 0x02-0xFF | Reserved |

**Event(s) generated (unless masked away):**

When the Read_Page_Scan_Type command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 734 of 1420
IPR2025-00103

### 7.3.52 Write Page Scan Type Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Scan_Type | 0x0047 | Page_Scan_Type | Status |

**Description:**

This command is used to write the Page Scan Type configuration parameter of the local Bluetooth device. See Section 6.11 on page 403. For details, see the Baseband Specification, "Page scan substate" on page 146 [Part B].

**Command Parameters:**

*Page_Scan_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Mandatory: Standard Scan (default) |
| 0x01 | Optional: Interlaced Scan |
| 0x02-0xFF | Reserved |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Page_Scan_Type command succeeded. |
| 0x01-0xFF | Write_Page_Scan_Type command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Page_Scan_Type command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 735 of 1420
IPR2025-00103

### 7.3.53 Read AFH Channel Assessment Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| Read_AFH_Channel_Assessment_Mode | 0x0048 | | Status, AFH_Channel_Assessment_Mode |

**Description:**

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 features.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_AFH_Channel_Assessment_Mode command succeeded. |
| 0x01-0xFF | Read_AFH_Channel_Assessment_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*AFH_Channel_Assessment_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Controller channel assessment disabled. |
| 0x01 | Controller channel assessment enabled. |
| 0x02-0xFF | Reserved for future use. |

**Event(s) generated (unless masked away):**

When the Read_AFH_Channel_Assessment_Mode command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 736 of 1420
IPR2025-00103

### 7.3.54 Write AFH Channel Assessment Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| Write_AFH_Channel _Assessment_Mode | 0x0049 | AFH_Channel _Assessment_Mode | Status |

**Description:**

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.

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 state 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 features.

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 of 'Channel Assessment Not Supported' should be returned. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions.

If the Controller supports a channel assessment scheme then the default AFH_Channel_Assessment_Mode is enabled, otherwise the default is disabled.

**Command Parameters:**

*AFH_Channel_Assessment_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Controller channel assessment disabled. |
| 0x01 | Controller channel assessment enabled. |
| 0x02-0xFF | Reserved for future use. |



LifeScan, Inc. Ex. 1019
Page 737 of 1420
IPR2025-00103

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_AFH_Channel_Assessment_Mode command succeeded. |
| 0x01-0xFF | Write_AFH_Channel_Assessment_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_AFH_Channel_Assessment_Mode command has completed, a Command Complete event will be generated.

### 7.3.55 Read Extended Inquiry Response Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Extended_Inquiry_Response | 0x0051 | | Status, FEC_Required, Extended_Inquiry_Response |

**Description:**

The Read_Extended_Inquiry_Response command will read the extended inquiry response to be sent during the extended inquiry response procedure. The FEC_Required parameter states if FEC encoding is required.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Extended_Inquiry_Response command succeeded. |
| 0x01 - 0xFF | Read_Extended_Inquiry_Response command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*FEC_Required:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | FEC is not required |
| 0x01 | FEC is required |
| 0x02-0xFF | Reserved |

*Extended_Inquiry_Response:* *Size: 240 Octets*

| Value | Parameter Description |
|---|---|
| | Extended inquiry response data as defined in [Vol 3 Part C] Section 8 on page 229. |

**Event(s) generated (unless masked away):**

When the Read_Extended_Inquiry_Response command has completed, a Command Complete event will be generated.

### 7.3.56 Write Extended Inquiry Response Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Extended_Inquiry_Response | 0x0052 | FEC_Required, Extended_Inquiry_Response | Status |

**Description:**

The Write_Extended_Inquiry_Response command will write the extended inquiry response to be sent during the extended inquiry response procedure. The FEC_Required command parameter states if FEC encoding is required. The extended inquiry response data is not preserved over a reset. The initial value of the inquiry response data is all zero octets. The controller shall not interpret the extended inquiry response data.

**Command Parameters:**

*FEC_Required:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | FEC is not required |
| 0x01 | FEC is required |
| 0x02-0xFF | Reserved |

*Extended Inquiry Response:* *Size: 240 Octets*

| Value | Parameter Description |
|---|---|
| | Extended inquiry response data as defined in [Vol 3 Part C] Section 8 on page 229. |
| | All octets zero. **Default** |

**Return Parameters:**

*Status:* *Size: 1 Octet*



LifeScan, Inc. Ex. 1019
Page 739 of 1420
IPR2025-00103

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Extended_Inquiry_Response command succeeded |
| 0x01-0xFF | Write_Extended_Inquiry_Response command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Extended_Inquiry_Response command has completed, a Command Complete event will be generated.

### 7.3.57 Refresh Encryption Key

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Refresh_Encryption_Key | 0x0053 | Connection_Handle | Status |

**Description:**

This command is used by the Host to cause the Controller to refresh the encryption key by pausing and resuming encryption.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for the ACL connection to have the encryption key refreshed on. |

Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Refresh_Encryption_Key command succeeded |
| 0x01 - 0xFF | Refresh_Encryption_Key command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

A Command_Status event is sent from the Controller to the Host when the Controller has started the Refresh Encryption Key procedure. An Encryption Key Refresh Complete event is generated when the Refresh Encryption Key procedure has completed.



LifeScan, Inc. Ex. 1019
Page 740 of 1420
IPR2025-00103

### 7.3.58 Read Simple Pairing Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Simple_Pairing_mode | 0x0055 | | Status, Simple_Pairing_Mode |

**Description:**

This command reads the Simple Pairing mode setting in the Controller.

**Command Parameters:**

None

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Simple_Pairing_Mode command succeeded. |
| 0x01 - 0xFF | Read_Simple_Pairing_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Simple_Pairing_Mode* *Size: 1Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Simple Pairing not set (default) |
| 0x01 | Simple pairing enabled |
| 0x02 - 0xFF | Reserved for future use |

**Event(s) generated (unless masked away):**

When the Read_Simple_Pairing_Mode command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 741 of 1420
IPR2025-00103

### 7.3.59 Write Simple Pairing Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Simple_Pairing_mode | 0x0056 | Simple_Pairing_Mode | Status |

**Description:**

This command enables Simple Pairing mode in the Controller. When Simple Pairing Mode is set to 'enabled' the Link Manager will respond to an LMP_io_capability_req PDU with an LMP_io_capability_res PDU and continue with the subsequent pairing procedure. When Simple Pairing mode is set to 'disabled', the Link Manager will reject an IO capability request. A Host shall not set the Simple Pairing Mode to ‘disabled.’

Until Write_Simple_Pairing_mode is received by the Controller, it shall not support any Simple Pairing sequences, and shall respond with the error code Simple Pairing not Supported by Host. This command shall be written before initiating page scan or paging procedures.

The Link Manager Secure Simple Pairing (Host Support) feature bit shall be set to the Simple Pairing Mode parameter. The default value for Simple_Pairing_Mode shall be 'disabled.' When Simple Pairing Mode is set to 'enabled,' the bit in the LMP features mask indicating support for Secure Simple Pairing (Host Support) shall be set to enabled in subsequent responses to an LMP_feature_req from a remote device.

**Command Parameters:**

*Simple_Pairing_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Simple Pairing disabled (default) |
| 0x01 | Simple pairing enabled |
| 0x02 - 0xFF | Reserved for future use |

**Return Parameters***:*

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Simple_Pairing_Mode command succeeded. |
| 0x01 - 0xFF | Write_Simple_Pairing_Mode command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Simple_Pairing_Mode command has completed, a Command Complete event shall be generated.

### 7.3.60 Read Local OOB Data Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_OOB_Data | 0x0057 | | Status, C, R |

**Description:**

This command is used to obtain a Simple Pairing Hash C and Simple Pairing Randomizer R which are intended to be transferred to a remote device using an OOB mechanism. The controller shall create new values for C and R for each invocation of this command.

**Note**: Each OOB transfer will have unique C and R values so after each OOB transfer this command shall be used to obtain a new set of values for the next OOB transfer.

**Note**: The controller keeps information used to generate these values for later use in the simple pairing process. If the controller is powered off or reset then this information is lost and the values obtained before the power off or reset are invalid.

**Command Parameters:**

None

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Local_OOB_Data command succeeded. |
| 0x01 - 0xFF | Read_Local_OOB_Data command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*C:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX XXXXXXXXXXX XXXXXXXXXX | Simple Pairing Hash C |

*R:* *Size: 16 Octets*

**Event(s) generated (unless masked away):**

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX XXXXXXXXXXX XXXXXXXXXX | Simple Pairing Randomizer R |

When the Read_Local_OOB_Data command has completed, a Command Complete event shall be generated.

### 7.3.61 Read Inquiry Response Transmit Power Level Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Inquiry _Response_Transmit _Power_Level | 0x0058 | | Status, TX_Power |

**Description:**

This command will read the inquiry Transmit Power level used to transmit the FHS and EIR data packets. This can be used directly in the Tx Power Level EIR data type.

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Inquiry_Response_Transmit_Power_Level command succeeded. |
| 0x01 - 0xFF | Read_Inquiry_Response_Transmit_Power_Level command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*TX Power:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Size: 1 Octet (signed integer)<br>Range: $-70 \leq N \leq 20$<br>Units: dBm |

**Event(s) generated (unless masked away):**



LifeScan, Inc. Ex. 1019
Page 744 of 1420
IPR2025-00103

When the Read_Inquiry_Response_Transmit_Power_Level command has completed, a Command Complete event shall be generated.

### 7.3.62 Write Inquiry Transmit Power Level Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Inquiry_Transmit_Power_Level | 0x0059 | TX_Power | Status |

**Description:**

This command is used to write the inquiry transmit power level used to transmit the inquiry (ID) data packets. The Controller should use the supported TX power level closest to the Tx_Power parameter.

**Command Parameters**:

None

*TX Power:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Size: 1 Octet (signed integer)<br>Range: $-70 \leq N \leq 20$<br>Units: dBm |

When the Write_Inquiry_Transmit_Power_Level command has completed, a Command Complete event shall be generated.

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Inquiry_Transmit_Power_Level command succeeded |
| 0x01 - 0xFF | Write_Inquiry_Transmit_Power_Level command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Inquiry_Transmit_Power_Level command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 745 of 1420
IPR2025-00103

### 7.3.63 Send Keypress Notification Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Send_Keypress_Notification | 0x0060 | BD_ADDR, Notification_Type | Status, BD_ADDR |

**Description:**

This command is used during the Passkey Entry protocol by a device with KeyboardOnly IO capabilities. It is used by a Host to inform the remote device when keys have been entered or erased.

**Command Parameters:**

*(BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

*Notification_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0 | Passkey entry started |
| 1 | Passkey digit entered |
| 2 | Passkey digit erased |
| 3 | Passkey cleared |
| 4 | Passkey entry completed |
| 5-255 | Reserved for future use |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Send_Keypress_Notification command succeeded |
| 0x01 - 0xFF | Send_Keypress_Notification command failed |

*(BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

**Event(s) generated (unless masked away):**



LifeScan, Inc. Ex. 1019
Page 746 of 1420
IPR2025-00103

When the Send_Keypress_Notification command has completed, a Command Complete event shall be generated.

### 7.3.64 Read Default Erroneous Data Reporting

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Default_ Erroneous_Data_Reporting | 0x005A | | Status, Erroneous_Data_Reporting |

**Description:**

This command reads the Erroneous_Data_Reporting parameter.

**Command Parameters:**

None.

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Default_Erroneous_Data_Reporting command succeeded. |
| 0x01 - 0xFF | Read_Default_Erroneous_Data_Reporting command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Erroneous_Data_Reporting:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Erroneous data reporting disabled. |
| 0x01 | Erroneous data reporting enabled. |
| 0x02 - 0xFF | Reserved. |

**Event(s) generated (unless masked away):**

When the Write_Default_Erroneous_Data_Reporting command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 747 of 1420
IPR2025-00103

### 7.3.65 Write Default Erroneous Data Reporting

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Default_ Erroneous_Data_Reporting | 0x005B | Erroneous_Data_Reporting | Status |

**Description:**

This command writes the Erroneous_Data_Reporting parameter. The Controller shall set the Packet_Status_Flag as defined in Section 5.4.3, "HCI Synchronous Data Packets," on page 396, depending on the value of this parameter. The new value for the Erroneous_Data_Reporting parameter shall not apply to existing synchronous connections.

**Command Parameters:**

*Erroneous_Data_Reporting:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Erroneous Data reporting disabled. **Default.** |
| 0x01 | Erroneous data reporting enabled. |
| 0x02 - 0xFF | Reserved. |

**Return Parameters**:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Default_Erroneous_Data_Reporting command succeeded. |
| 0x01 - 0xFF | Write_Default_Erroneous_Data_Reporting command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Default_Erroneous_Data_Reporting command has completed, a Command Complete event shall be generated.



LifeScan, Inc. Ex. 1019
Page 748 of 1420
IPR2025-00103

### 7.3.66 Enhanced Flush Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Enhanced_Flush | 0x005F | Connection_Handle, Packet_Type | None |

**Description:**

The Enhanced_Flush command is used to discard all L2CAP packets identified by Packet_Type 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 of the same type in the Controller. The only packet type defined is automatically-flushable. Packets not identified by Packet_Type will not be flushed and will be processed normally by the Controller.

After flushing the packets, all data that is sent to the Controller for the same Connection Handle and packet type will be discarded by the Controller until an HCI Data Packet with the start Packet_Boundary_Flag (0x00 or 0x02) is received. This command will allow higher-level software to control how long the baseband should try to retransmit a baseband packet of a specific type for a Connection Handle before all data of that type currently pending for transmission in the Controller should be flushed. Note that the Enhanced Flush command is used for ACL-U connections ONLY. The Flush command can be used to flush all packets (see Section 7.3.4 on page 508). In addition to the Enhanced Flush and Flush commands, the automatic flush timers (see Section 7.3.29 on page 536) can be used to automatically flush an automatically-flushable L2CAP packet that is currently being transmitted after the specified flush timer has expired.

This command shall be supported by a device that declares support for the Non-Flushable Packet Boundary Flag feature.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection.<br>Range: 0x0000 - 0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 749 of 1420
IPR2025-00103

*Packet_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Automatically flushable only. |
| 0x01 - 0xFF | Reserved for future use. |

**Return Parameters**:

None

**Event(s) generated (unless masked away):**

When the controller receives the Enhanced_Flush command, the Controller shall send the Command Status event to the host. In addition, when all the packets identified by Packet_Type have been flushed for the specified Connection Handle, the controller shall send an Enhanced Flush Complete event to the host. The controller may send the Enhanced Flush Complete event immediately after flushing all the packets of type Packet_Type for the specified Connection Handle, or it may wait until all packets for the specified Connection Handle, independent of Packet_Type, buffered in the controller at the time of the receipt of the Enhanced Flush Command, have been either flushed or transmitted.

Note: No Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Enhanced Flush Complete event will indicate that this command has been completed.

## 7.4 INFORMATIONAL PARAMETERS

The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the Bluetooth device and the capabilities of the Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters. For Informational Parameters Commands, the OGF is defined as 0x04.

### 7.4.1 Read Local Version Information Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Version_ Information | 0x0001 | | Status,<br>HCI Version,<br>HCI Revision,<br>LMP Version,<br>Manufacturer_Name,<br>LMP Subversion |

**Description:**

This command will read the values for the version information for the local Bluetooth device.

The HCI Version information defines the version information of the HCI layer. The LMP Version information defines the version of the LMP. The Manufacturer_Name information indicates the manufacturer of the local device.

The HCI Revision and LMP Subversion are implementation dependent.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Local_Version_Information command succeeded. |
| 0x01-0xFF | Read_Local_Version_Information command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |



LifeScan, Inc. Ex. 1019
Page 751 of 1420
IPR2025-00103

*HCI_Version:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Bluetooth Assigned Numbers | |

*HCI_Revision:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Revision of the Current HCI in the Bluetooth device. |

*LMP_Version:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Version of the Current LMP in the Bluetooth device.<br>See Bluetooth Assigned Numbers |

*Manufacturer_Name:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Manufacturer Name of the Bluetooth device.<br>See Bluetooth Assigned Numbers |

*LMP_Subversion:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Subversion of the Current LMP in the Bluetooth device, see Table 5.2 on page 313 in the Link Manager Protocol for assigned values (SubVersNr). |

**Event(s) generated (unless masked away):**

When the Read_Local_Version_Information command has completed, a Command Complete event will be generated.

### 7.4.2 Read Local Supported Commands Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_ Read_Local_Supported_Commands | 0x0002 | | Status, Supported Commands |

**Description:**

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 Section 6.26 on page 410 for more information.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0 | Read Local Supported Commands command succeeded |
| 0x01-0xff | Read Local Supported Commands command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Supported Commands:* *Size: 64 Octets*

| Value | Parameter Description |
|---|---|
| | 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.<br>See section 6.26, "Supported Commands," on page 410. |

**Event(s) generated (unless masked away):**

When the Read Local Supported Commands command has completed a Command Complete event will be generated.

### 7.4.3 Read Local Supported Features Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Local_Supported_Features | 0x0003 | | Status,<br>LMP_Features |

**Description:**

This command requests a list of the supported features for the local device. This command will return a list of the LMP features. For details see "Link Manager Protocol" on page 203 [Part C].

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Local_Supported_Features command succeeded. |
| 0x01-0xFF | Read_Local_Supported_Features command failed. See "Error Codes" on page 331 [Part D]. |

*LMP_Features:* *Size: 8 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX XXXXXXXX | Bit Mask List of LMP features. For details see "Link Manager Protocol" on page 203 [Part C]. |

**Event(s) generated (unless masked away):**

When the Read_Local_Supported_Features command has completed, a Command Complete event will be generated.

### 7.4.4 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 |

**Description:**

The HCI_Read_Local_Extended_Features command returns the requested page of the extended LMP features.

**Command Parameters:**

*Page Number:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Requests the normal LMP features as returned by HCI_Read_Local_Supported_Features. |
| 0x01-0xFF | Return the corresponding page of features. |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | HCI_Read_Local_Extended_Features command succeeded |
| 0x01-0xFF | HCI_Read_Local_Extended_Features command failed. See "Error Codes" on page 331 [Part D] for list of error codes. |

*Page Number:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The normal LMP features as returned by HCI_Read_Local_Supported_Features. |
| 0x01-0xFF | The page number of the features returned. |

*Maximum Page Number:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00-0xFF | The highest features page number which contains non-zero bits for the local device. |

*Extended_LMP_Features:* *Size: 8 Octets*

| Value | Parameter Description |
|---|---|



LifeScan, Inc. Ex. 1019
Page 755 of 1420
IPR2025-00103

| | |
|---|---|
| 0xFFFFFFFFFFFFFFFF | Bit map of requested page of LMP features. See LMP specification for details. |

**Event(s) generated (unless masked away):**

When the Controller receives the HCI_Read_Local_Extended_Features command the Controller sends the Command Complete command to the Host containing the requested information.



LifeScan, Inc. Ex. 1019
Page 756 of 1420
IPR2025-00103

### 7.4.5 Read Buffer Size Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Buffer_Size | 0x0005 | | Status,<br>HC_ACL_Data_Packet_Length,<br>HC_Synchronous_Data_Packet_Length,<br>HC_Total_Num_ACL_Data_Packets,<br>HC_Total_Num_Synchronous_Data_Packets |

**Description:**

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 baseband packets by the Link Manager. 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.

Note: 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.

**Command Parameters:**

None.



LifeScan, Inc. Ex. 1019
Page 757 of 1420
IPR2025-00103

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Buffer_Size command succeeded. |
| 0x01-0xFF | Read_Buffer_Size command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*HC_ACL_Data_Packet_Length:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Total number of HCI Synchronous Data Packets that can be stored in the data buffers of the Controller. |

**Event(s) generated (unless masked away):**

When the Read_Buffer_Size command has completed, a Command Complete event will be generated.

### 7.4.6 Read BD_ADDR Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_BD_ADDR | 0x0009 | | Status,<br>BD_ADDR |

**Description:**

This command shall read the Bluetooth device address (BD_ADDR). See the "Baseband Specification" on page 53 [Part B] for details of how BD_ADDR is used.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_BD_ADDR command succeeded. |
| 0x01-0xFF | Read_BD_ADDR command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device |

**Event(s) generated (unless masked away):**

When the Read_BD_ADDR command has completed, a Command Complete event will be generated.

## 7.5 STATUS PARAMETERS

The Controller modifies all status parameters. These parameters provide information about the current state of the Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters other than to reset certain specific parameters. For the Status and baseband, the OGF is defined as 0x05.

### 7.5.1 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 |

**Description:**

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 Section 6.15 on page 405.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the Failed Contact Counter should be read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Failed_Contact_Counter command succeeded. |
| 0x01-0xFF | Read_Failed_Contact_Counter command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |



LifeScan, Inc. Ex. 1019
Page 760 of 1420
IPR2025-00103

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the Failed Contact Counter has been read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Failed_Contact_Counter:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Number of consecutive failed contacts for a connection corresponding to the Connection Handle. |

**Event(s) generated (unless masked away):**

When the Read_Failed_Contact_Counter command has completed, a Command Complete event will be generated.

### 7.5.2 Reset Failed Contact Counter Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Reset_Failed_ Contact_Counter | 0x0002 | Connection_Handle | Status, Connection_Handle |

**Description:**

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 Section 6.15 on page 405.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the Failed Contact Counter should be reset.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Reset_Failed_Contact_Counter command succeeded. |
| 0x01-0xFF | Reset_Failed_Contact_Counter command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the Failed Contact Counter has been reset.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Event(s) generated (unless masked away):**

When the Reset_Failed_Contact_Counter command has completed, a Command Complete event will be generated.

### 7.5.3 Read Link Quality Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Link_Quality | 0x0003 | Connection_Handle | Status,<br>Connection_Handle,<br>Link_Quality |

**Description:**

This command will return the value for the Link_Quality for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection. This command will return a Link_Quality value from 0-255, which represents the quality of the link between two Bluetooth devices. The higher the value, the better the link quality is. Each Bluetooth module vendor will determine how to measure the link quality.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the connection for which link quality parameters are to be read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Link_Quality command succeeded. |
| 0x01-0xFF | Read_Link_Quality command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the connection for which the link quality parameter has been read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 763 of 1420
IPR2025-00103

*Link_Quality:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The current quality of the Link connection between the local device and the remote device specified by the Connection Handle.<br>Range: 0x00 – 0xFF<br>The higher the value, the better the link quality is. |

**Event(s) generated (unless masked away):**

When the Read_Link_Quality command has completed, a Command Complete event will be generated.

### 7.5.4 Read RSSI Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_RSSI | 0x0005 | Connection_Handle | Status,<br>Connection_Handle, RSSI |

**Description:**

This command will read the value for the difference between the measured Received Signal Strength Indication (RSSI) and the limits of the Golden Receive Power Range (see Radio Specification Section 4.1.6 on page 43) for a Connection Handle to another Bluetooth device. The Connection_Handle must be a Connection_Handle for an ACL connection. Any positive RSSI value returned by the Controller indicates how many dB the RSSI is above the upper limit, any negative value indicates how many dB the RSSI is below the lower limit. The value zero indicates that the RSSI is inside the Golden Receive Power Range.

Note: how accurate the dB values will be depends on the Bluetooth hardware. The only requirements for the hardware are that the Bluetooth 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, which 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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the RSSI is to be read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

### Return Parameters:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_RSSI command succeeded. |
| 0x01-0xFF | Read_RSSI command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the RSSI has been read.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*RSSI:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | Size: 1 Octet (signed integer)<br>Range: -128 ≤ N ≤ 127<br>Units: dB |

### Event(s) generated (unless masked away):

When the Read_RSSI command has completed, a Command Complete event will be generated.

### 7.5.5 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 |

**Description:**

This command will return the values for the AFH_Mode and AFH_Channel_Map 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 baseband 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.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the Channel Map is to be read.<br>Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use) |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_AFH_Channel_Map command succeeded. |
| 0x01-0xFF | Read_AFH_Channel_Map command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |



LifeScan, Inc. Ex. 1019
Page 766 of 1420
IPR2025-00103

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the Channel Map is to be read.<br>Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use) |

*AFH_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | AFH disabled. |
| 0x01 | AFH enabled. |
| 0x02-0xFF | Reserved for future use. |

*AFH_Channel_Map:* *Size: 10 Octets (79 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX XXXXXXXX XXX | If AFH_Mode is AFH enabled then this parameter contains 79 1-bit fields, otherwise the contents are reserved.<br>The $n^{th}$ such field (in the range 0 to 78) contains the value for channel *n*:<br>Channel *n* is unused = 0<br>Channel *n* is used = 1<br>Range: 0x00000000000000000000-0x7FFFFFFFFFFFFFFFFFFF (0x80000000000000000000-0xFFFFFFFFFFFFFFFFFFFF Reserved for future use) |

**Event(s) generated (unless masked away):**

When the Read_AFH_Channel_Map command has completed, a Command Complete event will be generated.

### 7.5.6 Read Clock Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Clock | 0x0007 | Connection_Handle<br>Which_Clock | Status, Connection_Handle, Clock, Accuracy |

**Description:**

This command will read the estimate of the value of the Bluetooth Clock.

If the Which_Clock value is 0, then the Connection_Handle shall be ignored, and the local Bluetooth Clock value shall be returned, and the accuracy parameter shall be set to 0.

If the Which_Clock 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 Bluetooth Clock of this device shall be returned. If the current role is Slave, then an estimate of the Bluetooth Clock 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.

Note: The Bluetooth Clock has a minimum accuracy of 250ppm, or about 22 seconds drift in one day.

Note: See Baseband Specification, Section 1.1, on page 60 for more information about the Bluetooth Clock.

**Command Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify which connection to be used for reading the masters Bluetooth Clock.<br>Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Which_Clock:* *Size 1 Octet)*

| Value | Parameter Description |
|---|---|
| 0xXX | 0x00 = Local Clock (Connection Handle does not have to be a valid handle)<br>0x01 = Piconet Clock (Connection Handle shall be a valid ACL Handle)<br>0x02 to 0xFF = Reserved |



LifeScan, Inc. Ex. 1019
Page 768 of 1420
IPR2025-00103

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_BD_CLOCK command succeeded. |
| 0x01 – 0xFF | Read_BD_CLOCK command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle for the Connection for which the master clock has been read. If the Which_Clock was 0, then the Connection_Handle shall be set to 0 and ignored upon receipt.<br>Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Clock:* *Size: 4 Octets (28 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Bluetooth Clock of the device requested. |

*Accuracy:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | +/- maximum Bluetooth Clock error. Value of 0xFFFF means Unknown.<br>Accuracy = +/- N * 0.3125 msec (1 Bluetooth Clock)<br>Range for N: 0x0000 - 0xFFFE<br>Time Range for N: 0 - 20479.375 msec |

**Event(s) generated (unless masked away):**

When the Read_Clock command has completed, a Command Complete event will be generated.

## 7.6 TESTING COMMANDS

The Testing commands are used to provide the ability to test various functional capabilities of the Bluetooth hardware. These commands provide the ability to arrange various conditions for testing. For the Testing Commands, the OGF is defined as 0x06.

### 7.6.1 Read Loopback Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Loopback_Mode | 0x0001 | | Status, Loopback_Mode |

**Description:**

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 Non-testing Mode operation, the Loopback Mode is set to Non-testing Mode and the path of the information is as specified by the Bluetooth specifications. 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 7.1 on page 602. For details of loopback modes see Section 7.6.2 on page 601.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Loopback_Mode command succeeded. |
| 0x01-0xFF | Read_Loopback_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Loopback_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Loopback mode enabled. **Default.** |
| 0x01 | Enable Local Loopback. |
| 0x02 | Enable Remote Loopback. |
| 0x03-0xFF | Reserved for Future Use. |

**Event(s) generated (unless masked away):**

When the Read_Loopback_Mode command has completed, a Command Complete event will be generated.

### 7.6.2 Write Loopback Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Loopback_Mode | 0x0002 | Loopback_Mode | Status |

**Description:**

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 Non-testing Mode operation, the Loopback Mode is set to Non-testing Mode and the path of the information as specified by the Bluetooth specifications. 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 7.1 on page 602.

When the Bluetooth 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 commands Reset and Write_Loopback_Mode can be used to exit local loopback mode.

If Write_Loopback_Mode 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 comes over the air. It will only allow 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 7.2 on page 602, where the rightmost device is set to Remote Loopback Mode and the leftmost device is set to Non-testing Mode. This allows the Bluetooth Air link to be tested without any other variables.

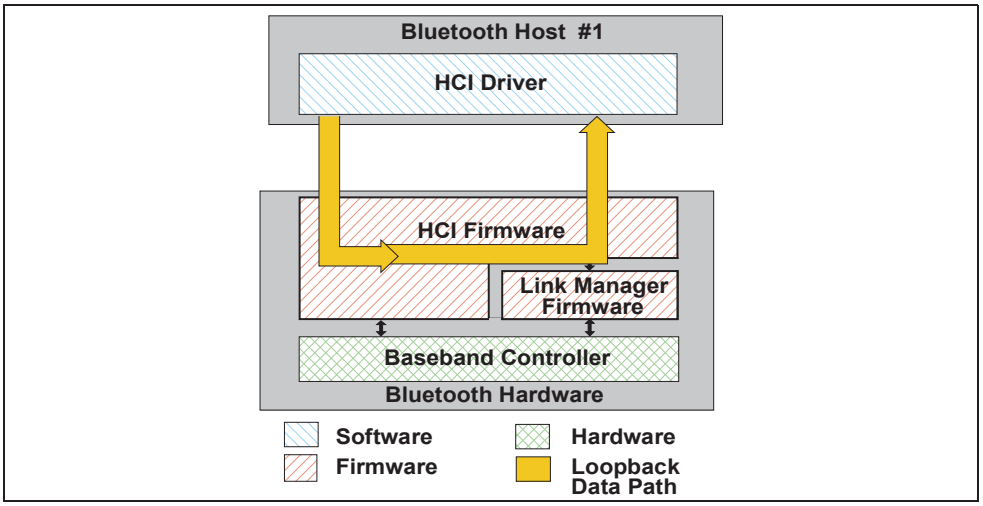



*Figure 7.1: Local Loopback Mode*

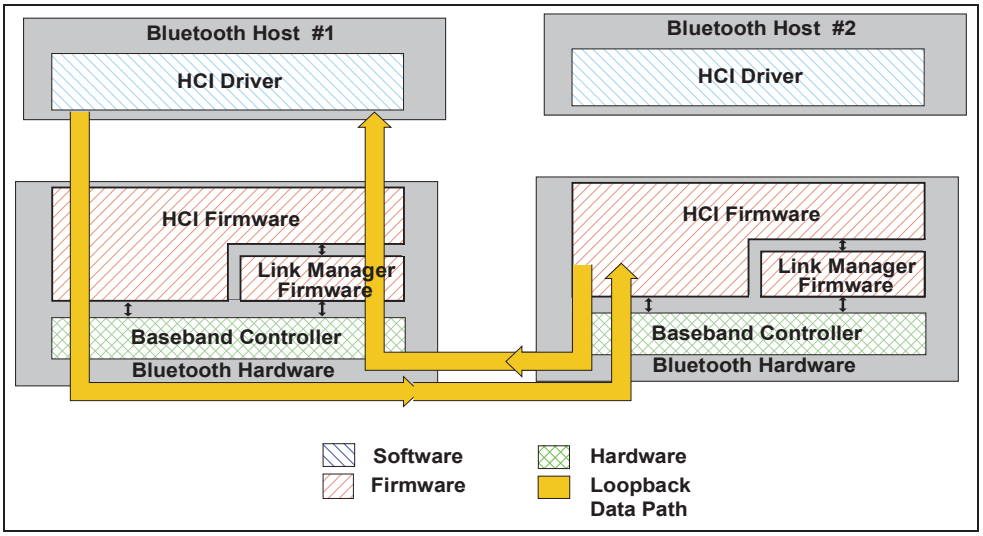



*Figure 7.2: Remote Loopback Mode*



LifeScan, Inc. Ex. 1019
Page 772 of 1420
IPR2025-00103

### Command Parameters:

*Loopback_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Loopback mode enabled. **Default.** |
| 0x01 | Enable Local Loopback. |
| 0x02 | Enable Remote Loopback. |
| 0x03-0xFF | Reserved for Future Use. |

### Return Parameters:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Loopback_Mode command succeeded. |
| 0x01-0xFF | Write_Loopback_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the Write_Loopback_Mode command has completed, a Command Complete event will be generated.



LifeScan, Inc. Ex. 1019
Page 773 of 1420
IPR2025-00103

### 7.6.3 Enable Device Under Test Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Enable_Device_Under_ Test_Mode | 0x0003 | | Status |

**Description:**

The Enable_Device_Under_Test_Mode command will allow the local Bluetooth module to enter test mode via LMP test commands. For details see "Link Manager Protocol" on page 203 [Part C]. The Host issues this command when it wants the local device to be the DUT for the Testing scenarios as described in the "Test Methodology" on page 249[vol. 3]. 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 Device Under Test mode. To disable and exit the Device Under Test Mode, the Host can issue the HCI_Reset command. This command prevents remote Bluetooth devices from causing the local Bluetooth device to enter test mode without first issuing this command.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Enter_Device_Under_Test_Mode command succeeded. |
| 0x01-0xFF | Enter_Device_Under_Test_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Enter_Device_Under_Test_Mode command has completed, a Command Complete event will be generated.

### 7.6.4 Write Simple Pairing Debug Mode Command

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Simple_Pairing_ Debug_mode | 0x0004 | Simple_Pairing_Debug_ Mode | Status |

**Description:**

This command configures the Controller to use a predefined Diffie Hellman private key for Simple Pairing to enable debug equipment to monitor the encrypted connection.

Note: only one side (initiator or responder) needs to set simple pairing debug mode in order for debug equipment to be able to determine the link key and, therefore, be able to monitor the encrypted connection.

When the Simple_Pairing_Debug_Mode parameter is set to enabled the Controller shall use the predefined Diffie Hellman private key. The Controller shall also set the resulting Link_Key type to "Debug Combination Key."

When in Simple Pairing debug mode, the Link Manager shall use the following Diffie Hellman private / public key pair:

- Private key: 07915f86918ddc27005df1d6cf0c142b625ed2eff4a518ff
- Public key (X): 15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
- Public key (Y): b09d42b81bc5bd009f79e4b59dbbaa857fca856fb9f7ea25

**Command Parameters:**

*Simple_Pairing_Debug_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Simple Pairing debug mode disabled (default) |
| 0x01 | Simple pairing debug mode enabled |
| 0x02 - 0xFF | Reserved for future use |

**Return Parameters***:*

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Simple_Pairing_Mode command succeeded. |
| 0x01 - 0xFF | Write_Simple_Pairing_Mode command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Simple_Pairing_Debug_Mode command has completed, a Command Complete event shall be generated.

## 7.7 EVENTS

### 7.7.1 Inquiry Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Inquiry Complete | 0x01 | Status |

**Description:**

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.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Inquiry command completed successfully. |
| 0x01-0xFF | Inquiry command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |



LifeScan, Inc. Ex. 1019
Page 776 of 1420
IPR2025-00103

### 7.7.2 Inquiry Result Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Inquiry Result | 0x02 | Num_Responses,<br>BD_ADDR[i],<br>Page_Scan_Repetition_Mode[i],<br>Reserved[i],<br>Reserved[i],<br>Class_of_Device[i]<br>Clock_Offset[i] |

**Description:**

The Inquiry Result event indicates that a Bluetooth device or multiple Bluetooth 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 Bluetooth devices information in one Inquiry Result event. The event can be used to return one or more Inquiry responses in one event.

**Event Parameters:**

*Num_Responses:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of responses from the Inquiry. |

*BD_ADDR[i]:* *Size: 6 Octets * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for each device which responded. |



LifeScan, Inc. Ex. 1019
Page 777 of 1420
IPR2025-00103

*Page_Scan_Repetition_Mode[i]:* *Size: 1 Octet * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| 0x03 – 0xFF | Reserved |

*Reserved[i]:* [1] *Size: 1 Octet * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXX | Reserved. |

*Reserved[i]:* [2] *Size: 1 Octet * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXX | Reserved, must be set to 0x00. |

*Class_of_Device[i]:* *Size: 3 Octets * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device |

*Clock_Offset[i]:* *Size: 2 Octets * Num_Responses*

| Bit format | Parameter Description |
|---|---|
| Bit 14-0 | Bit 16-2 of CLKslave-CLKmaster. |
| Bit 15 | Reserved |

1. This was the Page_Scan_Period_Mode parameter in the v1.1 specification. This parameter has no meaning in v1.2 or later and no default value.
2. This was the Page_Scan_Mode parameter in the v1.1 specification.

### 7.7.3 Connection Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Connection Complete | 0x03 | Status,<br>Connection_Handle,<br>BD_ADDR,<br>Link_Type,<br>Encryption_Enabled |

**Description:**

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, which issued the Create Connection, or Accept_Connection_Request or Reject_Connection_Request command and then received a Command Status event, if the issued command failed or was successful.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully completed. |
| 0x01-0xFF | Connection failed to Complete. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection between two Bluetooth devices. The Connection Handle is used as an identifier for transmitting and receiving voice or data.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the other connected Device forming the connection. |

*Link_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SCO connection. |
| 0x01 | ACL connection (Data Channels). |
| 0x02-0xFF | Reserved for future use. |

*Encryption_Enabled:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Link level encryption disabled. |
| 0x01 | Link level encryption enabled. |
| 0x02-0xFF | Reserved for future use. |

### 7.7.4 Connection Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Connection Request | 0x04 | BD_ADDR,<br>Class_of_Device,<br>Link_Type |

**Description:**

The Connection Request event is used to indicate that a new incoming connection is trying to be established. The connection may either be accepted or rejected. If this event is masked away and there is an incoming connection attempt and the Controller is not set to auto-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 timer Conn_Accept_Timeout 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 Accept_Connection_Request. If the Event is responded to with Accept_Connection_Request, then the default parameter settings of the Accept_Synchronous_Connection_Request (see Section 7.1.27 on page 459) should be used by the local LM when negotiating the SCO or eSCO link parameters. In that case, the Connection_Complete Event and not the Synchronous_Connection_Complete Event, shall be returned on completion of the connection.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device that requests the connection. |



LifeScan, Inc. Ex. 1019
Page 780 of 1420
IPR2025-00103

*Class_of_Device:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device, which requests the connection. |
| 0x000000 | Unknown Class of Device |

*Link_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SCO Connection requested |
| 0x01 | ACL Connection requested |
| 0x02 | eSCO Connection requested |
| 0x03-0xFF | Reserved for Future Use. |



LifeScan, Inc. Ex. 1019
Page 781 of 1420
IPR2025-00103

### 7.7.5 Disconnection Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Disconnection Complete | 0x05 | Status,<br>Connection_Handle, Reason |

**Description:**

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 and there was a parameter error, or the command was not presently allowed, or a Connection Handle that didn't correspond to a connection was given.

Note: 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.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Disconnection has occurred. |
| 0x01-0xFF | Disconnection failed to complete. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle which was disconnected.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Reason:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reason for disconnection. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |

### 7.7.6 Authentication Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Authentication Complete | 0x06 | Status, Connection_Handle |

**Description:**

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.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication Request successfully completed. |
| 0x01-0xFF | Authentication Request failed to complete. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for which Authentication has been performed.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 783 of 1420
IPR2025-00103

### 7.7.7 Remote Name Request Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Remote Name Request Complete | 0x07 | Status,<br>BD_ADDR, Remote_Name |

**Description:**

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 which device the user-friendly name was obtained from.

Note: The Remote_Name Parameter is a string parameter. Endian-ness does therefore not apply to the Remote_Name Parameter. The first octet of the name is received first.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Remote_Name_Request command succeeded. |
| 0x01-0xFF | Remote_Name_Request command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the device whose name was requested. |

*Remote_Name:* *Size: 248 Octets*

| Value | Parameter Description |
|---|---|
| Name[248] | A UTF-8 encoded user-friendly descriptive name for the remote device.<br>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. |



LifeScan, Inc. Ex. 1019
Page 784 of 1420
IPR2025-00103

### 7.7.8 Encryption Change Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Encryption Change | 0x08 | Status,<br>Connection_Handle,<br>Encryption_Enabled |

**Description:**

The Encryption Change event is used to indicate that the change of the encryption mode has been completed. The Connection_Handle will be a Connection_Handle for an ACL connection. The Encryption_Enable event parameter specifies the new Encryption Enabled 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. Note that this event shall not be generated if encryption is paused or resumed; during a role switch, for example.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Encryption Change has occurred. |
| 0x01-0xFF | Encryption Change failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for which the link layer encryption has been enabled/disabled for all Connection Handles with the same Bluetooth device endpoint as the specified Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Encryption_Enabled:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Link Level Encryption is OFF. |
| 0x01 | Link Level Encryption is ON. |
| 0x02-0xFF | Reserved. |

### 7.7.9 Change Connection Link Key Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Change Connection Link Key Complete | 0x09 | Status, Connection_Handle |

**Description:**

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 which issued the Change_Connection_Link_Key command.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Change_Connection_Link_Key command succeeded. |
| 0x01-0xFF | Change_Connection_Link_Key command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle which the Link Key has been change for all Connection Handles with the same Bluetooth device end point as the specified Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 786 of 1420
IPR2025-00103

### 7.7.10 Master Link Key Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Master Link Key Complete | 0x0A | Status,<br>Connection_Handle, Key_Flag |

**Description:**

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. The Key_Flag event parameter is used to indicate which Link Key (temporary link key of the Master, or the semi-permanent link keys) is now being used in the piconet.

Note: 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.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Master_Link_Key command succeeded. |
| 0x01-0xFF | Master_Link_Key command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for which the Link Key has been changed for all devices in the same piconet.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Key_Flag:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Using Semi-permanent Link Key. |
| 0x01 | Using Temporary Link Key. |



LifeScan, Inc. Ex. 1019
Page 787 of 1420
IPR2025-00103

### 7.7.11 Read Remote Supported Features Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Read Remote Supported Features Complete | 0x0B | Status, Connection_Handle, LMP_Features |

**Description:**

The Read Remote Supported Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote Bluetooth 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. For details see "Link Manager Protocol" on page 203[vol. 2].

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Remote_Supported_Features command succeeded. |
| 0x01-0xFF | Read_Remote_Supported_Features command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle which is used for the Read_Remote_Supported_Features command.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*LMP_Features:* *Size: 8 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX XXXXXXXX | Bit Mask List of LMP features. See "Link Manager Protocol" on page 203 [Part C]. |



LifeScan, Inc. Ex. 1019
Page 788 of 1420
IPR2025-00103

### 7.7.12 Read Remote Version Information Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Read Remote Version Information Complete | 0x0C | Status,<br>Connection_Handle,<br>LMP_Version,<br>Manufacturer_Name,<br>LMP_Subversion |

**Description:**

The Read Remote Version Information Complete event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle is for an ACL connection. The LMP_Version event parameter defines the specification version of the Bluetooth device. The Manufacturer_Name event parameter indicates the manufacturer of the remote Bluetooth device. The LMP_Subversion event parameter is controlled by the manufacturer and is implementation dependant. The LMP_Subversion event parameter defines the various revisions that each version of the Bluetooth hardware will go through as design processes change and errors are fixed. This allows the software to determine what Bluetooth hardware is being used and, if necessary, to work around various bugs in the hardware.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Remote_Version_Information command succeeded. |
| 0x01-0xFF | Read_Remote_Version_Information command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle which is used for the Read_Remote_Version_Information command.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 789 of 1420
IPR2025-00103

*LMP_Version:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Version of the Current LMP in the remote Bluetooth device. For LMP_Version information, see Bluetooth Assigned Numbers. |

*Manufacturer_Name:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Manufacturer Name of the remote Bluetooth device, see Table 5.2 on page 313 in the Link Manager Protocol for assigned values (CompId). |

*LMP_Subversion:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Subversion of the Current LMP in the remote Bluetooth device, see Table 5.2 on page 313 in the Link Manager Protocol for assigned values (SubVersNr). |

### 7.7.13 QoS Setup Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| QoS Setup Complete | 0x0D | Status,<br>Connection_Handle,<br>Flags,<br>Service_Type,<br>Token_Rate,<br>Peak_Bandwidth,<br>Latency,<br>Delay_Variation |

**Description:**

The QoS Setup Complete event is used to indicate the completion of the process of the Link Manager setting up QoS with the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. For more detail see "Logical Link Control and Adaptation Protocol Specification" on page 13[vol. 3].

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | QoS_Setup command succeeded. |
| 0x01-0xFF | QoS_Setup command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle which is used for the QoS_Setup command.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Flags:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 – 0xFF | Reserved for Future Use. |

*Service_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic Available. |
| 0x01 | Best Effort Available. |
| 0x02 | Guaranteed Available. |
| 0x03-0xFF | Reserved for Future Use. |

*Token_Rate:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Token Rate, in octets per second. |

*Peak_Bandwidth:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Peak Bandwidth, in octets per second. |

*Latency:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Latency, in microseconds. |

*Delay_Variation:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Available Delay Variation, in microseconds. |

### 7.7.14 Command Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Command Complete | 0x0E | Num_HCI_Command_Packets, Command_Opcode, Return_Parameters |

**Description:**

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 the Num_HCI_Command_Packets event parameter is set to 1 or more. Command_Opcode, 0x0000 is a NOP (No OPeration), 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.

**Event Parameters:**

*Num_HCI_Command_Packets:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | The Number of HCI command packets which are allowed to be sent to the Controller from the Host.<br>Range for N: 0 – 255 |

*Command_Opcode:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Opcode of the command which caused this event. |

*Return_Parameter(s):* *Size: Depends on Command*

| Value | Parameter Description |
|---|---|
| 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. |



LifeScan, Inc. Ex. 1019
Page 792 of 1420
IPR2025-00103

### 7.7.15 Command Status Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Command Status | 0x0F | Status,<br>Num_HCI_Command_Packets,<br>Command_Opcode |

**Description:**

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 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 can not begin to execute (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 the Num_HCI_Command_Packets event parameter is set to 1 or more. Command_Opcode, 0x0000 is a NOP (No OPeration) and can be used to change the number of outstanding HCI command packets that the Host can send before waiting.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Command currently in pending. |
| 0x01-0xFF | Command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Num_HCI_Command_Packets:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXX | The Number of HCI command packets which are allowed to be sent to the Controller from the Host.<br>Range for N: 0 – 255 |

*Command_Opcode:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Opcode of the command which caused this event and is pending completion. |



LifeScan, Inc. Ex. 1019
Page 793 of 1420
IPR2025-00103

### 7.7.16 Hardware Error Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Hardware Error | 0x10 | Hardware_Code |

**Description:**

The Hardware Error event is used to indicate some type of hardware failure for the Bluetooth device. This event is used to notify the Host that a hardware failure has occurred in the Bluetooth device.

**Event Parameters:**

*Hardware_Code:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00-0xFF | These Hardware_Codes will be implementation-specific, and can be assigned to indicate various hardware problems. |

### 7.7.17 Flush Occurred Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Flush Occurred | 0x11 | Connection_Handle |

**Description:**

The Flush Occurred event is used to indicate that, for the specified Connection Handle, the current user data to be transmitted has 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 baseband packet part of an L2CAP packet is flushed, then the rest of the HCI data packets for the L2CAP packet must also be flushed.

**Event Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle which was flushed.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 794 of 1420
IPR2025-00103

### 7.7.18 Role Change Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Role Change | 0x12 | Status,<br>BD_ADDR, New_Role |

**Description:**

The Role Change event is used to indicate that the current Bluetooth role related to the particular connection has changed. This event only occurs when both the remote and local Bluetooth devices have completed their role change for the Bluetooth device associated with the BD_ADDR event parameter. This event allows both affected Hosts to be notified when the Role has been changed.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Role change has occurred. |
| 0x01-0xFF | Role change failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which a role change has completed. |

*New_Role:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Currently the Master for specified BD_ADDR. |
| 0x01 | Currently the Slave for specified BD_ADDR. |



LifeScan, Inc. Ex. 1019
Page 795 of 1420
IPR2025-00103

### 7.7.19 Number Of Completed Packets Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Number Of Completed Packets | 0x13 | Number_of_Handles,<br>Connection_Handle[i],<br>HC_Num_Of_Completed_Packets[i] |

**Description:**

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 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.
Note that Number Of Completed Packets events will not report on synchronous Connection Handles if synchronous Flow Control is disabled. (See Read/Write_Synchronous_Flow_Control_Enable on page 544 and page 545.)

**Event Parameters:**

*Number_of_Handles:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The number of Connection Handles and Num_HCI_Data_Packets parameters pairs contained in this event.<br>Range: 0-255 |

*Connection_Handle[i]: Size: Number_of_Handles * 2 Octets(12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 796 of 1420
IPR2025-00103

*HC_Num_Of_Completed_Packets [i]: Size: Number_of_Handles * 2 Octets*

| Value | Parameter Description |
|---|---|
| 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.<br>Range for N: 0x0000-0xFFFF |

### 7.7.20 Mode Change Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Mode Change | 0x14 | Status,<br>Connection_Handle,<br>]Current_Mode,<br>Interval |

**Description:**

The Mode Change event is used to indicate when the device associated with the Connection Handle changes between Active mode, Hold mode, and 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 which connection the Mode Change event is for. The Current_Mode event parameter is used to indicate which state the connection is currently in. The Interval parameter is used to specify a time amount specific to each state. Each Controller that is associated with the Connection Handle which has changed Modes will send the Mode Change event to its Host.

**Event Parameters:**

*Status: Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | A Mode Change has occurred. |
| 0x01-0xFF | Hold_Mode, Sniff_Mode, Exit_Sniff_Mode, Park_State, or Exit_Park_State command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle: Size: 2 Octets(12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 797 of 1420
IPR2025-00103

*Current_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Active Mode. |
| 0x01 | Hold Mode. |
| 0x02 | Sniff Mode. |
| 0x03 | Park State |
| 0x04-0xFF | Reserved for future use. |

*Interval:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Hold:<br>Number of Baseband slots to wait in Hold Mode.<br>Hold Interval = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0002-0xFFFE<br>Time Range: 1.25 msec-40.9 sec<br><br>Sniff:<br>Number of Baseband slots between sniff intervals.<br>Time between sniff intervals = 0.625 msec (1 Baseband slot)<br>Range for N: 0x0002-0xFFFE<br>Time Range: 1.25 msec-40.9 sec<br><br>Park:<br>Number of Baseband slots between consecutive beacons.<br>Interval Length = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0002-0xFFFE<br>Time Range: 1.25 msec-40.9 Seconds |

### 7.7.21 Return Link Keys Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Return Link Keys | 0x15 | Num_Keys,<br>BD_ADDR [i], Link_Key[i] |

**Description:**

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. This event shall never return the value of the link keys. The link keys value parameter shall always contain the value of zero.

**Event Parameters:**

*Num_Keys:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of Link Keys contained in this event.<br>Range: 0x01 – 0x0B |

*BD_ADDR [i]:* *Size: 6 Octets * Num_Keys*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the associated Link Key. |

*Link_Key[i]:* *Size:16 Octets * Num_Keys*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX<br>XXXXXXXXXX<br>XXXXXXXXXX | Link Key for the associated BD_ADDR. |

### 7.7.22 PIN Code Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| PIN Code Request | 0x16 | BD_ADDR |

**Description:**

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 or not.

Note: 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 Link Manager to respond to the request from the remote Link Manager (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 Link Manager detects LMP response timeout. (See “Link Manager Protocol” on page 203 [Part C].)

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which a new link key is being created for. |



LifeScan, Inc. Ex. 1019
Page 800 of 1420
IPR2025-00103

### 7.7.23 Link Key Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Link Key Request | 0x17 | BD_ADDR |

**Description:**

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, or the stored Link Key does not meet the security requirements for the requested service, 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.

Note: If the Link Key Request event is masked away, then the Controller will assume that the Host has no additional link keys.

If the Host uses the Link_Key_Request_Negative_Reply Command when the requested service requires an authenticated Link Key and the current Link Key is unauthenticated, the Host should set the Authentication_Requirements parameter one of the to MITM Protection Required options.

When the Controller generates a Link Key Request event in order for the local Link Manager to respond to the request from the remote Link Manager (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 Link Manager detects LMP response timeout. (See “Link Manager Protocol” on page 203 [Part C].)

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device which a stored link key is being requested. |



LifeScan, Inc. Ex. 1019
Page 801 of 1420
IPR2025-00103

### 7.7.24 Link Key Notification Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Link Key Notification | 0x18 | BD_ADDR, Link_Key, Key_Type |

**Description:**

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, debug combination key, unauthenticated combination key, authenticated combination key or changed combination key) that was used during pairing. If pairing with unit key is not supported, the Host can for instance discard the key or disconnect the link.

The combination key Key_Type is used when standard pairing was used. The debug combination key Key_Type is used when Simple Pairing was used and the debug public key is sent or received. The unauthenticated combination key Key_Type is used when the Just Works Simple Pairing association model was used. The authenticated combination key Key_Type is used when Simple Pairing was used and the Just Works association mode was not used. The changed combination key Key_Type is used when the link key has been changed using the Change Connection Link Key procedure and Simple Pairing Mode is set to enabled. Note: it is the responsibility of the Host to remember whether the combination key was a debug, unauthenticated or authenticated Key_Type prior to changing the link key.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the Device for which the new link key has been generated. |

*Link_Key:* *Size: 16 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX | Link Key for the associated BD_ADDR. |

*Key_Type:* *Size: 1 Octets*

| Value | Parameter Description |
|---|---|

| | |
|---|---|
| 0x00 | Combination Key |
| 0x01 | Local Unit Key |
| 0x02 | Remote Unit Key |
| 0x03 | Debug Combination Key |
| 0x04 | Unauthenticated Combination Key |
| 0x05 | Authenticated Combination Key |
| 0x06 | Changed Combination Key |
| 0x07-0xFF | Reserved |



LifeScan, Inc. Ex. 1019
Page 803 of 1420
IPR2025-00103

### 7.7.25 Loopback Command Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Loopback Command | 0x19 | HCI_Command_Packet |

**Description:**

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 section 7.6.1, “Read Loopback Mode Command,” on page 600 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.

Note: 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.

**Event Parameters:**

*HCI_Command_Packet:* *Size: Depends on Command*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | HCI Command Packet, including header. |

### 7.7.26 Data Buffer Overflow Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Data Buffer Overflow | 0x1A | Link_Type |

**Description:**

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.

**Event Parameters:**

*Link_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Synchronous Buffer Overflow (Voice Channels). |
| 0x01 | ACL Buffer Overflow (Data Channels). |
| 0x02-0xFF | Reserved for Future Use. |

### 7.7.27 Max Slots Change Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Max Slots Change | 0x1B | Connection_Handle,<br>LMP_Max_Slots |

**Description:**

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 baseband packets transmitted by the local device, changes. The Connection_Handle will be a Connection_Handle for an ACL connection.

**Event Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*LMP_Max_Slots:* *Size: 1 octet*

| Value | Parameter Description |
|---|---|
| 0x01, 0x03, 0x05 | Maximum number of slots allowed to use for baseband packets, see Section 4.1.10 on page 237 and Section 5.2 on page 313 in "Link Manager Protocol" on page 203 [Part C]. |

### 7.7.28 Read Clock Offset Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Read Clock Offset Complete | 0x1C | Status,<br>Connection_Handle, Clock_Offset |

**Description:**

The Read Clock Offset Complete event is used to indicate the completion of the process of the Link Manager obtaining the Clock Offset information of the Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Clock_Offset command succeeded. |
| 0x01-0xFF | Read_Clock_Offset command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Specifies which Connection Handle's Clock Offset parameter is returned.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Clock_Offset:* *Size: 2 Octets*

| Bit format | Parameter Description |
|---|---|
| Bit 14-0 | Bit 16-2 of CLKslave-CLKmaster. |
| Bit 15 | Reserved. |



LifeScan, Inc. Ex. 1019
Page 806 of 1420
IPR2025-00103

### 7.7.29 Connection Packet Type Changed Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Connection Packet Type Changed | 0x1D | Status,<br>Connection_Handle, Packet_Type |

**Description:**

The Connection Packet Type Changed event is used to indicate that the process has completed of the Link Manager changing 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 Link Manager 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.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection Packet Type changed successfully. |
| 0x01-0xFF | Connection Packet Type Changed failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 807 of 1420
IPR2025-00103

*Packet_Type:* *Size: 2 Octets*

*For ACL_Link_Type*

| Value | Parameter Description |
|---|---|
| 0x0001 | Reserved for future use. |
| 0x0002 | 2-DH1 may not be used. |
| 0x0004 | 3-DH1 may not be used. |
| 0x0008[1] | DM1 may be used. |
| 0x0010 | DH1 may be used. |
| 0x0020 | Reserved for future use. |
| 0x0040 | Reserved for future use. |
| 0x0080 | Reserved for future use. |
| 0x0100 | 2-DH3 may not be used. |
| 0x0200 | 3-DH3 may not be used. |
| 0x0400 | DM3 may be used. |
| 0x0800 | DH3 may be used. |
| 0x1000 | 2-DH5 may not be used. |
| 0x2000 | 3-DH5 may not be used. |
| 0x4000 | DM5 may be used. |
| 0x8000 | DH5 may be used. |

1. This bit will be interpreted as set to 1 by Bluetooth V1.2 or later controllers.

*For SCO_Link_Type*

| Value | Parameter Description |
|---|---|
| 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. |

### 7.7.30 QoS Violation Event

| Event | Event Code | Event Parameters |
|---|---|---|
| QoS Violation | 0x1E | Connection_Handle |

**Description:**

The QoS Violation event is used to indicate the Link Manager is unable to provide the current QoS requirement for the Connection Handle. This event indicates that the Link Manager 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.

**Event Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle that the LM is unable to provide the current QoS requested for.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

### 7.7.31 Page Scan Repetition Mode Change Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Page Scan Repetition Mode Change | 0x20 | BD_ADDR,<br>Page_Scan_Repetition_Mode |

**Description:**

The Page Scan Repetition Mode Change event indicates that the remote Bluetooth device with the specified BD_ADDR has successfully changed the Page_Scan_Repetition_Mode (SR).

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the remote device. |

*Page_Scan_Repetition_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| 0x03 – 0xFF | Reserved. |

### 7.7.32 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 |

**Description:**

The Flow Specification Complete event is used to inform the Host about the Quality of Service 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 field. The flow parameters are defined in the L2CAP specification "Quality of Service (QoS) Option" on page 58[vol. 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 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 success completion).

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Flow Specification command succeeded |
| 0x01 – 0xFF | Flow Specification command failed. See "Error Codes" on page 331 [Part D] for list of Error Codes |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle used to identify for which ACL connection the Flow is specified.<br>Range: 0x0000 - 0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Flags:* *Size: 1 Octet)*

| Value | Parameter Description |
|---|---|
| 0x00 – 0xFF | Reserved for Future Use. |

*Flow_direction:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 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:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | No Traffic |
| 0x01 | Best Effort |
| 0x02 | Guaranteed |
| 0x03 – 0xFF | Reserved for Future Use |

*Token Rate:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Rate in octets per second |

*Token Bucket Size 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Token Bucket Size in octets |

*Peak_Bandwidth:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Peak Bandwidth in octets per second |

*Access Latency:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXX | Access Latency in microseconds |

### 7.7.33 Inquiry Result with RSSI Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Inquiry Result with RSSI | 0x22 | Num_responses,<br>BD_ADDR[i],<br>Page_Scan_Repetition_Mode[i],<br>Reserved[i],<br>Class_of_Device[i],<br>Clock_Offset[i],<br>RSSI[i] |

**Description:**

The Inquiry Result with RSSI event indicates that a Bluetooth device or multiple Bluetooth 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 Bluetooth devices information in one Inquiry Result event. The event can be used to return one or more Inquiry responses in one event. The RSSI parameter is measured during the FHS packet returned by each responding slave.

This event shall only be generated if the Inquiry Mode parameter of the last Write_Inquiry_Mode command was set to 0x01 (Inquiry Result format with RSSI).

**Event Parameters:**

*Num_Responses:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Number of responses from the Inquiry. |

*BD_ADDR[i]:* *Size: 6 Octets * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for each device which responded. |

*Page_Scan_Repetition_Mode[i]:* *Size: 1 Octet* Num_Responses*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| 0x03 – 0xFF | Reserved |

*Reserved[i]:*[1] *Size: 1 Octet* Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXX | Reserved. |

*Class_of_Device[i]:* *Size: 3 Octets * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device |

*Clock_Offset[i]:* *Size: 2 Octets * Num_Responses*

| Bit format | Parameter Description |
|---|---|
| Bit 14-0 | Bit 16-2 of CLKslave-CLKmaster. |
| Bit 15 | Reserved |

*RSSI[i]:* *Size: 1 Octet * Num_Responses*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20<br>Units: dBm |

1. This was the Page_Scan_Period_Mode parameter in the v1.1 specification. This parameter has no meaning in v1.2 or later and no default value.

### 7.7.34 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 |

**Description:**

The Read Remote Extended Features Complete event is used to indicate the completion of the process of the Link Manager obtaining 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. For details see "Link Manager Protocol" on page 203 [Part C].

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Request for remote extended features succeeded |
| 0x01-0xFF | Request for remote extended features failed – standard HCI error value |

*Connection_Handle:* *Size: 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | The Connection Handle identifying the device to which the remote features apply. Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use) |

*Page Number:* *Size: 1Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The normal LMP features as returned by HCI_Read_Remote_Supported_Features |
| 0x01-0xFF | The page number of the features returned |

*Maximum Page Number:* *Size: 1Octet*

| Value | Parameter Description |
|---|---|
| 0x00-0xFF | The highest features page number which contains non-zero bits for the local device |



LifeScan, Inc. Ex. 1019
Page 815 of 1420
IPR2025-00103

*Extended_LMP_Features:* *Size: 8 Octets*

| Value | Parameter Description |
|---|---|
| 0xFFFFFFFFFFFFFFFF | Bit map of requested page of LMP features.<br>See LMP specification for details |

### 7.7.35 Synchronous Connection Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Synchronous Connection Complete | 0x2C | Status,<br>Connection_Handle,<br>BD_ADDR,<br>Link Type,<br>Transmission_Interval,<br>Retransmission Window,<br>Rx_Packet_Length,<br>Tx_Packet_Length<br>Air Mode |

**Description:**

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, which issued the Setup_Synchronous_Connection, or Accept_Synchronous_Connection_Request or Reject_Synchronous_Connection_Request command and then received a Command Status event, if the issued command failed or was successful.

**Event Parameters:**

*Status:* *1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully completed. |
| 0x01 –0xFF | Connection failed to complete. See “Error Codes” on page 331 [Part D] for error codes and descriptions. |

*Connection_Handle:* *2 octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection between two Bluetooth devices.<br>Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |



LifeScan, Inc. Ex. 1019
Page 816 of 1420
IPR2025-00103

*BD_ADDR:* *6 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the other connected device forming the connection. |

*Link_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | SCO Connection |
| 0x01 | Reserved |
| 0x02 | eSCO Connection |
| 0x03 – 0xFF | Reserved |

*Transmission_Interval:* *1 octets*

| Value | Parameter Description |
|---|---|
| 0xXX | Time between two consecutive eSCO instants measured in slots. Must be zero for SCO links. |

*Retransmission window:* *1 octets*

| Value | Parameter Description |
|---|---|
| 0xXX | The size of the retransmission window measured in slots. Must be zero for SCO links. |

*Rx_Packet_Length:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the eSCO payload in the receive direction. Must be zero for SCO links. |

*Tx_Packet_Length:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the eSCO payload in the transmit direction. Must be zero for SCO links. |



LifeScan, Inc. Ex. 1019
Page 817 of 1420
IPR2025-00103

*Air Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | μ-law log |
| 0x01 | A-law log |
| 0x02 | CVSD |
| 0x03 | Transparent Data |
| 0x04 – 0xFF | Reserved |



LifeScan, Inc. Ex. 1019
Page 818 of 1420
IPR2025-00103

### 7.7.36 Synchronous Connection Changed event

| Event | Event Code | Event Parameters |
|---|---|---|
| Synchronous Connection Changed | 0x2D | Status,<br>Connection_Handle,<br>Transmission_Interval,<br>Retransmission Window,<br>Rx_Packet_Length,<br>Tx_Packet_Length |

**Description**:

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.

**Command Parameters:**

*Status:* *1 octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Connection successfully reconfigured. |
| 0x01 –0xFF | Reconfiguration failed to complete.<br>See "Error Codes" on page 331 [Part D] for error codes and descriptions. |

*Connection_Handle:* *2 octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle to be used to identify a connection between two Bluetooth devices.<br>Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Transmission_Interval:* *1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Time between two consecutive SCO/eSCO instants measured in slots. |

*Retransmission window:* *1 octet*

| Value | Parameter Description |
|---|---|
| 0xXX | The size of the retransmission window measured in slots. Must be zero for SCO links. |



LifeScan, Inc. Ex. 1019
Page 819 of 1420
IPR2025-00103

*Rx_Packet_Length:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the SCO/eSCO payload in the receive direction. |

*Tx_Packet_Length:* *2 octets*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Length in bytes of the SCO/eSCO payload in the transmit direction. |

### 7.7.37 Sniff Subrating Event

| Event | Event Code | Event Parameters |
|---|---|---|
| HCI Sniff Subrating | 0x2E | Status<br>Connection_Handle<br>Maximum_Transmit_Latency<br>Maximum_Receive_Latency<br>Minimum_Remote_Timeout<br>Minimum_Local_Timeout |

**Description:**

The Sniff Subrating event indicates that the device associated with the Connection Handle has either enabled sniff subrating or the sniff subrating parameters have been renegotiated by the link manager. The Connection Handle will be a Connection Handle for an ACL connection. The Connection Handle event parameter indicates which connection the Sniff Subrating event is for.

Each Controller that is associated with the Connection Handle that has changed its subrating parameters will send the Sniff Subrating event to its host.

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | The sniff subrating parameters have changed. |
| 0x01 – 0xFF | Sniff Subrating Command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|



LifeScan, Inc. Ex. 1019
Page 820 of 1420
IPR2025-00103

| | |
|---|---|
| 0xXXXX | Connection Handle.<br>Range: 0x0000 – 0x0EFF (0x0F00 – 0x0FFF Reserved for future use) |

*Maximum_Transmit_Latency:* *Size: 2 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum latency for data being transmitted from the local device to the remote device.<br>Latency = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0xFFFE<br>Time Range: 0 sec - 40.9 sec |

*Maximum_Receive_Latency:* *Size: 2 Octet*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Maximum latency for data being received by the local device from the remote device.<br>Latency = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0xFFFE<br>Time Range: 0 sec - 40.9 sec |

*Minimum_Remote_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The base sniff subrate timeout in baseband slots that the remote device shall use.<br>Timeout = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0x8000<br>Time Range: 0 sec – 20.5 sec |

*Minimum_Local_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | The base sniff subrate timeout in baseband slots that the local device will use.<br>Timeout = N * 0.625 msec (1 Baseband slot)<br>Range for N: 0x0000 – 0x8000<br>Time Range: 0 sec – 20.5 sec |



LifeScan, Inc. Ex. 1019
Page 821 of 1420
IPR2025-00103

### 7.7.38 Extended Inquiry Result Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Extended Inquiry Result | 0x2F | Num_Responses,<br>BD_ADDR,<br>Page_Scan_Repetition_Mode,<br>Reserved,<br>Class_of_Device,<br>Clock_Offset,<br>RSSI,<br>Extended_Inquiry_Response |

**Description:**

The Extended Inquiry Result event indicates that a Bluetooth device has responded during the current inquiry process with extended inquiry response data. This event will be sent from the Controller to the Host upon reception of an Extended Inquiry Response from a remote device. One single Extended Inquiry Response is returned per event. This event contains RSSI and inquiry response data for the Bluetooth device that responded to the latest inquiry. The RSSI parameter is measured during the FHS packet returned by each responding slave. The Num_Responses parameter shall be set to one.

This event is only generated if the Inquiry_Mode parameter of the last Write_Inquiry_Mode command was set to 0x02 (Inquiry Result with RSSI format or Extended Inquiry Result format).

Note: This ensures that an old host will never receive the Extended Inquiry Result event.

If an inquiry response packet with the EIR field set to zero is received, the Inquiry Result with RSSI event format is used. If the EIR bit is set to one the Extended Inquiry Result event format is used. If the EIR bit is set to one but the controller failed to receive the extended inquiry response packet, the Extended_Inquiry_Response parameter is set to zeros. If an extended inquiry response packet from the same device is correctly received in a later response, another event shall be generated.

Note: The only difference between the Extended Inquiry Result event and the Inquiry Result with RSSI event is the additional Extended_Inquiry_Response parameter.

Note: The Extended_Inquiry_Response parameter is not interpreted by the controller. The tagged data set by the other host should be passed unaltered if it has been correctly received.

**Event Parameters:**

*Num_Responses:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x01 | Number of responses from the inquiry. The Extended Inquiry Result event always contains a single response. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR for the device that responded. |

*Page_Scan_Repetition_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | R0 |
| 0x01 | R1 |
| 0x02 | R2 |
| 0x03 - 0xFF | Reserved |

*Reserved:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Reserved. |

*Class_of_Device:* *Size: 3 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXX | Class of Device for the device that responded. |

*Clock_Offset* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| Bit 14-0 | Bit 16-2 of CLKslave-CLKmaster. |
| Bit 15 | Reserved. |

*RSSI:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0xXX | Range: -127 to +20 |
| Units: dBm | |

*Extended_Inquiry_Response:* *Size: 240 Octets*

| Value | Parameter Description |
|---|---|
| | Extended inquiry response data as defined in [Vol 3 Part C] Section 8 on page 229 |

### 7.7.39 Encryption Key Refresh Complete

| Command | Event Code | Event Parameters |
|---|---|---|
| Encryption Key Refresh Complete | 0x30 | Status<br>Connection_Handle |

**Description:**

The Encryption Key Refresh Complete event is used to indicate to the Host that the encryption key was refreshed on the given Connection_Handle any time encryption is paused and then resumed. The Controller shall send this event when the encryption key has been refreshed due to encryption being started or resumed.

If the Encryption Key Refresh Complete event was generated due to an encryption pause and resume operation embedded within a change connection link key procedure, the Encryption Key Refresh Complete event shall be sent prior to the Change Connection Link Key Complete event.

If the Encryption Key Refresh Complete event was generated due to an encryption pause and resume operation embedded within a role switch procedure, the Encryption Key Refresh Complete event shall be sent prior to the Role Change event.

**Event parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Encryption Key Refresh completed successfully |
| 0x01 - 0xFF | Encryption Key Refresh failed. See "Error Codes" on page 331 [Part D] for list of Error Codes |

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for the ACL connection to have the encryption key refreshed on.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use). |



LifeScan, Inc. Ex. 1019
Page 824 of 1420
IPR2025-00103

### 7.7.40 IO Capability Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| IO Capability Request | 0x31 | BD_ADDR |

**Description:**

The IO Capability Request event is used to indicate that the IO capabilities of the host are required for a simple pairing process. The host shall respond with an HCI_IO_Capability_Reqeust_Reply command. This event will only be generated if simple pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |



LifeScan, Inc. Ex. 1019
Page 825 of 1420
IPR2025-00103

### 7.7.41 IO Capability Response Event

| Event | Event Code | Event Parameters |
|---|---|---|
| IO Capability Response | 0x32 | BD_ADDR,<br>IO_Capability,<br>OOB_Data_Present,<br>Authentication_Requirements |

**Description:**

The IO Capability Response event is used to indicate to the host that IO capabilities from a remote device specified by BD_ADDR have been received during a simple pairing process. This event will only be generated if simple pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR identifying the device to which the IO capabilities apply. |

*IO_Capability:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | DisplayOnly |
| 0x01 | DisplayYesNo |
| 0x02 | KeyboardOnly |
| 0x03 | NoInputNoOutput |
| 0x04 – 0xFF | Reserved for future use |

*OOB_Data_Present:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | OOB authentication data not present |
| 0x01 | OOB authentication data from remote device present |
| 0x02 – 0xFF | Reserved for future use |

*Authentication_Requirements:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | MITM Protection Not Required – No Bonding. Numeric comparison with automatic accept allowed. |
| 0x01 | MITM Protection Required – No Bonding. Use IO Capabilities to determine authentication procedure |
| 0x02 | MITM Protection Not Required – Dedicated Bonding. Numeric comparison with automatic accept allowed. |
| 0x03 | MITM Protection Required – Dedicated Bonding. Use IO Capabilities to determine authentication procedure |
| 0x04 | MITM Protection Not Required – General Bonding. Numeric Comparison with automatic accept allowed. |
| 0x05 | MITM Protection Required – General Bonding. Use IO capabilities to determine authentication procedure. |
| 0x06 - 0xFF | Reserved for future use |

### 7.7.42 User Confirmation Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| User Confirmation Request | 0x33 | BD_ADDR,<br>Numeric_Value |

**Description:**

The User Confirmation Request event is used to indicate that user confirmation of a numeric value is required. The host shall reply with either the User Confirmation Request Reply or the User Confirmation Request Negative Reply command. If the host has output capability it shall display the Numeric_Value until the Simple Pairing Complete event is received. It shall reply based on the yes/no response from the user. If the host has no input and no output it shall reply with the User Confirmation Request Reply command.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of device involved in Simple Pairing process. |

*Numeric_Value:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|
| 0x00000000 – 0x000F423F | Numeric value to be displayed. Valid values are decimal 000000 – 999999. |



LifeScan, Inc. Ex. 1019
Page 827 of 1420
IPR2025-00103

### 7.7.43 User Passkey Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| User Passkey Request | 0x34 | BD_ADDR |

**Description:**

The User Passkey Request event is used to indicate that a passkey is required as part of a Simple Pairing process. The Host shall respond with either a User_Passkey_Request_Reply or User_Passkey_Request_Negative_Reply command. This event will only be generated if simple pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device involved in simple pairing process. |

### 7.7.44 Remote OOB Data Request Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Remote OOB Data Request | 0x35 | BD_ADDR |

**Description:**

The Remote OOB Data Request event is used to indicate that the Simple Pairing Hash C and the Simple Pairing Randomizer R is required for the Secure Simple Pairing process involving the device identified by BD_ADDR. The C and R values were transferred to the host from the remote device via an OOB mechanism. This event is sent by the controller because the host previously set the OOB Data Present parameter to "OOB authentication data from remote device present" in an IO Capability Request Reply command. The Host shall respond with the Remote OOB Data Request Reply command.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device from which the C and R values were received. |

### 7.7.45 Simple Pairing Complete Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Simple Pairing Complete | 0x36 | Status,<br>BD_ADDR |

**Description:**

The Simple Pairing Complete event is used to indicate that the simple pairing process has completed. A host that is displaying a numeric value can use this event to change its UI.

When the LMP simple pairing sequences fail for any reason, the Simple Pairing Complete event shall be sent to the Host. When Simple Pairing Complete event is sent in response to the IO capability exchange failing, the Status parameter shall be set to the error code received from the remote device. Otherwise, the Status shall be set to the error code “Authentication Failure.”

**Event Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Simple Pairing succeeded |
| 0x01 - 0xFF | Simple Pairing failed. See “Error Codes” on page 331 [Part D] for a list of Error Codes. |

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device involved in simple pairing process. |

### 7.7.46 Link Supervision Timeout Changed Event

| Event | Event Code | Event Parameters |
|---|---|---|
| Link Supervision Timeout Changed | 0x38 | Connection_Handle,<br>Link_Supervision_Timeout |

**Description:**

The Link_Supervision_Timeout_Changed event is used to notify the slave's Host when the Link_Supervision_Timeout parameter is changed in the slave Controller. This event shall only be sent to the Host by the slave controller upon receiving an LMP_supervision_timeout PDU from the master.

Note: the Connection_Handle used for this command shall be the ACL connection of the appropriate device.

**Event Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Link_Supervision_Timeout:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| N = 0xXXXX | Measured in Number of Baseband slots<br>Link_Supervision_Timeout = N*0.625 msec (1 Baseband slot)<br>Range for N: 0x0001 - 0xFFFF<br>Time Range: 0.625ms - 40.9 sec<br>(0 means infinite timeout) |

### 7.7.47 Enhanced Flush Complete Event

| Event | Event Code | Event parameter |
|---|---|---|
| Enhanced Flush Complete | 0x39 | Connection_Handle |

**Description:**

The Enhanced Flush Complete event is used to indicate that an Enhanced Flush is complete for the specified Connection Handle.

**Event Parameters:**

*Connection_Handle:* *Size: 2 Octets (12 bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle.<br>Range: 0x0000 - 0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

### 7.7.48 User Passkey Notification Event

| Event | Event Code | Event Parameters |
|---|---|---|
| User Passkey Notification | 0x3B | BD_ADDR, Passkey |

**Description:**

The User Passkey Notification event is used to provide a passkey for the Host to display to the user as required as part of a Simple Pairing process. The Passkey parameter shall be a randomly generated number (see [Part H], Security Specification, Section 2 on page 857) generated by the Controller modulo 1,000,000.

This event will be generated if the IO capabilities of the local device are DisplayOnly or DisplayYesNo and the IO capabilities of the remote device are KeyboardOnly.

This event will only be generated if simple pairing has been enabled with the HCI_Write_Simple_Pairing_Mode command.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the device involved in simple pairing process. |

*Passkey:* *Size: 4 Octets*

| Value | Parameter Description |
|---|---|

| | |
|---|---|
| 0x00000000 - 0x000F423F | Passkey to be displayed. Valid values are decimal 000000 - 999999. |

### 7.7.49 Keypress Notification Event

| Command | Event Code | Event Parameters |
|---|---|---|
| Keypress Notification | 0x3C | BD_ADDR, Notification_Type |

**Description:**

The Keypress Notification event is sent to the Host after a passkey notification has been received by the Link Manager on the given BD_ADDR. The Notification_Type parameter may be used by the Host's user interface.

**Event parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device involved in simple pairing process |

*Notification_Type:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0 | Passkey entry started |
| 1 | Passkey digit entered |
| 2 | Passkey digit erased |
| 3 | Passkey cleared |
| 4 | Passkey entry completed |
| 5-255 | Reserved for future use |

### 7.7.50 Remote Host Supported Features Notification Event

| Command | Event Code | Event Parameters |
|---|---|---|
| Remote Host Supported Features Notification | 0x3D | BD_ADDR<br>Host_Supported_Features |

**Description**

The Remote Host Supported Features Notification event is used to return the LMP extended features page containing the Host features. The BD_ADDR is the address of the remote device.

This event shall only be generated after the LMP extended features are read from the remote device during a connection initiated by a Remote Name Request command.

Note: this event is not generated during a connection initiated by the Create Connection command.

**Event parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of remote device. |

*Host_Supported_Features:* *Size: 8 Octets*

| Value | Parameter Description |
|---|---|
| 0xFFFFFFFFFFFFFFFF | Bit map of Host Supported Features page of LMP extended features.<br>For more information, see Link Manager Protocol on page 203. |



LifeScan, Inc. Ex. 1019
Page 833 of 1420
IPR2025-00103

# 8 LIST OF FIGURES

Figure 1.1: Overview of the Lower Software Layers .....357
Figure 1.2: End to End Overview of Lower Software Layers to Transfer Data .....358
Figure 5.1: HCI Command Packet .....393
Figure 5.2: HCI ACL Data Packet .....394
Figure 5.3: HCI Synchronous Data Packet .....396
Figure 5.4: HCI Event Packet .....398
Figure 7.1: Local Loopback Mode .....601
Figure 7.2: Remote Loopback Mode .....601



LifeScan, Inc. Ex. 1019
Page 834 of 1420
IPR2025-00103

# 9 LIST OF TABLES

Table 3.1: Overview of commands and events ....360
Table 3.2: Generic events ....361
Table 3.3: Controller flow control ....362
Table 3.4: Controller information ....362
Table 3.5: Controller configuration ....363
Table 3.6: Device discovery ....364
Table 3.7: Connection setup ....366
Table 3.8: Remote information ....368
Table 3.9: Synchronous connections ....369
Table 3.10: Connection state ....370
Table 3.11: Piconet structure ....371
Table 3.12: Quality of service ....372
Table 3.13: Physical links ....374
Table 3.14: Controller flow control. ....375
Table 3.15: Link information ....376
Table 3.16: Authentication and encryption ....377
Table 3.17: Testing ....380
Table 3.18: Alphabetical list of commands and events. ....381



LifeScan, Inc. Ex. 1019
Page 835 of 1420
IPR2025-00103

# 10 APPENDIX A: DEPRECATED COMMANDS, EVENTS AND CONFIGURATION PARAMETERS

Commands, events and configuration parameters in this section were in prior versions of the specification, but have been determined not to be 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.

10.1 Read Page Scan Mode Command ..... 669
10.2 Write Page Scan Mode Command ..... 670
10.3 Read Page Scan Period Mode Command ..... 671
10.4 Write Page Scan Period Mode Command ..... 672
10.5 Add SCO Connection Command ..... 673
10.6 Page Scan Mode Change Event ..... 675
10.7 Read Country Code Command ..... 676
10.8 Read Encryption Mode Command ..... 677
10.9 Write Encryption Mode Command ..... 677
10.10 Deprecated Parameters ..... 678
10.10.1 Authentication Enable ..... 678
10.10.2 Encryption Mode ..... 678
10.10.3 Page Scan Mode ..... 679



LifeScan, Inc. Ex. 1019
Page 836 of 1420
IPR2025-00103

## 10.1 READ PAGE SCAN MODE COMMAND

| Command | OGF | OCF | Command Parameters | Return Parameters |
|---|---|---|---|---|
| HCI_Read_Page_Scan_Mode | 0x03 | 0x003D | | Status,<br>Page_Scan_Mode |

**Description:**

This command is used to read the default Page Scan Mode configuration parameter of the local Bluetooth device. See “Page Scan Mode” on page 677.

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Page_Scan_Mode command succeeded. |
| 0x01-0xFF | Read_Page_Scan_Mode command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

*Page_Scan_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Appendix A, “Page Scan Mode” on page 677. | |

**Event(s) generated (unless masked away):**

When the Read_Page_Scan_Mode command has completed, a Command Complete event will be generated.

## 10.2 WRITE PAGE SCAN MODE COMMAND

OGF: 0x03 (Controller and baseband commands)

| Command | OGF | OCF | Command Parameters | Return Parameters |
|---|---|---|---|---|
| HCI_Write_Page_Scan_Mode | 0x03 | 0x003E | Page_Scan_Mode | Status |

**Description:**

This command is used to write the default Page Scan Mode configuration parameter of the local Bluetooth device. See “Page Scan Mode” on page 677.

**Command Parameters:**

*Page_Scan_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Appendix A, See “Page Scan Mode” on page 677.. | |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Page_Scan_Mode command succeeded. |
| 0x01-0xFF | Write_Page_Scan_Mode command failed. See “Error Codes” on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Page_Scan_Mode command has completed, a Command Complete event will be generated.

## 10.3 READ PAGE SCAN PERIOD MODE COMMAND

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Page_Scan_Period_Mode | 0x003B | | Status, Page_Scan_Period_Mode |

**Description:**

This command is used to read the mandatory Page_Scan_Period_Mode configuration parameter of the local Bluetooth device. See Section 6.10 on page 402.

**Command Parameters:**

None

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Page_Scan_Period_Mode command succeeded. |
| 0x01-0xFF | Read_Page_Scan_Period_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Page_Scan_Period_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | P0 |
| 0x01 | P1 |
| 0x02 | P2 |
| 0x03-0xFF | Reserved. |

**Event(s) generated (unless masked away):**

When the Read_Page_Scan_Period_Mode command has completed, a Command Complete event will be generated.

## 10.4 WRITE PAGE SCAN PERIOD MODE COMMAND

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Page_Scan_Period_Mode | 0x003C | Page_Scan_Period_Mode | Status |

### Description:

This command is used to write the mandatory Page_Scan_Period_Mode configuration parameter of the local Bluetooth device. See Section 6.10 on page 402.

### Command Parameters:

*Page_Scan_Period_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | P0 |
| 0x01 | P1 |
| 0x02 | P2. **Default.** |
| 0x03-0xFF | Reserved. |

### Return Parameters:

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Page_Scan_Period_Mode command succeeded. |
| 0x01-0xFF | Write_Page_Scan_Period_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

### Event(s) generated (unless masked away):

When the Write_Page_Scan_Period_Mode command has completed, a Command Complete event will be generated.

## 10.5 ADD SCO CONNECTION COMMAND

| Command | OGF | OCF | Command Parameters | Return Parameters |
|---|---|---|---|---|
| HCI_Add_SCO_Connection | 0x01 | 0x0007 | Connection_Handle, Packet_Type | |

**Description:**

This command will cause the Link Manager to create a SCO connection using the ACL connection specified by the Connection_Handle command parameter. This command causes the local Bluetooth device to create a SCO connection. The Link Manager 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 Link Manager should use for the connection. The Link Manager must only use 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 Link Manager 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).

Note: An SCO connection can only be created when an ACL connection already exists and when it is not put in park. For a definition of the different packet types, see the “Baseband Specification” on page 53 [Part B].

Note: At least one packet type must be specified. The Host should enable as many packet types as possible for the Link Manager to perform efficiently. However, the Host must not enable packet types that the local device does not support.

**Command Parameters:**

*Connection_Handle* *Size 2 Octets (12 Bits meaningful)*

| Value | Parameter Description |
|---|---|
| 0xXXXX | Connection Handle for the ACL connection being used to create an SCO connection.<br>Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use) |

*Packet_Type:* *Size: 2 Octets*

| Value | Parameter Description |
|---|---|
| 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. |

**Return Parameters:**

None.

**Event(s) generated (unless masked away):**

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.

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.

## 10.6 PAGE SCAN MODE CHANGE EVENT

| Event | Event Code | Event Parameters |
|---|---|---|
| Page Scan Mode Change | 0x1F | BD_ADDR, Page_Scan_Mode |

**Description:**

The Page Scan Mode Change event indicates that the connected remote Bluetooth device with the specified BD_ADDR has successfully changed the Page_Scan_Mode.

**Event Parameters:**

*BD_ADDR:* *Size: 6 Octets*

| Value | Parameter Description |
|---|---|
| 0xXXXXXXXXXXXX | BD_ADDR of the remote device. |

*Page_Scan_Mode:* *Size: 1 Octet*

| 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. |

## 10.7 READ COUNTRY CODE COMMAND

| Command | OGF | OCF | Command Parameters | Return Parameters |
|---|---|---|---|---|
| HCI_Read_Country_Code | 0x04 | 0x0007 | | Status,<br>Country_Code |

**Description:**

This command will read the value for the Country_Code return parameter. The Country_Code defines which range of frequency band of the ISM 2.4 GHz band will be used by the device. Each country has local regulatory bodies regulating which ISM 2.4 GHz frequency ranges can be used.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Country_Code command succeeded. |
| 0x01-0xFF | Read_Country_Code command failed. See "Error Codes" on page 331 [Part D] for error codes and descriptions. |

*Country_Code:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | North America & Europe* and Japan |
| 0x01 | France |
| 0x04-FF | Reserved for Future Use. |

*. Except France

**Event(s) generated (unless masked away):**

When the Read_Country_Code command has completed, a Command Complete event will be generated.

## 10.8 READ ENCRYPTION MODE COMMAND

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Read_Encryption_Mode | 0x0021 | | Status, Encryption_Mode |

**Description:**

This command will read the value for the Encryption_Mode configuration parameter. See Section 10.10.2 on page 676.

**Command Parameters:**

None.

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Read_Encryption_Mode command succeeded. |
| 0x01-0xFF | Read_Encryption_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

*Encryption_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Section 10.10.2 on page 676 | |

**Event(s) generated (unless masked away):**

When the Read_Encryption_Mode command has completed, a Command Complete event will be generated.

## 10.9 WRITE ENCRYPTION MODE COMMAND

| Command | OCF | Command Parameters | Return Parameters |
|---|---|---|---|
| HCI_Write_Encryption_Mode | 0x0022 | Encryption_Mode | Status |

**Description:**

This command will write the value for the Encryption_Mode configuration parameter. See Section 10.10.2 on page 676.

**Command Parameters:**



LifeScan, Inc. Ex. 1019
Page 845 of 1420
IPR2025-00103

*Encryption_Mode:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| See Section 10.10.2 on page 676. | |

**Return Parameters:**

*Status:* *Size: 1 Octet*

| Value | Parameter Description |
|---|---|
| 0x00 | Write_Encryption_Mode command succeeded. |
| 0x01-0xFF | Write_Encryption_Mode command failed. See "Error Codes" on page 331 [Part D] for a list of error codes and descriptions. |

**Event(s) generated (unless masked away):**

When the Write_Encryption_Mode command has completed, a Command Complete event will be generated.

## 10.10 DEPRECATED PARAMETERS

### 10.10.1 Authentication Enable

The Authentication_Enable parameter controls if the local device requires to authenticate 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.

Note: Changing this parameter does not affect existing connections.

| Value | Parameter Description |
|---|---|
| 0x00 | Authentication not required. |
| 0x01 | Authentication required for all connections. |
| 0x02-0xFF | Reserved |

### 10.10.2 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.

Note: 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.

Note: for encryption to be used, both devices must support and enable encryption.

| Value | Parameter Description |
|---|---|
| 0x00 | Encryption not required. |
| 0x01 | Encryption required for all connections. |
| 0x02-0xFF | Reserved. |

Note: 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.

### 10.10.3 Page Scan Mode

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 Baseband timer T_mandatory_pscan has not expired, the mandatory page scan mode must be applied. For details see the “Baseband Specification” on page 53 [Part B] (Keyword: Page-Scan-Mode, FHS-Packet, T_mandatory_pscan)

| 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 |



LifeScan, Inc. Ex. 1019
Page 847 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 848 of 1420
IPR2025-00103

# MESSAGE SEQUENCE CHARTS

## Between Host and Host Controller/Link Manager

***Examples of interactions between Host Controller Interface Commands and Events and Link Manager Protocol Data Units are represented in the form of message sequence charts. These charts show typical interactions and do not indicate all possible protocol behavior.***



LifeScan, Inc. Ex. 1019
Page 849 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 850 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 683**
1.1 Notation ..... 683
1.2 Flow of Control ..... 684
1.3 Example MSC ..... 684
**2 Services Without Connection Request ..... 685**
2.1 Remote Name Request ..... 685
2.2 One-time Inquiry ..... 687
2.3 Periodic Inquiry ..... 689
**3 ACL Connection Establishment and Detachment ..... 691**
3.1 Connection Setup ..... 692
**4 Optional Activities After ACL Connection Establishment ..... 699**
4.1 Authentication Requested ..... 699
4.2 Simple Pairing Message Sequence Charts ..... 700
4.2.1 Optional OOB Information Collection ..... 700
4.2.2 Enable Simple Pairing ..... 701
4.2.3 Connection Establishment ..... 701
4.2.4 L2CAP Connection Request for a Secure Service ..... 701
4.2.5 Optional OOB Information Transfer ..... 702
4.2.6 Start Simple Pairing ..... 702
4.2.7 IO Capability Exchange ..... 703
4.2.8 Public Key Exchange ..... 703
4.2.9 Authentication ..... 704
4.2.10 Numeric Comparison ..... 704
4.2.11 Numeric Comparison Failure on Initiating Side ..... 705
4.2.12 Numeric Comparison Failure on Responding Side ..... 706
4.2.13 Passkey Entry ..... 707
4.2.14 Passkey Entry Failure on Responding Side ..... 708
4.2.15 Passkey Entry Failure on Initiator Side ..... 709
4.2.16 Out of Band ..... 709
4.2.17 OOB Failure on Initiator Side ..... 710
4.2.18 DHKey Checks ..... 711
4.2.19 Calculate Link Key ..... 711
4.2.20 Enable Encryption ..... 712
4.2.21 L2CAP Connection Response ..... 713
4.3 Link Supervision Timeout Changed Event ..... 713
4.4 Set Connection Encryption ..... 714



LifeScan, Inc. Ex. 1019
Page 851 of 1420
IPR2025-00103

4.5 Change Connection Link Key ..... 715
4.6 Change Connection Link Key with Encryption Pause and Resume ..... 715
4.7 Master Link Key ..... 717
4.8 Read Remote Supported Features ..... 719
4.9 Read Remote Extended Features ..... 719
4.10 Read Clock Offset ..... 720
4.11 Role Switch on an Encrypted Link using Encryption Pause and Resume ..... 720
4.12 Refreshing Encryption Keys ..... 721
4.13 Read Remote Version Information ..... 722
4.14 QOS Setup ..... 722
4.15 Switch Role ..... 723

**5 Synchronous Connection Establishment and Detachment ..... 725**
5.1 Synchronous Connection Setup ..... 725

**6 Sniff, Hold and Park ..... 730**
6.1 Sniff Mode ..... 730
6.2 Hold Mode ..... 731
6.3 Park State ..... 733

**7 Buffer Management, Flow Control ..... 736**

**8 Loopback Mode ..... 738**
8.1 Local Loopback Mode ..... 738
8.2 Remote Loopback Mode ..... 740

**9 List of Figures ..... 742**

# 1 INTRODUCTION

This section shows typical interactions between Host Controller Interface (HCI) Commands and Events and Link Manager (LM) Protocol Data Units (PDU). It focuses on the message sequence charts (MSCs) for the procedures specified in [3] “Bluetooth Host Controller Interface Functional Specification” with regard to LM Procedures from [2] “Link Manager Protocol”.

This section illustrates only the most useful scenarios, it does not cover all possible alternatives. Furthermore, the message sequence charts do not consider errors over the air interface or host interface. In all message sequence charts 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 message sequence charts is for illustrative purposes. The messages may be sent in a different order where allowed by the Link Manager or HCI sections. If any of these charts differ with text in the Baseband, Link Manager, or HCI sections, the text in those sections shall be considered normative. This section is informative.

## 1.1 NOTATION

The notation used in the message sequence charts (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 following symbols represent interactions and states:

| | |
|---|---|
| **Oval** | Defines the context for the message sequence chart. |
| **Hexagon** | Indicates a condition needed to start the transactions below this hexagon. The location and width of the Hexagon indicates which entity or entities make this decision. |
| **Box** | Replaces a group of transactions. May indicate a user action, or a procedure in the baseband. |
| **Dashed Box** | Optional group of transactions. |
| **Solid Arrow** | Represents a message, signal or transaction. Can be used to show LMP and HCI traffic.<br>Some baseband 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. |



LifeScan, Inc. Ex. 1019
Page 853 of 1420
IPR2025-00103

## 1.2 FLOW OF CONTROL

Some message sequences are split into several charts. These charts are marked in sequence with different step numbers with multiple paths through with optional 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.

## 1.3 EXAMPLE MSC

The protocol entities represented in the example shown in Figure 1.1 on page 684 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 section may show the interactions of more than two devices.

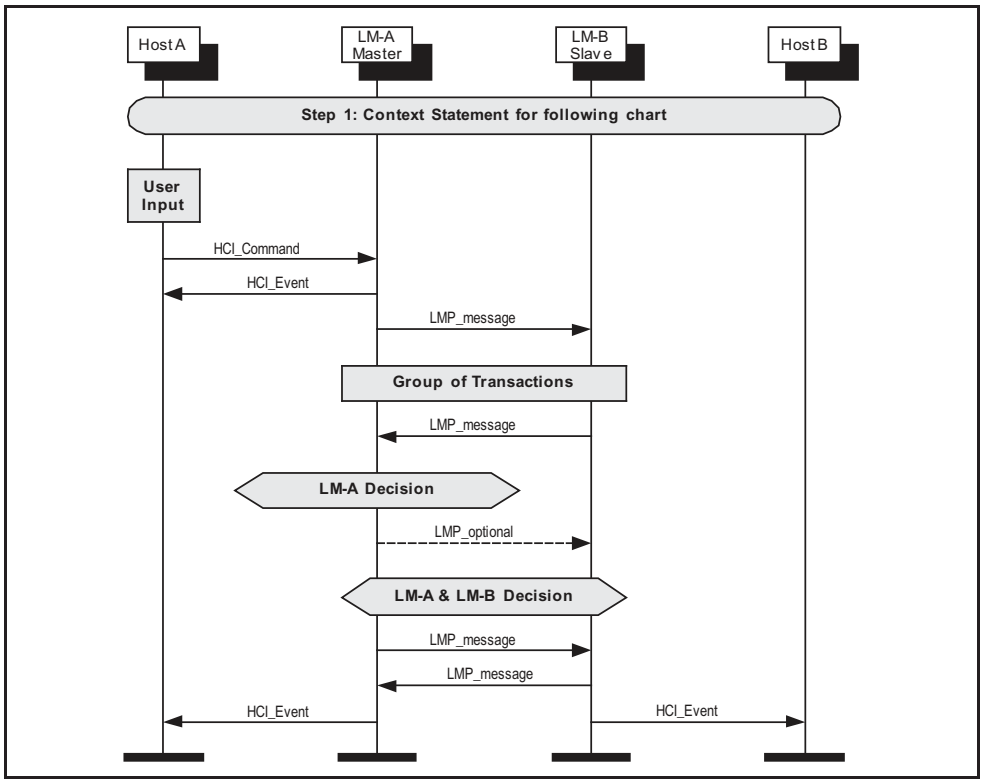



*Figure 1.1: Example MSC.*

# 2 SERVICES WITHOUT CONNECTION REQUEST

## 2.1 REMOTE NAME REQUEST

The service Remote Name Request is used to find out the name of the remote device without requiring an explicit ACL Connection.

**Step 1:** The host sends an HCI_Set_Event_Mask with the bit of Remote Host Supported Features Notification Event (0x1000000000000000) set and an HCI_Remote_Name_Request command expecting that its local device will automatically try to connect to the remote device.(See Figure 2.1 on page 685.)

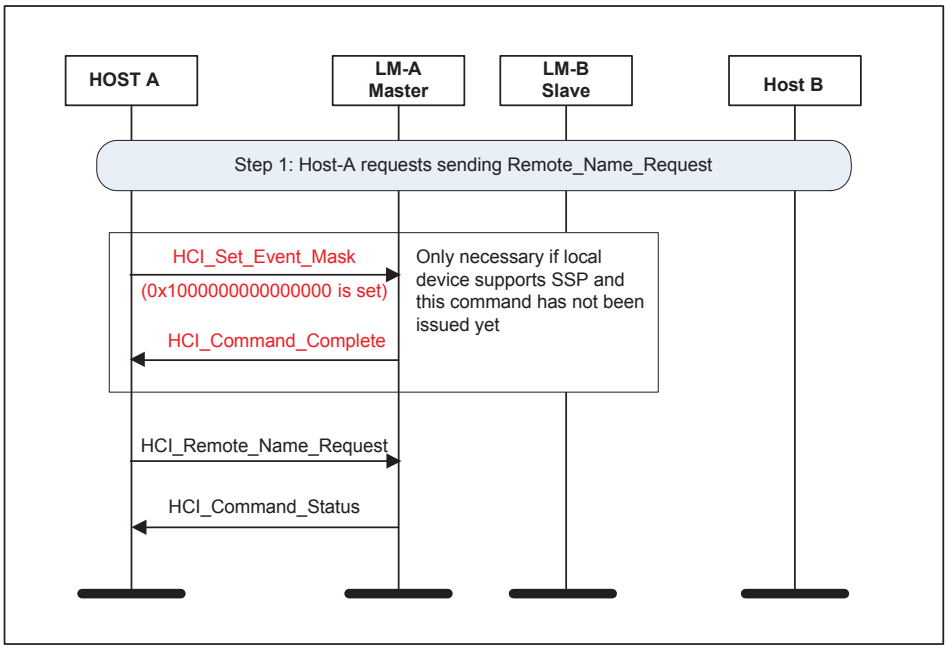



*Figure 2.1: Remote name request.*

**Step 2a:** If an ACL Connection does not exist device A pages device B. After the Baseband paging procedure, the local device attempts to get the remote device's extended features, send an HCI_Remote_Device_Supported_Features_Notification event, get the remote name, disconnect, and return the name of the remote device to the Host.(See Figure 2.2 on page 686.)



LifeScan, Inc. Ex. 1019
Page 855 of 1420
IPR2025-00103

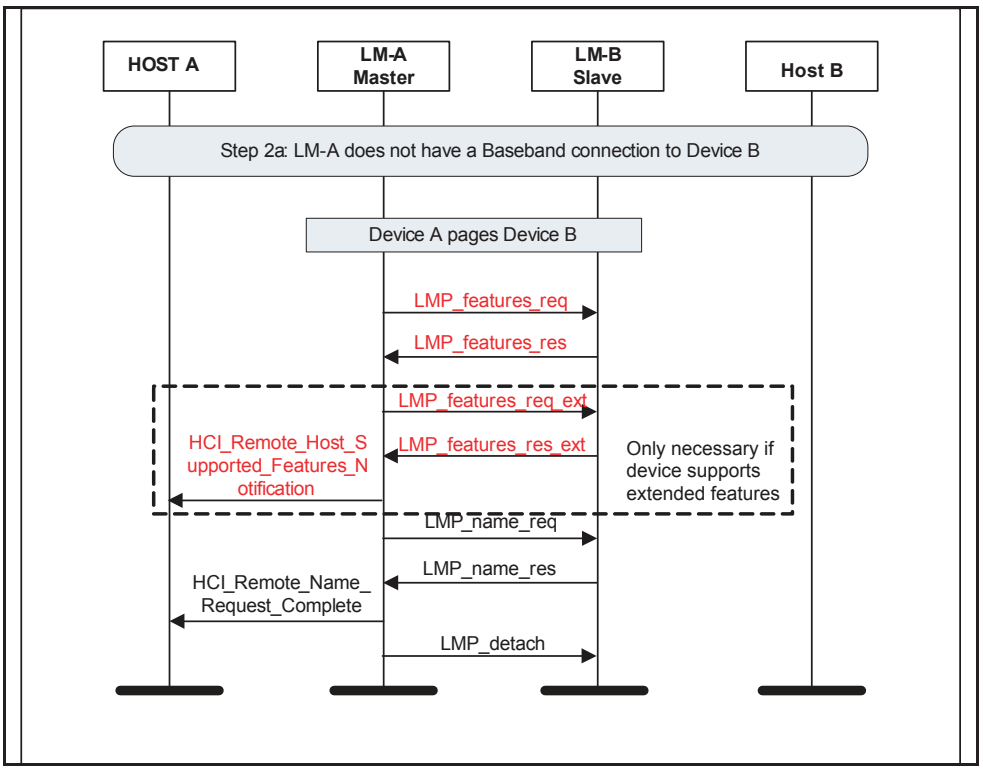



*Figure 2.2: Remote name request if no current baseband connection.*

**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 2.3 on page 687.)

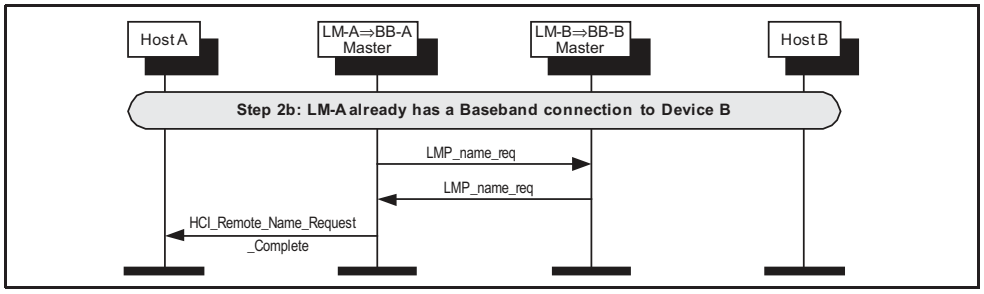



*Figure 2.3: Remote name request with baseband connection.*

## 2.2 ONE-TIME INQUIRY

Inquiry is used to detect and collect nearby devices.

**Step 1:** The host sends an HCI_Inquiry command. (See Figure 2.4 on page 687.)

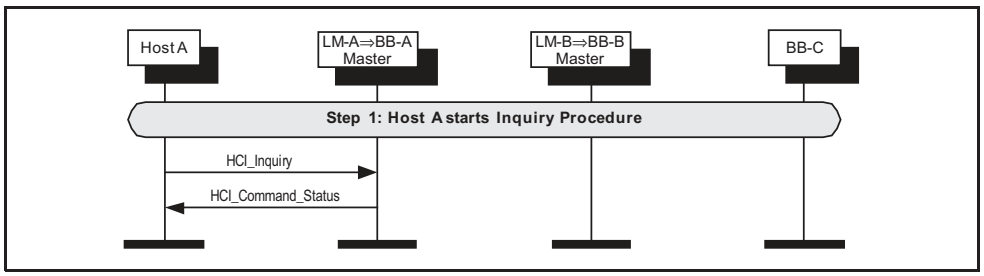



*Figure 2.4: Host A starts inquiry procedure.*



LifeScan, Inc. Ex. 1019
Page 857 of 1420
IPR2025-00103

**Step 2:** The Controller will start the Baseband inquiry procedure with the specified Inquiry Access Code 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 2.5 on page 688.)

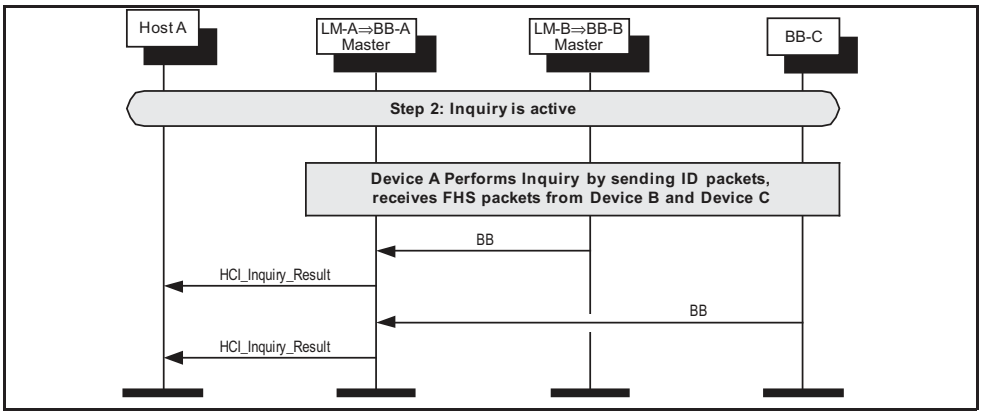



*Figure 2.5: LM-A performs inquiry and reports result.*

**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 2.6 on page 688.)

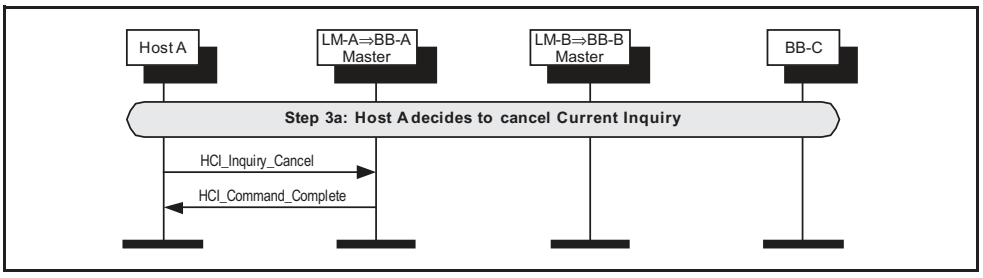



*Figure 2.6: Host A cancels inquiry.*

**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 2.7 on page 689.)

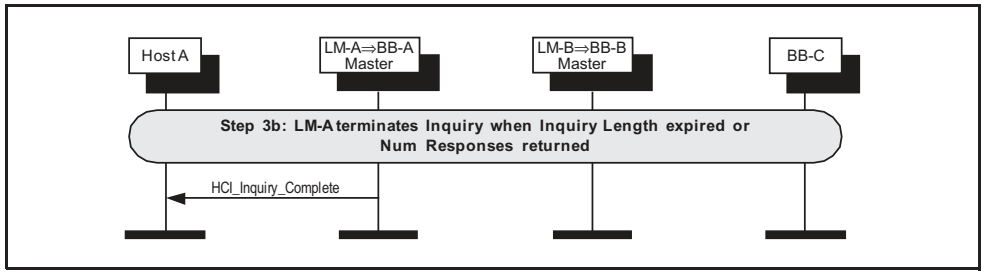



*Figure 2.7: LM-A terminates current inquiry.*

## 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 2.8 on page 689.)

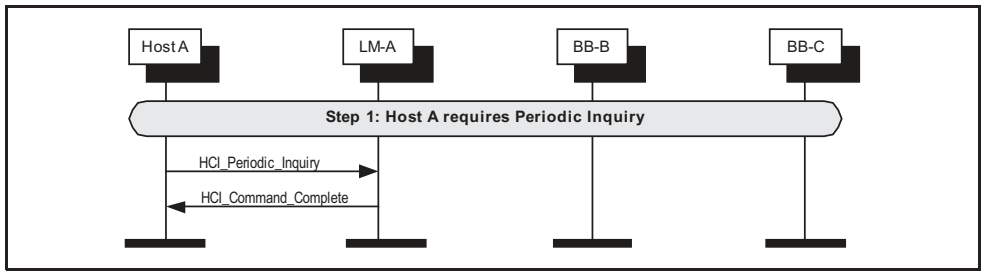



*Figure 2.8: Host A starts periodic inquiry.*

**Step 2:** The Controller will start a periodic Inquiry. In the inquiry cycle, one or several Inquiry Result events will be returned. (See Figure 2.9 on page 690.)

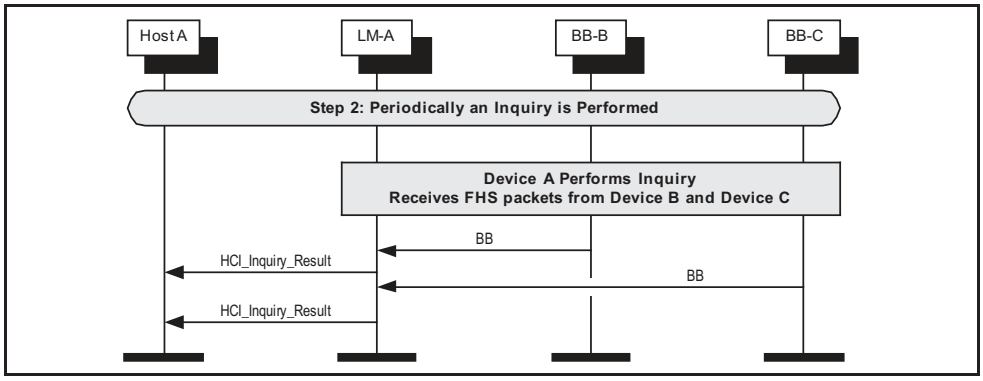



*Figure 2.9: LM-A periodically performs an inquiry and reports result.*

**Step 3:** An Inquiry Complete event will be returned to the Host when the current periodic inquiry has finished. (See Figure 2.10 on page 690.)

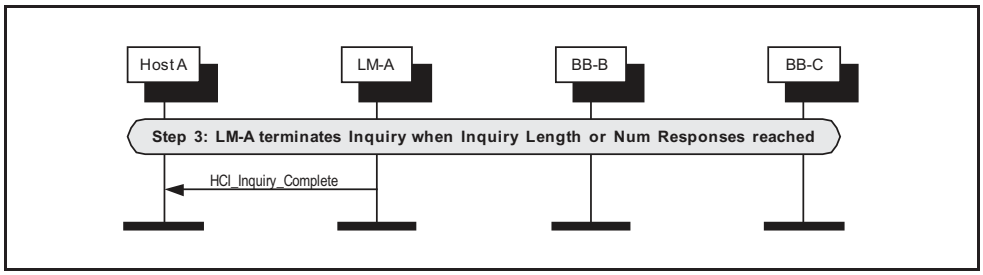



*Figure 2.10: LM-A terminates current inquiry.*

**Step 4:** The periodic Inquiry can be stopped using the HCI_Exit_Periodic_Inquiry_Mode command. (See Figure 2.11 on page 690.)

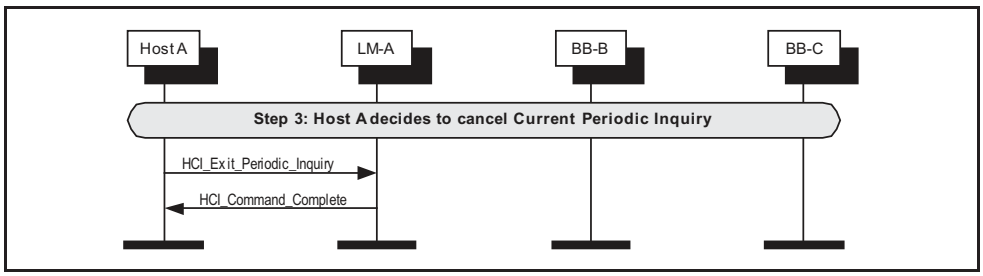



*Figure 2.11: Host A decides to exit periodic inquiry.*

# 3 ACL CONNECTION ESTABLISHMENT AND DETACHMENT

A flow diagram of the establishment and detachment of a connection between two devices is shown in Figure 3.1 on page 691. The process is illustrated in 9 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 following message sequence charts.

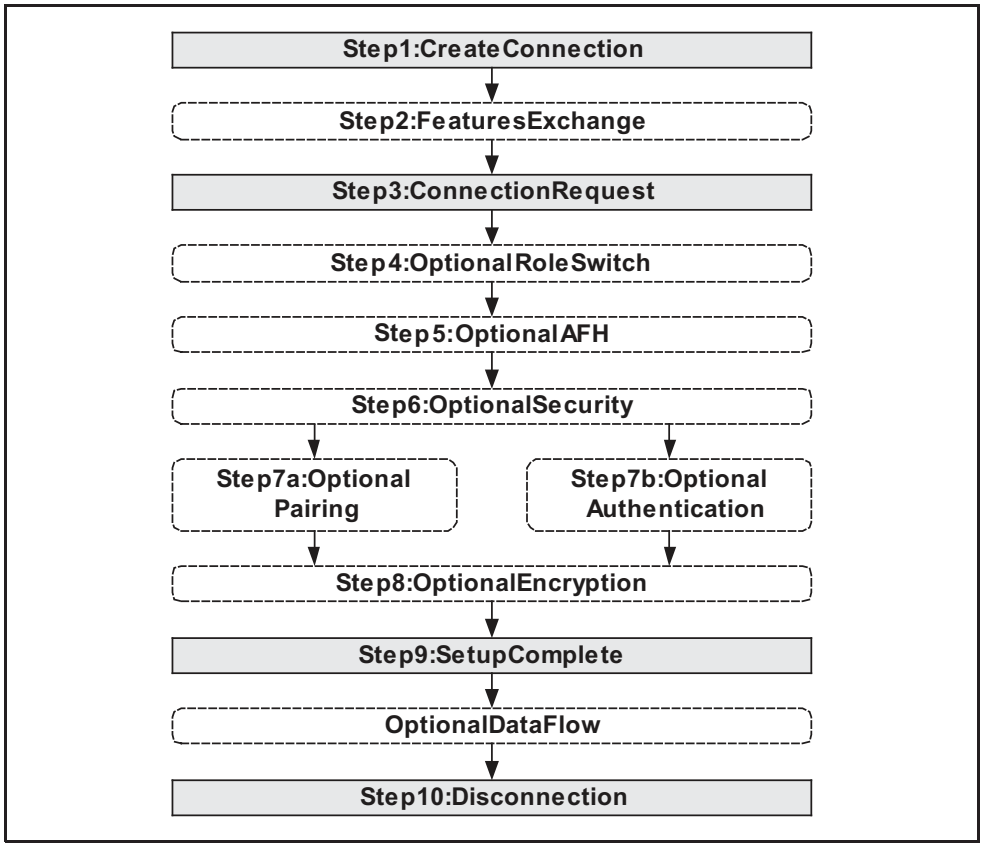



*Figure 3.1: Overview diagram for connection setup.*



LifeScan, Inc. Ex. 1019
Page 861 of 1420
IPR2025-00103

## 3.1 CONNECTION SETUP

**Step 1:** The host sends an HCI_Create_Connection command to the Controller. The Controller then performs a Baseband paging procedure with the specified BD_ADDR. (See Figure 3.2 on page 692.)

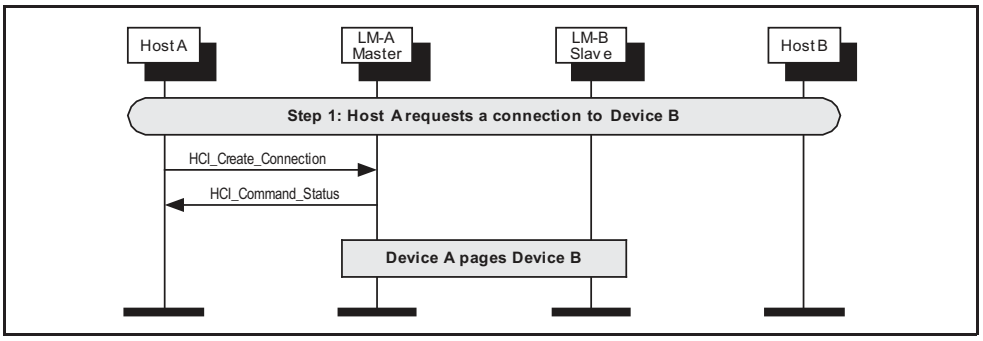



*Figure 3.2: Host A requests connection with device B.*

**Step 2:** Optionally, the LM may decide to exchange features. (See Figure 3.3 on page 692.)

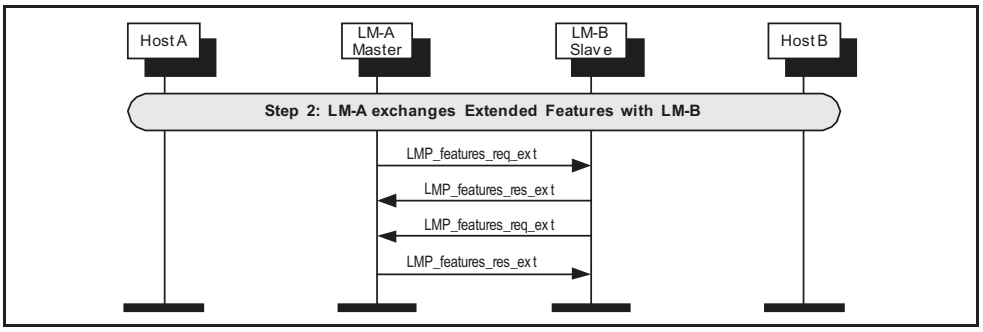



*Figure 3.3: LM-A and LM-B exchange features.*

**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 3.4 on page 692)

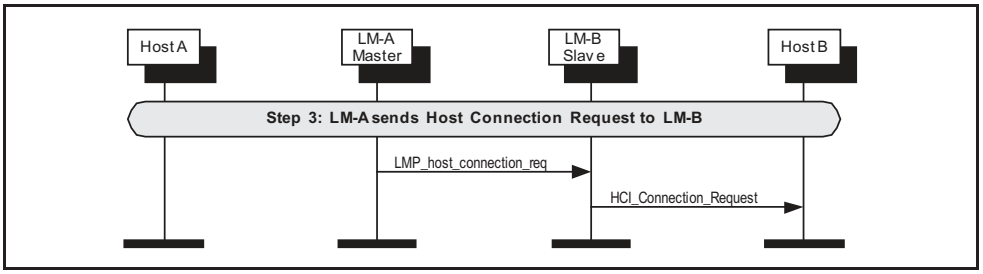



*Figure 3.4: LM-A requests host connection.*

**Step 4a:** The remote host rejects this connection, and the link is terminated. (See Figure 3.5 on page 693.)

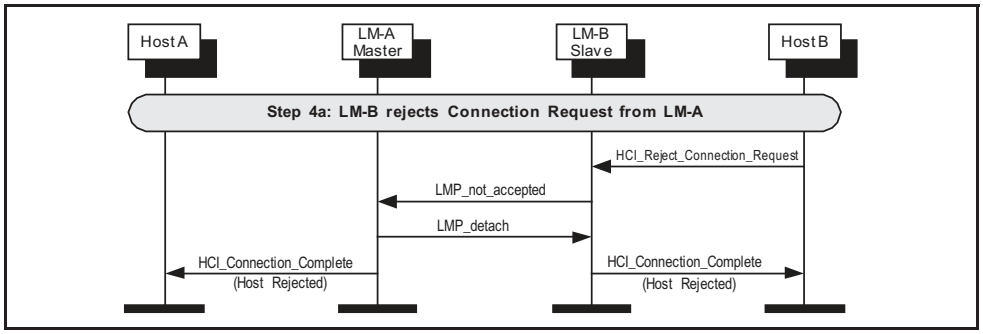



*Figure 3.5: Device B rejects connection request.*

**Step 4b:** The remote host accepts this connection. (See Figure 3.6 on page 693.)

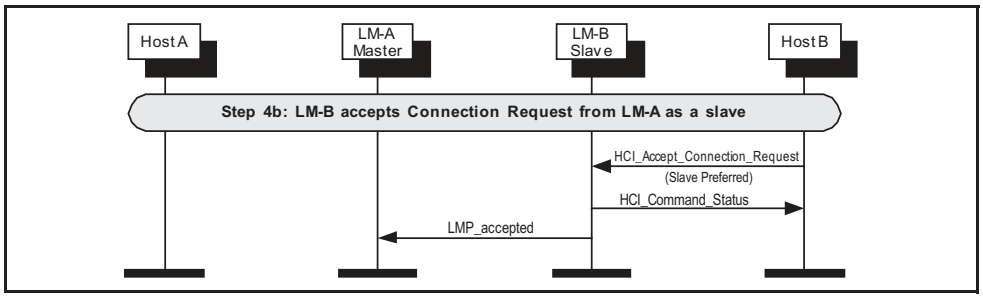



*Figure 3.6: Device B accepts connection request.*

**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 for the LMP_host_connection_req PDU is sent. (See Figure 3.7 on page 694.)

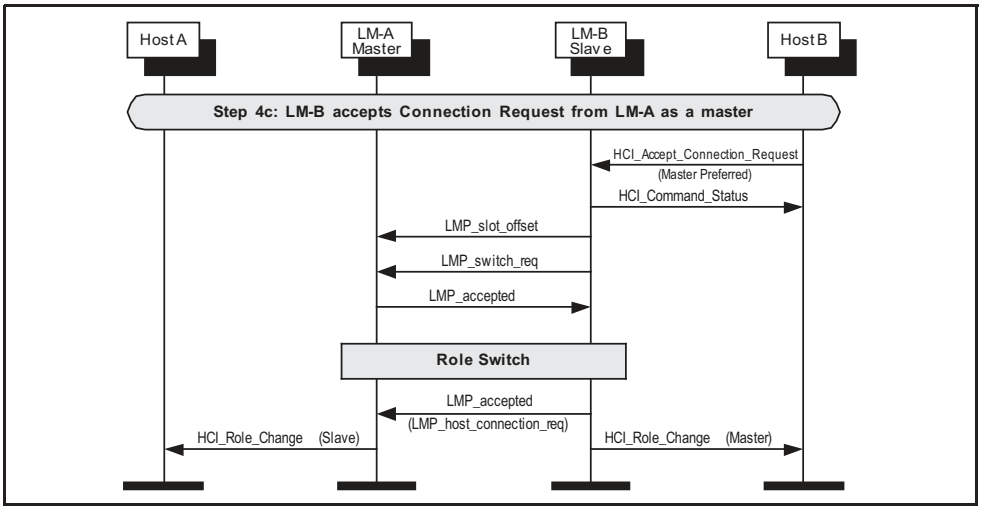



*Figure 3.7: Device B accepts connection requests as master.*

**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 and LMP_channel_classification_req PDU. (See Figure 3.8 on page 694.)

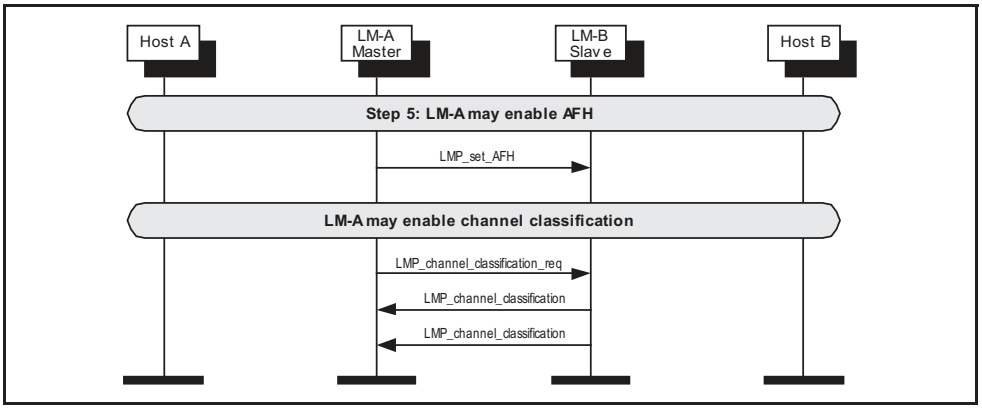



*Figure 3.8: LM-A starts adaptive frequency hopping.*



LifeScan, Inc. Ex. 1019
Page 864 of 1420
IPR2025-00103

**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 3.9 on page 695.)

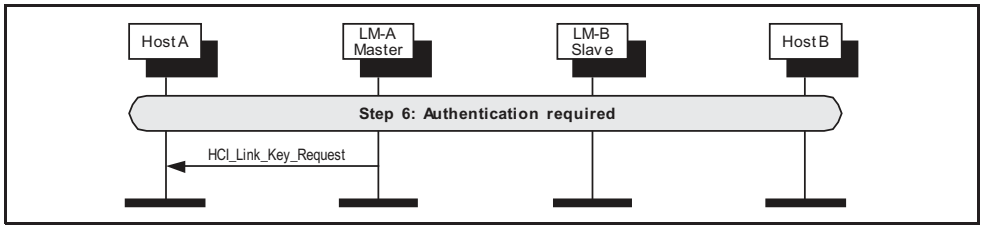



*Figure 3.9: Authentication initiated.*



LifeScan, Inc. Ex. 1019
Page 865 of 1420
IPR2025-00103

**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 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 3.10 on page 696.)

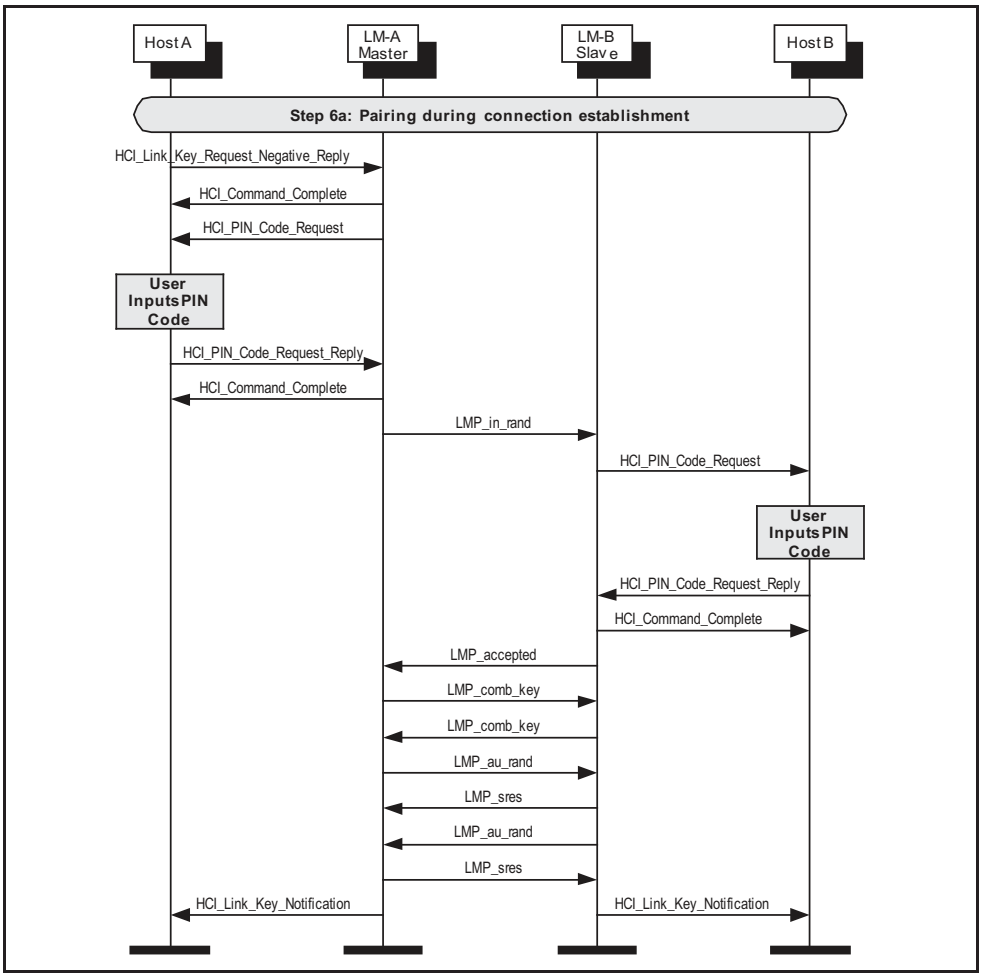



*Figure 3.10: Pairing during connection setup.*



LifeScan, Inc. Ex. 1019
Page 866 of 1420
IPR2025-00103

**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 configuration parameter Authentication_Enable is set, then the authentication procedure must be executed. This MSC only shows the case when Authentication_Enable is set on both sides. (See Figure 3.11 on page 697.)

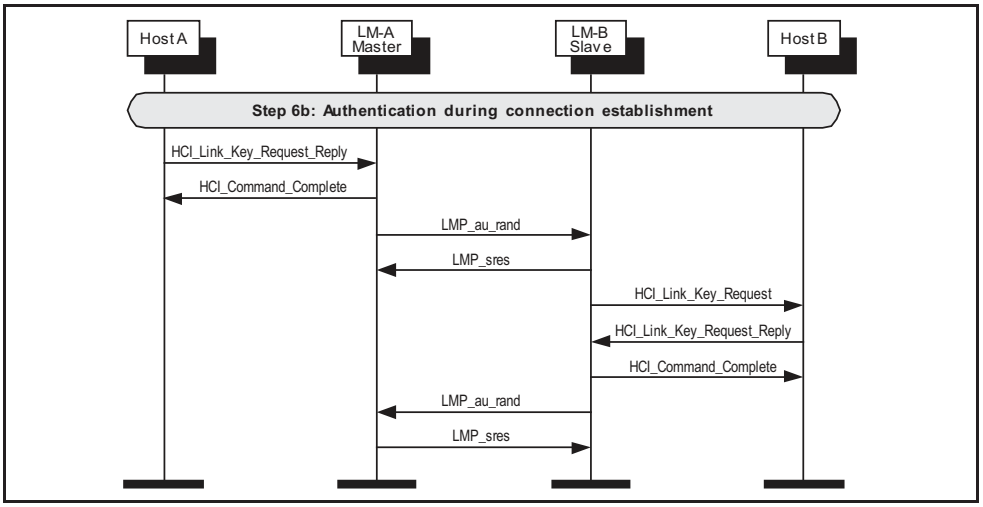



*Figure 3.11: Authentication during connection setup.*

**Step 8:** Once the pairing or authentication procedure is successful, the encryption procedure may be started. This MSC only shows the set up of an encrypted point-to-point connection. (See Figure 3.12 on page 697.)

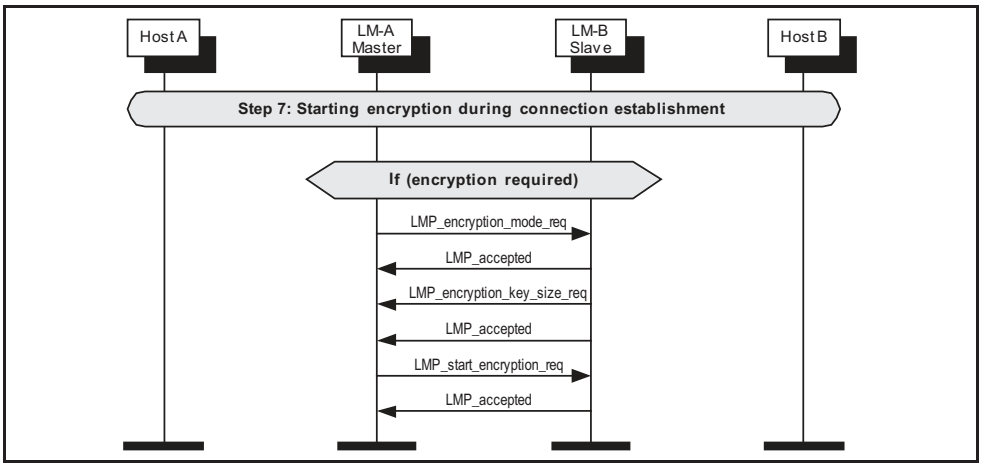



*Figure 3.12: Starting encryption during connection setup.*

**Step 9:** The LMs indicate that the connection is setup 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 3.13 on page 698.)

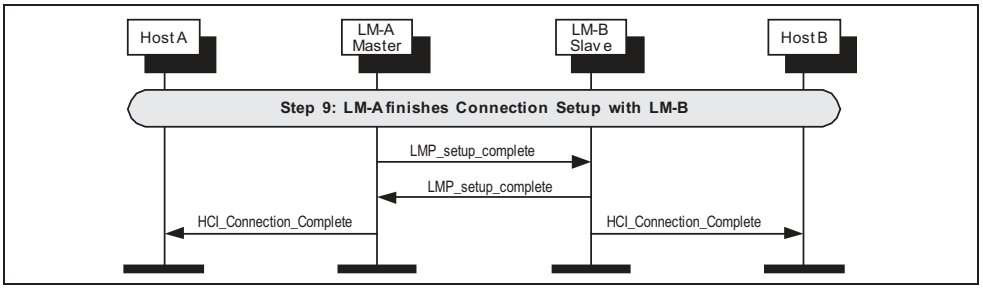



*Figure 3.13: LM-A and LM-B finishes connection setup.*

**Step 10:** Once the connection is no longer needed, either device may terminate the connection using the HCI_Disconnect command and LMP_detach message 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 Baseband is needed to ensure that the remote LM has received the LMP_detach PDU. (See Figure 3.13 on page 698.)

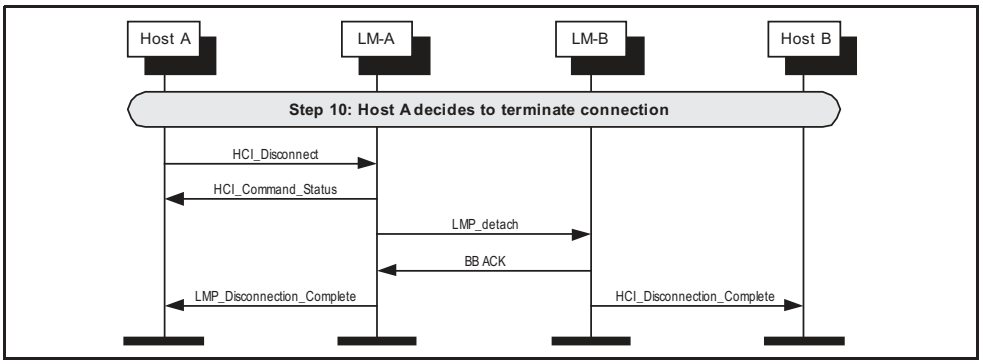



*Figure 3.14: Host A decides to disconnect.*

# 4 OPTIONAL ACTIVITIES AFTER ACL CONNECTION ESTABLISHMENT

## 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 4.1 on page 699.)

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 (Section 3.1, Step 7a:) will be used. (See Figure 3.9 on page 695.)

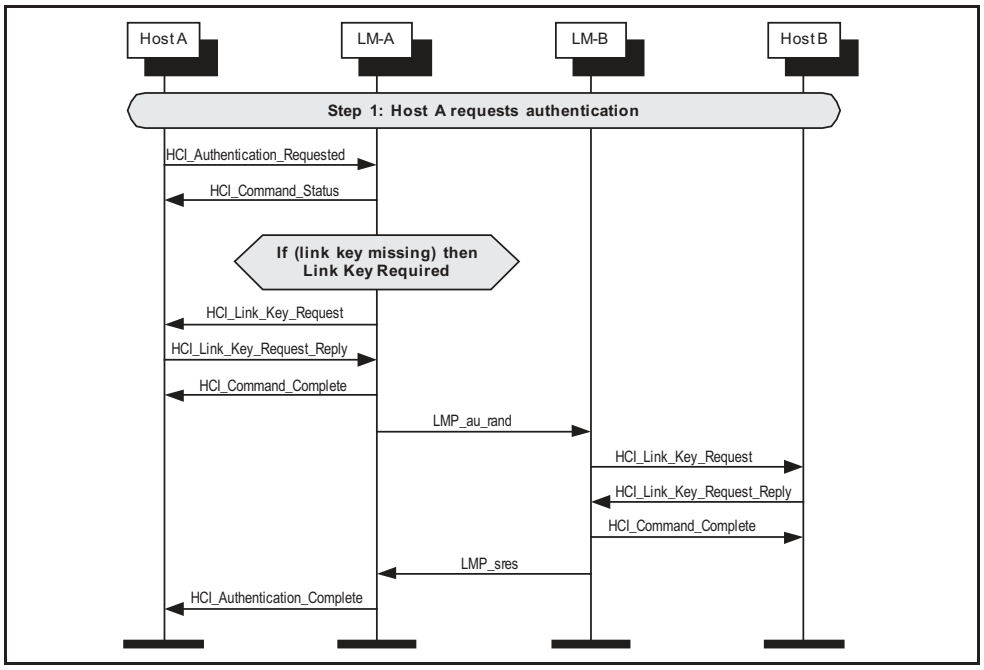



*Figure 4.1: Authentication requested.*

## 4.2 SIMPLE PAIRING MESSAGE SEQUENCE CHARTS

A flow diagram of simple pairing between two devices is shown in Figure 4.2. The process is illustrated in 11 distinct steps. A number of these steps have a number of different options.

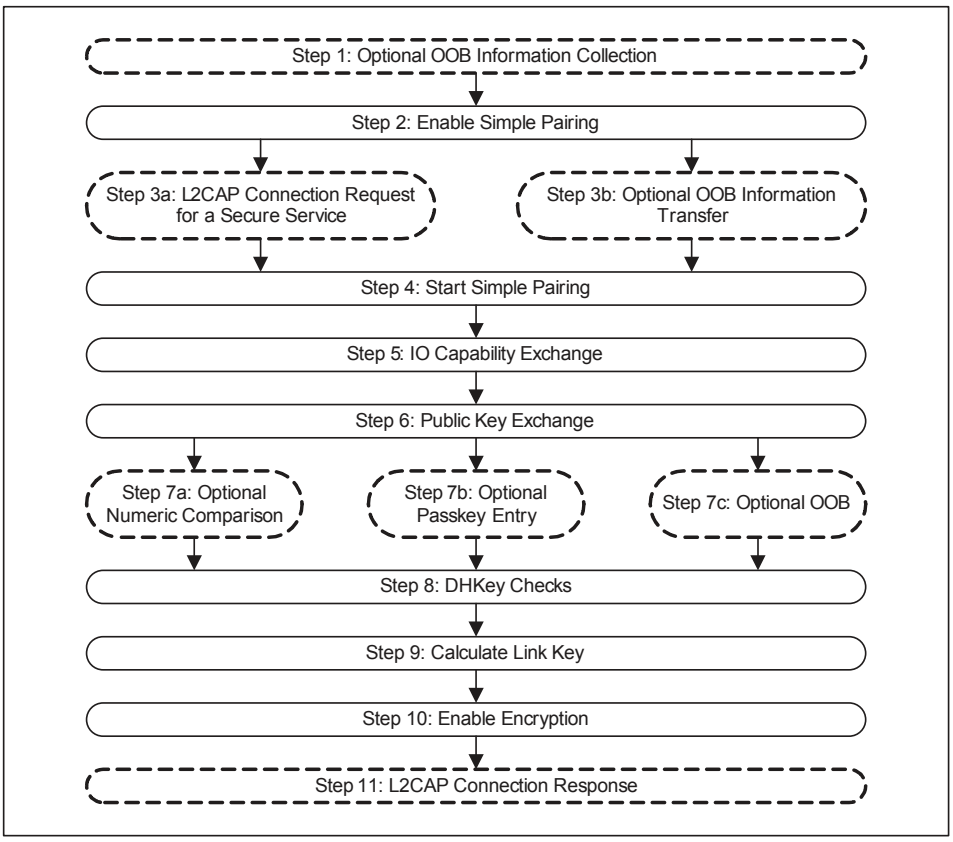



*Figure 4.2: Simple Pairing Flow Diagram*

### 4.2.1 Optional OOB Information Collection

If a device supports OOB information exchange, then the Host should request the C and R values from the controller that need to be sent by OOB. It is then assumed that the host transfers this information to the OOB system. Note: This could occur a long time before, for example at the factory for a passive tag.



LifeScan, Inc. Ex. 1019
Page 870 of 1420
IPR2025-00103

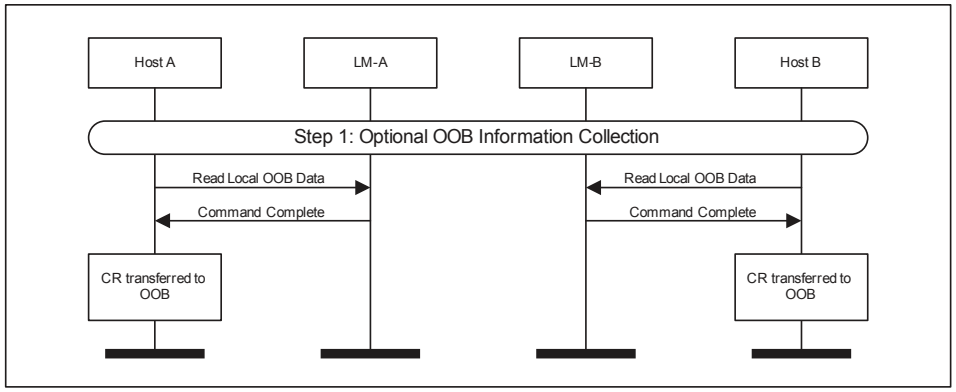



*Figure 4.3: Optional OOB Information Collection*

### 4.2.2 Enable Simple Pairing

To enable simple pairing, a device must use the Write Simple Pairing Mode command. This should be done before any other connections that may use simple pairing are created.

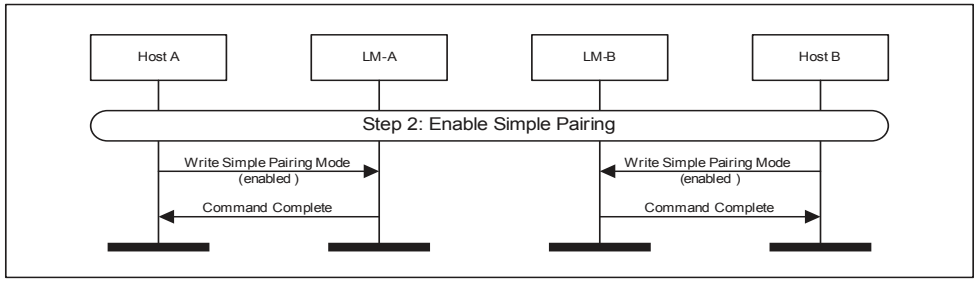



*Figure 4.4: Enable Simple Pairing*

### 4.2.3 Connection Establishment

Simple pairing, once it is enabled, is triggered by one of two possible actions. It could be triggered by an L2CAP connection request to a service that requires security, or it could be triggered by an OOB transfer of information.

### 4.2.4 L2CAP Connection Request for a Secure Service

Once a connection has been established between two devices, if a device requests an L2CAP connection to a service that requires authentication and encryption, then the device will start simple pairing.

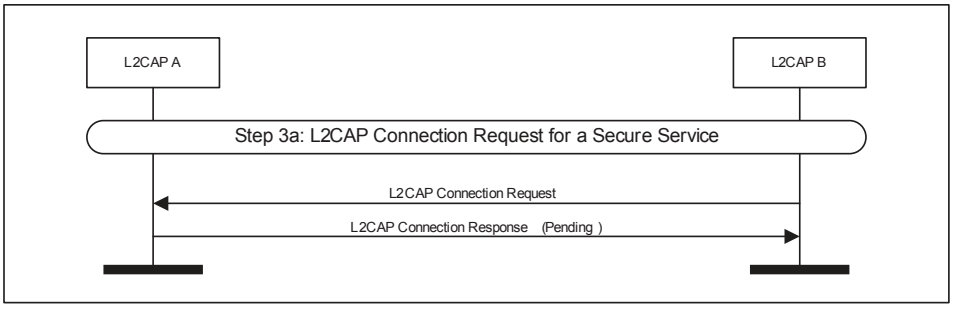



*Figure 4.5: L2CAP Connection Request for a Secure Service*

### 4.2.5 Optional OOB Information Transfer

Even if a Bluetooth connection has not been established between two devices, an OOB transfer can occur that transfers the Bluetooth Device Address of the device, and other OOB information for authentication. If an OOB transfer occurs, then the host can start simple pairing.

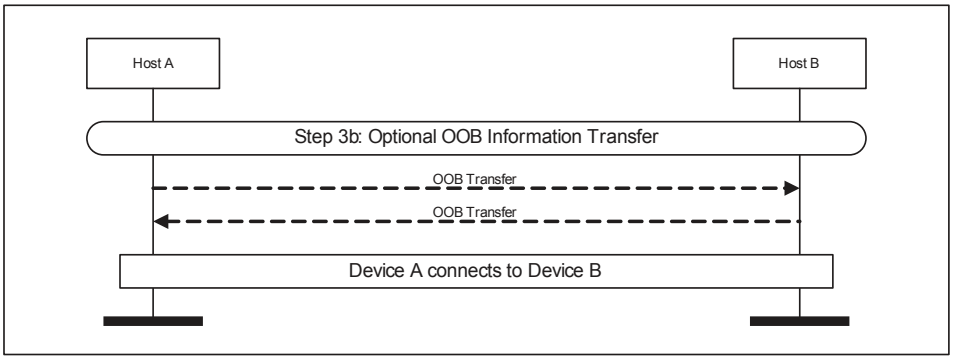



*Figure 4.6: Optional OOB Information Transfer*

### 4.2.6 Start Simple Pairing

Once the host has determined that simple pairing should start, it issues an Authentication Requested command to the controller. This will cause the controller to generate a request for a link key. If the host has a link key for this connection, then pairing is not required, and the link key can be used immediately once it has been authenticated. Simple Pairing will only be used if a Link_Key_Request_Negative_Reply Command is sent from the Host to the Controller on the initiating side.

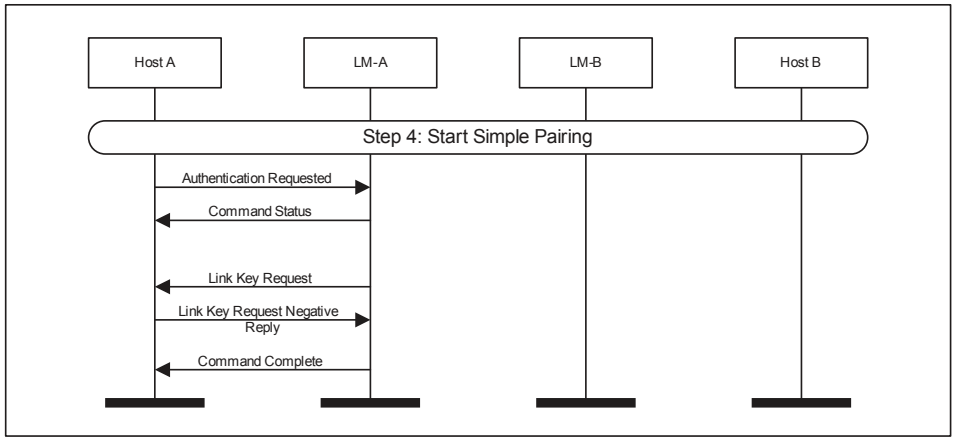



*Figure 4.7: Start Simple Pairing*

### 4.2.7 IO Capability Exchange

To be able to determine the correct authentication algorithm to use, the input / output capabilities of the two devices are exchanged.

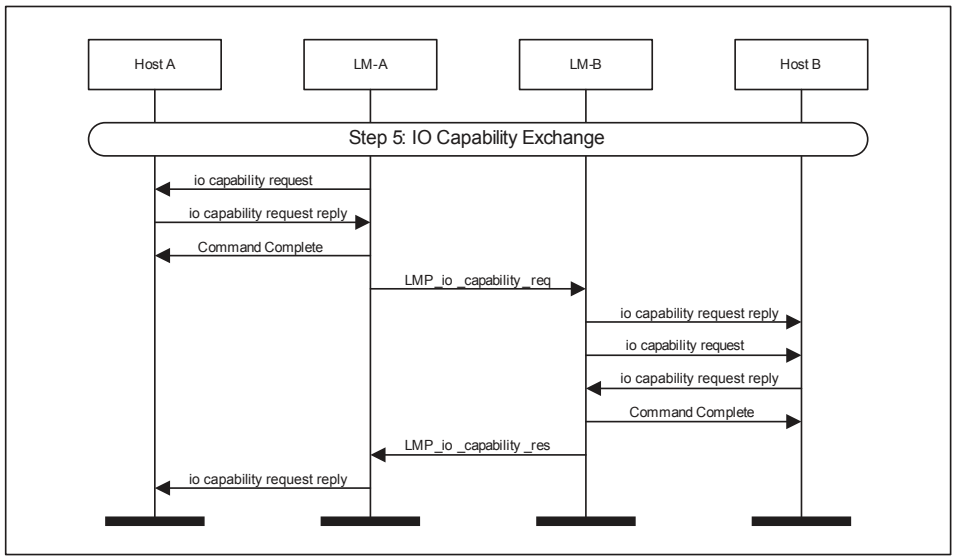



*Figure 4.8: IO Capability Exchange*

### 4.2.8 Public Key Exchange

Next the public keys are exchanged between the two devices. Once a device has received the public key of the peer device, it can start to calculate the Diffie Hellman Key (DHKey). This may take a long time, and should be started early, so that user interaction can hide the calculation time. The DHKey is not required until step 8.

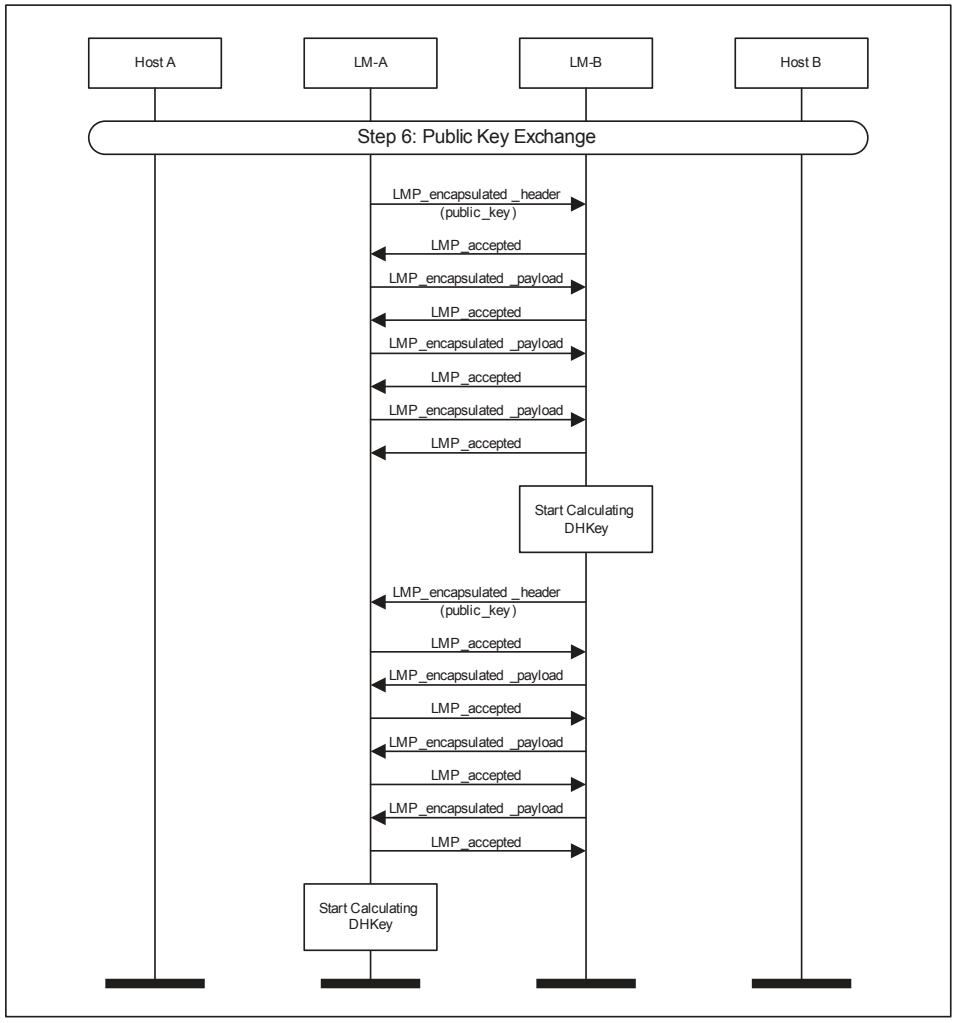



*Figure 4.9: Public Key Exchange*

### 4.2.9 Authentication

A device can be authenticated by using one of three algorithms. The choice of algorithm is determined by the combination of the IO capabilities of the two devices.

### 4.2.10 Numeric Comparison

The numeric comparison step will be done when both devices have output capabilities, or if one of the devices has no input or output capabilities. If both devices have output capabilities, this step requires the displaying of a user confirmation value. This value should be displayed until the end of step 8. If one or both devices do not have output capabilities, the same protocol is used but the Hosts will skip the step asking for the user confirmation.



LifeScan, Inc. Ex. 1019
Page 874 of 1420
IPR2025-00103

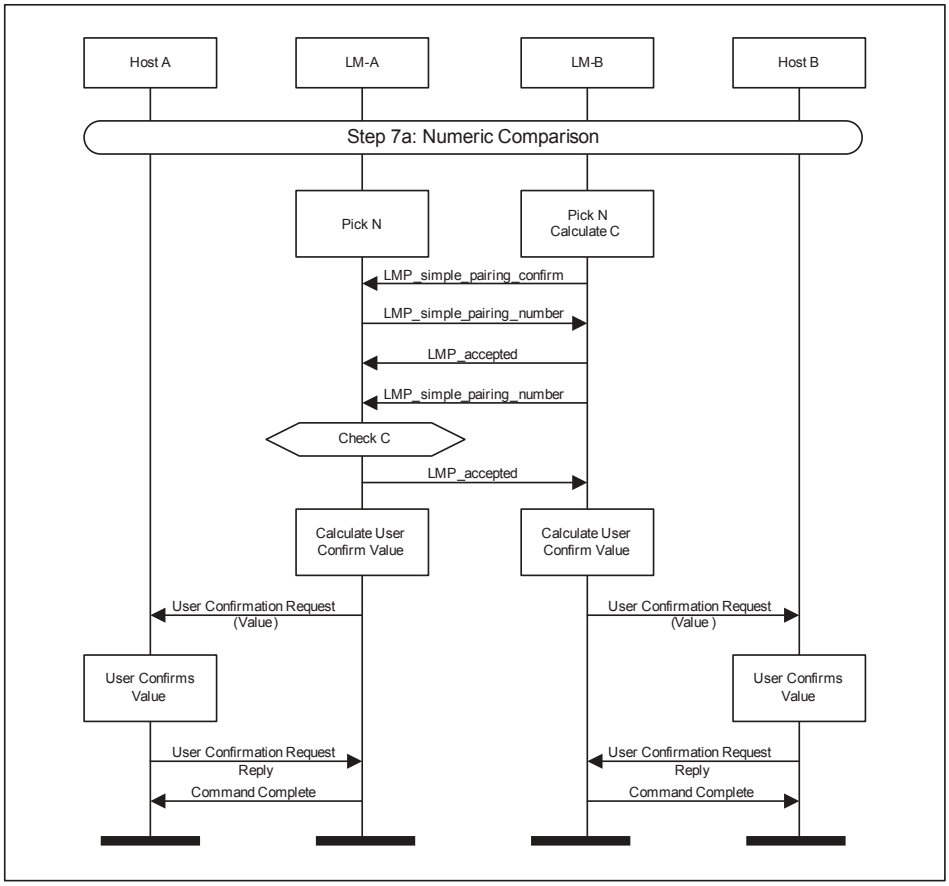



*Figure 4.10: Numeric Comparison Authentication*

### 4.2.11 Numeric Comparison Failure on Initiating Side

If the numeric comparison fails on the initiating side due to the user indicating that the confirmation values do not match, Simple Pairing is terminated.



LifeScan, Inc. Ex. 1019
Page 875 of 1420
IPR2025-00103

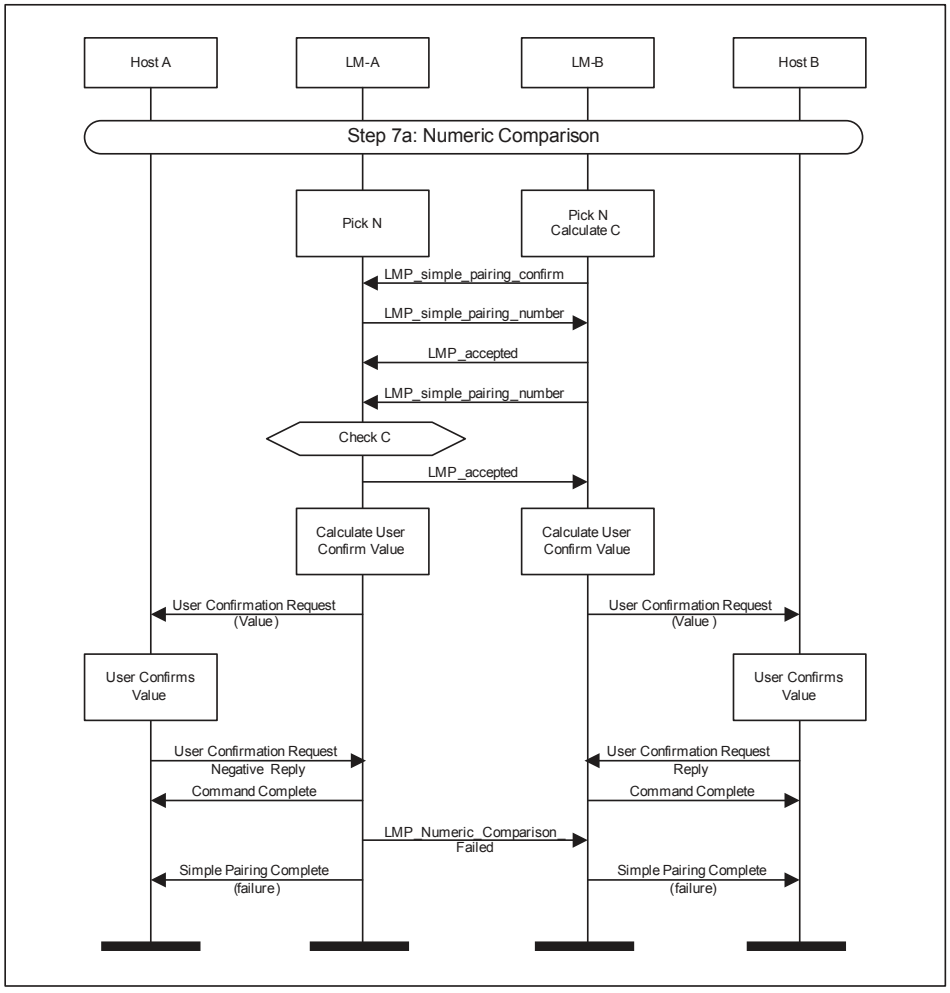



*Figure 4.11: Numeric Comparison Authentication (Failure on Initiating Side)*

### 4.2.12 Numeric Comparison Failure on Responding Side

If the numeric comparison fails on the responding side due to the user indicating that the confirmation values do not match, Simple Pairing is terminated.

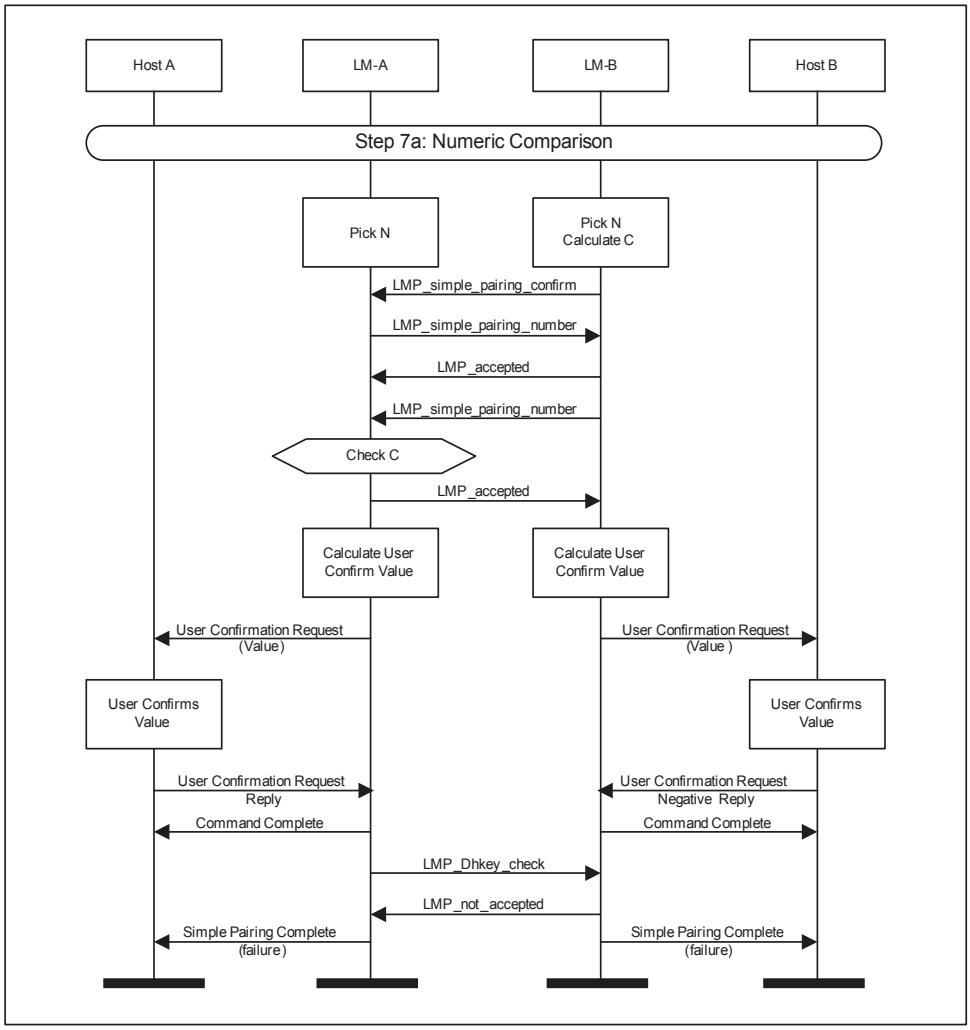



*Figure 4.12: Numeric Comparison Failure on Responding Side*

### 4.2.13 Passkey Entry

The Passkey entry step is used in two cases: when one device has numeric input only and the other device has either a display or numeric input capability or when both devices only have numeric input capability. In this step, one device display a number to be entered by the other device or the user enters a number on both devices. This number should be displayed until the end of step 8. Key press notification messages are shown during the user input phase.

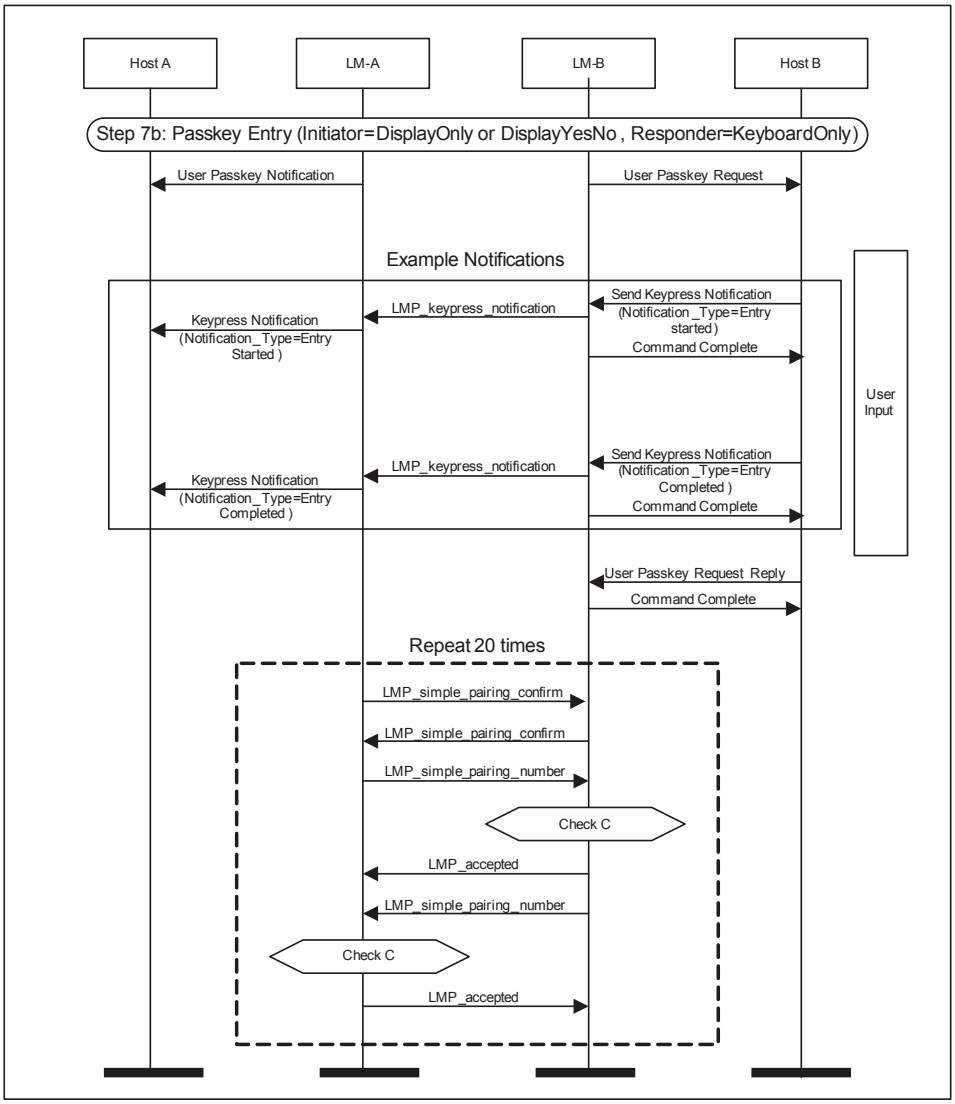



*Figure 4.13: Passkey Entry Authentication*

### 4.2.14 Passkey Entry Failure on Responding Side

If the passkey entry fails on the responding side, Simple Pairing is terminated.

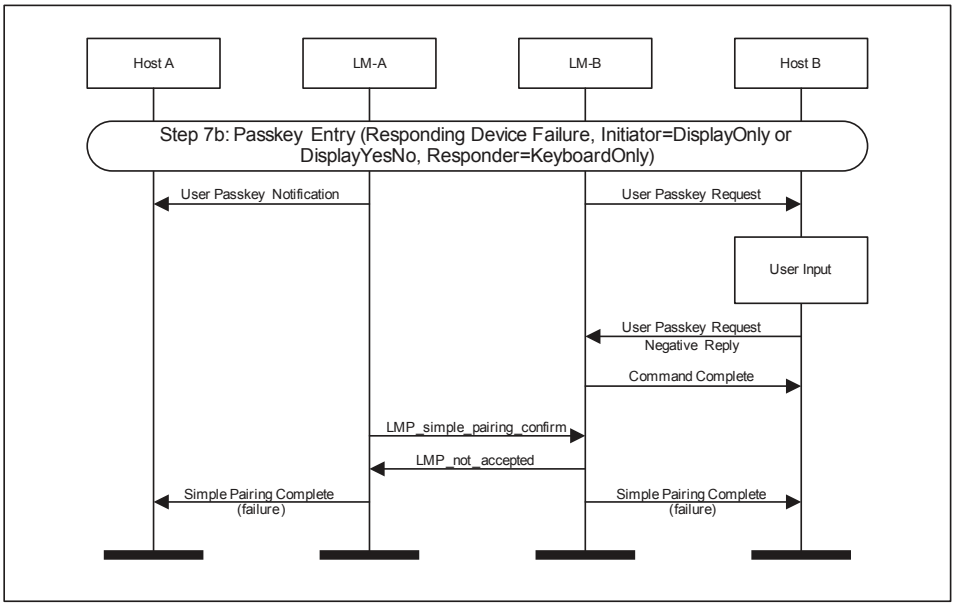



*Figure 4.14: Passkey Entry Failure on Responding Side*

### 4.2.15 Passkey Entry Failure on Initiator Side

If the passkey entry fails on the initiating side, Simple Pairing is terminated. Note that this is only possible if the initiating LM side sends an HCI User Passkey Request event.

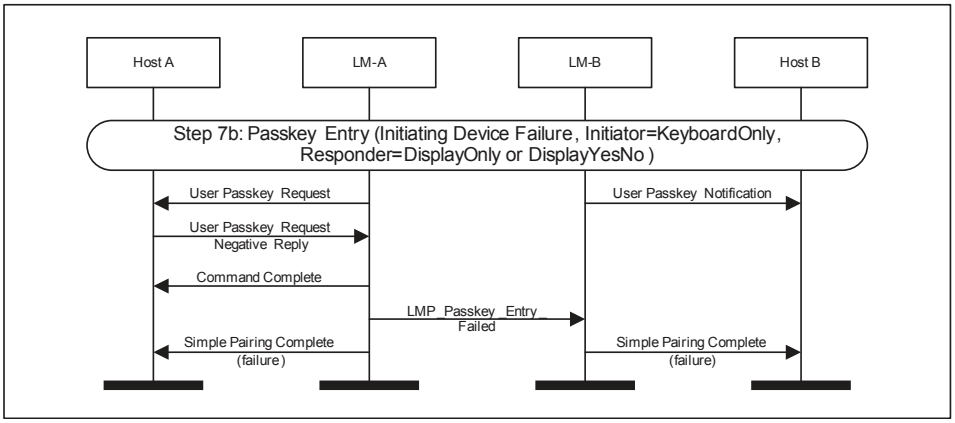



*Figure 4.15: Passkey Entry Failure on Initiating Side*

### 4.2.16 Out of Band

The OOB authentication will only be done when both devices have some OOB information to use. This step requires no user interaction.

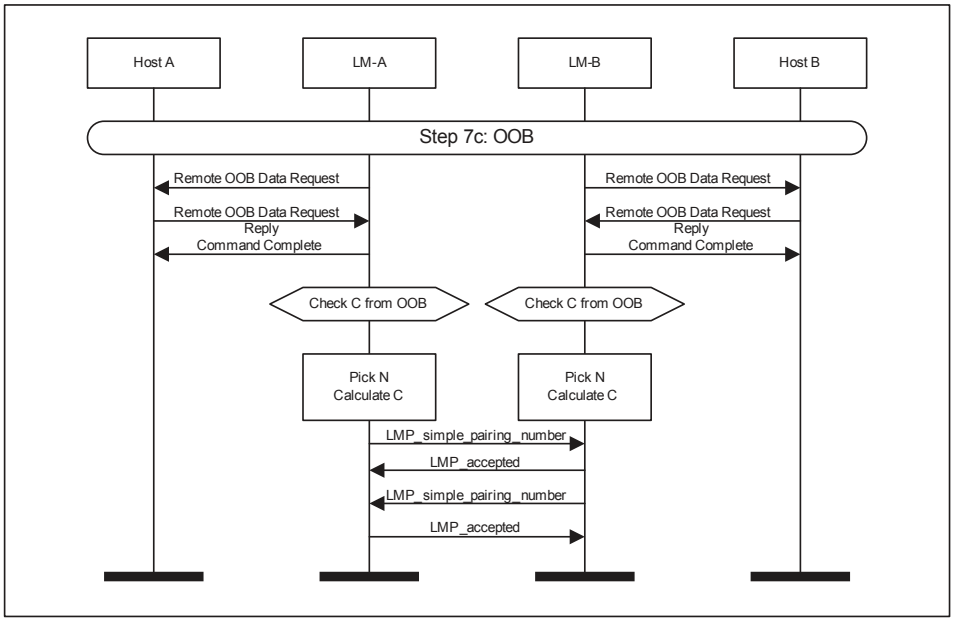



*Figure 4.16: OOB Authentication*

### 4.2.17 OOB Failure on Initiator Side

If the initiating side does not have OOB information, Simple Pairing is terminated.

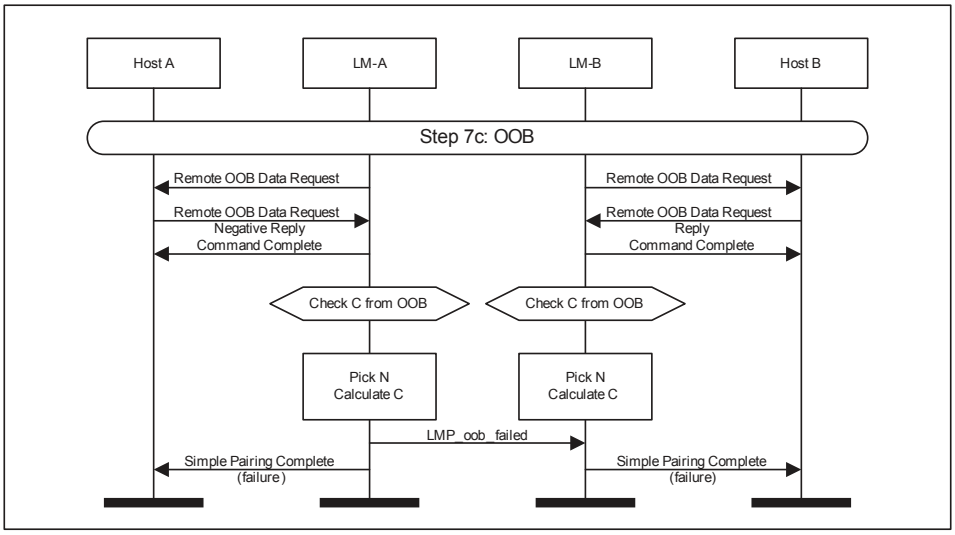



*Figure 4.17: OOB Authentication failure on initiating side*



LifeScan, Inc. Ex. 1019
Page 880 of 1420
IPR2025-00103

### 4.2.18 DHKey Checks

Once the devices have been authenticated, and the DHKey calculation has completed, the DHKey value generated is checked. If this succeeds, then both devices would have finished displaying information to the user about the process, and therefore a message is sent from the controller to the host to notify it to stop displaying this information.

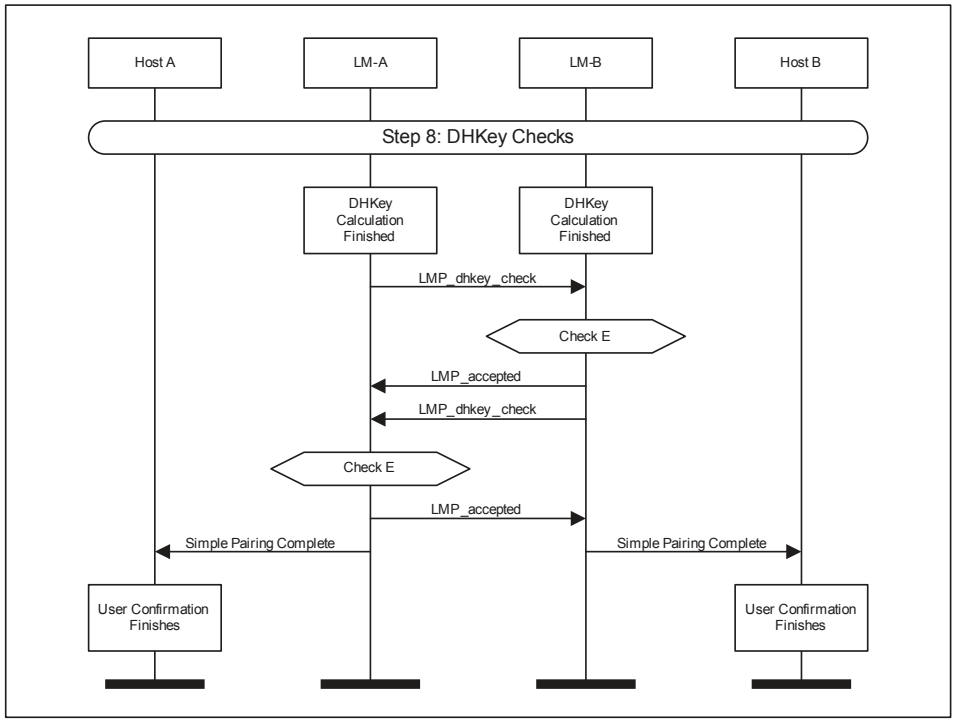



*Figure 4.18: DHKey Checks*

### 4.2.19 Calculate Link Key

Once simple pairing is complete, the link key can be calculated from the DHKey, and this should be used as input to a standard mutual authentication. Once this is complete, a Link Key Notification event will be generated.

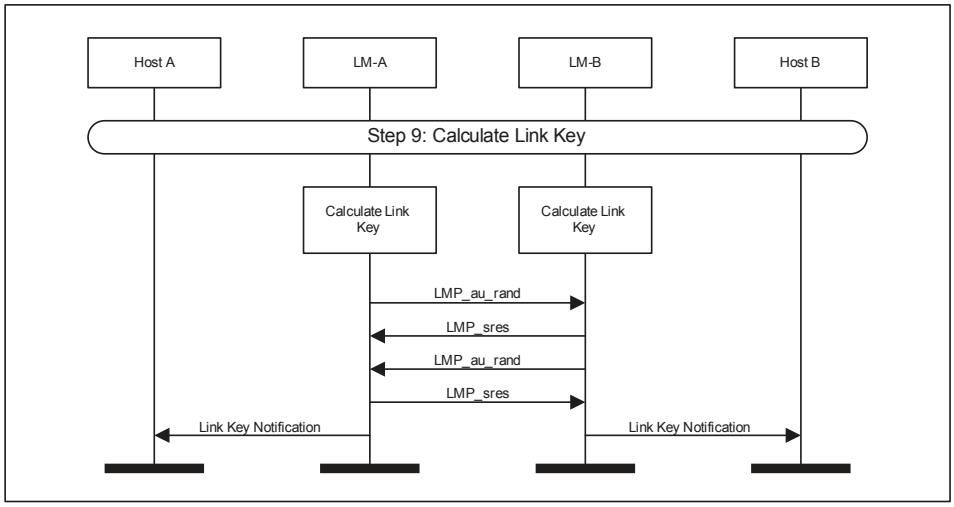



*Figure 4.19: Calculate Link Key*

### 4.2.20 Enable Encryption

Once the link key has been notified to the host, the Authentication Requested command will complete with an Authentication Complete event. The host can then turn on encryption using the standard methods.

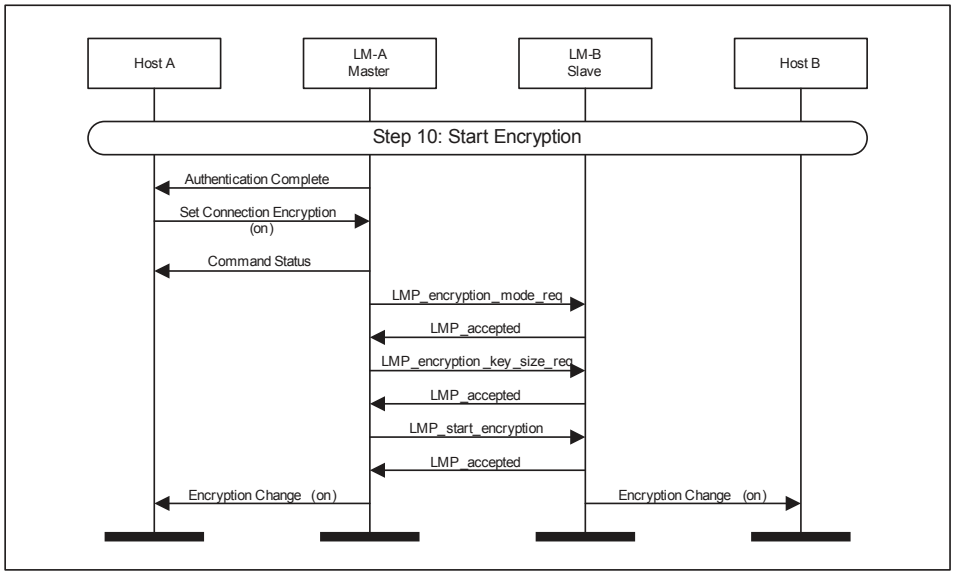



*Figure 4.20: Start Encryption*

### 4.2.21 L2CAP Connection Response

If this simple pairing was triggered by an L2CAP Connection Request, then only after all of the above steps have completed can the L2CAP Connection Response message be sent.

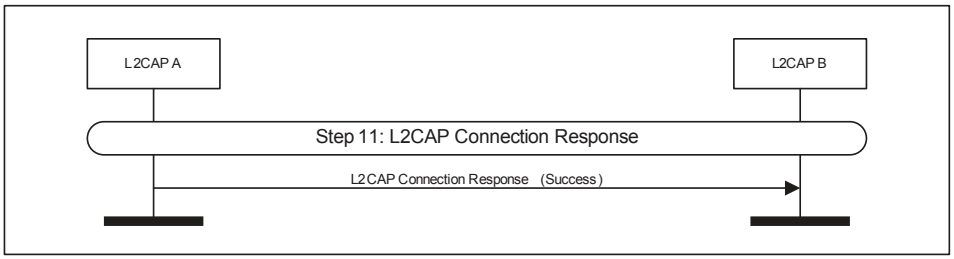



*Figure 4.21: L2CAP Connection Response*

## 4.3 LINK SUPERVISION TIMEOUT CHANGED EVENT

When enabled by the Host, the slave generates an HCI_Link_Supervision_Timeout_Changed event after the LMP_supervision_timeout PDU is received.

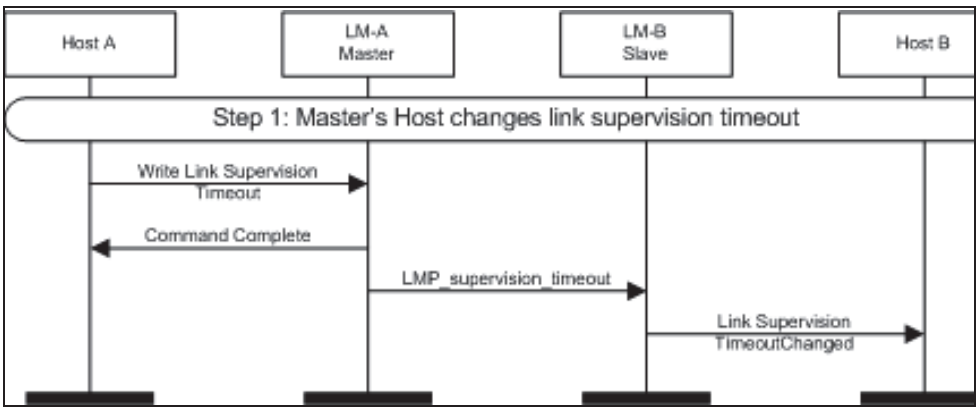



*Figure 4.22: Link Supervision Timeout Event*



LifeScan, Inc. Ex. 1019
Page 883 of 1420
IPR2025-00103

## 4.4 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 4.23 on page 714. 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 4.23 on page 714.)

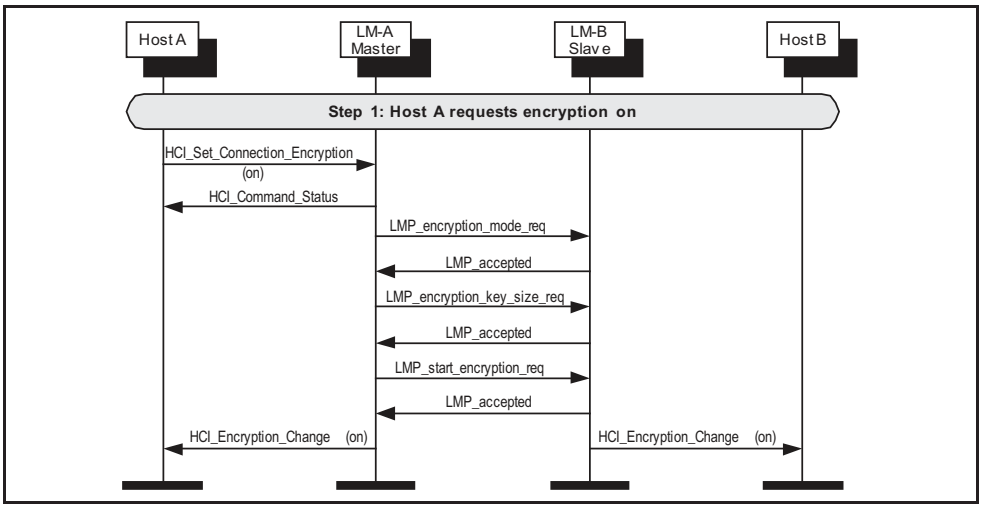



*Figure 4.23: Encryption requested.*

**Step 2:** To terminate the use of encryption, The HCI_Set_Connection_Encryption command is used. (See Figure 4.24 on page 714.)

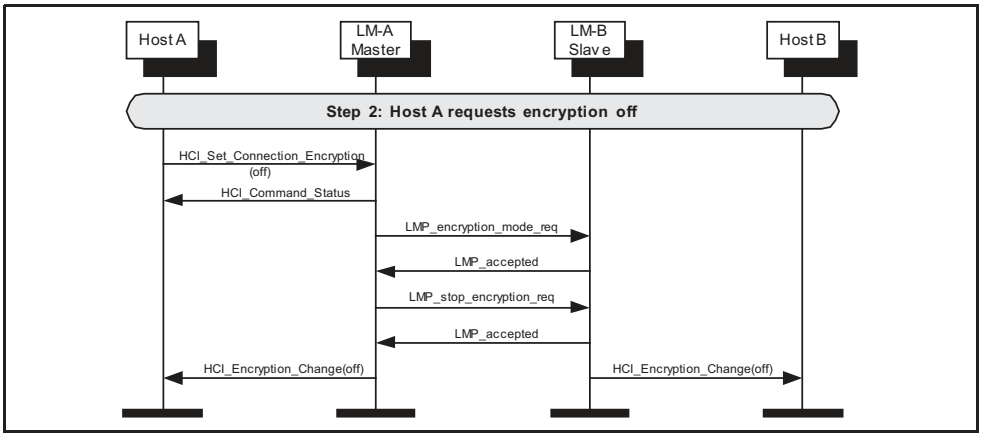



*Figure 4.24: Encryption off requested.*



LifeScan, Inc. Ex. 1019
Page 884 of 1420
IPR2025-00103

## 4.5 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 4.25 on page 715.)

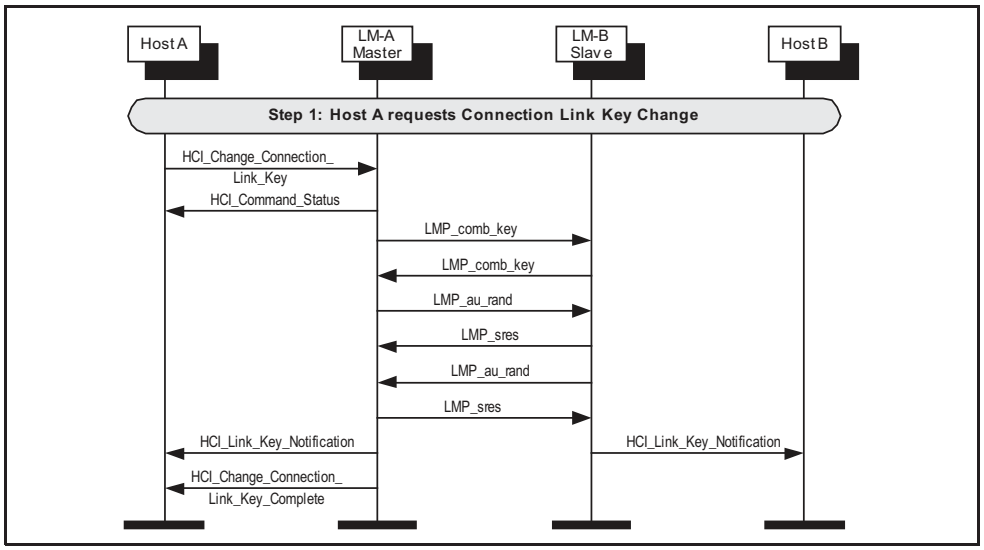



*Figure 4.25: Change connection link key.*

## 4.6 CHANGE CONNECTION LINK KEY WITH ENCRYPTION PAUSE AND RESUME

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. Encryption will then be paused and resumed, immediately using this new link key to generate a new encryption key. (See Figure 4.26 on page 716.)

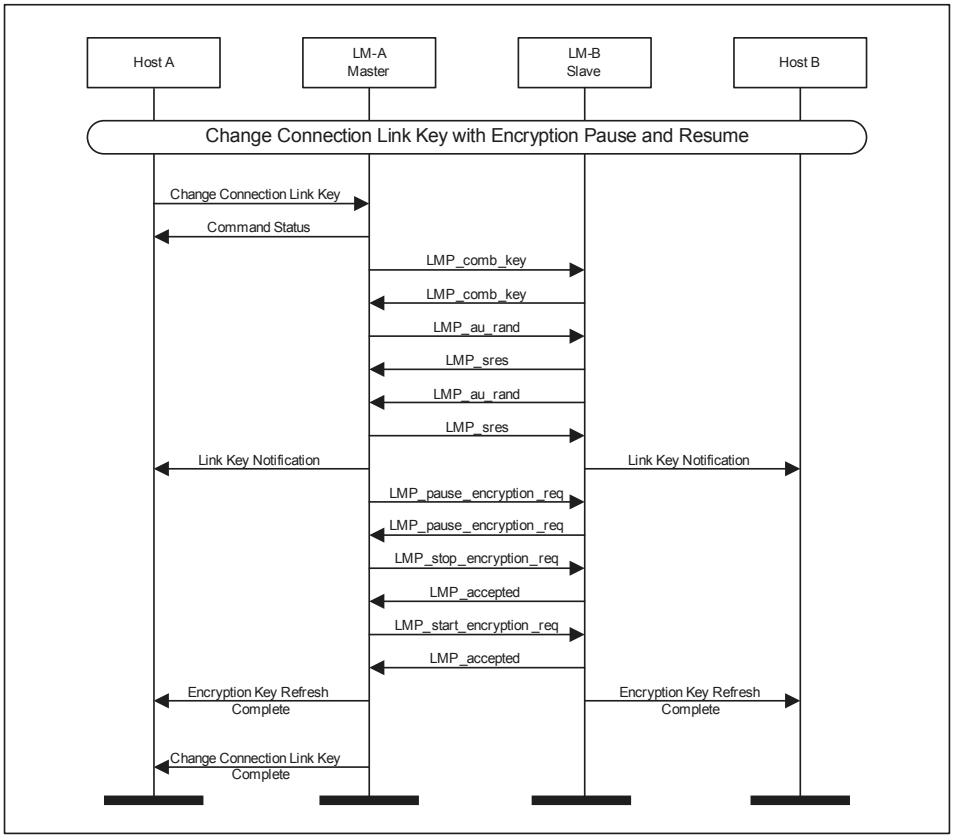



*Figure 4.26: Change Connection Link Key with Encryption Pause Resume*

## 4.7 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 4.27 on page 717.)

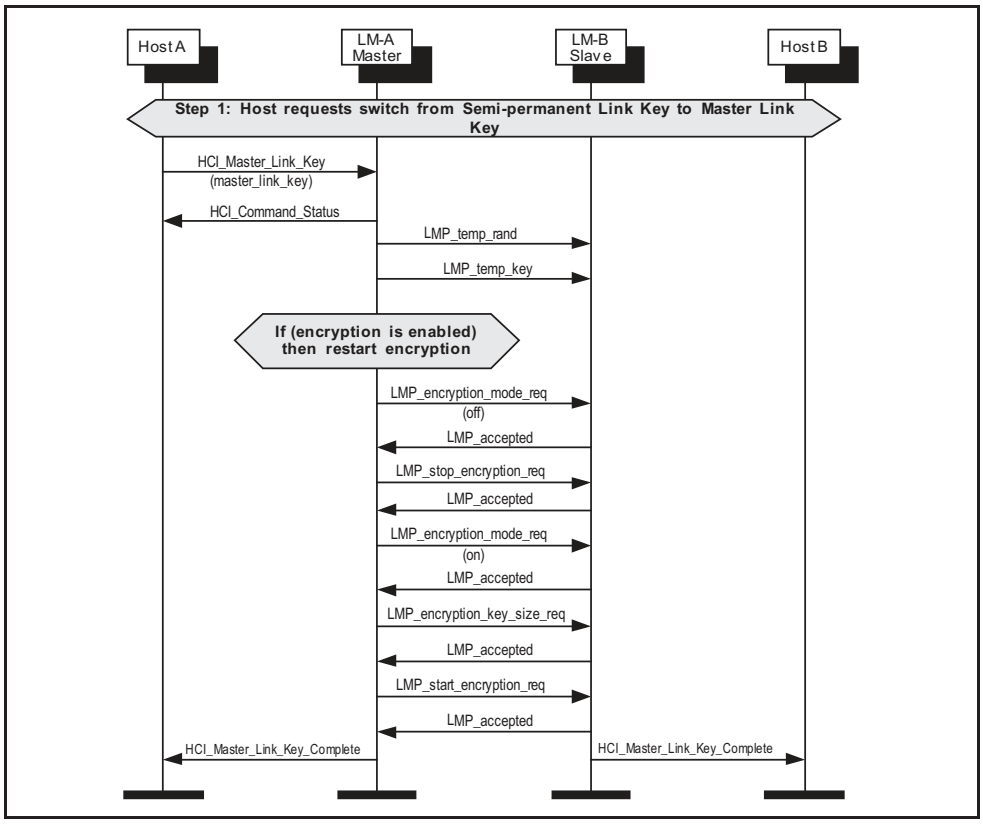



*Figure 4.27: Change to master link key.*

**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 4.28 on page 718.)

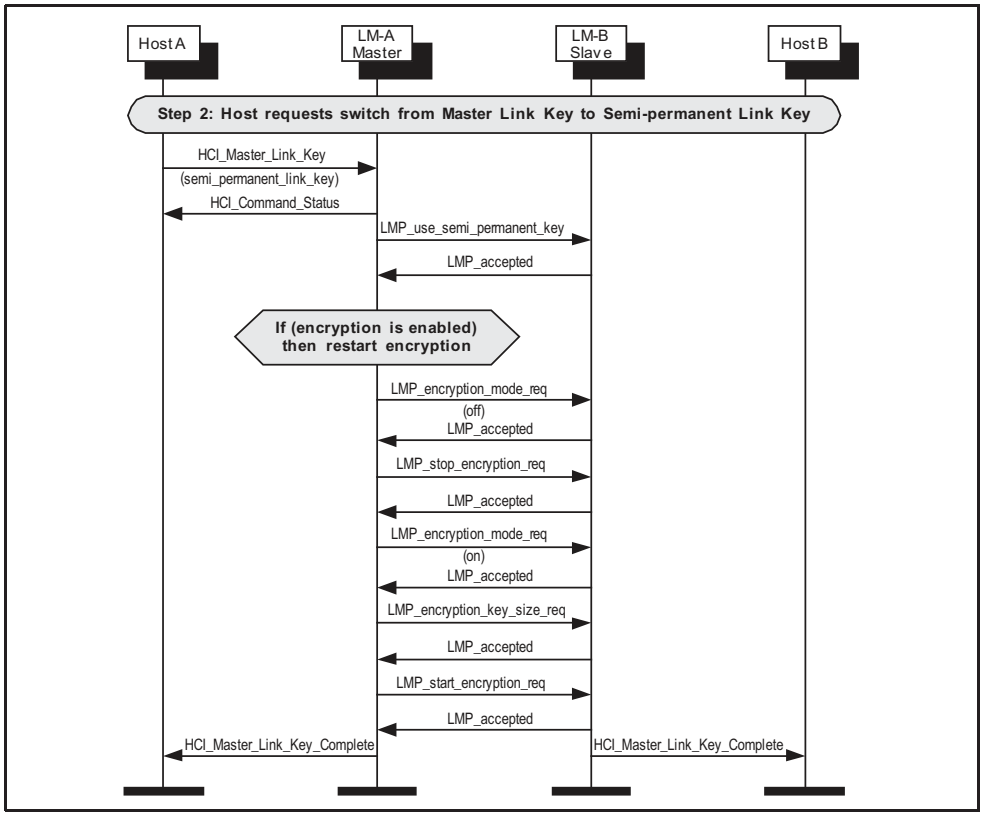



*Figure 4.28: Change to semi permanent link key.*



LifeScan, Inc. Ex. 1019
Page 888 of 1420
IPR2025-00103

## 4.8 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 4.29 on page 719.)

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.

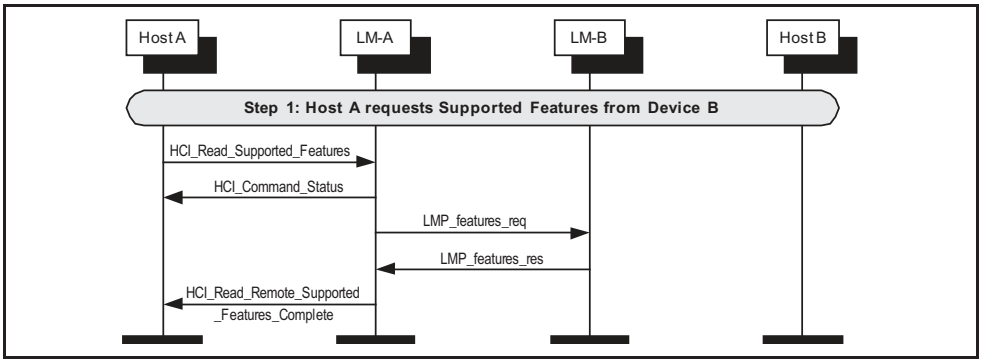



*Figure 4.29: Read remote supported features.*

## 4.9 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 4.30 on page 719.)

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.

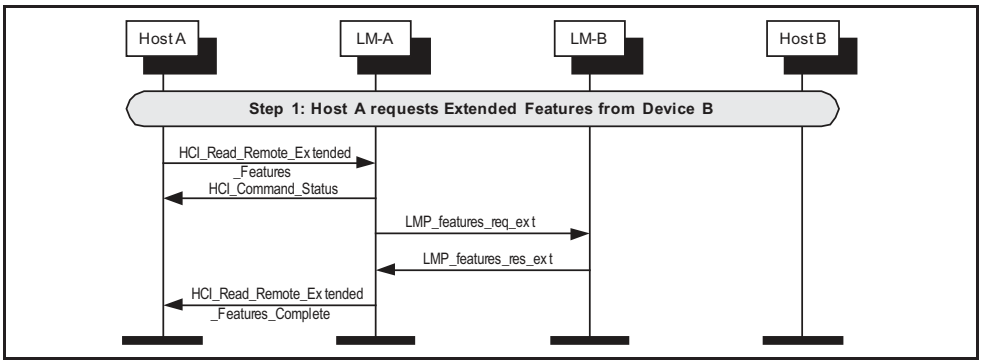



*Figure 4.30: Read remote extended features.*

## 4.10 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 4.31 on page 720.)

**Step 1:** The host requests the clock offset of a remote device.

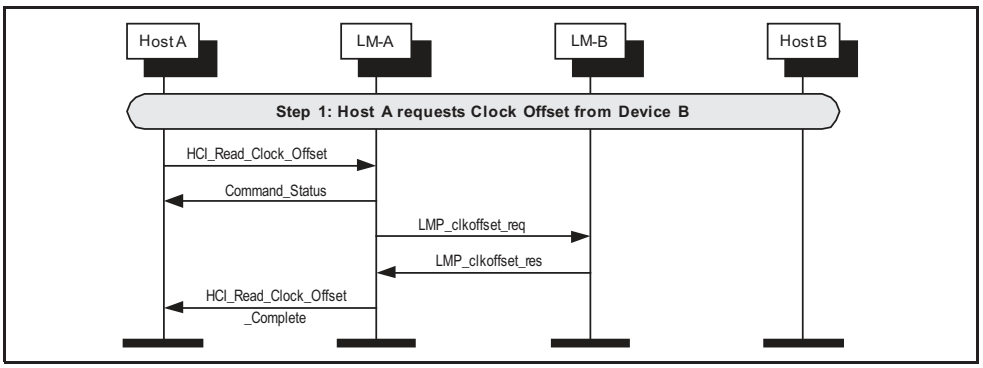



*Figure 4.31: Read clock offset.*

## 4.11 ROLE SWITCH ON AN ENCRYPTED LINK USING ENCRYPTION PAUSE AND RESUME

The HCI_Switch_Role command can be used to explicitly switch the current master / slave role of the local device with the specified device. The master host (A) requests a role switch with a slave. This will first pause encryption, and then send the switch request, and the slave will respond with the slot offset and accepted. The role switch is performed by doing the TDD switch and piconet switch. Encryption is resumed, and finally an HCI_Role_Change event is sent on both sides. (Section 4.12 on page 721)

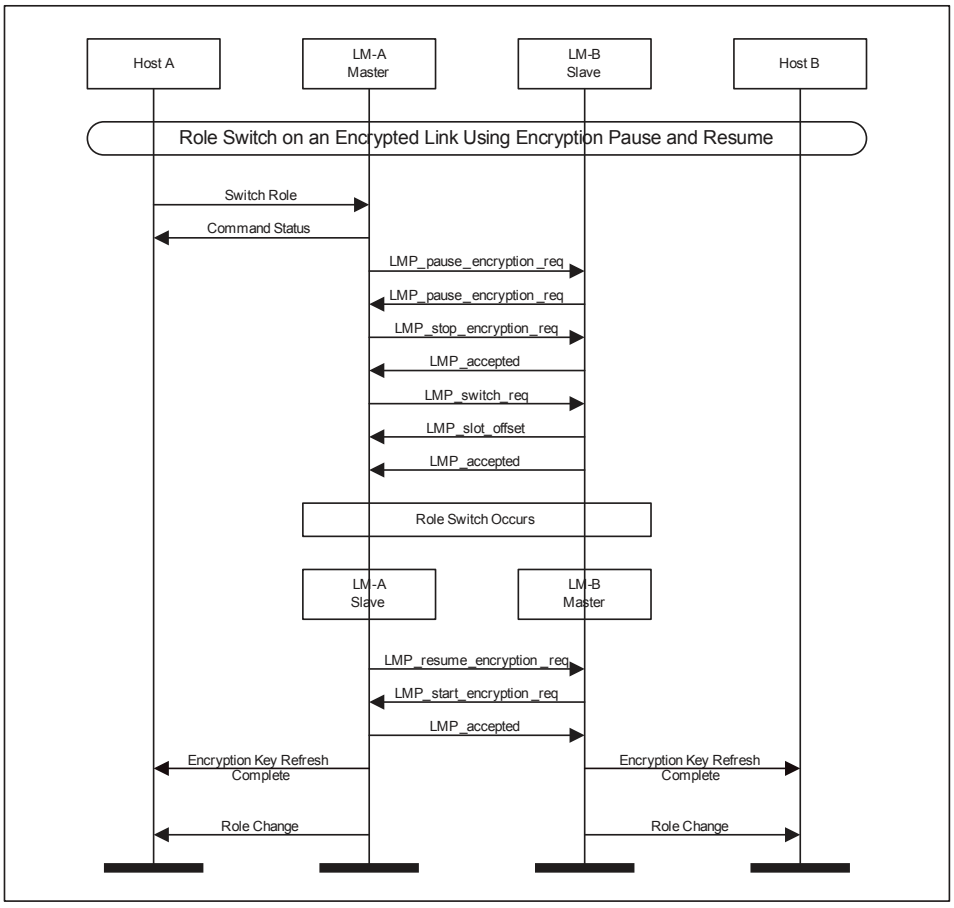



*Figure 4.32: Role Switch on an Encrypted link using encryption pause and resume.*

## 4.12 REFRESHING ENCRYPTION KEYS

The HCI_Refresh_Encryption_Key command may be used by the master's Host to explicitly pause and resuming encryption to refresh the encryption key. After encryption is resumed an HCI_Encryption_Key_Freshness event is sent on both sides. (Section 4.13 on page 708).

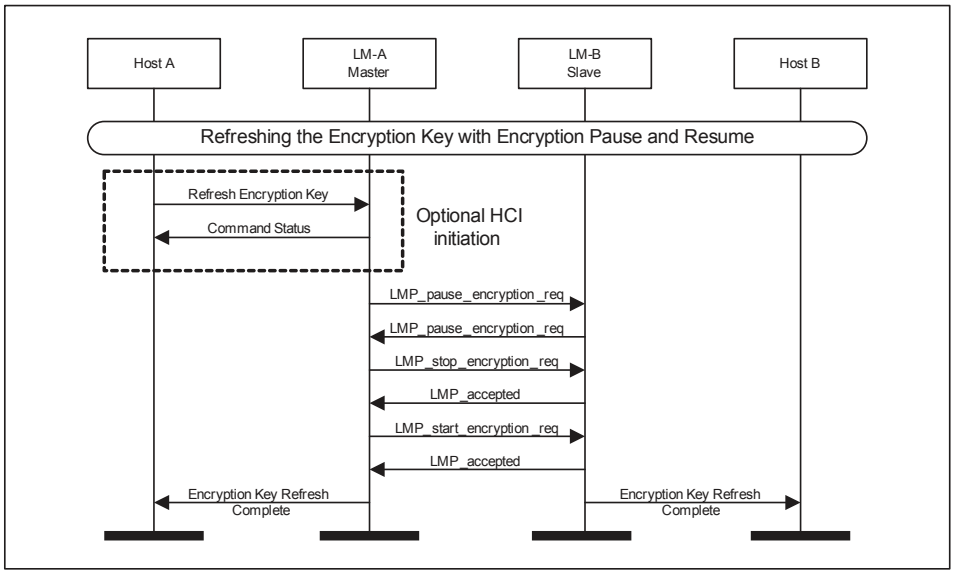



*Figure 4.33: Refreshing encryption keys using encryption pause and resume*

## 4.13 READ REMOTE VERSION INFORMATION

Using the HCI_Read_Remote_Version_Information command the version information of a remote device can be read. (See Figure 4.34 on page 722.)

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.

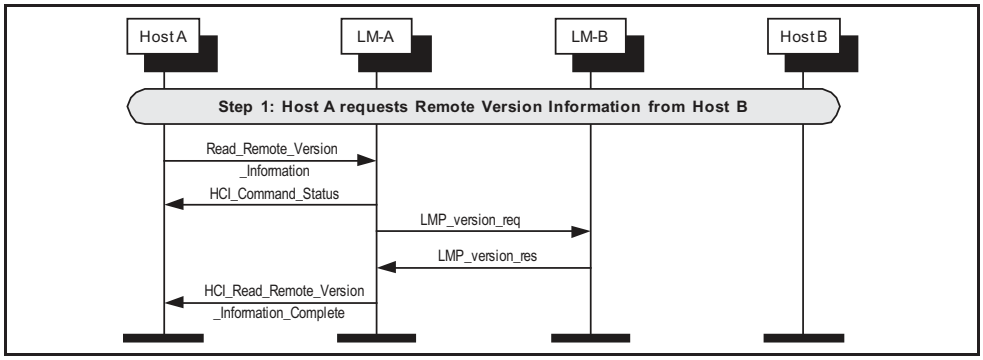



*Figure 4.34: Read remote version information.*

## 4.14 QOS SETUP

Using the HCI_Flow_Specification command the Quality of Service (QoS) and Flow Specification requirements of a connection can be notified to a Controller.



LifeScan, Inc. Ex. 1019
Page 892 of 1420
IPR2025-00103

The Controller may then change the quality of service parameters with a remote device. (See Figure 4.35 on page 723.)

**Step 1:** The host sends QoS parameters to a remote device.

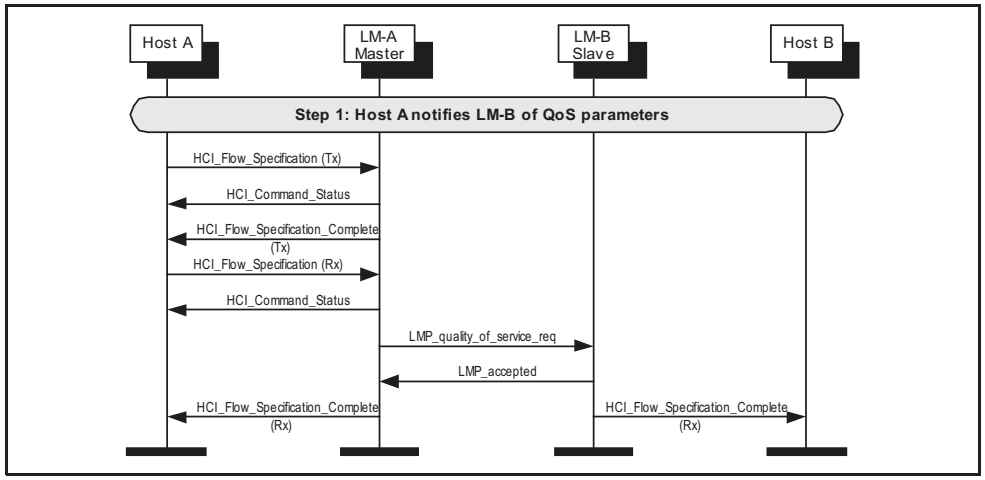



*Figure 4.35: QoS flow specification.*

## 4.15 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 accepted. (See Figure 4.36 on page 723.)

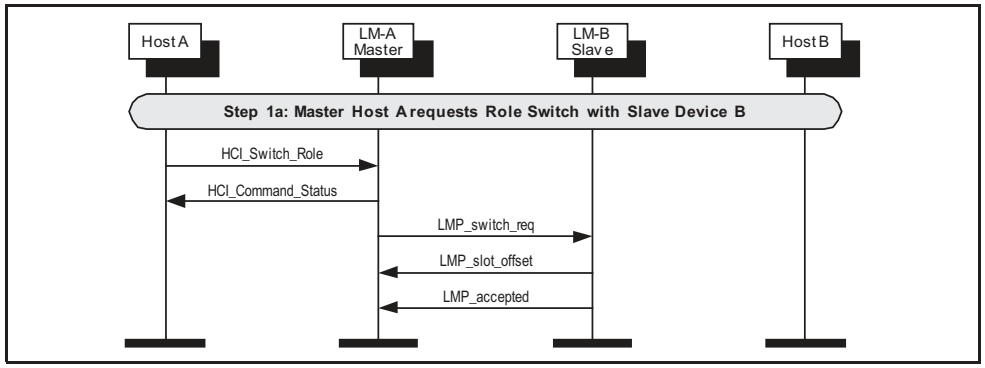



*Figure 4.36: Master requests role switch.*



LifeScan, Inc. Ex. 1019
Page 893 of 1420
IPR2025-00103

**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 4.37 on page 724.)

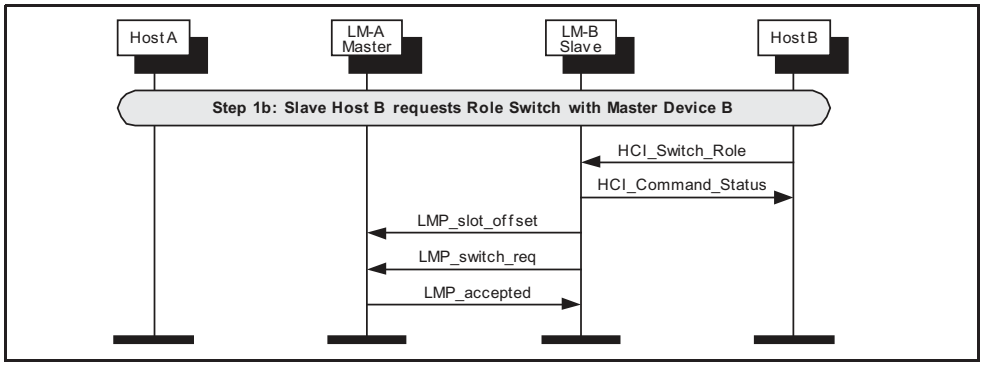



*Figure 4.37: Slave requests role switch.*

**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 4.38 on page 724.)

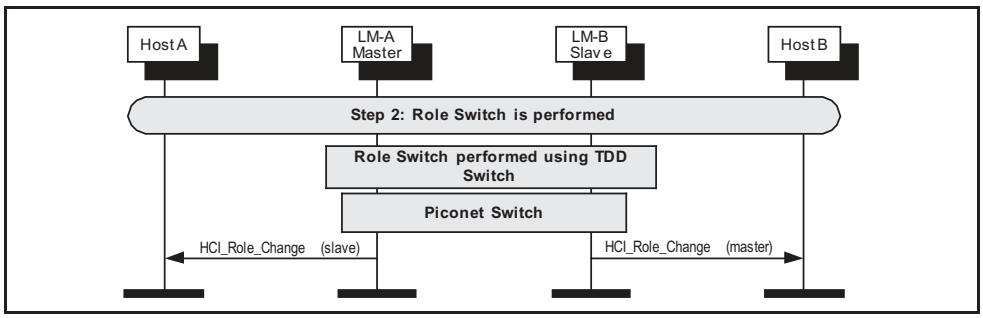



*Figure 4.38: Role switch is performed.*

# 5 SYNCHRONOUS CONNECTION ESTABLISHMENT AND DETACHMENT

## 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 synchronous connection can be created.

**Step 1a:** Master device requests a synchronous connection with a device. (See Figure 5.1 on page 725.)

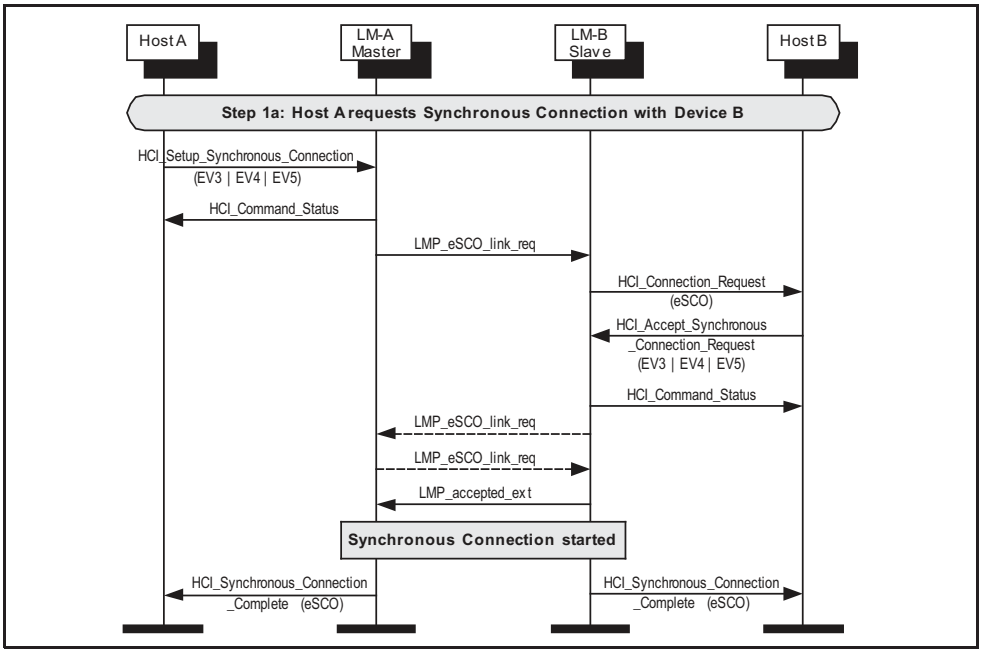



*Figure 5.1: Master requests synchronous EV3, EV4 OR EV5 connection.*



LifeScan, Inc. Ex. 1019
Page 895 of 1420
IPR2025-00103

**Step 1b:** Slave device requests a synchronous connection with a device. (See Figure 5.2 on page 726.)

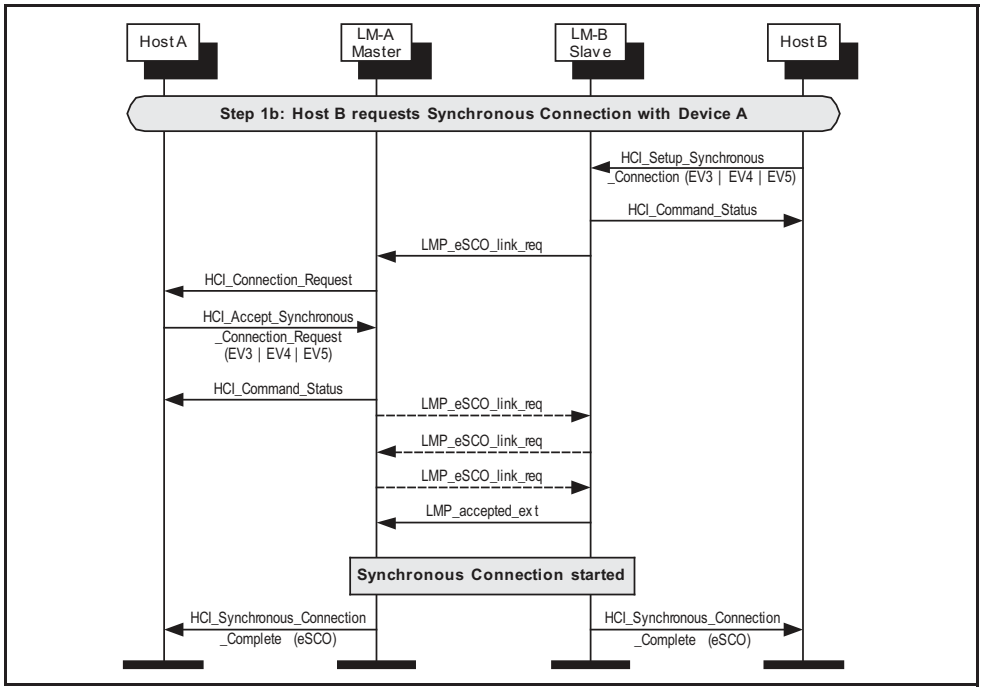



*Figure 5.2: Slave requests synchronous EV3, EV4 OR EV5 connection.*

**Step 1c:** Master device requests a SCO connection with a device. (See Figure 5.3 on page 726.)

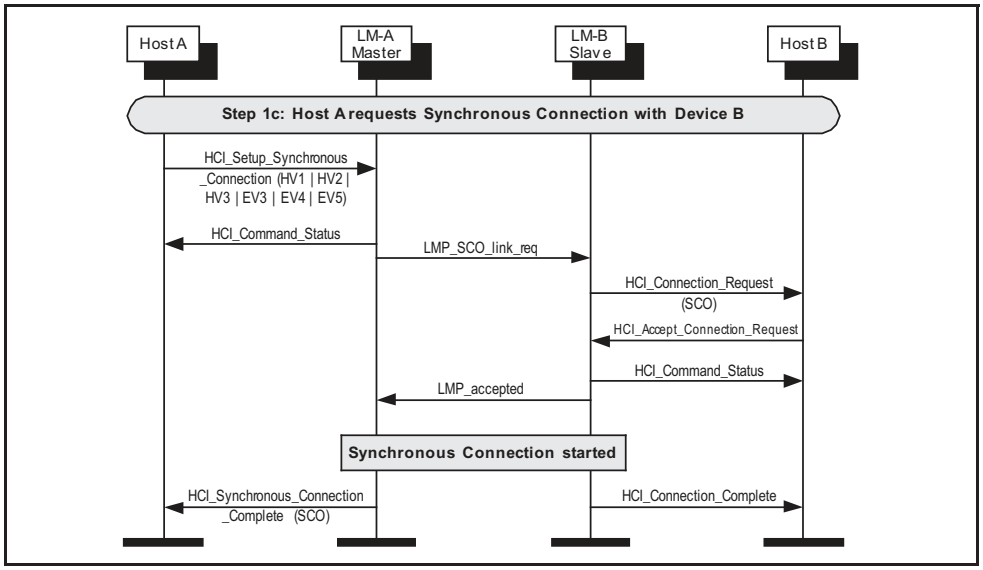



*Figure 5.3: Master requests synchronous connection using SCO.*



LifeScan, Inc. Ex. 1019
Page 896 of 1420
IPR2025-00103

**Step 1d:** Master device requests a SCO connection with a device.
(See Figure 5.4 on page 727.)

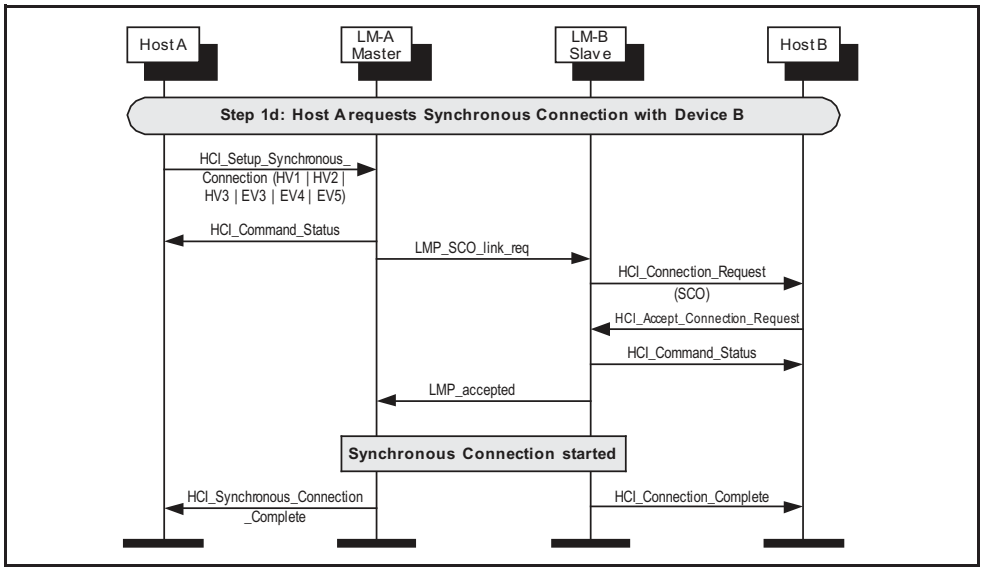



*Figure 5.4: Master requests synchronous connection with legacy slave.*

**Step 1e:** Host device requests a SCO connection with a device.
(See Figure 5.5 on page 727.)

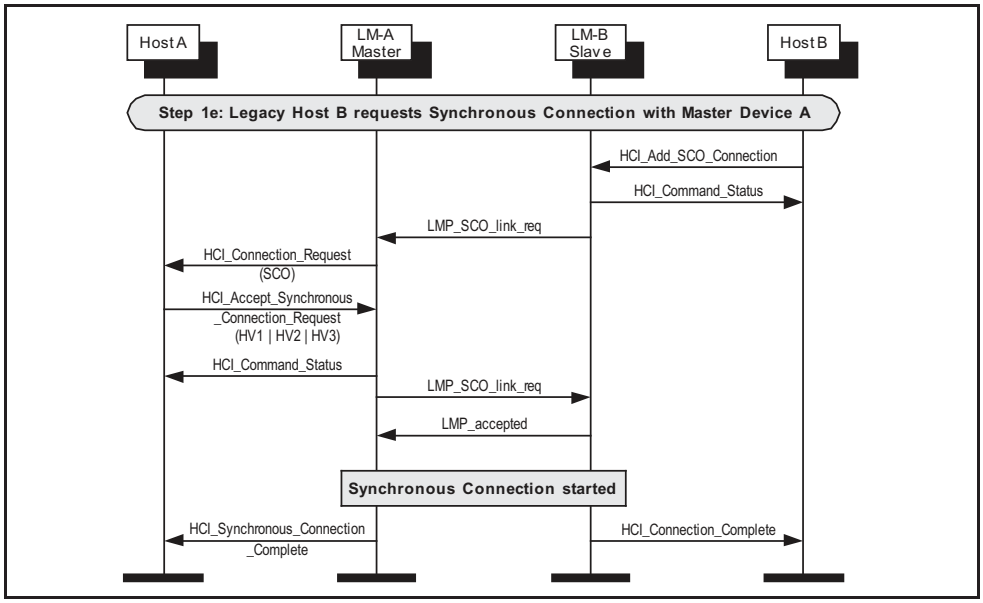



*Figure 5.5: Any device that supports only SCO connections requests a synchronous connection with a device.*

**Step 2a:** Master renegotiates eSCO connection (See Figure 5.6 on page 728.)

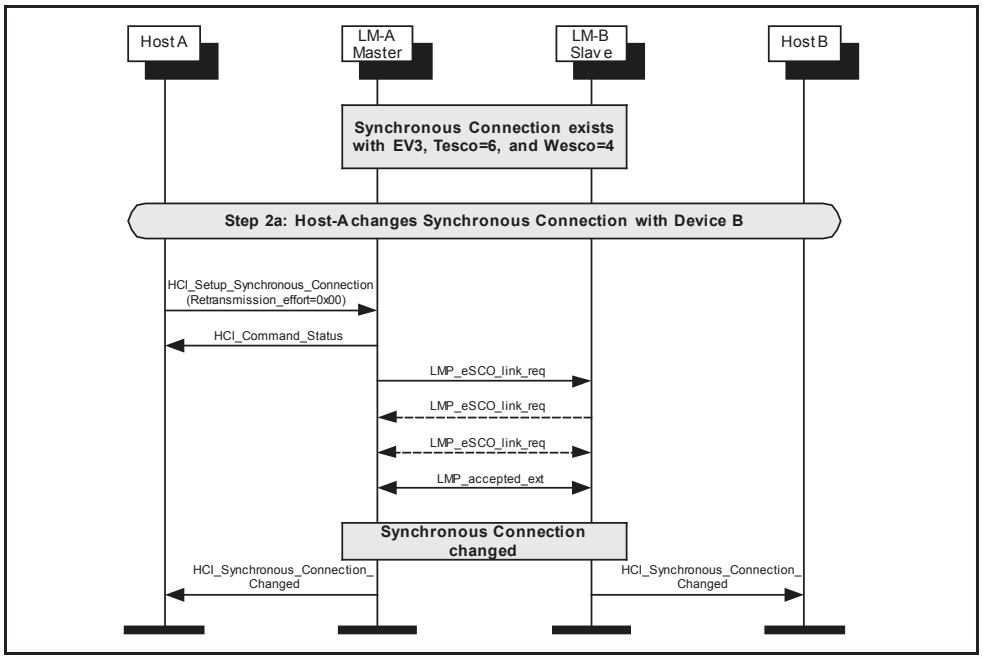



*Figure 5.6: Master renegotiates eSCO connection.*

**Step 2b:** Slave renegotiates eSCO connection (See Figure 5.7 on page 728.)

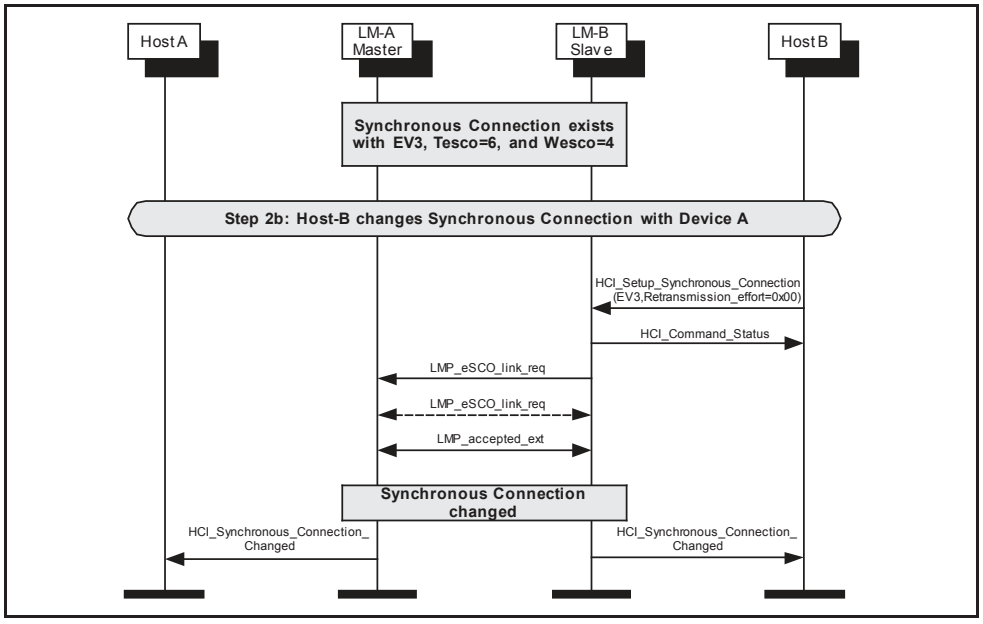



*Figure 5.7: Slave renegotiates eSCO connection.*

**Step 3a:** eSCO Disconnection. (See Figure 5.8 on page 729.)

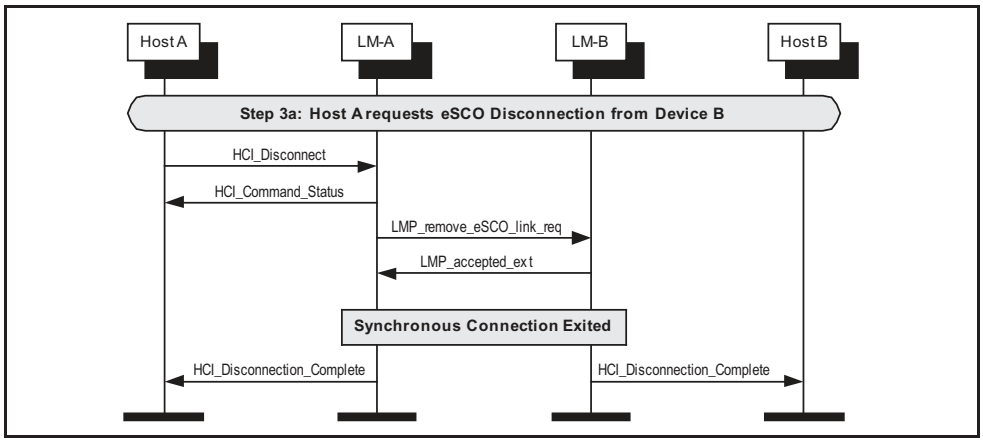



*Figure 5.8: Synchronous disconnection of eSCO connection.*

**Step 3b:** SCO Disconnection. (See Figure 5.9 on page 729.)

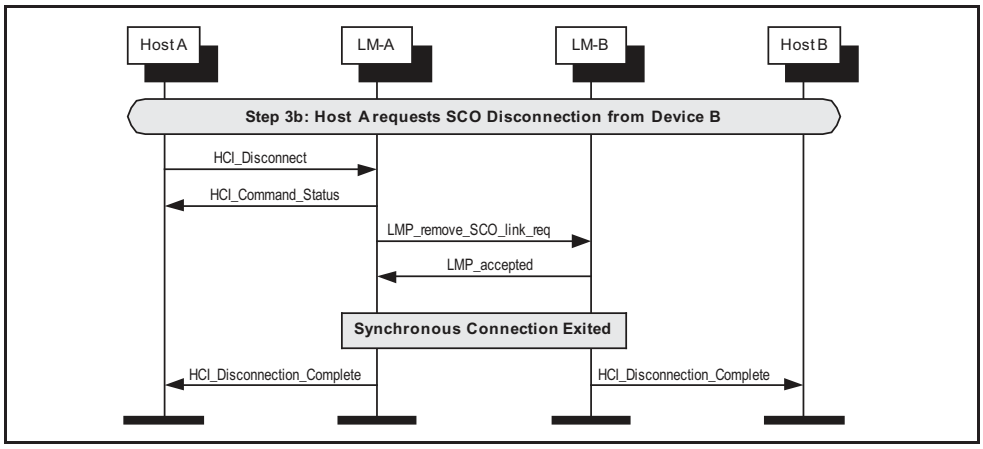



*Figure 5.9: Synchronous disconnection of SCO connection.*

# 6 SNIFF, HOLD AND PARK

Entry into Sniff mode, Hold mode or Park state requires an established ACL Connection.

## 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 6.1 on page 730.)

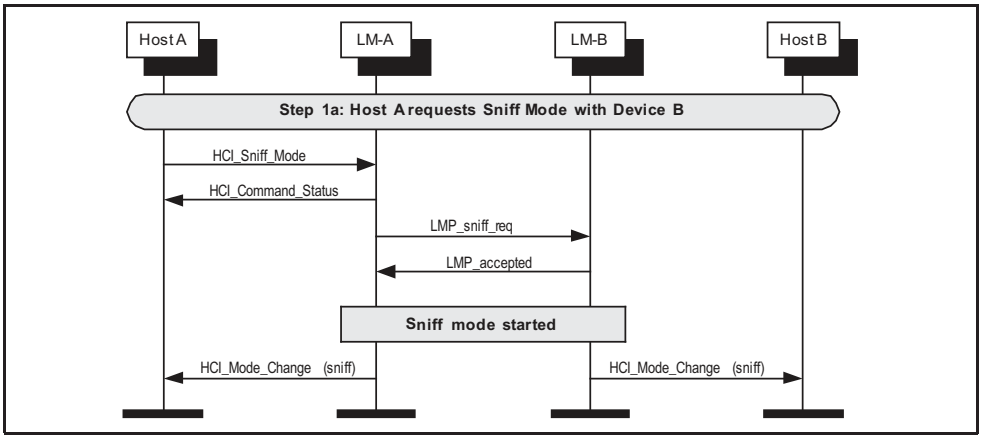



*Figure 6.1: Sniff mode request.*

**Step 2:** Host requests to exit sniff mode. (See Figure 6.2 on page 730.)

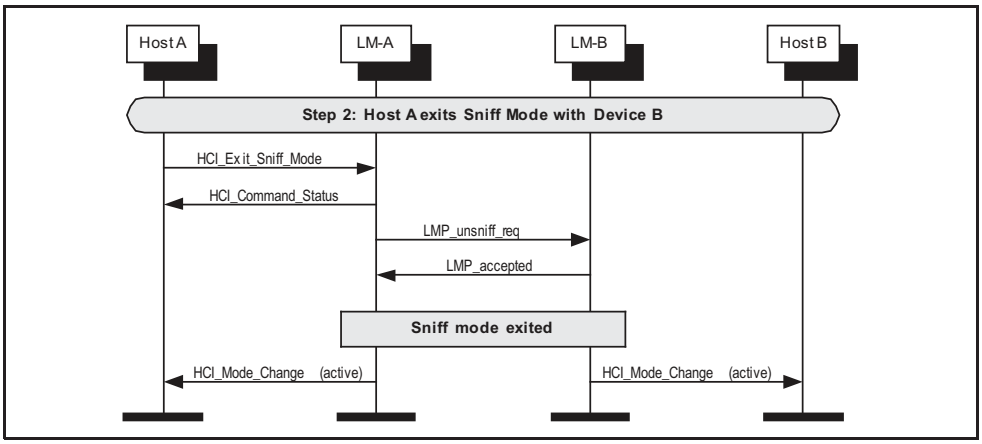



*Figure 6.2: Exit sniff mode request.*



LifeScan, Inc. Ex. 1019
Page 900 of 1420
IPR2025-00103

## 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 6.3 on page 731.)

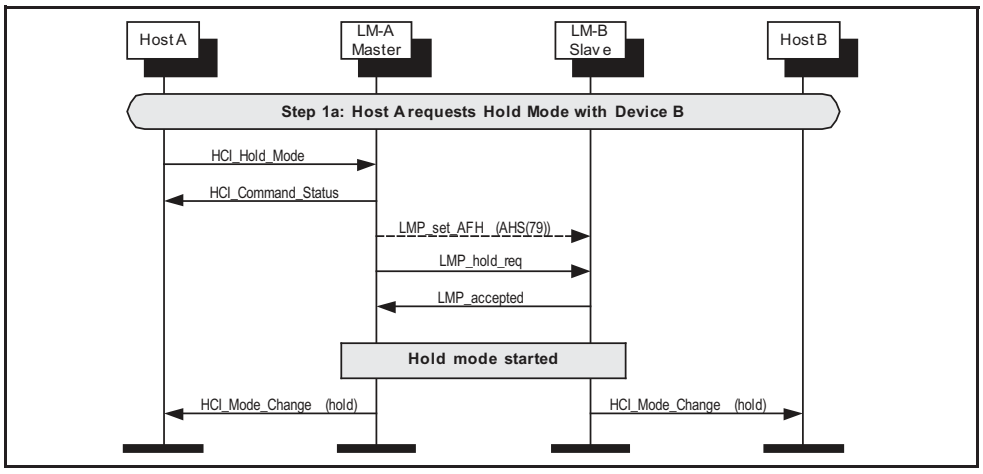



*Figure 6.3: Hold request.*

**Step 1b:** A host may force hold mode. (See Figure 6.4 on page 731.)

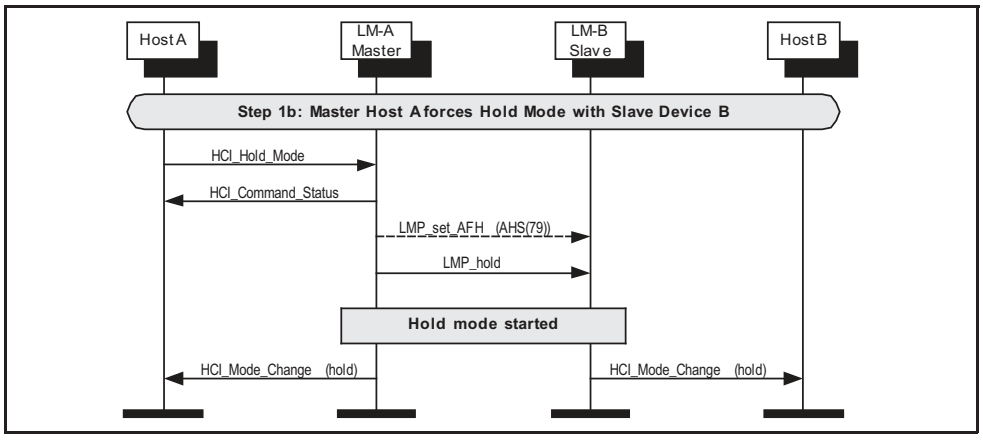



*Figure 6.4: Master forces hold mode.*



LifeScan, Inc. Ex. 1019
Page 901 of 1420
IPR2025-00103

**Step 1c:** A slave device requests hold mode. (See Figure 6.5 on page 732.)

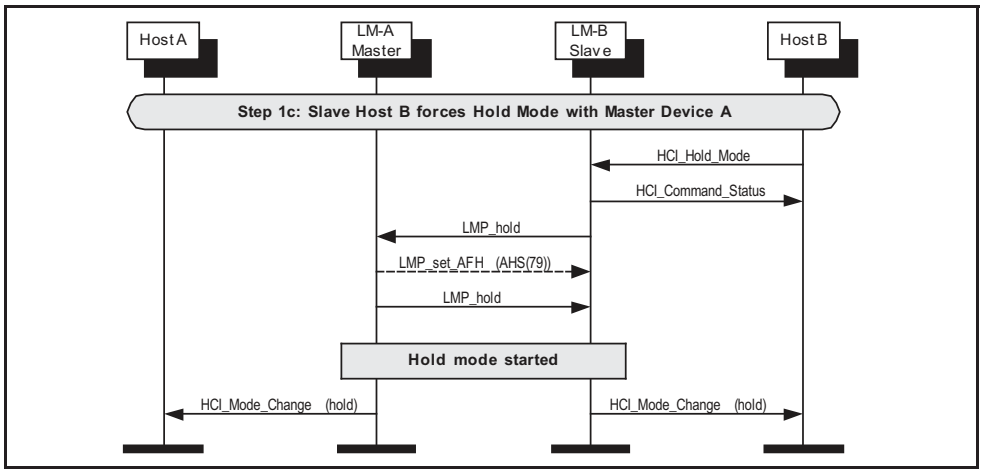



*Figure 6.5: Slave forces hold mode.*

**Step 2:** When hold mode completes the hosts are notified using the HCI_Mode_Change event. (See Figure 6.6 on page 732.)

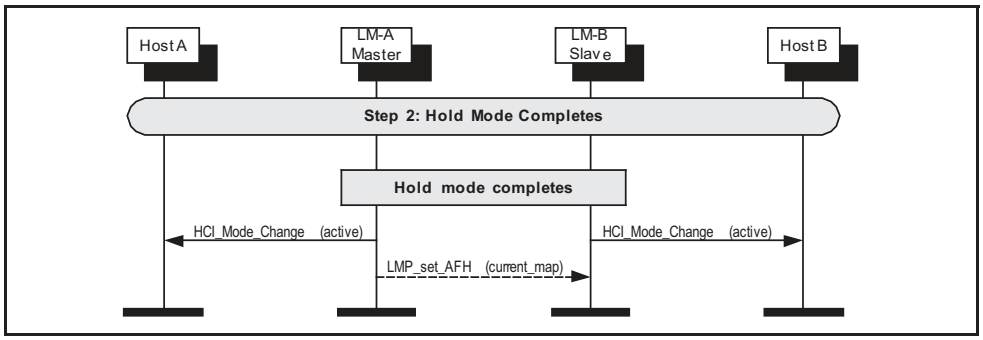



*Figure 6.6: Hold mode completes.*

## 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 the park state. Before sending the LMP_park_req PDU, the master may disable AFH by setting the connection into AHS(79). (See Figure 6.7 on page 733.)

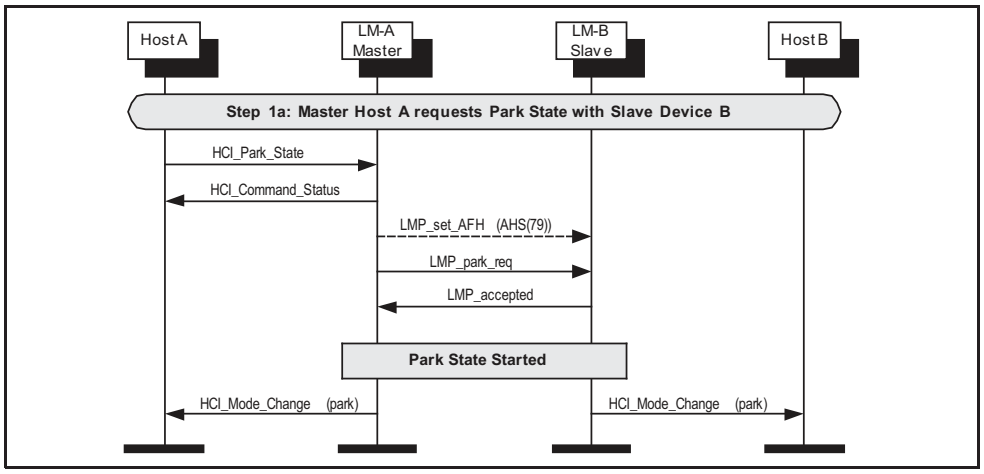



*Figure 6.7: Park state request from master.*

**Step 1b:** The slave requests to be placed in the 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 6.8 on page 733.)

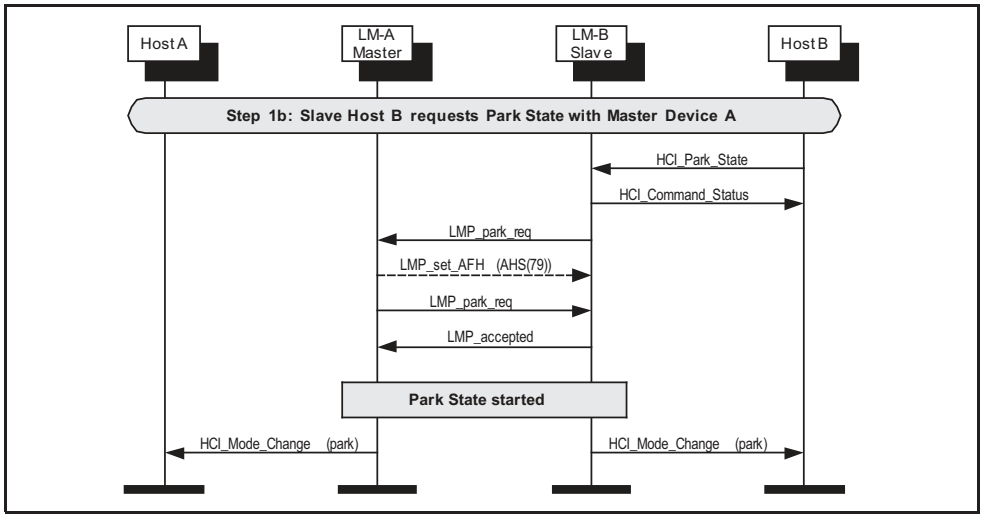



*Figure 6.8: Park state request from slave.*

**Step 2:** When in the park state, a slave still needs to 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 the figure. (See Figure 6.9 on page 734.)

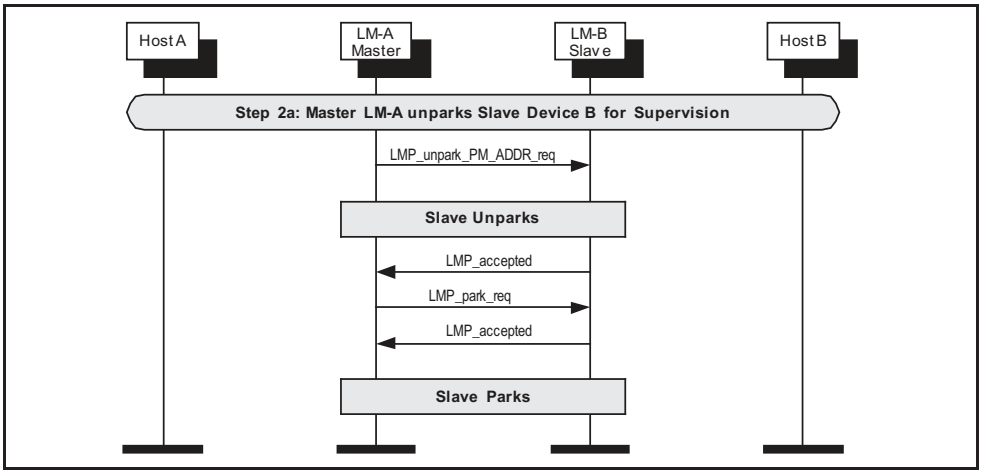



*Figure 6.9: Master unparks slave for supervision.*

**Step 3a:** A master may unpark a slave to exit park state. The master should re-enable AFH by setting the current AFH channel map to the unparked slave. (See Figure 6.10 on page 734.)

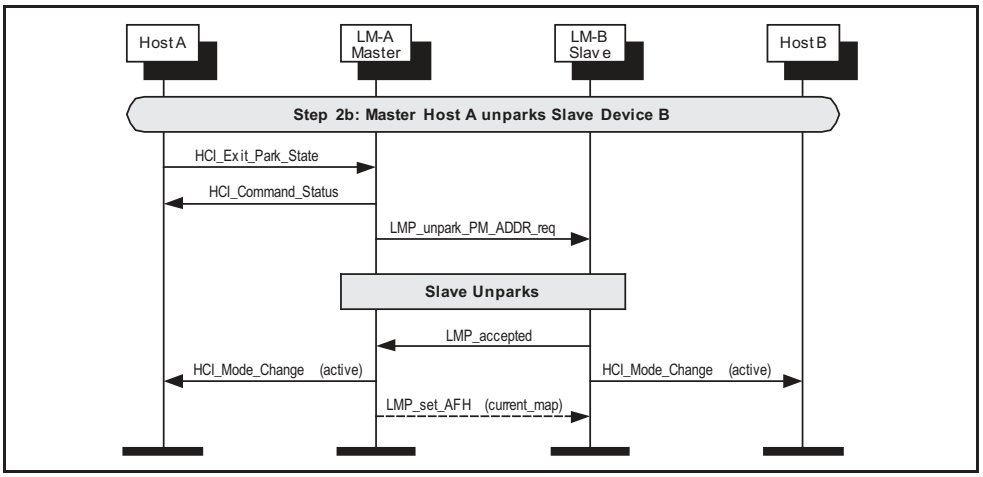



*Figure 6.10: Master exits park state with slave.*

**Step 3b:** A slave may request to be unparked by sending a message in an access window. It will then receive instructions from the master to unpark. The master should re-enable AFH by setting the current AFH channel map to the unparked slave. (See Figure 6.11 on page 735.)

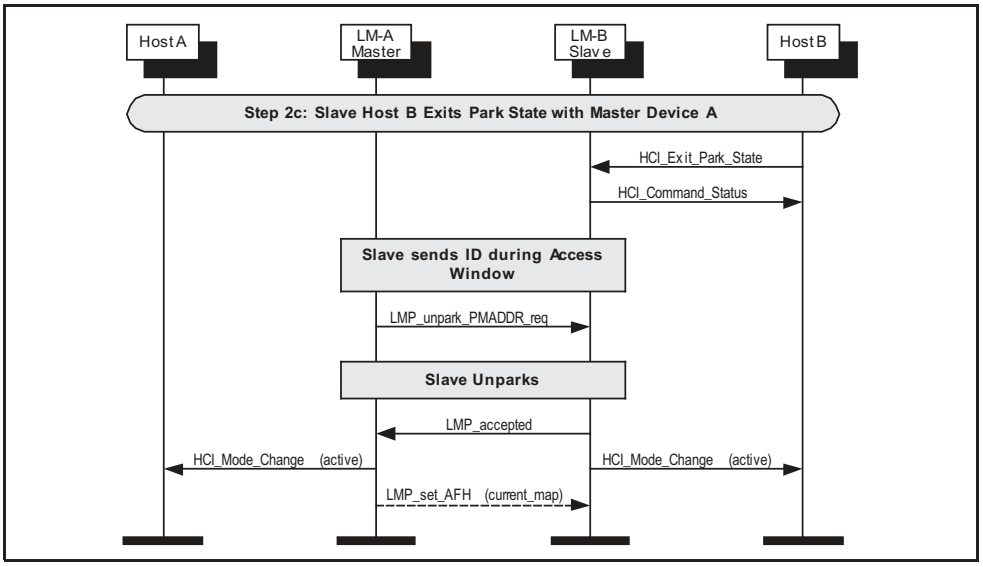



*Figure 6.11: Slave exits park state with master.*



LifeScan, Inc. Ex. 1019
Page 905 of 1420
IPR2025-00103

# 7 BUFFER MANAGEMENT, FLOW CONTROL

Buffer management is very important for resource limited devices. This can be achieved on the Host Controller Interface using the HCI_Read_Buffer_Size command, and the HCI_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 Baseband acknowledgement has been received for this data, then an HCI_Number_Of_Completed_Packets event will be generated. (See Figure 7.1 on page 736.)

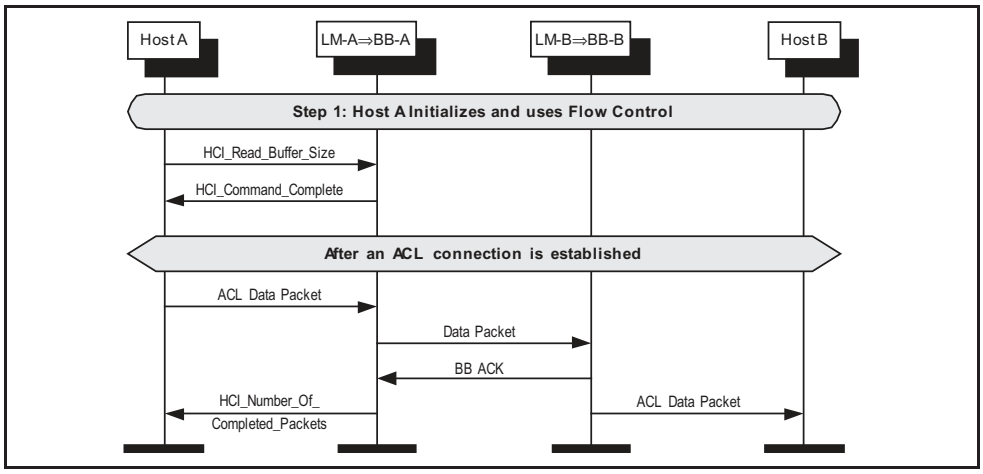



*Figure 7.1: Host to Controller flow control.*

**Step 2:** During initialization, the host notifies the Controller that host flow control shall be used, and then 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 HCI_Host_Number_Of_Completed_Packets. (See Figure 7.2 on page 737.)

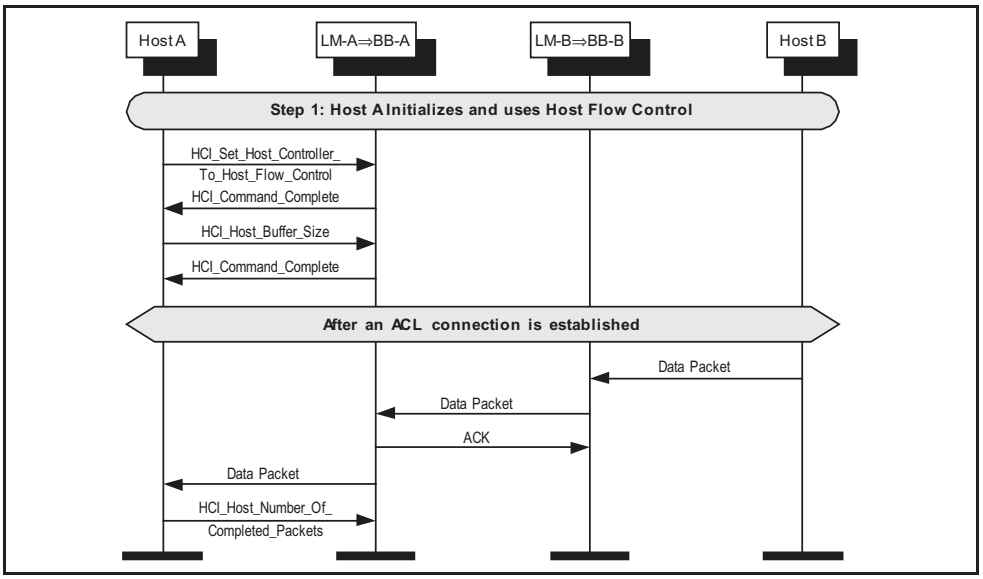



*Figure 7.2: Controller to Host flow control.*

# 8 LOOPBACK MODE

The loopback modes are used for testing of a device only.

## 8.1 LOCAL LOOPBACK MODE

The local loopback mode is used to loopback 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 8.1 on page 738.)

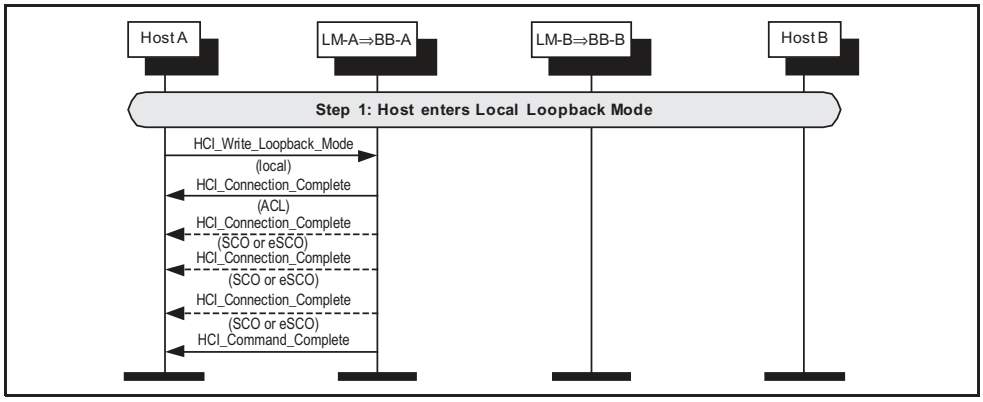



*Figure 8.1: Entering local loopback mode.*

**Step 2a:** The host sending HCI Data Packet will receive the exact same data back in HCI Data Packets from the Controller. (See Figure 8.2 on page 738.)

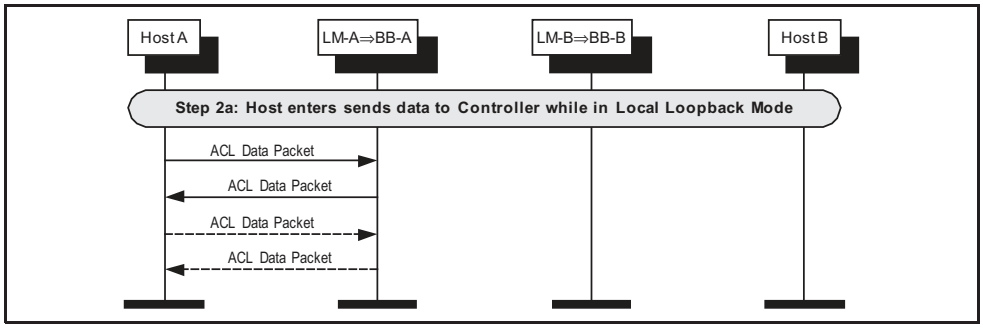



*Figure 8.2: Looping back data in local loopback mode.*

**Step 2b:** The host sending most HCI Command Packets to the Controller will receive an HCI_Loopback_Command event with the contents of the HCI Command Packet in the payload. (See Figure 8.3 on page 739.)

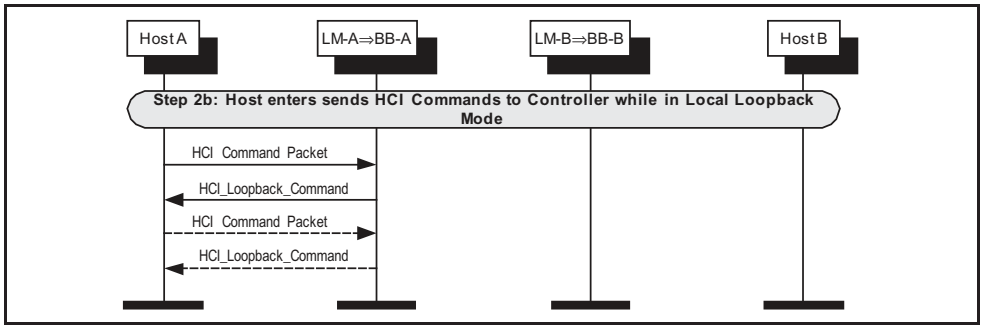



*Figure 8.3: Looping back commands in local loopback mode.*

**Step 3:** The host exits local loopback mode. Multiple disconnection complete events are generated before the command complete event.
(See Figure 8.4 on page 739.)

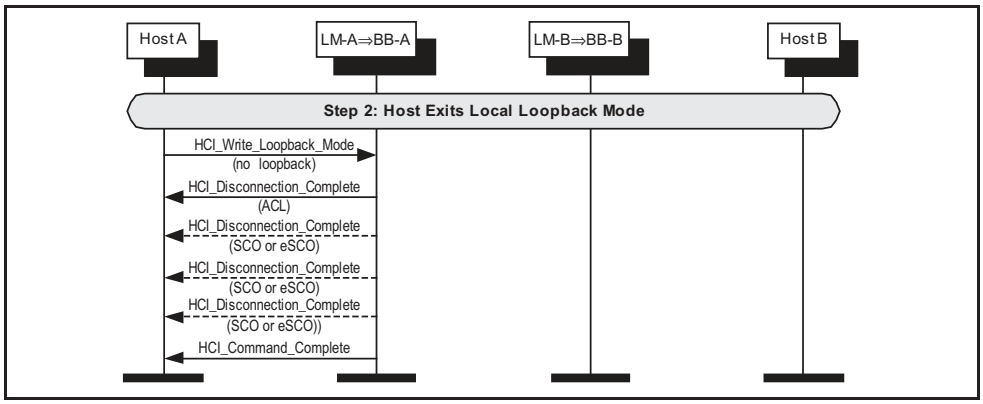



*Figure 8.4: Exiting local loopback mode.*



LifeScan, Inc. Ex. 1019
Page 909 of 1420
IPR2025-00103

## 8.2 REMOTE LOOPBACK MODE

The remote loopback mode is used to loopback 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 8.5 on page 740.)

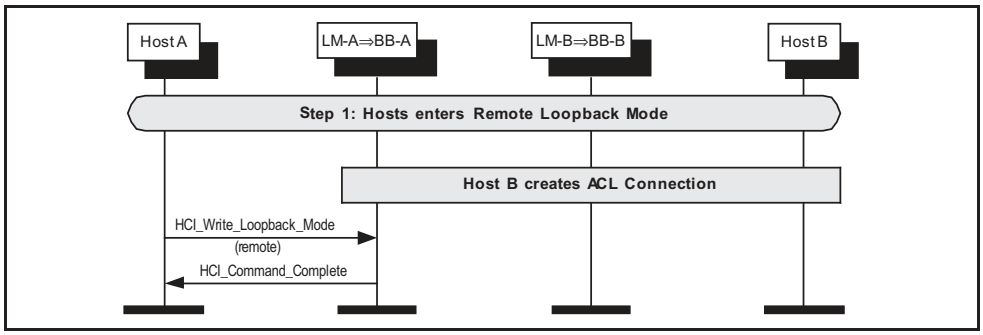



*Figure 8.5: Entering remote loopback mode.*

**Step 2:** Any data received from the remote host will be looped back in the Controller of the local device. (See Figure 8.6 on page 740.)

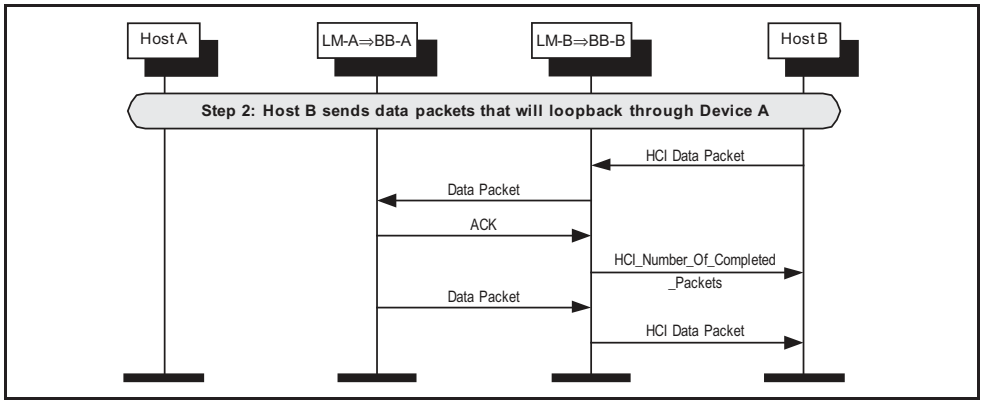



*Figure 8.6: Looping back data in Remote Loopback Mode.*

**Step 3:** The local host exits remote loopback mode. Any connections can then be disconnected by the remote device. (See Figure 8.7 on page 741.)

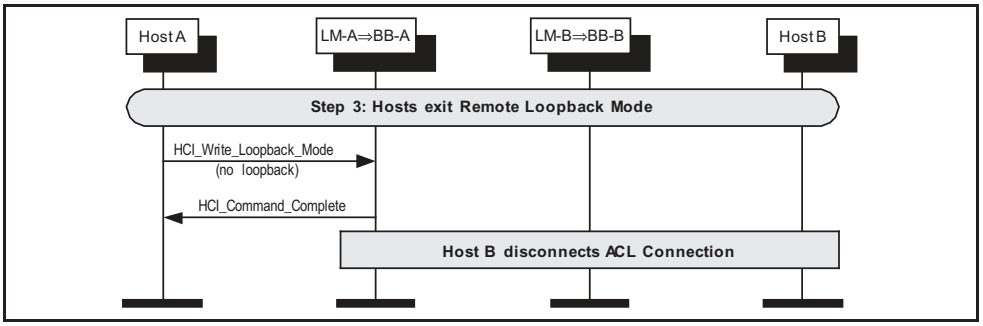



*Figure 8.7: Exiting remote loopback mode.*



LifeScan, Inc. Ex. 1019
Page 911 of 1420
IPR2025-00103

# 9 LIST OF FIGURES

Figure 1.1: Example MSC. ....680
Figure 2.1: Remote name request. ....681
Figure 2.2: Remote name request if no current baseband connection. ....681
Figure 2.3: Remote name request with baseband connection. ....682
Figure 2.4: Host A starts inquiry procedure. ....682
Figure 2.5: LM-A performs inquiry and reports result. ....683
Figure 2.6: Host A cancels inquiry. ....683
Figure 2.7: LM-A terminates current inquiry. ....684
Figure 2.8: Host A starts periodic inquiry. ....684
Figure 2.9: LM-A periodically performs an inquiry and reports result. ....685
Figure 2.10: LM-A terminates current inquiry. ....685
Figure 2.11: Host A decides to exit periodic inquiry. ....685
Figure 3.1: Overview diagram for connection setup. ....686
Figure 3.2: Host A requests connection with device B. ....687
Figure 3.3: LM-A and LM-B exchange features. ....687
Figure 3.4: LM-A requests host connection. ....687
Figure 3.5: Device B rejects connection request. ....688
Figure 3.6: Device B accepts connection request. ....688
Figure 3.7: Device B accepts connection requests as master. ....689
Figure 3.8: LM-A starts adaptive frequency hopping. ....689
Figure 3.9: Authentication initiated. ....690
Figure 3.10: Pairing during connection setup. ....691
Figure 3.11: Authentication during connection setup. ....692
Figure 3.12: Starting encryption during connection setup. ....692
Figure 3.13: LM-A and LM-B finishes connection setup. ....693
Figure 3.14: Host A decides to disconnect. ....693
Figure 4.1: Authentication requested. ....694
Figure 4.2: Simple Pairing Flow Diagram ....695
Figure 4.3: Optional OOB Information Collection ....696
Figure 4.4: Enable Simple Pairing ....696
Figure 4.5: L2CAP Connection Request for a Secure Service ....697
Figure 4.6: Optional OOB Information Transfer ....697
Figure 4.7: Start Simple Pairing ....698
Figure 4.8: IO Capability Exchange ....698
Figure 4.9: Public Key Exchange ....699
Figure 4.10: Numeric Comparison Authentication ....700
Figure 4.11: Numeric Comparison Authentication (Failure on Initiating Side) .. 701
Figure 4.12: Numeric Comparison Failure on Responding Side ....702
Figure 4.13: Passkey Entry Authentication ....703

Figure 4.14: Passkey Entry Failure on Responding Side .....704
Figure 4.15: Passkey Entry Failure on Initiating Side .....704
Figure 4.16: OOB Authentication .....705
Figure 4.17: OOB Authentication failure on initiating side .....705
Figure 4.18: DHKey Checks .....706
Figure 4.19: Calculate Link Key .....707
Figure 4.20: Start Encryption .....707
Figure 4.21: L2CAP Connection Response .....708
Figure 4.22: Link Supervision Timeout Event .....708
Figure 4.23: Encryption requested. .....709
Figure 4.24: Encryption off requested. .....709
Figure 4.25: Change connection link key. .....710
Figure 4.26: NOTE: Taken from Encryption Pause Resume CR. Change Connection Link Key with Encryption Pause Resume 711
Figure 4.27: Change to master link key. .....712
Figure 4.28: Change to semi permanent link key. .....713
Figure 4.29: Read remote supported features. .....714
Figure 4.30: Read remote extended features. .....714
Figure 4.31: Read clock offset. .....715
Figure 4.32: Role Switch on an Encrypted link using encryption pause and resume. 716
Figure 4.33: Refreshing encryption keys using encryption pause and resume . 717
Figure 4.34: Read remote version information. .....717
Figure 4.35: QoS flow specification. .....718
Figure 4.36: Master requests role switch. .....718
Figure 4.37: Slave requests role switch. .....719
Figure 4.38: Role switch is performed. .....719
Figure 5.1: Master requests synchronous EV3, EV4 OR EV5 connection. ..... 720
Figure 5.2: Slave requests synchronous EV3, EV4 OR EV5 connection. .721
Figure 5.3: Master requests synchronous connection using SCO. .....721
Figure 5.4: Master requests synchronous connection with legacy slave. .722
Figure 5.5: Any device that supports only SCO connections requests a synchronous connection with a device. 722
Figure 5.6: Master renegotiates eSCO connection. .....723
Figure 5.7: Slave renegotiates eSCO connection. .....723
Figure 5.8: Synchronous disconnection of eSCO connection. .....724
Figure 5.9: Synchronous disconnection of SCO connection. .....724
Figure 6.1: Sniff mode request. .....725
Figure 6.2: Exit sniff mode request. .....725
Figure 6.3: Hold request. .....726
Figure 6.4: Master forces hold mode. .....726

Figure 6.5: Slave forces hold mode. ....727
Figure 6.6: Hold mode completes. ....727
Figure 6.7: Park state request from master. ....728
Figure 6.8: Park state request from slave. ....728
Figure 6.9: Master unparks slave for supervision. ....729
Figure 6.10: Master exits park state with slave. ....729
Figure 6.11: Slave exits park state with master. ....730
Figure 7.1: Host to Controller flow control. ....731
Figure 7.2: Controller to Host flow control. ....732
Figure 8.1: Entering local loopback mode. ....733
Figure 8.2: Looping back data in local loopback mode. ....733
Figure 8.3: Looping back commands in local loopback mode. ....734
Figure 8.4: Exiting local loopback mode. ....734
Figure 8.5: Entering remote loopback mode. ....735
Figure 8.6: Looping back data in Remote Loopback Mode. ....735
Figure 8.7: Exiting remote loopback mode. ....736

# SAMPLE DATA

***This appendix contains sample data for various parts of the Bluetooth baseband specification. All sample data are provided for reference purpose only; they are intended as a complement to the definitions provided elsewhere in the specification. They can be used to check the behavior of an implementation and avoid misunderstandings. Fulfilling these sample data is a necessary but not sufficient condition for an implementation to be fully Bluetooth compliant.***



LifeScan, Inc. Ex. 1019
Page 915 of 1420
IPR2025-00103

*Sample Data*



LifeScan, Inc. Ex. 1019
Page 916 of 1420
IPR2025-00103

# CONTENTS

**Contents** ..... **747**

**1 Encryption Sample Data** ..... **749**
- 1.1 Generating Kc' from Kc, ..... 749
- 1.2 First Set of Sample Data ..... 752
- 1.3 Second Set of Sample Data ..... 760
- 1.4 Third Set of Samples ..... 768
- 1.5 Fourth Set of Samples ..... 776

**2 Frequency Hopping Sample Data** ..... **784**
- 2.1 First set ..... 785
- 2.2 Second set ..... 791
- 2.3 Third set ..... 797

**3 Access Code Sample Data** ..... **803**

**4 HEC and Packet Header Sample Data** ..... **806**

**5 CRC Sample Data** ..... **807**

**6 Complete Sample Packets** ..... **808**
- 6.1 Example of DH1 Packet ..... 808
- 6.2 Example of DM1 Packet ..... 809

**7 Simple Pairing Sample Data** ..... **810**
- 7.1 P-192 Sample Data ..... 810
  - 7.1.1 Data Set 1 ..... 810
  - 7.1.2 Data Set 2 ..... 810
  - 7.1.3 Data Set 3 ..... 810
  - 7.1.4 Data Set 4 ..... 810
  - 7.1.5 Data Set 5 ..... 810
  - 7.1.6 Data Set 6 ..... 811
  - 7.1.7 Data Set 7 ..... 811
  - 7.1.8 Data Set 8 ..... 811
  - 7.1.9 Data Set 9 ..... 811
  - 7.1.10 Data Set 10 ..... 811
- 7.2 Hash Functions Sample Data ..... 811
  - 7.2.1 f1() ..... 812
  - 7.2.2 g() ..... 813
  - 7.2.3 f2() ..... 813
  - 7.2.4 f3() ..... 813

**8 Whitening Sequence Sample Data** ..... **821**

**9 FEC Sample Data** ..... **824**

**10 Encryption Key Sample Data ..... 825**
10.1 Four Tests of E1 ..... 825
10.2 Four Tests of E21 ..... 830
10.3 Three Tests of E22 ..... 832
10.4 Tests of E22 With Pin Augmenting ..... 834
10.5 Four Tests of E3 ..... 844



LifeScan, Inc. Ex. 1019
Page 918 of 1420
IPR2025-00103

# 1 ENCRYPTION SAMPLE DATA

This section contains four sets of sample data for the encryption process.

With respect to the functional description of the encryption engine in the Bluetooth baseband specification, the contents of registers and resulting concurrent values are listed as well. This by no means excludes different implementations (as far as they produce the same encryption stream) but is intended to describe the functional behavior.

In case of misunderstandings or inconsistencies, these sample data form the normative reference.

## 1.1 GENERATING KC' FROM KC,

where Kc'(x) = g2(x)(Kc(x) mod g1(x)).

```
Note: All polynomials are in hexadecimal notation.
      'L' is the effective key length in bytes.
      The notation 'p: [m]' implies that deg(p(x)) = m.

                              MSB                                     LSB
L = 1
g1:                 [8]       00000000 00000000 00000000 0000011d
g2:                 [119]     00e275a0 abd218d4 cf928b9b bf6cb08f
Kc:                           a2b230a4 93f281bb 61a85b82 a9d4a30e
Kc mod g1:          [7]       00000000 00000000 00000000 0000009f
g2(Kc mod g1):      [126]     7aa16f39 59836ba3 22049a7b 87f1d8a5
-------------------------------------------------------------
L = 2
g1:                 [16]      00000000 00000000 00000000 0001003f
g2:                 [112]     0001e3f6 3d7659b3 7f18c258 cff6efef
Kc:                           64e7df78 bb7ccaa4 61433123 5b3222ad
Kc mod g1:          [12]      00000000 00000000 00000000 00001ff0
g2(Kc mod g1):      [124]     142057bb 0bceac4c 58bd142e 1e710a50
-------------------------------------------------------------
L = 3
g1:                 [24]      00000000 00000000 00000000 010000db
g2:                 [104]     000001be f66c6c3a b1030a5a 1919808b
Kc:                           575e5156 ba685dc6 112124ac edb2c179
Kc mod g1:          [23]      00000000 00000000 00000000 008ddbc8
g2(Kc mod g1):      [127]     d56d0adb 8216cb39 7fe3c591 1ff95618
-------------------------------------------------------------
L = 4
g1:                 [32]      00000000 00000000 00000001 000000af
g2:                 [96]      00000001 6ab89969 de17467f d3736ad9
Kc:                           8917b4fc 403b6db2 1596b86d 1cb8adab
Kc mod g1:          [31]      00000000 00000000 00000000 aa1e78aa
g2(Kc mod g1):      [127]     91910128 b0e2f5ed a132a03e af3d8cda
-------------------------------------------------------------
```

*Sample Data*

```
L = 5
g1:                [40]        00000000 00000000 00000100 00000039
g2:                [88]        00000000 01630632 91da50ec 55715247
Kc:                            785c915b dd25b9c6 0102ab00 b6cd2a68
Kc mod g1:         [38]        00000000 00000000 0000007f 13d44436
g2(Kc mod g1):     [126]       6fb5651c cb80c8d7 ea1ee56d f1ec5d02
-------------------------------------------------------------
L = 6
g1:                [48]        00000000 00000000 00010000 00000291
g2:                [77]        00000000 00002c93 52aa6cc0 54468311
Kc:                            5e77d19f 55ccd7d5 798f9a32 3b83e5d8
Kc mod g1:         [47]        00000000 00000000 000082eb 4af213ed
g2(Kc mod g1):     [124]       16096bcb afcf8def 1d226a1b 4d3f9a3d
-------------------------------------------------------------
```

*Sample Data*

```
L = 7
g1:                [56]        00000000 00000000 01000000 00000095
g2:                [71]        00000000 000000b3 f7fffce2 79f3a073
Kc:                            05454e03 8ddcfbe3 ed024b2d 92b7f54c
Kc mod g1:         [55]        00000000 00000000 0095b8a4 8eb816da
g2(Kc mod g1):     [126]       50f9c0d4 e3178da9 4a09fe0d 34f67b0e
------------------------------------------------------------
L = 8
g1:                [64]        00000000 00000001 00000000 0000001b
g2:                [63]        00000000 00000000 a1ab815b c7ec8025
Kc:                            7ce149fc f4b38ad7 2a5d8a41 eb15ba31
Kc mod g1:         [63]        00000000 00000000 8660806c 1865deec
g2(Kc mod g1):     [126]       532c36d4 5d0954e0 922989b6 826f78dc
------------------------------------------------------------
L = 9
g1:                [72]        00000000 00000100 00000000 00000609
g2:                [49]        00000000 00000000 0002c980 11d8b04d
Kc:                            5eeff7ca 84fc2782 9c051726 3df6f36e
Kc mod g1:         [71]        00000000 00000083 58ccb7d0 b95d3c71
g2(Kc mod g1):     [120]       016313f6 0d3771cf 7f8e4bb9 4aa6827d
------------------------------------------------------------
L = 10
g1:                [80]        00000000 00010000 00000000 00000215
g2:                [42]        00000000 00000000 0000058e 24f9a4bb
Kc:                            7b13846e 88beb4de 34e7160a fd44dc65
Kc mod g1:         [79]        00000000 0000b4de 34171767 f36981c3
g2(Kc mod g1):     [121]       023bc1ec 34a0029e f798dcfb 618ba58d
------------------------------------------------------------
L = 11
g1:                [88]        00000000 01000000 00000000 0000013b
g2:                [35]        00000000 00000000 0000000c a76024d7
Kc:                            bda6de6c 6e7d757e 8dfe2d49 9a181193
Kc mod g1:         [86]        00000000 007d757e 8dfe88aa 2fcee371
g2(Kc mod g1):     [121]       022e08a9 3aa51d8d 2f93fa78 85cc1f87
------------------------------------------------------------
L = 12
g1:                [96]        00000001 00000000 00000000 000000dd
g2:                [28]        00000000 00000000 00000000 1c9c26b9
Kc:                            e6483b1c 2cdb1040 9a658f97 c4efd90d
Kc mod g1:         [93]        00000000 2cdb1040 9a658fd7 5b562e41
g2(Kc mod g1):     [121]       030d752b 216fe29b b880275c d7e6f6f9
------------------------------------------------------------
L = 13
g1:                [104]       00000100 00000000 00000000 0000049d
g2:                [21]        00000000 00000000 00000000 0026d9e3
Kc:                            d79d281d a2266847 6b223c46 dc0ab9ee
Kc mod g1:         [100]       0000001d a2266847 6b223c45 e1fc5fa6
g2(Kc mod g1):     [121]       03f11138 9cebf919 00b93808 4ac158aa
------------------------------------------------------------
```

*Sample Data*

```
L = 14
g1:                [112]       00010000 00000000 00000000 0000014f
g2:                [14]        00000000 00000000 00000000 00004377
Kc:                            cad9a65b 9fca1c1d a2320fcf 7c4ae48e
Kc mod g1:         [111]       0000a65b 9fca1c1d a2320fcf 7cb6a909
g2(Kc mod g1):     [125]       284840fd f1305f3c 529f5703 76adf7cf
------------------------------------------------------------
L = 15
g1:                [120]       01000000 00000000 00000000 000000e7
g2:                [7]         00000000 00000000 00000000 00000089
Kc:                            21f0cc31 049b7163 d375e9e1 06029809
Kc mod g1:         [119]       00f0cc31 049b7163 d375e9e1 0602840e
g2(Kc mod g1):     [126]       7f10b53b 6df84b94 f22e566a 3754a37e
------------------------------------------------------------
L = 16
g1:                [128]       00000001 00000000 00000000 00000000 00000000
g2:                [0]         00000000 00000000 00000000 00000001
Kc:                            35ec8fc3 d50ccd32 5f2fd907 bde206de
Kc mod g1:         [125]       35ec8fc3 d50ccd32 5f2fd907 bde206de
g2(Kc mod g1):     [125]       35ec8fc3 d50ccd32 5f2fd907 bde206de
------------------------------------------------------------
```

## 1.2 FIRST SET OF SAMPLE DATA

```
Initial values for the key, pan address and clock

K'c1[0]   = 00   K'c1[1]   = 00    K'c1[2]   = 00    K'c1[3]  = 00
K'c1[4]   = 00   K'c1[5]   = 00    K'c1[6]   = 00    K'c1[7]  = 00
K'c1[8]   = 00   K'c1[9]   = 00    K'c1[10]  = 00    K'c1[11] = 00
K'c1[12]  = 00   K'c1[13]  = 00    K'c1[14]  = 00    K'c1[15] = 00

Addr1[0] = 00   Addr1[1] = 00    Addr1[2] = 00
Addr1[3] = 00   Addr1[4] = 00    Addr1[5] = 00

CL1[0]  = 00   CL1[1]  = 00    CL1[2]  = 00    CL1[3]  = 00


=============================================================================================
         Fill LFSRs with initial data
=============================================================================================

  t  clk#    LFSR1    LFSR2     LFSR3      LFSR4           X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

  0     0    0000000* 00000000* 000000000* 0000000000*     0  0  0  0    0  00     00   00
  1     1    0000000* 00000001* 000000000* 0000000001*     0  0  0  0    0  00     00   00
  2     2    0000000* 00000002* 000000000* 0000000003*     0  0  0  0    0  00     00   00
  3     3    0000000* 00000004* 000000000* 0000000007*     0  0  0  0    0  00     00   00
  4     4    0000000* 00000008* 000000000* 000000000E*     0  0  0  0    0  00     00   00
  5     5    0000000* 00000010* 000000000* 000000001C*     0  0  0  0    0  00     00   00
  6     6    0000000* 00000020* 000000000* 0000000038*     0  0  0  0    0  00     00   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 7 | 0000000* | 00000040* | 000000000* | 0000000070* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 8 | 8 | 0000000* | 00000080* | 000000000* | 00000000E0* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 9 | 9 | 0000000* | 00000100* | 000000000* | 00000001C0* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 10 | 10 | 0000000* | 00000200* | 000000000* | 0000000380* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 11 | 11 | 0000000* | 00000400* | 000000000* | 0000000700* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 12 | 12 | 0000000* | 00000800* | 000000000* | 0000000E00* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 13 | 13 | 0000000* | 00001000* | 000000000* | 0000001C00* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 14 | 14 | 0000000* | 00002000* | 000000000* | 0000003800* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 15 | 15 | 0000000* | 00004000* | 000000000* | 0000007000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 16 | 16 | 0000000* | 00008000* | 000000000* | 000000E000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 17 | 17 | 0000000* | 00010000* | 000000000* | 000001C000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 18 | 18 | 0000000* | 00020000* | 000000000* | 0000038000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 19 | 19 | 0000000* | 00040000* | 000000000* | 0000070000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 20 | 20 | 0000000* | 00080000* | 000000000* | 00000E0000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 21 | 21 | 0000000* | 00100000* | 000000000* | 00001C0000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 22 | 22 | 0000000* | 00200000* | 000000000* | 0000380000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 23 | 23 | 0000000* | 00400000* | 000000000* | 0000700000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 24 | 24 | 0000000* | 00800000* | 000000000* | 0000E00000* | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 25 | 25 | 0000000* | 01000000* | 000000000* | 0001C00000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 26 | 26 | 0000000 | 02000000* | 000000000* | 0003800000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 27 | 27 | 0000000 | 04000000* | 000000000* | 0007000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 28 | 28 | 0000000 | 08000000* | 000000000* | 000E000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 29 | 29 | 0000000 | 10000000* | 000000000* | 001C000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 30 | 30 | 0000000 | 20000000* | 000000000* | 0038000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 31 | 31 | 0000000 | 40000000* | 000000000* | 0070000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 32 | 32 | 0000000 | 00000001 | 000000000* | 00E0000000* | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 33 | 33 | 0000000 | 00000002 | 000000000* | 01C0000000* | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 34 | 34 | 0000000 | 00000004 | 000000000 | 0380000000* | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 35 | 35 | 0000000 | 00000008 | 000000000 | 0700000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 36 | 36 | 0000000 | 00000010 | 000000000 | 0E00000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 37 | 37 | 0000000 | 00000020 | 000000000 | 1C00000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 38 | 38 | 0000000 | 00000040 | 000000000 | 3800000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 39 | 39 | 0000000 | 00000080 | 000000000 | 7000000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |

Start clocking Summation Combiner

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 0000000 | 00000100 | 000000000 | 6000000001 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 41 | 2 | 0000000 | 00000200 | 000000000 | 4000000003 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 42 | 3 | 0000000 | 00000400 | 000000000 | 0000000007 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 43 | 4 | 0000000 | 00000800 | 000000000 | 000000000E | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 44 | 5 | 0000000 | 00001001 | 000000000 | 000000001D | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 45 | 6 | 0000000 | 00002002 | 000000000 | 000000003B | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 46 | 7 | 0000000 | 00004004 | 000000000 | 0000000077 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 47 | 8 | 0000000 | 00008008 | 000000000 | 00000000EE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 48 | 9 | 0000000 | 00010011 | 000000000 | 00000001DD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 49 | 10 | 0000000 | 00020022 | 000000000 | 00000003BB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 50 | 11 | 0000000 | 00040044 | 000000000 | 0000000777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 51 | 12 | 0000000 | 00080088 | 000000000 | 0000000EEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 52 | 13 | 0000000 | 00100110 | 000000000 | 0000001DDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 53 | 14 | 0000000 | 00200220 | 000000000 | 0000003BBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 54 | 15 | 0000000 | 00400440 | 000000000 | 0000007777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 55 | 16 | 0000000 | 00800880 | 000000000 | 000000EEEE | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 56 | 17 | 0000000 | 01001100 | 000000000 | 000001DDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 57 | 18 | 0000000 | 02002200 | 000000000 | 000003BBBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 58 | 19 | 0000000 | 04004400 | 000000000 | 0000077777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 59 | 20 | 0000000 | 08008800 | 000000000 | 00000EEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 60 | 21 | 0000000 | 10011000 | 000000000 | 00001DDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 61 | 22 | 0000000 | 20022000 | 000000000 | 00003BBBBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 62 | 23 | 0000000 | 40044000 | 000000000 | 0000777777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 63 | 24 | 0000000 | 00088001 | 000000000 | 0000EEEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 64 | 25 | 0000000 | 00110003 | 000000000 | 0001DDDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 65 | 26 | 0000000 | 00220006 | 000000000 | 0003BBBBBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 66 | 27 | 0000000 | 0044000C | 000000000 | 0007777777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 67 | 28 | 0000000 | 00880018 | 000000000 | 000EEEEEEE | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 68 | 29 | 0000000 | 01100031 | 000000000 | 001DDDDDDC | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 69 | 30 | 0000000 | 02200062 | 000000000 | 003BBBBBB8 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 70 | 31 | 0000000 | 044000C4 | 000000000 | 0077777770 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 71 | 32 | 0000000 | 08800188 | 000000000 | 00EEEEEEE0 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 72 | 33 | 0000000 | 11000311 | 000000000 | 01DDDDDDC1 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 73 | 34 | 0000000 | 22000622 | 000000000 | 03BBBBBB83 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 74 | 35 | 0000000 | 44000C44 | 000000000 | 0777777707 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 75 | 36 | 0000000 | 08001888 | 000000000 | 0EEEEEEE0E | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 76 | 37 | 0000000 | 10003111 | 000000000 | 1DDDDDDC1D | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 77 | 38 | 0000000 | 20006222 | 000000000 | 3BBBBBB83B | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 78 | 39 | 0000000 | 4000C444 | 000000000 | 7777777077 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 79 | 40 | 0000000 | 00018888 | 000000000 | 6EEEEEE0EF | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 80 | 41 | 0000000 | 00031110 | 000000000 | 5DDDDDC1DE | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 81 | 42 | 0000000 | 00062220 | 000000000 | 3BBBBB83BC | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 82 | 43 | 0000000 | 000C4440 | 000000000 | 7777770779 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 83 | 44 | 0000000 | 00188880 | 000000000 | 6EEEEE0EF2 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 84 | 45 | 0000000 | 00311100 | 000000000 | 5DDDDC1DE5 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 85 | 46 | 0000000 | 00622200 | 000000000 | 3BBBB83BCB | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 86 | 47 | 0000000 | 00C44400 | 000000000 | 7777707797 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 87 | 48 | 0000000 | 01888801 | 000000000 | 6EEEE0EF2F | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 88 | 49 | 0000000 | 03111003 | 000000000 | 5DDDC1DE5E | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 89 | 50 | 0000000 | 06222006 | 000000000 | 3BBB83BCBC | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 90 | 51 | 0000000 | 0C44400C | 000000000 | 7777077979 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 91 | 52 | 0000000 | 18888018 | 000000000 | 6EEE0EF2F2 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 92 | 53 | 0000000 | 31110030 | 000000000 | 5DDC1DE5E5 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 93 | 54 | 0000000 | 62220060 | 000000000 | 3BB83BCBCB | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 94 | 55 | 0000000 | 444400C1 | 000000000 | 7770779797 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 11 |
| 95 | 56 | 0000000 | 08880183 | 000000000 | 6EE0EF2F2F | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 96 | 57 | 0000000 | 11100307 | 000000000 | 5DC1DE5E5F | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 97 | 58 | 0000000 | 2220060E | 000000000 | 3B83BCBCBF | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 98 | 59 | 0000000 | 44400C1C | 000000000 | 770779797E | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 99 | 60 | 0000000 | 08801838 | 000000000 | 6E0EF2F2FC | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 100 | 61 | 0000000 | 11003070 | 000000000 | 5C1DE5E5F8 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 101 | 62 | 0000000 | 220060E0 | 000000000 | 383BCBCBF0 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 102 | 63 | 0000000 | 4400C1C0 | 000000000 | 70779797E0 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 103 | 64 | 0000000 | 08018380 | 000000000 | 60EF2F2FC1 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 104 | 65 | 0000000 | 10030701 | 000000000 | 41DE5E5F82 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 105 | 66 | 0000000 | 20060E02 | 000000000 | 03BCBCBF04 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 106 | 67 | 0000000 | 400C1C05 | 000000000 | 0779797E09 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 107 | 68 | 0000000 | 0018380A | 000000000 | 0EF2F2FC12 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 108 | 69 | 0000000 | 00307015 | 000000000 | 1DE5E5F825 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 109 | 70 | 0000000 | 0060E02A | 000000000 | 3BCBCBF04B | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 110 | 71 | 0000000 | 00C1C055 | 000000000 | 779797E097 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 111 | 72 | 0000000 | 018380AA | 000000000 | 6F2F2FC12F | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 112 | 73 | 0000000 | 03070154 | 000000000 | 5E5E5F825E | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 113 | 74 | 0000000 | 060E02A8 | 000000000 | 3CBCBF04BC | 0 | 0 | 0 | 1 | 0 | 11 | 11 | 11 |
| 114 | 75 | 0000000 | 0C1C0550 | 000000000 | 79797E0979 | 0 | 0 | 0 | 0 | 1 | 00 | 11 | 11 |
| 115 | 76 | 0000000 | 18380AA0 | 000000000 | 72F2FC12F2 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 116 | 77 | 0000000 | 30701541 | 000000000 | 65E5F825E5 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 117 | 78 | 0000000 | 60E02A82 | 000000000 | 4BCBF04BCB | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |

*Sample Data*

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 118 | 79 | 0000000 | 41C05505 | 000000000 | 1797E09796 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 119 | 80 | 0000000 | 0380AA0A | 000000000 | 2F2FC12F2C | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 120 | 81 | 0000000 | 07015415 | 000000000 | 5E5F825E59 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 121 | 82 | 0000000 | 0E02A82A | 000000000 | 3CBF04BCB2 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 122 | 83 | 0000000 | 1C055054 | 000000000 | 797E097964 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 123 | 84 | 0000000 | 380AA0A8 | 000000000 | 72FC12F2C9 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 124 | 85 | 0000000 | 70154151 | 000000000 | 65F825E593 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 125 | 86 | 0000000 | 602A82A3 | 000000000 | 4BF04BCB26 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 126 | 87 | 0000000 | 40550546 | 000000000 | 17E097964C | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 127 | 88 | 0000000 | 00AA0A8D | 000000000 | 2FC12F2C99 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 128 | 89 | 0000000 | 0154151A | 000000000 | 5F825E5932 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 129 | 90 | 0000000 | 02A82A34 | 000000000 | 3F04BCB264 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 130 | 91 | 0000000 | 05505468 | 000000000 | 7E097964C9 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 131 | 92 | 0000000 | 0AA0A8D0 | 000000000 | 7C12F2C992 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 132 | 93 | 0000000 | 154151A1 | 000000000 | 7825E59324 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 133 | 94 | 0000000 | 2A82A342 | 000000000 | 704BCB2648 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 134 | 95 | 0000000 | 55054684 | 000000000 | 6097964C91 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 135 | 96 | 0000000 | 2A0A8D09 | 000000000 | 412F2C9923 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 136 | 97 | 0000000 | 54151A12 | 000000000 | 025E593246 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 137 | 98 | 0000000 | 282A3424 | 000000000 | 04BCB2648D | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 138 | 99 | 0000000 | 50546848 | 000000000 | 097964C91A | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 139 | 100 | 0000000 | 20A8D090 | 000000000 | 12F2C99235 | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 140 | 101 | 0000000 | 4151A120 | 000000000 | 25E593246A | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 141 | 102 | 0000000 | 02A34240 | 000000000 | 4BCB2648D5 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 142 | 103 | 0000000 | 05468481 | 000000000 | 17964C91AB | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 143 | 104 | 0000000 | 0A8D0903 | 000000000 | 2F2C992357 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 144 | 105 | 0000000 | 151A1206 | 000000000 | 5E593246AE | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 145 | 106 | 0000000 | 2A34240C | 000000000 | 3CB2648D5C | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 146 | 107 | 0000000 | 54684818 | 000000000 | 7964C91AB8 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 147 | 108 | 0000000 | 28D09030 | 000000000 | 72C9923571 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 148 | 109 | 0000000 | 51A12060 | 000000000 | 6593246AE2 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 149 | 110 | 0000000 | 234240C0 | 000000000 | 4B2648D5C5 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 150 | 111 | 0000000 | 46848180 | 000000000 | 164C91AB8A | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 151 | 112 | 0000000 | 0D090301 | 000000000 | 2C99235714 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 152 | 113 | 0000000 | 1A120602 | 000000000 | 593246AE28 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 153 | 114 | 0000000 | 34240C04 | 000000000 | 32648D5C51 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 154 | 115 | 0000000 | 68481809 | 000000000 | 64C91AB8A2 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 155 | 116 | 0000000 | 50903012 | 000000000 | 4992357144 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 156 | 117 | 0000000 | 21206024 | 000000000 | 13246AE288 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 157 | 118 | 0000000 | 4240C048 | 000000000 | 2648D5C511 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 158 | 119 | 0000000 | 04818090 | 000000000 | 4C91AB8A23 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 159 | 120 | 0000000 | 09030120 | 000000000 | 1923571446 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 160 | 121 | 0000000 | 12060240 | 000000000 | 3246AE288D | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 161 | 122 | 0000000 | 240C0480 | 000000000 | 648D5C511B | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 162 | 123 | 0000000 | 48180900 | 000000000 | 491AB8A237 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 163 | 124 | 0000000 | 10301200 | 000000000 | 123571446F | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 164 | 125 | 0000000 | 20602400 | 000000000 | 246AE288DF | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 165 | 126 | 0000000 | 40C04800 | 000000000 | 48D5C511BE | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 166 | 127 | 0000000 | 01809001 | 000000000 | 11AB8A237D | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 167 | 128 | 0000000 | 03012002 | 000000000 | 23571446FA | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 168 | 129 | 0000000 | 06024004 | 000000000 | 46AE288DF5 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 169 | 130 | 0000000 | 0C048008 | 000000000 | 0D5C511BEA | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 170 | 131 | 0000000 | 18090011 | 000000000 | 1AB8A237D5 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 171 | 132 | 0000000 | 30120022 | 000000000 | 3571446FAA | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 172 | 133 | 0000000 | 60240044 | 000000000 | 6AE288DF55 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 173 | 134 | 0000000 | 40480089 | 000000000 | 55C511BEAA | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 174 | 135 | 0000000 | 00900113 | 000000000 | 2B8A237D54 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 175 | 136 | 0000000 | 01200227 | 000000000 | 571446FAA8 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 176 | 137 | 0000000 | 0240044E | 000000000 | 2E288DF550 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 177 | 138 | 0000000 | 0480089C | 000000000 | 5C511BEAA0 | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 178 | 139 | 0000000 | 09001138 | 000000000 | 38A237D540 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 179 | 140 | 0000000 | 12002270 | 000000000 | 71446FAA81 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 180 | 141 | 0000000 | 240044E0 | 000000000 | 6288DF5503 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 181 | 142 | 0000000 | 480089C0 | 000000000 | 4511BEAA06 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 182 | 143 | 0000000 | 10011381 | 000000000 | 0A237D540D | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 183 | 144 | 0000000 | 20022702 | 000000000 | 1446FAA81A | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 184 | 145 | 0000000 | 40044E04 | 000000000 | 288DF55035 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 185 | 146 | 0000000 | 00089C08 | 000000000 | 511BEAA06A | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 186 | 147 | 0000000 | 00113810 | 000000000 | 2237D540D5 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 187 | 148 | 0000000 | 00227021 | 000000000 | 446FAA81AA | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 188 | 149 | 0000000 | 0044E042 | 000000000 | 08DF550355 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 189 | 150 | 0000000 | 0089C085 | 000000000 | 11BEAA06AA | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 190 | 151 | 0000000 | 0113810A | 000000000 | 237D540D54 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 191 | 152 | 0000000 | 02270215 | 000000000 | 46FAA81AA9 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 192 | 153 | 0000000 | 044E042A | 000000000 | 0DF5503553 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 193 | 154 | 0000000 | 089C0854 | 000000000 | 1BEAA06AA7 | 0 | 1 | 0 | 1 | 0 | 01 | 10 | 10 |
| 194 | 155 | 0000000 | 113810A8 | 000000000 | 37D540D54E | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 195 | 156 | 0000000 | 22702150 | 000000000 | 6FAA81AA9D | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 196 | 157 | 0000000 | 44E042A0 | 000000000 | 5F5503553A | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 197 | 158 | 0000000 | 09C08540 | 000000000 | 3EAA06AA75 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 198 | 159 | 0000000 | 13810A80 | 000000000 | 7D540D54EA | 0 | 1 | 0 | 0 | 1 | 10 | 10 | 10 |
| 199 | 160 | 0000000 | 27021500 | 000000000 | 7AA81AA9D5 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 200 | 161 | 0000000 | 4E042A00 | 000000000 | 75503553AB | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 201 | 162 | 0000000 | 1C085400 | 000000000 | 6AA06AA756 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 202 | 163 | 0000000 | 3810A800 | 000000000 | 5540D54EAC | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 203 | 164 | 0000000 | 70215000 | 000000000 | 2A81AA9D58 | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 204 | 165 | 0000000 | 6042A001 | 000000000 | 5503553AB0 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 205 | 166 | 0000000 | 40854002 | 000000000 | 2A06AA7561 | 0 | 1 | 0 | 0 | 1 | 10 | 10 | 10 |
| 206 | 167 | 0000000 | 010A8004 | 000000000 | 540D54EAC3 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 207 | 168 | 0000000 | 02150009 | 000000000 | 281AA9D586 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 208 | 169 | 0000000 | 042A0012 | 000000000 | 503553AB0C | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 209 | 170 | 0000000 | 08540024 | 000000000 | 206AA75618 | 0 | 0 | 0 | 0 | 0 | 10 | 10 | 10 |
| 210 | 171 | 0000000 | 10A80048 | 000000000 | 40D54EAC30 | 0 | 1 | 0 | 1 | 0 | 01 | 10 | 10 |
| 211 | 172 | 0000000 | 21500091 | 000000000 | 01AA9D5861 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 212 | 173 | 0000000 | 42A00122 | 000000000 | 03553AB0C3 | 0 | 1 | 0 | 0 | 0 | 11 | 01 | 01 |
| 213 | 174 | 0000000 | 05400244 | 000000000 | 06AA756186 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 214 | 175 | 0000000 | 0A800488 | 000000000 | 0D54EAC30D | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 215 | 176 | 0000000 | 15000911 | 000000000 | 1AA9D5861A | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 216 | 177 | 0000000 | 2A001223 | 000000000 | 3553AB0C35 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 217 | 178 | 0000000 | 54002446 | 000000000 | 6AA756186A | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 218 | 179 | 0000000 | 2800488D | 000000000 | 554EAC30D5 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 219 | 180 | 0000000 | 5000911B | 000000000 | 2A9D5861AA | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 220 | 181 | 0000000 | 20012236 | 000000000 | 553AB0C355 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 221 | 182 | 0000000 | 4002446C | 000000000 | 2A756186AA | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 222 | 183 | 0000000 | 000488D9 | 000000000 | 54EAC30D54 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 223 | 184 | 0000000 | 000911B2 | 000000000 | 29D5861AA8 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 224 | 185 | 0000000 | 00122364 | 000000000 | 53AB0C3550 | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 225 | 186 | 0000000 | 002446C8 | 000000000 | 2756186AA0 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 226 | 187 | 0000000 | 00488D90 | 000000000 | 4EAC30D540 | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 227 | 188 | 0000000 | 00911B20 | 000000000 | 1D5861AA81 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 228 | 189 | 0000000 | 01223640 | 000000000 | 3AB0C35502 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 229 | 190 | 0000000 | 02446C80 | 000000000 | 756186AA05 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 230 | 191 | 0000000 | 0488D901 | 000000000 | 6AC30D540B | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 231 | 192 | 0000000 | 0911B203 | 000000000 | 55861AA817 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 232 | 193 | 0000000 | 12236407 | 000000000 | 2B0C35502F | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 233 | 194 | 0000000 | 2446C80E | 000000000 | 56186AA05F | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 234 | 195 | 0000000 | 488D901C | 000000000 | 2C30D540BF | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 235 | 196 | 0000000 | 111B2039 | 000000000 | 5861AA817E | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 236 | 197 | 0000000 | 22364072 | 000000000 | 30C35502FD | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 237 | 198 | 0000000 | 446C80E4 | 000000000 | 6186AA05FB | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 238 | 199 | 0000000 | 08D901C8 | 000000000 | 430D540BF6 | 0 | 1 | 0 | 0 | 0 | 11 | 01 | 01 |
| 239 | 200 | 0000000 | 11B20391 | 000000000 | 061AA817EC | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |

```
Z[0]  = 3D
Z[1]  = C1
Z[2]  = F0
Z[3]  = BB
Z[4]  = 58
Z[5]  = 1E
Z[6]  = 42
Z[7]  = 42
Z[8]  = 4B
Z[9]  = 8E
Z[10] = C1
Z[11] = 2A
Z[12] = 40
Z[13] = 63
Z[14] = 7A
Z[15] = 1E

================================================================================================
         Reload this pattern into the LFSRs
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 04B583D
LFSR2  <= 208E1EC1
LFSR3  <= 063C142F0
LFSR4  <= 0F7A2A42BB
C[t+1] <= 10

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
```

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 240 | 1 | 04B583D | 208E1EC1 | 063C142F0 | 0F7A2A42BB | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 241 | 2 | 096B07A | 411C3D82 | 0C78285E1 | 1EF4548577 | 1 | 0 | 1 | 1 | 1 | 10 | 10 | 11 |
| 242 | 3 | 12D60F4 | 02387B04 | 18F050BC3 | 3DE8A90AEF | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 243 | 4 | 05AC1E9 | 0470F609 | 11E0A1786 | 7BD15215DF | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 244 | 5 | 0B583D2 | 08E1EC13 | 03C142F0C | 77A2A42BBF | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 245 | 6 | 16B07A5 | 11C3D827 | 078285E18 | 6F4548577E | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 246 | 7 | 0D60F4B | 2387B04F | 0F050BC30 | 5E8A90AEFD | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 247 | 8 | 1AC1E97 | 470F609E | 1E0A17860 | 3D15215DFA | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 248 | 9 | 1583D2E | 0E1EC13D | 1C142F0C0 | 7A2A42BBF4 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 249 | 10 | 0B07A5D | 1C3D827B | 18285E181 | 74548577E9 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 250 | 11 | 160F4BB | 387B04F7 | 1050BC302 | 68A90AEFD2 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 251 | 12 | 0C1E976 | 70F609EE | 00A178605 | 515215DFA5 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 252 | 13 | 183D2ED | 61EC13DD | 0142F0C0B | 22A42BBF4B | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 253 | 14 | 107A5DA | 43D827BA | 0285E1817 | 4548577E97 | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 254 | 15 | 00F4BB4 | 07B04F74 | 050BC302F | 0A90AEFD2E | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 255 | 16 | 01E9769 | 0F609EE8 | 0A178605E | 15215DFA5C | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 256 | 17 | 03D2ED3 | 1EC13DD0 | 142F0C0BD | 2A42BBF4B9 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 257 | 18 | 07A5DA7 | 3D827BA0 | 085E1817B | 548577E972 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 258 | 19 | 0F4BB4F | 7B04F740 | 10BC302F6 | 290AEFD2E5 | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 259 | 20 | 1E9769F | 7609EE80 | 0178605ED | 5215DFA5CA | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 260 | 21 | 1D2ED3F | 6C13DD01 | 02F0C0BDA | 242BBF4B94 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 261 | 22 | 1A5DA7E | 5827BA03 | 05E1817B4 | 48577E9729 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 262 | 23 | 14BB4FC | 304F7407 | 0BC302F69 | 10AEFD2E53 | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 263 | 24 | 09769F9 | 609EE80E | 178605ED2 | 215DFA5CA7 | 1 | 1 | 0 | 0 | 0 | 10 | 00 | 01 |
| 264 | 25 | 12ED3F2 | 413DD01C | 0F0C0BDA4 | 42BBF4B94F | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 265 | 26 | 05DA7E5 | 027BA038 | 1E1817B49 | 0577E9729F | 0 | 0 | 1 | 0 | 1 | 01 | 00 | 10 |
| 266 | 27 | 0BB4FCA | 04F74071 | 1C302F693 | 0AEFD2E53F | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 267 | 28 | 1769F95 | 09EE80E3 | 18605ED27 | 15DFA5CA7F | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 268 | 29 | 0ED3F2B | 13DD01C6 | 10C0BDA4F | 2BBF4B94FE | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 269 | 30 | 1DA7E56 | 27BA038D | 01817B49F | 577E9729FD | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 270 | 31 | 1B4FCAD | 4F74071B | 0302F693E | 2EFD2E53FB | 1 | 0 | 0 | 1 | 0 | 01 | 10 | 10 |
| 271 | 32 | 169F95B | 1EE80E37 | 0605ED27D | 5DFA5CA7F7 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 272 | 33 | 0D3F2B7 | 3DD01C6E | 0C0BDA4FB | 3BF4B94FEF | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 273 | 34 | 1A7E56F | 7BA038DC | 1817B49F6 | 77E9729FDE | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 274 | 35 | 14FCADF | 774071B9 | 102F693ED | 6FD2E53FBD | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 275 | 36 | 09F95BE | 6E80E373 | 005ED27DB | 5FA5CA7F7B | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 276 | 37 | 13F2B7C | 5D01C6E7 | 00BDA4FB6 | 3F4B94FEF7 | 0 | 0 | 0 | 0 | 0 | 11 | 10 | 00 |
| 277 | 38 | 07E56F9 | 3A038DCE | 017B49F6C | 7E9729FDEE | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 278 | 39 | 0FCADF2 | 74071B9C | 02F693ED8 | 7D2E53FBDD | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 279 | 40 | 1F95BE5 | 680E3738 | 05ED27DB0 | 7A5CA7F7BA | 1 | 0 | 0 | 0 | 1 | 11 | 10 | 00 |
| 280 | 41 | 1F2B7CA | 501C6E71 | 0BDA4FB60 | 74B94FEF74 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 281 | 42 | 1E56F94 | 2038DCE2 | 17B49F6C0 | 69729FDEE8 | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 282 | 43 | 1CADF29 | 4071B9C4 | 0F693ED80 | 52E53FBDD1 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 283 | 44 | 195BE53 | 00E37389 | 1ED27DB01 | 25CA7F7BA3 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 284 | 45 | 12B7CA6 | 01C6E713 | 1DA4FB602 | 4B94FEF747 | 0 | 1 | 1 | 1 | 0 | 01 | 01 | 11 |
| 285 | 46 | 056F94C | 038DCE26 | 1B49F6C04 | 1729FDEE8E | 0 | 1 | 1 | 0 | 1 | 11 | 01 | 01 |
| 286 | 47 | 0ADF299 | 071B9C4D | 1693ED808 | 2E53FBDD1C | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 287 | 48 | 15BE532 | 0E37389A | 0D27DB011 | 5CA7F7BA38 | 0 | 0 | 1 | 1 | 0 | 10 | 10 | 11 |
| 288 | 49 | 0B7CA64 | 1C6E7135 | 1A4FB6022 | 394FEF7471 | 1 | 0 | 1 | 0 | 0 | 01 | 10 | 10 |
| 289 | 50 | 16F94C9 | 38DCE26A | 149F6C044 | 729FDEE8E2 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 290 | 51 | 0DF2993 | 71B9C4D4 | 093ED8089 | 653FBDD1C4 | 1 | 1 | 1 | 0 | 0 | 00 | 01 | 01 |
| 291 | 52 | 1BE5327 | 637389A9 | 127DB0112 | 4A7F7BA388 | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 292 | 53 | 17CA64E | 46E71353 | 04FB60224 | 14FEF74710 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 293 | 54 | 0F94C9C | 0DCE26A6 | 09F6C0448 | 29FDEE8E21 | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 294 | 55 | 1F29939 | 1B9C4D4D | 13ED80890 | 53FBDD1C42 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 295 | 56 | 1E53272 | 37389A9A | 07DB01121 | 27F7BA3884 | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 296 | 57 | 1CA64E5 | 6E713534 | 0FB602242 | 4FEF747108 | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 297 | 58 | 194C9CB | 5CE26A69 | 1F6C04485 | 1FDEE8E210 | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 298 | 59 | 1299397 | 39C4D4D3 | 1ED80890A | 3FBDD1C420 | 0 | 1 | 1 | 1 | 0 | 00 | 11 | 00 |
| 299 | 60 | 053272F | 7389A9A6 | 1DB011214 | 7F7BA38840 | 0 | 1 | 1 | 0 | 0 | 11 | 00 | 11 |
| 300 | 61 | 0A64E5E | 6713534C | 1B6022428 | 7EF7471081 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 301 | 62 | 14C9CBD | 4E26A699 | 16C044850 | 7DEE8E2102 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 302 | 63 | 099397A | 1C4D4D32 | 0D80890A0 | 7BDD1C4205 | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 303 | 64 | 13272F4 | 389A9A65 | 1B0112141 | 77BA38840B | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 304 | 65 | 064E5E8 | 713534CB | 160224283 | 6F74710817 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 305 | 66 | 0C9CBD1 | 626A6997 | 0C0448507 | 5EE8E2102E | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 306 | 67 | 19397A3 | 44D4D32E | 180890A0E | 3DD1C4205C | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 307 | 68 | 1272F46 | 09A9A65D | 10112141D | 7BA38840B8 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 308 | 69 | 04E5E8C | 13534CBA | 00224283A | 7747108171 | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 309 | 70 | 09CBD19 | 26A69975 | 004485075 | 6E8E2102E3 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 310 | 71 | 1397A32 | 4D4D32EB | 00890A0EA | 5D1C4205C7 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 311 | 72 | 072F465 | 1A9A65D7 | 0112141D5 | 3A38840B8F | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 312 | 73 | 0E5E8CA | 3534CBAF | 0224283AA | 747108171F | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 313 | 74 | 1CBD194 | 6A69975E | 044850755 | 68E2102E3E | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 314 | 75 | 197A329 | 54D32EBC | 0890A0EAB | 51C4205C7D | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 315 | 76 | 12F4653 | 29A65D79 | 112141D56 | 238840B8FA | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 316 | 77 | 05E8CA6 | 534CBAF2 | 024283AAD | 47108171F4 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 317 | 78 | 0BD194D | 269975E5 | 04850755B | 0E2102E3E9 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 318 | 79 | 17A329A | 4D32EBCB | 090A0EAB6 | 1C4205C7D2 | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 319 | 80 | 0F46535 | 1A65D797 | 12141D56D | 38840B8FA5 | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 320 | 81 | 1E8CA6A | 34CBAF2F | 04283AADA | 7108171F4B | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 321 | 82 | 1D194D5 | 69975E5F | 0850755B4 | 62102E3E97 | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 322 | 83 | 1A329AA | 532EBCBF | 10A0EAB68 | 44205C7D2F | 1 | 0 | 0 | 0 | 0 | 11 | 01 | 01 |
| 323 | 84 | 1465355 | 265D797F | 0141D56D1 | 0840B8FA5E | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 324 | 85 | 08CA6AB | 4CBAF2FF | 0283AADA2 | 108171F4BC | 1 | 1 | 0 | 1 | 0 | 01 | 01 | 11 |
| 325 | 86 | 1194D56 | 1975E5FF | 050755B45 | 2102E3E979 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 326 | 87 | 0329AAD | 32EBCBFF | 0A0EAB68A | 4205C7D2F3 | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 327 | 88 | 065355A | 65D797FF | 141D56D14 | 040B8FA5E7 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 328 | 89 | 0CA6AB4 | 4BAF2FFF | 083AADA28 | 08171F4BCF | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 329 | 90 | 194D569 | 175E5FFF | 10755B450 | 102E3E979E | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 330 | 91 | 129AAD3 | 2EBCBFFF | 00EAB68A1 | 205C7D2F3C | 0 | 1 | 0 | 0 | 0 | 10 | 01 | 11 |
| 331 | 92 | 05355A6 | 5D797FFF | 01D56D142 | 40B8FA5E78 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 332 | 93 | 0A6AB4D | 3AF2FFFE | 03AADA285 | 0171F4BCF1 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 333 | 94 | 14D569B | 75E5FFFD | 0755B450A | 02E3E979E2 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 334 | 95 | 09AAD37 | 6BCBFFFA | 0EAB68A15 | 05C7D2F3C4 | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 335 | 96 | 1355A6E | 5797FFF4 | 1D56D142A | 0B8FA5E788 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 336 | 97 | 06AB4DC | 2F2FFFE8 | 1AADA2854 | 171F4BCF11 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 337 | 98 | 0D569B8 | 5E5FFFD0 | 155B450A9 | 2E3E979E23 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 338 | 99 | 1AAD370 | 3CBFFFA1 | 0AB68A153 | 5C7D2F3C46 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 339 | 100 | 155A6E0 | 797FFF43 | 156D142A7 | 38FA5E788D | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 340 | 101 | 0AB4DC0 | 72FFFE87 | 0ADA2854E | 71F4BCF11B | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 341 | 102 | 1569B81 | 65FFFD0E | 15B450A9D | 63E979E236 | 0 | 1 | 0 | 1 | 0 | 11 | 10 | 01 |
| 342 | 103 | 0AD3703 | 4BFFFA1C | 0B68A153B | 47D2F3C46C | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 343 | 104 | 15A6E07 | 17FFF438 | 16D142A76 | 0FA5E788D8 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 344 | 105 | 0B4DC0F | 2FFFE870 | 0DA2854EC | 1F4BCF11B0 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 345 | 106 | 169B81F | 5FFFD0E1 | 1B450A9D8 | 3E979E2360 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 346 | 107 | 0D3703F | 3FFFA1C3 | 168A153B0 | 7D2F3C46C1 | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 347 | 108 | 1A6E07E | 7FFF4386 | 0D142A761 | 7A5E788D83 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 348 | 109 | 14DC0FD | 7FFE870C | 1A2854EC2 | 74BCF11B07 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 349 | 110 | 09B81FB | 7FFD0E19 | 1450A9D84 | 6979E2360E | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 350 | 111 | 13703F6 | 7FFA1C33 | 08A153B09 | 52F3C46C1C | 0 | 1 | 1 | 1 | 1 | 11 | 10 | 01 |
| 351 | 112 | 06E07EC | 7FF43867 | 1142A7612 | 25E788D838 | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |
| 352 | 113 | 0DC0FD8 | 7FE870CF | 02854EC25 | 4BCF11B071 | 1 | 1 | 0 | 1 | 1 | 11 | 00 | 11 |
| 353 | 114 | 1B81FB1 | 7FD0E19E | 050A9D84B | 179E2360E3 | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 354 | 115 | 1703F62 | 7FA1C33D | 0A153B096 | 2F3C46C1C7 | 0 | 1 | 1 | 0 | 0 | 11 | 00 | 11 |
| 355 | 116 | 0E07EC4 | 7F43867B | 142A7612C | 5E788D838E | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 356 | 117 | 1C0FD88 | 7E870CF6 | 0854EC259 | 3CF11B071C | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 357 | 118 | 181FB11 | 7D0E19ED | 10A9D84B3 | 79E2360E38 | 1 | 0 | 0 | 1 | 1 | 11 | 01 | 01 |
| 358 | 119 | 103F622 | 7A1C33DA | 0153B0967 | 73C46C1C71 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 359 | 120 | 007EC45 | 743867B5 | 02A7612CE | 6788D838E3 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 360 | 121 | 00FD88B | 6870CF6B | 054EC259C | 4F11B071C6 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 361 | 122 | 01FB117 | 50E19ED7 | 0A9D84B38 | 1E2360E38C | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |
| 362 | 123 | 03F622F | 21C33DAE | 153B09671 | 3C46C1C718 | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 363 | 124 | 07EC45F | 43867B5C | 0A7612CE2 | 788D838E30 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 364 | 125 | 0FD88BF | 070CF6B9 | 14EC259C4 | 711B071C61 | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |



LifeScan, Inc. Ex. 1019
Page 929 of 1420
IPR2025-00103

## 1.3 SECOND SET OF SAMPLE DATA

```
Initial values for the key, BD_ADDR and clock

K'c2[0]   = 00   K'c2[1]   = 00    K'c2[2]   = 00    K'c2[3]  = 00
K'c2[4]   = 00   K'c2[5]   = 00    K'c2[6]   = 00    K'c2[7]  = 00
K'c2[8]   = 00   K'c2[9]   = 00    K'c2[10]  = 00    K'c2[11] = 00
K'c2[12]  = 00   K'c2[13]  = 00    K'c2[14]  = 00    K'c2[15] = 00

Addr2[0] = 00   Addr2[1] = 00    Addr2[2] = 00
Addr2[3] = 00   Addr2[4] = 00    Addr2[5] = 00

CL2[0]  = 00   CL2[1]  = 00    CL2[2]  = 00    CL2[3]  = 03
```

```
================================================================================================
         Fill LFSRs with initial data
================================================================================================

  t  clk#    LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

  0     0    0000000* 00000000* 000000000* 0000000000*     0  0  0  0    0  00     00   00
  1     1    0000001* 00000001* 000000001* 0000000001*     0  0  0  0    0  00     00   00
  2     2    0000002* 00000002* 000000002* 0000000003*     0  0  0  0    0  00     00   00
  3     3    0000004* 00000004* 000000004* 0000000007*     0  0  0  0    0  00     00   00
  4     4    0000008* 00000008* 000000008* 000000000E*     0  0  0  0    0  00     00   00
  5     5    0000010* 00000010* 000000010* 000000001C*     0  0  0  0    0  00     00   00
  6     6    0000020* 00000020* 000000020* 0000000038*     0  0  0  0    0  00     00   00
  7     7    0000040* 00000040* 000000040* 0000000070*     0  0  0  0    0  00     00   00
  8     8    0000080* 00000080* 000000080* 00000000E0*     0  0  0  0    0  00     00   00
  9     9    0000100* 00000100* 000000100* 00000001C0*     0  0  0  0    0  00     00   00
 10    10    0000200* 00000200* 000000200* 0000000380*     0  0  0  0    0  00     00   00
 11    11    0000400* 00000400* 000000400* 0000000700*     0  0  0  0    0  00     00   00
 12    12    0000800* 00000800* 000000800* 0000000E00*     0  0  0  0    0  00     00   00
 13    13    0001000* 00001000* 000001000* 0000001C00*     0  0  0  0    0  00     00   00
 14    14    0002000* 00002000* 000002000* 0000003800*     0  0  0  0    0  00     00   00
 15    15    0004000* 00004000* 000004000* 0000007000*     0  0  0  0    0  00     00   00
 16    16    0008000* 00008000* 000008000* 000000E000*     0  0  0  0    0  00     00   00
 17    17    0010000* 00010000* 000010000* 000001C000*     0  0  0  0    0  00     00   00
 18    18    0020000* 00020000* 000020000* 0000038000*     0  0  0  0    0  00     00   00
 19    19    0040000* 00040000* 000040000* 0000070000*     0  0  0  0    0  00     00   00
 20    20    0080000* 00080000* 000080000* 00000E0000*     0  0  0  0    0  00     00   00
 21    21    0100000* 00100000* 000100000* 00001C0000*     0  0  0  0    0  00     00   00
 22    22    0200000* 00200000* 000200000* 0000380000*     0  0  0  0    0  00     00   00
 23    23    0400000* 00400000* 000400000* 0000700000*     0  0  0  0    0  00     00   00
 24    24    0800000* 00800000* 000800000* 0000E00000*     1  1  0  0    0  01     00   00
 25    25    1000000* 01000000* 001000000* 0001C00000*     0  0  0  0    0  00     00   00
 26    26    0000001  02000000* 002000000* 0003800000*     0  0  0  0    0  00     00   00
 27    27    0000002  04000000* 004000000* 0007000000*     0  0  0  0    0  00     00   00
 28    28    0000004  08000000* 008000000* 000E000000*     0  0  0  0    0  00     00   00
 29    29    0000008  10000000* 010000000* 001C000000*     0  0  0  0    0  00     00   00
 30    30    0000010  20000000* 020000000* 0038000000*     0  0  0  0    0  00     00   00
 31    31    0000020  40000000* 040000000* 0070000000*     0  0  0  0    0  00     00   00
 32    32    0000040  00000001  080000000* 00E0000000*     0  0  1  1    0  01     00   00
 33    33    0000080  00000002  100000000* 01C0000000*     0  0  0  1    1  00     00   00
 34    34    0000101  00000004  000000001  0380000000*     0  0  0  1    1  00     00   00
```

*Sample Data*

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | 35 | 0000202 | 00000008 | 000000002 | 0700000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 36 | 36 | 0000404 | 00000010 | 000000004 | 0E00000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 37 | 37 | 0000808 | 00000020 | 000000008 | 1C00000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 38 | 38 | 0001011 | 00000040 | 000000011 | 3800000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 39 | 39 | 0002022 | 00000080 | 000000022 | 7000000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |

Start clocking Summation Combiner

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 0004044 | 00000100 | 000000044 | 6000000001 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 41 | 2 | 0008088 | 00000200 | 000000088 | 4000000003 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 42 | 3 | 0010111 | 00000400 | 000000111 | 0000000007 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 43 | 4 | 0020222 | 00000800 | 000000222 | 000000000E | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 44 | 5 | 0040444 | 00001001 | 000000444 | 000000001D | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 45 | 6 | 0080888 | 00002002 | 000000888 | 000000003B | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 46 | 7 | 0101111 | 00004004 | 000001111 | 0000000077 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 47 | 8 | 0202222 | 00008008 | 000002222 | 00000000EE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 48 | 9 | 0404444 | 00010011 | 000004444 | 00000001DD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 49 | 10 | 0808888 | 00020022 | 000008888 | 00000003BB | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 50 | 11 | 1011110 | 00040044 | 000011111 | 0000000777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 51 | 12 | 0022221 | 00080088 | 000022222 | 0000000EEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 52 | 13 | 0044442 | 00100110 | 000044444 | 0000001DDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 53 | 14 | 0088884 | 00200220 | 000088888 | 0000003BBB | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 54 | 15 | 0111109 | 00400440 | 000111111 | 0000007777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 55 | 16 | 0222212 | 00800880 | 000222222 | 000000EEEE | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 56 | 17 | 0444424 | 01001100 | 000444444 | 000001DDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 57 | 18 | 0888848 | 02002200 | 000888888 | 000003BBBB | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 58 | 19 | 1111090 | 04004400 | 001111110 | 0000077777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 59 | 20 | 0222120 | 08008800 | 002222220 | 00000EEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 60 | 21 | 0444240 | 10011000 | 004444440 | 00001DDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 61 | 22 | 0888480 | 20022000 | 008888880 | 00003BBBBB | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 62 | 23 | 1110900 | 40044000 | 011111100 | 0000777777 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 63 | 24 | 0221200 | 00088001 | 022222200 | 0000EEEEEE | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 64 | 25 | 0442400 | 00110003 | 044444400 | 0001DDDDDD | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 65 | 26 | 0884800 | 00220006 | 088888800 | 0003BBBBBB | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 66 | 27 | 1109000 | 0044000C | 111111000 | 0007777777 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 67 | 28 | 0212001 | 00880018 | 022222001 | 000EEEEEEE | 0 | 1 | 0 | 0 | 0 | 11 | 01 | 01 |
| 68 | 29 | 0424002 | 01100031 | 044444002 | 001DDDDDDC | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 69 | 30 | 0848004 | 02200062 | 088888004 | 003BBBBBB8 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 70 | 31 | 1090008 | 044000C4 | 111110008 | 0077777770 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 71 | 32 | 0120010 | 08800188 | 022220010 | 00EEEEEEE0 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 72 | 33 | 0240020 | 11000311 | 044440020 | 01DDDDDDC1 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 73 | 34 | 0480040 | 22000622 | 088880040 | 03BBBBBB83 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 74 | 35 | 0900081 | 44000C44 | 111100080 | 0777777707 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 75 | 36 | 1200103 | 08001888 | 022200101 | 0EEEEEEE0E | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 76 | 37 | 0400207 | 10003111 | 044400202 | 1DDDDDDC1D | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 77 | 38 | 080040E | 20006222 | 088800404 | 3BBBBBB83B | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 78 | 39 | 100081C | 4000C444 | 111000808 | 7777777077 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 79 | 40 | 0001038 | 00018888 | 022001010 | 6EEEEEE0EF | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 80 | 41 | 0002070 | 00031110 | 044002020 | 5DDDDDC1DE | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 81 | 42 | 00040E0 | 00062220 | 088004040 | 3BBBBB83BC | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 82 | 43 | 00081C1 | 000C4440 | 110008081 | 7777770779 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 83 | 44 | 0010383 | 00188880 | 020010103 | 6EEEEE0EF2 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 84 | 45 | 0020707 | 00311100 | 040020206 | 5DDDDC1DE5 | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 85 | 46 | 0040E0E | 00622200 | 08004040C | 3BBBB83BCB | 0 | 0 | 1 | 1 | 0 | 11 | 10 | 01 |
| 86 | 47 | 0081C1D | 00C44400 | 100080819 | 7777707797 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 87 | 48 | 010383A | 01888801 | 000101032 | 6EEEE0EF2F | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 88 | 49 | 0207075 | 03111003 | 000202064 | 5DDDC1DE5E | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |



LifeScan, Inc. Ex. 1019
Page 931 of 1420
IPR2025-00103

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 89 | 50 | 040E0EA | 06222006 | 0004040C8 | 3BBB83BCBC | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 90 | 51 | 081C1D5 | 0C44400C | 000808191 | 7777077979 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 91 | 52 | 10383AB | 18888018 | 001010323 | 6EEE0EF2F2 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 92 | 53 | 0070756 | 31110030 | 002020646 | 5DDC1DE5E5 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 93 | 54 | 00E0EAC | 62220060 | 004040C8C | 3BB83BCBCB | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 94 | 55 | 01C1D59 | 444400C1 | 008081919 | 7770779797 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 95 | 56 | 0383AB2 | 08880183 | 010103232 | 6EE0EF2F2F | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 96 | 57 | 0707565 | 11100307 | 020206464 | 5DC1DE5E5F | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 97 | 58 | 0E0EACA | 2220060E | 04040C8C8 | 3B83BCBCBF | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 98 | 59 | 1C1D594 | 44400C1C | 080819191 | 770779797E | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 99 | 60 | 183AB28 | 08801838 | 101032323 | 6E0EF2F2FC | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 100 | 61 | 1075650 | 11003070 | 002064647 | 5C1DE5E5F8 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 101 | 62 | 00EACA1 | 220060E0 | 0040C8C8E | 383BCBCBF0 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 102 | 63 | 01D5943 | 4400C1C0 | 00819191D | 70779797E0 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 103 | 64 | 03AB286 | 08018380 | 01032323A | 60EF2F2FC1 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 104 | 65 | 075650C | 10030701 | 020646475 | 41DE5E5F82 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 105 | 66 | 0EACA18 | 20060E02 | 040C8C8EA | 03BCBCBF04 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 106 | 67 | 1D59430 | 400C1C05 | 0819191D4 | 0779797E09 | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 107 | 68 | 1AB2861 | 0018380A | 1032323A9 | 0EF2F2FC12 | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 108 | 69 | 15650C3 | 00307015 | 006464752 | 1DE5E5F825 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 109 | 70 | 0ACA186 | 0060E02A | 00C8C8EA4 | 3BCBCBF04B | 1 | 0 | 0 | 1 | 1 | 00 | 11 | 10 |
| 110 | 71 | 159430C | 00C1C055 | 019191D48 | 779797E097 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 111 | 72 | 0B28618 | 018380AA | 032323A90 | 6F2F2FC12F | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 112 | 73 | 1650C30 | 03070154 | 064647520 | 5E5E5F825E | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 113 | 74 | 0CA1860 | 060E02A8 | 0C8C8EA40 | 3CBCBF04BC | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 114 | 75 | 19430C0 | 0C1C0550 | 19191D480 | 79797E0979 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 115 | 76 | 1286180 | 18380AA0 | 12323A900 | 72F2FC12F2 | 0 | 0 | 0 | 1 | 0 | 11 | 11 | 11 |
| 116 | 77 | 050C301 | 30701541 | 046475201 | 65E5F825E5 | 0 | 0 | 0 | 1 | 0 | 11 | 11 | 11 |
| 117 | 78 | 0A18602 | 60E02A82 | 08C8EA402 | 4BCBF04BCB | 1 | 1 | 1 | 1 | 1 | 10 | 11 | 11 |
| 118 | 79 | 1430C04 | 41C05505 | 1191D4804 | 1797E09796 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 119 | 80 | 0861808 | 0380AA0A | 0323A9008 | 2F2FC12F2C | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 120 | 81 | 10C3011 | 07015415 | 064752011 | 5E5F825E59 | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 121 | 82 | 0186022 | 0E02A82A | 0C8EA4022 | 3CBF04BCB2 | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 122 | 83 | 030C045 | 1C055054 | 191D48044 | 797E097964 | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 123 | 84 | 061808A | 380AA0A8 | 123A90088 | 72FC12F2C9 | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 124 | 85 | 0C30115 | 70154151 | 047520111 | 65F825E593 | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 125 | 86 | 186022A | 602A82A3 | 08EA40222 | 4BF04BCB26 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 126 | 87 | 10C0455 | 40550546 | 11D480444 | 17E097964C | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 127 | 88 | 01808AA | 00AA0A8D | 03A900888 | 2FC12F2C99 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 128 | 89 | 0301155 | 0154151A | 075201111 | 5F825E5932 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 129 | 90 | 06022AA | 02A82A34 | 0EA402222 | 3F04BCB264 | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 130 | 91 | 0C04555 | 05505468 | 1D4804445 | 7E097964C9 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 131 | 92 | 1808AAA | 0AA0A8D0 | 1A900888A | 7C12F2C992 | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 132 | 93 | 1011555 | 154151A1 | 152011115 | 7825E59324 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 133 | 94 | 0022AAB | 2A82A342 | 0A402222B | 704BCB2648 | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 134 | 95 | 0045556 | 55054684 | 148044457 | 6097964C91 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 135 | 96 | 008AAAC | 2A0A8D09 | 0900888AE | 412F2C9923 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 136 | 97 | 0115559 | 54151A12 | 12011115D | 025E593246 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 137 | 98 | 022AAB2 | 282A3424 | 0402222BA | 04BCB2648D | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 138 | 99 | 0455564 | 50546848 | 080444575 | 097964C91A | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 139 | 100 | 08AAAC8 | 20A8D090 | 100888AEA | 12F2C99235 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 140 | 101 | 1155591 | 4151A120 | 0011115D5 | 25E593246A | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 141 | 102 | 02AAB22 | 02A34240 | 002222BAA | 4BCB2648D5 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 142 | 103 | 0555644 | 05468481 | 004445755 | 17964C91AB | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 143 | 104 | 0AAAC88 | 0A8D0903 | 00888AEAA | 2F2C992357 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 144 | 105 | 1555911 | 151A1206 | 011115D55 | 5E593246AE | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 145 | 106 | 0AAB222 | 2A34240C | 02222BAAA | 3CB2648D5C | 1 | 0 | 0 | 1 | 1 | 11 | 01 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 146 | 107 | 1556445 | 54684818 | 044457555 | 7964C91AB8 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 147 | 108 | 0AAC88B | 28D09030 | 0888AEAAA | 72C9923571 | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 148 | 109 | 1559117 | 51A12060 | 11115D555 | 6593246AE2 | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 149 | 110 | 0AB222F | 234240C0 | 0222BAAAB | 4B2648D5C5 | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 150 | 111 | 156445F | 46848180 | 044575557 | 164C91AB8A | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 151 | 112 | 0AC88BF | 0D090301 | 088AEAAAE | 2C99235714 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 152 | 113 | 159117F | 1A120602 | 1115D555D | 593246AE28 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 153 | 114 | 0B222FE | 34240C04 | 022BAAABA | 32648D5C51 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 154 | 115 | 16445FD | 68481809 | 045755574 | 64C91AB8A2 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 155 | 116 | 0C88BFA | 50903012 | 08AEAAAE8 | 4992357144 | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 156 | 117 | 19117F5 | 21206024 | 115D555D1 | 13246AE288 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 157 | 118 | 1222FEA | 4240C048 | 02BAAABA2 | 2648D5C511 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 158 | 119 | 0445FD5 | 04818090 | 057555744 | 4C91AB8A23 | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 159 | 120 | 088BFAA | 09030120 | 0AEAAAE88 | 1923571446 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 160 | 121 | 1117F55 | 12060240 | 15D555D11 | 3246AE288D | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 161 | 122 | 022FEAA | 240C0480 | 0BAAABA22 | 648D5C511B | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 162 | 123 | 045FD54 | 48180900 | 175557444 | 491AB8A237 | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 163 | 124 | 08BFAA9 | 10301200 | 0EAAAE889 | 123571446F | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 164 | 125 | 117F553 | 20602400 | 1D555D113 | 246AE288DF | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 165 | 126 | 02FEAA7 | 40C04800 | 1AAABA227 | 48D5C511BE | 0 | 1 | 1 | 1 | 1 | 10 | 00 | 01 |
| 166 | 127 | 05FD54F | 01809001 | 15557444F | 11AB8A237D | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 167 | 128 | 0BFAA9F | 03012002 | 0AAAE889E | 23571446FA | 1 | 0 | 1 | 0 | 0 | 00 | 00 | 10 |
| 168 | 129 | 17F553F | 06024004 | 1555D113D | 46AE288DF5 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 169 | 130 | 0FEAA7E | 0C048008 | 0AABA227A | 0D5C511BEA | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 170 | 131 | 1FD54FC | 18090011 | 1557444F5 | 1AB8A237D5 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 171 | 132 | 1FAA9F9 | 30120022 | 0AAE889EB | 3571446FAA | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 172 | 133 | 1F553F2 | 60240044 | 155D113D7 | 6AE288DF55 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 173 | 134 | 1EAA7E4 | 40480089 | 0ABA227AE | 55C511BEAA | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 174 | 135 | 1D54FC9 | 00900113 | 157444F5D | 2B8A237D54 | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 175 | 136 | 1AA9F93 | 01200227 | 0AE889EBA | 571446FAA8 | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 176 | 137 | 1553F26 | 0240044E | 15D113D75 | 2E288DF550 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 177 | 138 | 0AA7E4C | 0480089C | 0BA227AEA | 5C511BEAA0 | 1 | 1 | 1 | 0 | 0 | 00 | 11 | 00 |
| 178 | 139 | 154FC98 | 09001138 | 17444F5D4 | 38A237D540 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 179 | 140 | 0A9F931 | 12002270 | 0E889EBA9 | 71446FAA81 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 180 | 141 | 153F262 | 240044E0 | 1D113D753 | 6288DF5503 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 181 | 142 | 0A7E4C5 | 480089C0 | 1A227AEA7 | 4511BEAA06 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 182 | 143 | 14FC98B | 10011381 | 1444F5D4F | 0A237D540D | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 183 | 144 | 09F9316 | 20022702 | 0889EBA9E | 1446FAA81A | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 184 | 145 | 13F262D | 40044E04 | 1113D753D | 288DF55035 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 185 | 146 | 07E4C5A | 00089C08 | 0227AEA7A | 511BEAA06A | 0 | 0 | 0 | 0 | 0 | 01 | 10 | 11 |
| 186 | 147 | 0FC98B4 | 00113810 | 044F5D4F5 | 2237D540D5 | 1 | 0 | 0 | 0 | 0 | 01 | 01 | 10 |
| 187 | 148 | 1F93169 | 00227021 | 089EBA9EB | 446FAA81AA | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 188 | 149 | 1F262D2 | 0044E042 | 113D753D7 | 08DF550355 | 1 | 0 | 0 | 1 | 1 | 10 | 11 | 01 |
| 189 | 150 | 1E4C5A4 | 0089C085 | 027AEA7AE | 11BEAA06AA | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 190 | 151 | 1C98B48 | 0113810A | 04F5D4F5C | 237D540D54 | 1 | 0 | 0 | 0 | 1 | 10 | 10 | 10 |
| 191 | 152 | 1931691 | 02270215 | 09EBA9EB8 | 46FAA81AA9 | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 192 | 153 | 1262D22 | 044E042A | 13D753D71 | 0DF5503553 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 193 | 154 | 04C5A44 | 089C0854 | 07AEA7AE2 | 1BEAA06AA7 | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 194 | 155 | 098B488 | 113810A8 | 0F5D4F5C4 | 37D540D54E | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 195 | 156 | 1316910 | 22702150 | 1EBA9EB89 | 6FAA81AA9D | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 196 | 157 | 062D220 | 44E042A0 | 1D753D712 | 5F5503553A | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 197 | 158 | 0C5A440 | 09C08540 | 1AEA7AE25 | 3EAA06AA75 | 1 | 1 | 1 | 1 | 1 | 10 | 11 | 11 |
| 198 | 159 | 18B4880 | 13810A80 | 15D4F5C4B | 7D540D54EA | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 199 | 160 | 1169100 | 27021500 | 0BA9EB897 | 7AA81AA9D5 | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 200 | 161 | 02D2201 | 4E042A00 | 1753D712E | 75503553AB | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 201 | 162 | 05A4403 | 1C085400 | 0EA7AE25C | 6AA06AA756 | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 202 | 163 | 0B48807 | 3810A800 | 1D4F5C4B8 | 5540D54EAC | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |



LifeScan, Inc. Ex. 1019
Page 933 of 1420
IPR2025-00103

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 203 | 164 | 169100F | 70215000 | 1A9EB8971 | 2A81AA9D58 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 204 | 165 | 0D2201E | 6042A001 | 153D712E3 | 5503553AB0 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 205 | 166 | 1A4403C | 40854002 | 0A7AE25C6 | 2A06AA7561 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 206 | 167 | 1488079 | 010A8004 | 14F5C4B8D | 540D54EAC3 | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 207 | 168 | 09100F2 | 02150009 | 09EB8971B | 281AA9D586 | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 208 | 169 | 12201E5 | 042A0012 | 13D712E37 | 503553AB0C | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 209 | 170 | 04403CA | 08540024 | 07AE25C6E | 206AA75618 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 210 | 171 | 0880795 | 10A80048 | 0F5C4B8DD | 40D54EAC30 | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 211 | 172 | 1100F2A | 21500091 | 1EB8971BA | 01AA9D5861 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 212 | 173 | 0201E54 | 42A00122 | 1D712E374 | 03553AB0C3 | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 213 | 174 | 0403CA9 | 05400244 | 1AE25C6E9 | 06AA756186 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 214 | 175 | 0807952 | 0A800488 | 15C4B8DD3 | 0D54EAC30D | 1 | 1 | 0 | 0 | 1 | 11 | 11 | 11 |
| 215 | 176 | 100F2A5 | 15000911 | 0B8971BA6 | 1AA9D5861A | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 216 | 177 | 001E54A | 2A001223 | 1712E374C | 3553AB0C35 | 0 | 0 | 0 | 0 | 1 | 00 | 11 | 11 |
| 217 | 178 | 003CA94 | 54002446 | 0E25C6E98 | 6AA756186A | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 218 | 179 | 0079528 | 2800488D | 1C4B8DD31 | 554EAC30D5 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 219 | 180 | 00F2A50 | 5000911B | 18971BA62 | 2A9D5861AA | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 220 | 181 | 01E54A0 | 20012236 | 112E374C4 | 553AB0C355 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 221 | 182 | 03CA940 | 4002446C | 025C6E988 | 2A756186AA | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 222 | 183 | 0795280 | 000488D9 | 04B8DD310 | 54EAC30D54 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 223 | 184 | 0F2A500 | 000911B2 | 0971BA620 | 29D5861AA8 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 224 | 185 | 1E54A00 | 00122364 | 12E374C40 | 53AB0C3550 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 225 | 186 | 1CA9400 | 002446C8 | 05C6E9880 | 2756186AA0 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 226 | 187 | 1952800 | 00488D90 | 0B8DD3101 | 4EAC30D540 | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 227 | 188 | 12A5000 | 00911B20 | 171BA6202 | 1D5861AA81 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 228 | 189 | 054A000 | 01223640 | 0E374C404 | 3AB0C35502 | 0 | 0 | 1 | 1 | 0 | 10 | 10 | 11 |
| 229 | 190 | 0A94000 | 02446C80 | 1C6E98808 | 756186AA05 | 1 | 0 | 1 | 0 | 0 | 01 | 10 | 10 |
| 230 | 191 | 1528001 | 0488D901 | 18DD31011 | 6AC30D540B | 0 | 1 | 1 | 1 | 0 | 10 | 01 | 10 |
| 231 | 192 | 0A50003 | 0911B203 | 11BA62023 | 55861AA817 | 1 | 0 | 0 | 1 | 0 | 11 | 10 | 01 |
| 232 | 193 | 14A0006 | 12236407 | 0374C4047 | 2B0C35502F | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 233 | 194 | 094000C | 2446C80E | 06E98808E | 56186AA05F | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 234 | 195 | 1280018 | 488D901C | 0DD31011D | 2C30D540BF | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 235 | 196 | 0500030 | 111B2039 | 1BA62023A | 5861AA817E | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 236 | 197 | 0A00060 | 22364072 | 174C40475 | 30C35502FD | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 237 | 198 | 14000C0 | 446C80E4 | 0E98808EA | 6186AA05FB | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 238 | 199 | 0800180 | 08D901C8 | 1D31011D5 | 430D540BF6 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 239 | 200 | 1000301 | 11B20391 | 1A62023AB | 061AA817EC | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |

```
Z[0]  = 25
Z[1]  = 45
Z[2]  = 6B
Z[3]  = 55
Z[4]  = 5F
Z[5]  = C2
Z[6]  = 20
Z[7]  = E5
Z[8]  = C4
Z[9]  = F8
Z[10] = 3A
Z[11] = F1
Z[12] = FF
Z[13] = 89
Z[14] = 02
Z[15] = 35

================================================================================================
        Reload this pattern into the LFSRs
```



LifeScan, Inc. Ex. 1019
Page 934 of 1420
IPR2025-00103

*Sample Data*

Hold content of Summation Combiner regs and calculate new C[t+1] and Z values

```
LFSR1  <= 1C45F25
LFSR2  <= 7FF8C245
LFSR3  <= 1893A206B
LFSR4  <= 1A02F1E555
C[t+1] <= 10
```

Generating 125 key symbols (encryption/decryption sequence)

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 240 | 1 | 1C45F25 | 7FF8C245 | 1893A206B | 1A02F1E555 | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 241 | 2 | 188BE4A | 7FF1848B | 1127440D7 | 3405E3CAAB | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 242 | 3 | 1117C95 | 7FE30917 | 024E881AF | 680BC79557 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 243 | 4 | 022F92B | 7FC6122F | 049D1035E | 50178F2AAF | 0 | 1 | 0 | 0 | 0 | 11 | 01 | 01 |
| 244 | 5 | 045F257 | 7F8C245E | 093A206BD | 202F1E555E | 0 | 1 | 1 | 0 | 1 | 10 | 11 | 01 |
| 245 | 6 | 08BE4AE | 7F1848BC | 127440D7A | 405E3CAABC | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 246 | 7 | 117C95C | 7E309178 | 04E881AF4 | 00BC795579 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 247 | 8 | 02F92B8 | 7C6122F0 | 09D1035E8 | 0178F2AAF2 | 0 | 0 | 1 | 0 | 0 | 11 | 01 | 01 |
| 248 | 9 | 05F2570 | 78C245E1 | 13A206BD0 | 02F1E555E5 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 249 | 10 | 0BE4AE1 | 71848BC2 | 07440D7A0 | 05E3CAABCA | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 250 | 11 | 17C95C3 | 63091784 | 0E881AF40 | 0BC7955795 | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 251 | 12 | 0F92B87 | 46122F09 | 1D1035E80 | 178F2AAF2B | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 252 | 13 | 1F2570F | 0C245E12 | 1A206BD01 | 2F1E555E56 | 1 | 0 | 1 | 0 | 0 | 11 | 10 | 01 |
| 253 | 14 | 1E4AE1F | 1848BC25 | 1440D7A03 | 5E3CAABCAC | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 254 | 15 | 1C95C3E | 3091784A | 0881AF407 | 3C79557958 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 255 | 16 | 192B87D | 6122F094 | 11035E80F | 78F2AAF2B1 | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 256 | 17 | 12570FA | 4245E128 | 0206BD01E | 71E555E562 | 0 | 0 | 0 | 1 | 0 | 10 | 01 | 11 |
| 257 | 18 | 04AE1F4 | 048BC250 | 040D7A03D | 63CAABCAC5 | 0 | 1 | 0 | 1 | 0 | 11 | 10 | 01 |
| 258 | 19 | 095C3E8 | 091784A0 | 081AF407A | 479557958A | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 259 | 20 | 12B87D1 | 122F0941 | 1035E80F4 | 0F2AAF2B14 | 0 | 0 | 0 | 0 | 1 | 11 | 01 | 11 |
| 260 | 21 | 0570FA3 | 245E1283 | 006BD01E9 | 1E555E5628 | 0 | 0 | 0 | 0 | 1 | 01 | 11 | 01 |
| 261 | 22 | 0AE1F46 | 48BC2506 | 00D7A03D2 | 3CAABCAC50 | 1 | 1 | 0 | 1 | 0 | 01 | 01 | 11 |
| 262 | 23 | 15C3E8C | 11784A0C | 01AF407A5 | 79557958A0 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 263 | 24 | 0B87D18 | 22F09419 | 035E80F4A | 72AAF2B140 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 264 | 25 | 170FA30 | 45E12832 | 06BD01E94 | 6555E56280 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 265 | 26 | 0E1F460 | 0BC25065 | 0D7A03D28 | 4AABCAC501 | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 266 | 27 | 1C3E8C0 | 1784A0CB | 1AF407A50 | 1557958A03 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 267 | 28 | 187D181 | 2F094196 | 15E80F4A0 | 2AAF2B1406 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 268 | 29 | 10FA302 | 5E12832C | 0BD01E941 | 555E56280C | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 269 | 30 | 01F4604 | 3C250658 | 17A03D283 | 2ABCAC5019 | 0 | 0 | 0 | 1 | 0 | 01 | 11 | 00 |
| 270 | 31 | 03E8C09 | 784A0CB0 | 0F407A506 | 557958A033 | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 271 | 32 | 07D1812 | 70941960 | 1E80F4A0C | 2AF2B14066 | 0 | 1 | 1 | 1 | 1 | 11 | 10 | 01 |
| 272 | 33 | 0FA3024 | 612832C1 | 1D01E9419 | 55E56280CD | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 273 | 34 | 1F46049 | 42506583 | 1A03D2832 | 2BCAC5019A | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 274 | 35 | 1E8C093 | 04A0CB07 | 1407A5065 | 57958A0335 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 275 | 36 | 1D18127 | 0941960F | 080F4A0CB | 2F2B14066B | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 276 | 37 | 1A3024F | 12832C1F | 101E94196 | 5E56280CD7 | 1 | 1 | 0 | 0 | 0 | 00 | 10 | 00 |
| 277 | 38 | 146049F | 2506583E | 003D2832C | 3CAC5019AE | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 278 | 39 | 08C093E | 4A0CB07D | 007A50658 | 7958A0335D | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 279 | 40 | 118127C | 141960FA | 00F4A0CB0 | 72B14066BA | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 280 | 41 | 03024F8 | 2832C1F4 | 01E941961 | 656280CD74 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 281 | 42 | 06049F1 | 506583E9 | 03D2832C2 | 4AC5019AE9 | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 282 | 43 | 0C093E2 | 20CB07D2 | 07A506585 | 158A0335D3 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 283 | 44 | 18127C5 | 41960FA5 | 0F4A0CB0B | 2B14066BA7 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 284 | 45 | 1024F8A | 032C1F4B | 1E9419616 | 56280CD74F | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 285 | 46 | 0049F15 | 06583E97 | 1D2832C2C | 2C5019AE9F | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 286 | 47 | 0093E2B | 0CB07D2F | 1A5065859 | 58A0335D3E | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 287 | 48 | 0127C56 | 1960FA5E | 14A0CB0B2 | 314066BA7D | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 288 | 49 | 024F8AD | 32C1F4BC | 094196164 | 6280CD74FB | 0 | 1 | 1 | 1 | 0 | 11 | 01 | 00 |
| 289 | 50 | 049F15A | 6583E978 | 12832C2C8 | 45019AE9F6 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 290 | 51 | 093E2B5 | 4B07D2F0 | 050658591 | 0A0335D3ED | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 291 | 52 | 127C56B | 160FA5E0 | 0A0CB0B22 | 14066BA7DA | 0 | 0 | 1 | 0 | 0 | 01 | 01 | 10 |
| 292 | 53 | 04F8AD7 | 2C1F4BC1 | 141961645 | 280CD74FB5 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 293 | 54 | 09F15AF | 583E9783 | 0832C2C8A | 5019AE9F6A | 1 | 0 | 1 | 0 | 0 | 11 | 10 | 01 |
| 294 | 55 | 13E2B5E | 307D2F06 | 106585915 | 20335D3ED5 | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 295 | 56 | 07C56BD | 60FA5E0D | 00CB0B22B | 4066BA7DAA | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 296 | 57 | 0F8AD7A | 41F4BC1B | 019616457 | 00CD74FB54 | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 297 | 58 | 1F15AF4 | 03E97836 | 032C2C8AF | 019AE9F6A9 | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 298 | 59 | 1E2B5E9 | 07D2F06C | 06585915E | 0335D3ED52 | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 299 | 60 | 1C56BD2 | 0FA5E0D8 | 0CB0B22BC | 066BA7DAA4 | 1 | 1 | 1 | 0 | 0 | 10 | 01 | 10 |
| 300 | 61 | 18AD7A5 | 1F4BC1B0 | 196164578 | 0CD74FB549 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 301 | 62 | 115AF4B | 3E978361 | 12C2C8AF0 | 19AE9F6A92 | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |
| 302 | 63 | 02B5E96 | 7D2F06C2 | 0585915E0 | 335D3ED524 | 0 | 0 | 0 | 0 | 0 | 10 | 00 | 11 |
| 303 | 64 | 056BD2D | 7A5E0D85 | 0B0B22BC1 | 66BA7DAA49 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 304 | 65 | 0AD7A5B | 74BC1B0A | 161645783 | 4D74FB5493 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 305 | 66 | 15AF4B6 | 69783615 | 0C2C8AF07 | 1AE9F6A926 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 306 | 67 | 0B5E96D | 52F06C2B | 185915E0F | 35D3ED524C | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 307 | 68 | 16BD2DB | 25E0D857 | 10B22BC1F | 6BA7DAA499 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 308 | 69 | 0D7A5B7 | 4BC1B0AF | 01645783F | 574FB54933 | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 309 | 70 | 1AF4B6F | 1783615F | 02C8AF07F | 2E9F6A9266 | 1 | 1 | 0 | 1 | 1 | 01 | 10 | 10 |
| 310 | 71 | 15E96DF | 2F06C2BF | 05915E0FF | 5D3ED524CC | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 311 | 72 | 0BD2DBF | 5E0D857F | 0B22BC1FE | 3A7DAA4998 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 312 | 73 | 17A5B7F | 3C1B0AFE | 1645783FD | 74FB549331 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 313 | 74 | 0F4B6FF | 783615FD | 0C8AF07FA | 69F6A92662 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 314 | 75 | 1E96DFF | 706C2BFB | 1915E0FF5 | 53ED524CC4 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 315 | 76 | 1D2DBFE | 60D857F6 | 122BC1FEB | 27DAA49988 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 316 | 77 | 1A5B7FD | 41B0AFEC | 045783FD7 | 4FB5493310 | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 317 | 78 | 14B6FFA | 03615FD8 | 08AF07FAE | 1F6A926620 | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 318 | 79 | 096DFF4 | 06C2BFB1 | 115E0FF5D | 3ED524CC40 | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 319 | 80 | 12DBFE8 | 0D857F63 | 02BC1FEBB | 7DAA499881 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 320 | 81 | 05B7FD0 | 1B0AFEC6 | 05783FD77 | 7B54933103 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 321 | 82 | 0B6FFA1 | 3615FD8C | 0AF07FAEF | 76A9266206 | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 322 | 83 | 16DFF42 | 6C2BFB18 | 15E0FF5DE | 6D524CC40C | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 323 | 84 | 0DBFE85 | 5857F631 | 0BC1FEBBD | 5AA4998819 | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 324 | 85 | 1B7FD0B | 30AFEC62 | 1783FD77A | 3549331033 | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 325 | 86 | 16FFA16 | 615FD8C5 | 0F07FAEF5 | 6A92662067 | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 326 | 87 | 0DFF42D | 42BFB18B | 1E0FF5DEA | 5524CC40CE | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 327 | 88 | 1BFE85B | 057F6317 | 1C1FEBBD5 | 2A4998819C | 1 | 0 | 1 | 0 | 0 | 00 | 00 | 10 |
| 328 | 89 | 17FD0B7 | 0AFEC62E | 183FD77AA | 5493310339 | 0 | 1 | 1 | 1 | 1 | 01 | 00 | 00 |
| 329 | 90 | 0FFA16F | 15FD8C5C | 107FAEF55 | 2926620672 | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 330 | 91 | 1FF42DF | 2BFB18B9 | 00FF5DEAA | 524CC40CE5 | 1 | 1 | 0 | 0 | 0 | 10 | 00 | 01 |
| 331 | 92 | 1FE85BF | 57F63172 | 01FEBBD55 | 24998819CA | 1 | 1 | 0 | 1 | 1 | 00 | 10 | 00 |
| 332 | 93 | 1FD0B7F | 2FEC62E4 | 03FD77AAA | 4933103394 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 333 | 94 | 1FA16FF | 5FD8C5C9 | 07FAEF555 | 1266206728 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 334 | 95 | 1F42DFF | 3FB18B93 | 0FF5DEAAA | 24CC40CE51 | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 335 | 96 | 1E85BFF | 7F631727 | 1FEBBD554 | 4998819CA3 | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 336 | 97 | 1D0B7FE | 7EC62E4F | 1FD77AAA9 | 1331033947 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 337 | 98 | 1A16FFC | 7D8C5C9F | 1FAEF5553 | 266206728E | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 338 | 99 | 142DFF9 | 7B18B93F | 1F5DEAAA7 | 4CC40CE51D | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 339 | 100 | 085BFF3 | 7631727F | 1EBBD554E | 198819CA3B | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 340 | 101 | 10B7FE6 | 6C62E4FF | 1D77AAA9C | 3310339477 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 341 | 102 | 016FFCC | 58C5C9FE | 1AEF55538 | 66206728EE | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 342 | 103 | 02DFF98 | 318B93FC | 15DEAAA70 | 4C40CE51DC | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |



LifeScan, Inc. Ex. 1019
Page 936 of 1420
IPR2025-00103

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 343 | 104 | 05BFF31 | 631727F8 | 0BBD554E1 | 18819CA3B9 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 344 | 105 | 0B7FE62 | 462E4FF1 | 177AAA9C2 | 3103394772 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 345 | 106 | 16FFCC5 | 0C5C9FE2 | 0EF555384 | 6206728EE4 | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 346 | 107 | 0DFF98A | 18B93FC4 | 1DEAAA709 | 440CE51DC9 | 1 | 1 | 1 | 0 | 0 | 00 | 11 | 00 |
| 347 | 108 | 1BFF315 | 31727F88 | 1BD554E12 | 0819CA3B93 | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 348 | 109 | 17FE62A | 62E4FF11 | 17AAA9C24 | 1033947726 | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 349 | 110 | 0FFCC54 | 45C9FE22 | 0F5553849 | 206728EE4C | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 350 | 111 | 1FF98A8 | 0B93FC44 | 1EAAA7093 | 40CE51DC99 | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 351 | 112 | 1FF3150 | 1727F889 | 1D554E127 | 019CA3B933 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 352 | 113 | 1FE62A0 | 2E4FF112 | 1AAA9C24F | 0339477267 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 353 | 114 | 1FCC541 | 5C9FE225 | 15553849E | 06728EE4CF | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 354 | 115 | 1F98A82 | 393FC44B | 0AAA7093C | 0CE51DC99F | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 355 | 116 | 1F31504 | 727F8897 | 1554E1279 | 19CA3B933E | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 356 | 117 | 1E62A09 | 64FF112F | 0AA9C24F2 | 339477267D | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 357 | 118 | 1CC5412 | 49FE225E | 1553849E4 | 6728EE4CFB | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 358 | 119 | 198A824 | 13FC44BC | 0AA7093C9 | 4E51DC99F7 | 1 | 1 | 1 | 0 | 1 | 10 | 00 | 01 |
| 359 | 120 | 1315049 | 27F88979 | 154E12792 | 1CA3B933EE | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 360 | 121 | 062A093 | 4FF112F3 | 0A9C24F24 | 39477267DC | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 361 | 122 | 0C54127 | 1FE225E6 | 153849E48 | 728EE4CFB8 | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 362 | 123 | 18A824E | 3FC44BCD | 0A7093C91 | 651DC99F71 | 1 | 1 | 1 | 0 | 0 | 11 | 01 | 00 |
| 363 | 124 | 115049C | 7F88979A | 14E127922 | 4A3B933EE2 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 364 | 125 | 02A0938 | 7F112F35 | 09C24F244 | 1477267DC5 | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |



LifeScan, Inc. Ex. 1019
Page 937 of 1420
IPR2025-00103

## 1.4 THIRD SET OF SAMPLES

```
Initial values for the key, pan address and clock

K'c3[0]   = FF   K'c3[1]   = FF    K'c3[2]   = FF    K'c3[3]  = FF
K'c3[4]   = FF   K'c3[5]   = FF    K'c3[6]   = FF    K'c3[7]  = FF
K'c3[8]   = FF   K'c3[9]   = FF    K'c3[10]  = FF    K'c3[11] = FF
K'c3[12]  = FF   K'c3[13]  = FF    K'c3[14]  = FF    K'c3[15] = FF

Addr3[0] = FF   Addr3[1] = FF    Addr3[2] = FF
Addr3[3] = FF   Addr3[4] = FF    Addr3[5] = FF

CL3[0]  = FF   CL3[1]  = FF    CL3[2]  = FF    CL3[3]  = 03
```

```
================================================================================================
         Fill LFSRs with initial data
================================================================================================
```

| t | clk# | LFSR1 | LFSR2 | LFSR3 | LFSR4 | X1 | X2 | X3 | X4 | Z | C[t+1] | C[t] | C[t-1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0000000* | 00000000* | 000000000* | 0000000000* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 1 | 1 | 0000001* | 00000001* | 000000001* | 0000000001* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 2 | 2 | 0000003* | 00000002* | 000000003* | 0000000003* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 3 | 3 | 0000007* | 00000004* | 000000007* | 0000000007* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 4 | 4 | 000000F* | 00000009* | 00000000F* | 000000000F* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 5 | 5 | 000001F* | 00000013* | 00000001F* | 000000001F* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 6 | 6 | 000003F* | 00000027* | 00000003F* | 000000003F* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 7 | 7 | 000007F* | 0000004F* | 00000007F* | 000000007F* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 8 | 8 | 00000FF* | 0000009F* | 0000000FF* | 00000000FF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 9 | 9 | 00001FF* | 0000013F* | 0000001FF* | 00000001FF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 10 | 10 | 00003FF* | 0000027F* | 0000003FF* | 00000003FF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 11 | 11 | 00007FF* | 000004FF* | 0000007FF* | 00000007FF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 12 | 12 | 0000FFF* | 000009FF* | 000000FFF* | 0000000FFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 13 | 13 | 0001FFF* | 000013FF* | 000001FFF* | 0000001FFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 14 | 14 | 0003FFF* | 000027FF* | 000003FFF* | 0000003FFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 15 | 15 | 0007FFF* | 00004FFF* | 000007FFF* | 0000007FFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 16 | 16 | 000FFFF* | 00009FFF* | 00000FFFF* | 000000FFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 17 | 17 | 001FFFF* | 00013FFF* | 00001FFFF* | 000001FFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 18 | 18 | 003FFFF* | 00027FFF* | 00003FFFF* | 000003FFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 19 | 19 | 007FFFF* | 0004FFFF* | 00007FFFF* | 000007FFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 20 | 20 | 00FFFFF* | 0009FFFF* | 0000FFFFF* | 00000FFFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 21 | 21 | 01FFFFF* | 0013FFFF* | 0001FFFFF* | 00001FFFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 22 | 22 | 03FFFFF* | 0027FFFF* | 0003FFFFF* | 00003FFFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 23 | 23 | 07FFFFF* | 004FFFFF* | 0007FFFFF* | 00007FFFFF* | 0 | 0 | 0 | 0 | 0 | 00 | 00 | 00 |
| 24 | 24 | 0FFFFFF* | 009FFFFF* | 000FFFFFF* | 0000FFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 25 | 25 | 1FFFFFF* | 013FFFFF* | 001FFFFFF* | 0001FFFFFF* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 26 | 26 | 1FFFFFF | 027FFFFF* | 003FFFFFF* | 0003FFFFFF* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 27 | 27 | 1FFFFFF | 04FFFFFF* | 007FFFFFF* | 0007FFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 28 | 28 | 1FFFFFF | 09FFFFFF* | 00FFFFFFF* | 000FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 29 | 29 | 1FFFFFF | 13FFFFFF* | 01FFFFFFF* | 001FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 30 | 30 | 1FFFFFF | 27FFFFFF* | 03FFFFFFF* | 003FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 31 | 31 | 1FFFFFF | 4FFFFFFF* | 07FFFFFFF* | 007FFFFFFF* | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 32 | 32 | 1FFFFFF | 1FFFFFFF | 0FFFFFFFF* | 00FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 33 | 33 | 1FFFFFF | 3FFFFFFE | 1FFFFFFFF* | 01FFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 34 | 34 | 1FFFFFF | 7FFFFFFC | 1FFFFFFFF | 03FFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | 35 | 1FFFFFF | 7FFFFFF9 | 1FFFFFFFF | 07FFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 36 | 36 | 1FFFFFF | 7FFFFFF3 | 1FFFFFFFF | 0FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 37 | 37 | 1FFFFFF | 7FFFFFE7 | 1FFFFFFFF | 1FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 38 | 38 | 1FFFFFF | 7FFFFFCF | 1FFFFFFFF | 3FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 39 | 39 | 1FFFFFF | 7FFFFF9F | 1FFFFFFFF | 7FFFFFFFF* | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |

Start clocking Summation Combiner

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 1FFFFFF | 7FFFF3F | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 41 | 2 | 1FFFFFF | 7FFFE7F | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 42 | 3 | 1FFFFFF | 7FFFCFF | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 43 | 4 | 1FFFFFF | 7FFF9FF | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 10 | 10 |
| 44 | 5 | 1FFFFFF | 7FFF3FF | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 45 | 6 | 1FFFFFF | 7FFE7FE | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 46 | 7 | 1FFFFFF | 7FFCFFC | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 47 | 8 | 1FFFFFF | 7FF9FF9 | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 48 | 9 | 1FFFFFF | 7FF3FF3 | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 49 | 10 | 1FFFFFF | 7FE7FE6 | 1FFFFFFFF | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 50 | 11 | 1FFFFFE | 7FCFFCC | 1FFFFFFFE | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 51 | 12 | 1FFFFFC | 7FF9FF99 | 1FFFFFFFC | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 10 | 10 |
| 52 | 13 | 1FFFFF8 | 7FF3FF33 | 1FFFFFFF8 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 53 | 14 | 1FFFFF0 | 7FE7FE67 | 1FFFFFFF0 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 54 | 15 | 1FFFFE0 | 7FCFFCCF | 1FFFFFFE1 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 55 | 16 | 1FFFFC0 | 7F9FF99F | 1FFFFFFC3 | 7FFFFFFFF | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 56 | 17 | 1FFFF80 | 7F3FF33E | 1FFFFFF87 | 7FFFFFFFE | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 57 | 18 | 1FFFF00 | 7E7FE67C | 1FFFFFF0F | 7FFFFFFFC | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 58 | 19 | 1FFFE01 | 7CFFCCF8 | 1FFFFFE1E | 7FFFFFFF8 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 59 | 20 | 1FFFC03 | 79FF99F0 | 1FFFFFC3C | 7FFFFFFF0 | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 60 | 21 | 1FFF807 | 73FF33E0 | 1FFFFF878 | 7FFFFFFE1 | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 61 | 22 | 1FFF00F | 67FE67C0 | 1FFFFF0F0 | 7FFFFFFC3 | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 62 | 23 | 1FFE01E | 4FFCCF80 | 1FFFFE1E1 | 7FFFFFF87 | 1 | 1 | 1 | 1 | 0 | 00 | 10 | 10 |
| 63 | 24 | 1FFC03C | 1FF99F00 | 1FFFFC3C3 | 7FFFFFF0F | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 64 | 25 | 1FF8078 | 3FF33E01 | 1FFFF8787 | 7FFFFFE1E | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 65 | 26 | 1FF00F0 | 7FE67C02 | 1FFFF0F0F | 7FFFFFC3C | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 66 | 27 | 1FE01E1 | 7FCCF805 | 1FFFE1E1E | 7FFFFF878 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 67 | 28 | 1FC03C3 | 7F99F00A | 1FFFC3C3C | 7FFFFF0F0 | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 68 | 29 | 1F80787 | 7F33E015 | 1FFF87878 | 7FFFFE1E1 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 69 | 30 | 1F00F0F | 7E67C02A | 1FFF0F0F0 | 7FFFFC3C3 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 70 | 31 | 1E01E1E | 7CCF8054 | 1FFE1E1E1 | 7FFFFF8787 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 71 | 32 | 1C03C3C | 799F00A9 | 1FFC3C3C3 | 7FFFFF0F0F | 1 | 1 | 1 | 1 | 1 | 01 | 01 | 11 |
| 72 | 33 | 1807878 | 733E0152 | 1FF878787 | 7FFFFE1E1E | 1 | 0 | 1 | 1 | 0 | 00 | 01 | 01 |
| 73 | 34 | 100F0F0 | 667C02A5 | 1FF0F0F0F | 7FFFFC3C3C | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 74 | 35 | 001E1E0 | 4CF8054B | 1FE1E1E1F | 7FFFF87878 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 75 | 36 | 003C3C1 | 19F00A96 | 1FC3C3C3F | 7FFFF0F0F0 | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 76 | 37 | 0078783 | 33E0152C | 1F878787F | 7FFFE1E1E1 | 0 | 1 | 1 | 1 | 1 | 01 | 00 | 00 |
| 77 | 38 | 00F0F07 | 67C02A59 | 1F0F0F0FF | 7FFFC3C3C3 | 0 | 1 | 1 | 1 | 0 | 11 | 01 | 00 |
| 78 | 39 | 01E1E0E | 4F8054B3 | 1E1E1E1FF | 7FFF878787 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 79 | 40 | 03C3C1C | 1F00A966 | 1C3C3C3FF | 7FFF0F0F0F | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 80 | 41 | 0787838 | 3E0152CC | 1878787FF | 7FFE1E1E1E | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 81 | 42 | 0F0F070 | 7C02A598 | 10F0F0FFF | 7FFC3C3C3C | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 82 | 43 | 1E1E0E0 | 78054B30 | 01E1E1FFF | 7FF8787878 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 83 | 44 | 1C3C1C0 | 700A9660 | 03C3C3FFE | 7FF0F0F0F0 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 84 | 45 | 1878380 | 60152CC0 | 078787FFC | 7FE1E1E1E0 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 85 | 46 | 10F0700 | 402A5980 | 0F0F0FFF8 | 7FC3C3C3C0 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 86 | 47 | 01E0E00 | 0054B300 | 1E1E1FFF0 | 7F87878780 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 87 | 48 | 03C1C00 | 00A96601 | 1C3C3FFE0 | 7F0F0F0F00 | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 88 | 49 | 0783800 | 0152CC03 | 18787FFC0 | 7E1E1E1E01 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 89 | 50 | 0F07000 | 02A59806 | 10F0FFF80 | 7C3C3C3C03 | 1 | 1 | 0 | 0 | 1 | 11 | 11 | 11 |
| 90 | 51 | 1E0E000 | 054B300D | 01E1FFF00 | 7878787807 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 91 | 52 | 1C1C001 | 0A96601A | 03C3FFE01 | 70F0F0F00F | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 92 | 53 | 1838003 | 152CC035 | 0787FFC03 | 61E1E1E01E | 1 | 0 | 0 | 1 | 0 | 10 | 10 | 11 |
| 93 | 54 | 1070007 | 2A59806B | 0F0FFF807 | 43C3C3C03C | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 94 | 55 | 00E000F | 54B300D7 | 1E1FFF00F | 0787878078 | 0 | 1 | 1 | 1 | 0 | 10 | 01 | 10 |
| 95 | 56 | 01C001F | 296601AE | 1C3FFE01F | 0F0F0F00F1 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 96 | 57 | 038003F | 52CC035C | 187FFC03F | 1E1E1E01E2 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 97 | 58 | 070007F | 259806B8 | 10FFF807F | 3C3C3C03C4 | 0 | 1 | 0 | 0 | 1 | 00 | 00 | 00 |
| 98 | 59 | 0E000FE | 4B300D71 | 01FFF00FE | 7878780788 | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 99 | 60 | 1C001FD | 16601AE2 | 03FFE01FD | 70F0F00F10 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 100 | 61 | 18003FA | 2CC035C5 | 07FFC03FB | 61E1E01E21 | 1 | 1 | 0 | 1 | 0 | 11 | 01 | 00 |
| 101 | 62 | 10007F4 | 59806B8B | 0FFF807F7 | 43C3C03C43 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 102 | 63 | 0000FE8 | 3300D717 | 1FFF00FEE | 0787807887 | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 103 | 64 | 0001FD0 | 6601AE2F | 1FFE01FDC | 0F0F00F10E | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 104 | 65 | 0003FA0 | 4C035C5F | 1FFC03FB8 | 1E1E01E21D | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 105 | 66 | 0007F40 | 1806B8BE | 1FF807F70 | 3C3C03C43B | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 106 | 67 | 000FE81 | 300D717C | 1FF00FEE1 | 7878078877 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 107 | 68 | 001FD02 | 601AE2F8 | 1FE01FDC2 | 70F00F10EF | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 108 | 69 | 003FA05 | 4035C5F0 | 1FC03FB84 | 61E01E21DE | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 109 | 70 | 007F40B | 006B8BE0 | 1F807F708 | 43C03C43BC | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 110 | 71 | 00FE816 | 00D717C0 | 1F00FEE11 | 0780788778 | 0 | 1 | 1 | 1 | 0 | 10 | 11 | 11 |
| 111 | 72 | 01FD02C | 01AE2F81 | 1E01FDC23 | 0F00F10EF1 | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 112 | 73 | 03FA059 | 035C5F02 | 1C03FB847 | 1E01E21DE3 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 113 | 74 | 07F40B3 | 06B8BE05 | 1807F708F | 3C03C43BC7 | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 114 | 75 | 0FE8166 | 0D717C0B | 100FEE11E | 780788778F | 1 | 0 | 0 | 0 | 0 | 01 | 01 | 10 |
| 115 | 76 | 1FD02CD | 1AE2F817 | 001FDC23D | 700F10EF1F | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 116 | 77 | 1FA059B | 35C5F02F | 003FB847A | 601E21DE3F | 1 | 1 | 0 | 0 | 1 | 10 | 11 | 01 |
| 117 | 78 | 1F40B37 | 6B8BE05E | 007F708F4 | 403C43BC7F | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 118 | 79 | 1E8166E | 5717C0BD | 00FEE11E9 | 00788778FF | 1 | 0 | 0 | 0 | 1 | 10 | 10 | 10 |
| 119 | 80 | 1D02CDC | 2E2F817A | 01FDC23D3 | 00F10EF1FE | 1 | 0 | 0 | 1 | 0 | 01 | 10 | 10 |
| 120 | 81 | 1A059B9 | 5C5F02F5 | 03FB847A6 | 01E21DE3FD | 1 | 0 | 0 | 1 | 1 | 01 | 01 | 10 |
| 121 | 82 | 140B373 | 38BE05EB | 07F708F4C | 03C43BC7FB | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 122 | 83 | 08166E7 | 717C0BD7 | 0FEE11E98 | 0788778FF7 | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 123 | 84 | 102CDCF | 62F817AE | 1FDC23D31 | 0F10EF1FEF | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 124 | 85 | 0059B9F | 45F02F5C | 1FB847A63 | 1E21DE3FDE | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 125 | 86 | 00B373E | 0BE05EB9 | 1F708F4C7 | 3C43BC7FBC | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 126 | 87 | 0166E7D | 17C0BD72 | 1EE11E98F | 788778FF78 | 0 | 1 | 1 | 1 | 0 | 10 | 11 | 11 |
| 127 | 88 | 02CDCFB | 2F817AE5 | 1DC23D31F | 710EF1FEF1 | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 128 | 89 | 059B9F7 | 5F02F5CA | 1B847A63F | 621DE3FDE2 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 129 | 90 | 0B373EF | 3E05EB94 | 1708F4C7F | 443BC7FBC4 | 1 | 0 | 0 | 0 | 1 | 10 | 10 | 10 |
| 130 | 91 | 166E7DF | 7C0BD728 | 0E11E98FF | 08778FF788 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 131 | 92 | 0CDCFBE | 7817AE50 | 1C23D31FF | 10EF1FEF10 | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 132 | 93 | 19B9F7D | 702F5CA1 | 1847A63FE | 21DE3FDE21 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 133 | 94 | 1373EFB | 605EB942 | 108F4C7FC | 43BC7FBC43 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 134 | 95 | 06E7DF7 | 40BD7285 | 011E98FF8 | 0778FF7886 | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 135 | 96 | 0DCFBEF | 017AE50A | 023D31FF0 | 0EF1FEF10D | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 136 | 97 | 1B9F7DF | 02F5CA15 | 047A63FE1 | 1DE3FDE21A | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 137 | 98 | 173EFBF | 05EB942B | 08F4C7FC3 | 3BC7FBC434 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 138 | 99 | 0E7DF7F | 0BD72856 | 11E98FF87 | 778FF78869 | 1 | 1 | 0 | 1 | 1 | 00 | 00 | 10 |
| 139 | 100 | 1CFBEFF | 17AE50AC | 03D31FF0F | 6F1FEF10D3 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 140 | 101 | 19F7DFE | 2F5CA159 | 07A63FE1E | 5E3FDE21A7 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 141 | 102 | 13EFBFC | 5EB942B3 | 0F4C7FC3C | 3C7FBC434F | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |
| 142 | 103 | 07DF7F8 | 3D728566 | 1E98FF878 | 78FF78869F | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 143 | 104 | 0FBEFF0 | 7AE50ACD | 1D31FF0F0 | 71FEF10D3E | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 144 | 105 | 1F7DFE1 | 75CA159B | 1A63FE1E1 | 63FDE21A7D | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 145 | 106 | 1EFBFC3 | 6B942B36 | 14C7FC3C3 | 47FBC434FB | 1 | 1 | 0 | 1 | 1 | 11 | 00 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 146 | 107 | 1DF7F86 | 5728566D | 098FF8786 | 0FF78869F7 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 147 | 108 | 1BEFF0C | 2E50ACDB | 131FF0F0C | 1FEF10D3EF | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 148 | 109 | 17DFE19 | 5CA159B6 | 063FE1E19 | 3FDE21A7DF | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 149 | 110 | 0FBFC33 | 3942B36D | 0C7FC3C32 | 7FBC434FBF | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 150 | 111 | 1F7F866 | 728566DB | 18FF87865 | 7F78869F7E | 1 | 1 | 1 | 0 | 0 | 00 | 01 | 01 |
| 151 | 112 | 1EFF0CC | 650ACDB6 | 11FF0F0CB | 7EF10D3EFC | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 152 | 113 | 1DFE199 | 4A159B6D | 03FE1E196 | 7DE21A7DF9 | 1 | 0 | 0 | 1 | 0 | 00 | 10 | 00 |
| 153 | 114 | 1BFC333 | 142B36DB | 07FC3C32C | 7BC434FBF3 | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 154 | 115 | 17F8666 | 28566DB6 | 0FF878659 | 778869F7E6 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 155 | 116 | 0FF0CCC | 50ACDB6D | 1FF0F0CB3 | 6F10D3EFCC | 1 | 1 | 1 | 0 | 0 | 11 | 01 | 00 |
| 156 | 117 | 1FE1999 | 2159B6DA | 1FE1E1966 | 5E21A7DF99 | 1 | 0 | 1 | 0 | 1 | 10 | 11 | 01 |
| 157 | 118 | 1FC3332 | 42B36DB5 | 1FC3C32CC | 3C434FBF33 | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 158 | 119 | 1F86664 | 0566DB6B | 1F8786599 | 78869F7E67 | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 159 | 120 | 1F0CCC8 | 0ACDB6D6 | 1F0F0CB33 | 710D3EFCCE | 1 | 1 | 1 | 0 | 0 | 10 | 01 | 10 |
| 160 | 121 | 1E19991 | 159B6DAC | 1E1E19666 | 621A7DF99D | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 161 | 122 | 1C33323 | 2B36DB58 | 1C3C32CCC | 4434FBF33B | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 162 | 123 | 1866647 | 566DB6B0 | 187865999 | 0869F7E676 | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 163 | 124 | 10CCC8F | 2CDB6D60 | 10F0CB333 | 10D3EFCCEC | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 164 | 125 | 019991E | 59B6DAC0 | 01E196666 | 21A7DF99D9 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 165 | 126 | 033323C | 336DB580 | 03C32CCCD | 434FBF33B3 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 166 | 127 | 0666478 | 66DB6B01 | 07865999A | 069F7E6766 | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 167 | 128 | 0CCC8F0 | 4DB6D603 | 0F0CB3334 | 0D3EFCCECD | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 168 | 129 | 19991E1 | 1B6DAC07 | 1E1966669 | 1A7DF99D9B | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |
| 169 | 130 | 13323C3 | 36DB580E | 1C32CCCD3 | 34FBF33B37 | 0 | 1 | 1 | 1 | 1 | 10 | 00 | 01 |
| 170 | 131 | 0664786 | 6DB6B01C | 1865999A7 | 69F7E6766F | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 171 | 132 | 0CC8F0D | 5B6D6039 | 10CB3334F | 53EFCCECDF | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 172 | 133 | 1991E1A | 36DAC073 | 01966669E | 27DF99D9BF | 1 | 1 | 0 | 1 | 1 | 01 | 00 | 00 |
| 173 | 134 | 1323C35 | 6DB580E6 | 032CCCD3C | 4FBF33B37E | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 174 | 135 | 064786A | 5B6B01CD | 065999A78 | 1F7E6766FC | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 175 | 136 | 0C8F0D5 | 36D6039B | 0CB3334F0 | 3EFCCECDF9 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 176 | 137 | 191E1AA | 6DAC0737 | 1966669E1 | 7DF99D9BF3 | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 177 | 138 | 123C354 | 5B580E6E | 12CCCD3C3 | 7BF33B37E7 | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 178 | 139 | 04786A9 | 36B01CDC | 05999A787 | 77E6766FCE | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 179 | 140 | 08F0D53 | 6D6039B8 | 0B3334F0E | 6FCCECDF9C | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 180 | 141 | 11E1AA6 | 5AC07370 | 166669E1D | 5F99D9BF38 | 0 | 1 | 0 | 1 | 1 | 10 | 11 | 01 |
| 181 | 142 | 03C354C | 3580E6E0 | 0CCCD3C3A | 3F33B37E70 | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 182 | 143 | 0786A99 | 6B01CDC0 | 1999A7875 | 7E6766FCE1 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 183 | 144 | 0F0D533 | 56039B81 | 13334F0EB | 7CCECDF9C2 | 1 | 0 | 0 | 1 | 0 | 01 | 10 | 10 |
| 184 | 145 | 1E1AA66 | 2C073703 | 06669E1D6 | 799D9BF385 | 1 | 0 | 0 | 1 | 1 | 01 | 01 | 10 |
| 185 | 146 | 1C354CC | 580E6E06 | 0CCD3C3AC | 733B37E70B | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 186 | 147 | 186A998 | 301CDC0C | 199A78759 | 66766FCE17 | 1 | 0 | 1 | 0 | 1 | 10 | 11 | 01 |
| 187 | 148 | 10D5331 | 6039B818 | 1334F0EB2 | 4CECDF9C2F | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 188 | 149 | 01AA662 | 40737031 | 0669E1D65 | 19D9BF385E | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 189 | 150 | 0354CC5 | 00E6E063 | 0CD3C3ACB | 33B37E70BD | 0 | 1 | 1 | 1 | 0 | 00 | 01 | 01 |
| 190 | 151 | 06A998A | 01CDC0C6 | 19A787596 | 6766FCE17B | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |
| 191 | 152 | 0D53315 | 039B818C | 134F0EB2C | 4ECDF9C2F6 | 1 | 1 | 0 | 1 | 1 | 00 | 10 | 00 |
| 192 | 153 | 1AA662A | 07370318 | 069E1D659 | 1D9BF385ED | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 193 | 154 | 154CC54 | 0E6E0630 | 0D3C3ACB3 | 3B37E70BDB | 0 | 0 | 1 | 0 | 1 | 00 | 00 | 00 |
| 194 | 155 | 0A998A8 | 1CDC0C60 | 1A7875967 | 766FCE17B6 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 195 | 156 | 1533151 | 39B818C0 | 14F0EB2CE | 6CDF9C2F6C | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 196 | 157 | 0A662A3 | 73703180 | 09E1D659D | 59BF385ED8 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 197 | 158 | 14CC547 | 66E06301 | 13C3ACB3A | 337E70BDB0 | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 198 | 159 | 0998A8E | 4DC0C602 | 078759675 | 66FCE17B61 | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 199 | 160 | 133151D | 1B818C05 | 0F0EB2CEB | 4DF9C2F6C2 | 0 | 1 | 1 | 1 | 0 | 01 | 01 | 11 |
| 200 | 161 | 0662A3B | 3703180B | 1E1D659D6 | 1BF385ED85 | 0 | 0 | 1 | 1 | 1 | 11 | 01 | 01 |
| 201 | 162 | 0CC5477 | 6E063017 | 1C3ACB3AC | 37E70BDB0B | 1 | 0 | 1 | 1 | 0 | 11 | 11 | 01 |
| 202 | 163 | 198A8EF | 5C0C602F | 187596759 | 6FCE17B617 | 1 | 0 | 1 | 1 | 0 | 10 | 11 | 11 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 203 | 164 | 13151DE | 3818C05F | 10EB2CEB2 | 5F9C2F6C2F | 0 | 0 | 0 | 1 | 1 | 01 | 10 | 11 |
| 204 | 165 | 062A3BC | 703180BF | 01D659D65 | 3F385ED85E | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 205 | 166 | 0C54779 | 6063017E | 03ACB3ACB | 7E70BDB0BD | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 206 | 167 | 18A8EF2 | 40C602FD | 075967597 | 7CE17B617B | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 207 | 168 | 1151DE4 | 018C05FA | 0EB2CEB2F | 79C2F6C2F7 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 208 | 169 | 02A3BC9 | 03180BF5 | 1D659D65E | 7385ED85EE | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 209 | 170 | 0547793 | 063017EB | 1ACB3ACBC | 670BDB0BDC | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 210 | 171 | 0A8EF27 | 0C602FD6 | 159675978 | 4E17B617B9 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 211 | 172 | 151DE4E | 18C05FAD | 0B2CEB2F1 | 1C2F6C2F73 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 212 | 173 | 0A3BC9C | 3180BF5A | 1659D65E3 | 385ED85EE6 | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 213 | 174 | 1477938 | 63017EB5 | 0CB3ACBC6 | 70BDB0BDCC | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 214 | 175 | 08EF270 | 4602FD6A | 19675978D | 617B617B99 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 215 | 176 | 11DE4E1 | 0C05FAD5 | 12CEB2F1A | 42F6C2F733 | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 216 | 177 | 03BC9C3 | 180BF5AA | 059D65E34 | 05ED85EE67 | 0 | 0 | 0 | 1 | 0 | 00 | 11 | 10 |
| 217 | 178 | 0779387 | 3017EB55 | 0B3ACBC68 | 0BDB0BDCCF | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 218 | 179 | 0EF270F | 602FD6AA | 1675978D0 | 17B617B99F | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 219 | 180 | 1DE4E1F | 405FAD54 | 0CEB2F1A1 | 2F6C2F733F | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 220 | 181 | 1BC9C3F | 00BF5AA9 | 19D65E342 | 5ED85EE67F | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 221 | 182 | 179387F | 017EB552 | 13ACBC684 | 3DB0BDCCFE | 0 | 0 | 0 | 1 | 1 | 10 | 10 | 10 |
| 222 | 183 | 0F270FF | 02FD6AA5 | 075978D09 | 7B617B99FC | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 223 | 184 | 1E4E1FF | 05FAD54A | 0EB2F1A12 | 76C2F733F9 | 1 | 1 | 1 | 1 | 1 | 10 | 01 | 10 |
| 224 | 185 | 1C9C3FE | 0BF5AA94 | 1D65E3425 | 6D85EE67F2 | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 225 | 186 | 19387FD | 17EB5529 | 1ACBC684B | 5B0BDCCFE4 | 1 | 1 | 1 | 0 | 1 | 01 | 10 | 10 |
| 226 | 187 | 1270FFA | 2FD6AA53 | 15978D096 | 3617B99FC9 | 0 | 1 | 0 | 0 | 0 | 01 | 01 | 10 |
| 227 | 188 | 04E1FF5 | 5FAD54A7 | 0B2F1A12C | 6C2F733F93 | 0 | 1 | 1 | 0 | 1 | 11 | 01 | 01 |
| 228 | 189 | 09C3FEB | 3F5AA94E | 165E34258 | 585EE67F27 | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 229 | 190 | 1387FD7 | 7EB5529C | 0CBC684B1 | 30BDCCFE4F | 0 | 1 | 1 | 1 | 1 | 10 | 10 | 11 |
| 230 | 191 | 070FFAE | 7D6AA538 | 1978D0962 | 617B99FC9E | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 231 | 192 | 0E1FF5C | 7AD54A70 | 12F1A12C4 | 42F733F93D | 1 | 1 | 0 | 1 | 1 | 01 | 10 | 10 |
| 232 | 193 | 1C3FEB9 | 75AA94E1 | 05E342588 | 05EE67F27A | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 233 | 194 | 187FD73 | 6B5529C3 | 0BC684B10 | 0BDCCFE4F4 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 234 | 195 | 10FFAE6 | 56AA5386 | 178D09621 | 17B99FC9E8 | 0 | 1 | 0 | 1 | 1 | 00 | 11 | 10 |
| 235 | 196 | 01FF5CC | 2D54A70C | 0F1A12C43 | 2F733F93D0 | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |
| 236 | 197 | 03FEB98 | 5AA94E19 | 1E3425887 | 5EE67F27A1 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 237 | 198 | 07FD731 | 35529C33 | 1C684B10F | 3DCCFE4F42 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 238 | 199 | 0FFAE63 | 6AA53866 | 18D09621F | 7B99FC9E84 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 239 | 200 | 1FF5CC6 | 554A70CD | 11A12C43F | 7733F93D09 | 1 | 0 | 0 | 0 | 1 | 11 | 10 | 00 |

```
Z[0]  = 59
Z[1]  = 3B
Z[2]  = EF
Z[3]  = 07
Z[4]  = 13
Z[5]  = 70
Z[6]  = 9B
Z[7]  = B7
Z[8]  = 52
Z[9]  = 8F
Z[10] = 3E
Z[11] = B9
Z[12] = A5
Z[13] = AC
Z[14] = EA
Z[15] = 9E
```



LifeScan, Inc. Ex. 1019
Page 942 of 1420
IPR2025-00103

*Sample Data*

```
================================================================================================
         Reload this pattern into the LFSRs
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 1521359
LFSR2  <= 528F703B
LFSR3  <= 0AC3E9BEF
LFSR4  <= 4FEAB9B707
C[t+1] <= 00

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240     1    1521359  528F703B  0AC3E9BEF  4FEAB9B707      0  1  1  1    1  00      10   00
241     2    0A426B3  251EE076  1587D37DE  1FD5736E0F      1  0  0  1    0  00      00   10
242     3    1484D67  4A3DC0ED  0B0FA6FBD  3FAAE6DC1E      0  0  1  1    0  01      00   00
243     4    0909ACF  147B81DA  161F4DF7A  7F55CDB83D      1  0  0  0    0  00      01   00
244     5    121359E  28F703B5  0C3E9BEF5  7EAB9B707B      0  1  1  1    1  10      00   01
245     6    0426B3C  51EE076B  187D37DEB  7D5736E0F6      0  1  1  0    0  00      10   00
246     7    084D679  23DC0ED6  10FA6FBD7  7AAE6DC1EC      1  1  0  1    1  00      00   10
247     8    109ACF2  47B81DAC  01F4DF7AF  755CDB83D8      0  1  0  0    1  00      00   00
248     9    01359E4  0F703B59  03E9BEF5E  6AB9B707B1      0  0  0  1    1  00      00   00
249    10    026B3C8  1EE076B3  07D37DEBD  55736E0F63      0  1  0  0    1  00      00   00
250    11    04D6791  3DC0ED67  0FA6FBD7A  2AE6DC1EC7      0  1  1  1    1  01      00   00
251    12    09ACF22  7B81DACF  1F4DF7AF4  55CDB83D8F      1  1  1  1    1  11      01   00
252    13    1359E44  7703B59E  1E9BEF5E8  2B9B707B1F      0  0  1  1    1  10      11   01
253    14    06B3C88  6E076B3C  1D37DEBD0  5736E0F63F      0  0  1  0    1  01      10   11
254    15    0D67911  5C0ED678  1A6FBD7A1  2E6DC1EC7E      1  0  1  0    1  01      01   10
255    16    1ACF223  381DACF0  14DF7AF42  5CDB83D8FD      1  0  0  1    1  11      01   01
256    17    159E446  703B59E0  09BEF5E85  39B707B1FA      0  0  1  1    1  10      11   01
257    18    0B3C88C  6076B3C0  137DEBD0A  736E0F63F4      1  0  0  0    1  01      10   11
258    19    1679118  40ED6780  06FBD7A15  66DC1EC7E8      0  1  0  1    1  01      01   10
259    20    0CF2231  01DACF00  0DF7AF42A  4DB83D8FD1      1  1  1  1    1  00      01   01
260    21    19E4463  03B59E01  1BEF5E854  1B707B1FA3      1  1  1  0    1  10      00   01
261    22    13C88C6  076B3C03  17DEBD0A9  36E0F63F47      0  0  0  1    1  11      10   00
262    23    079118C  0ED67807  0FBD7A152  6DC1EC7E8E      0  1  1  1    0  01      11   10
263    24    0F22318  1DACF00E  1F7AF42A4  5B83D8FD1D      1  1  1  1    1  01      01   11
264    25    1E44630  3B59E01C  1EF5E8548  3707B1FA3B      1  0  1  0    1  11      01   01
265    26    1C88C61  76B3C039  1DEBD0A91  6E0F63F477      1  1  1  0    0  11      11   01
266    27    19118C3  6D678073  1BD7A1523  5C1EC7E8EF      1  0  1  0    1  11      11   11
267    28    1223187  5ACF00E6  17AF42A46  383D8FD1DE      0  1  0  0    0  11      11   11
268    29    044630E  359E01CC  0F5E8548D  707B1FA3BD      0  1  1  0    1  11      11   11
269    30    088C61C  6B3C0399  1EBD0A91A  60F63F477B      1  0  1  1    0  10      11   11
270    31    1118C39  56780733  1D7A15234  41EC7E8EF6      0  0  1  1    0  10      10   11
271    32    0231872  2CF00E67  1AF42A468  03D8FD1DEC      0  1  1  1    1  01      10   10
272    33    04630E5  59E01CCE  15E8548D1  07B1FA3BD8      0  1  0  1    1  01      01   10
273    34    08C61CB  33C0399D  0BD0A91A3  0F63F477B1      1  1  1  0    0  00      01   01
274    35    118C396  6780733A  17A152347  1EC7E8EF63      0  1  0  1    0  10      00   01
275    36    031872D  4F00E674  0F42A468E  3D8FD1DEC7      0  0  1  1    0  00      10   00
276    37    0630E5A  1E01CCE8  1E8548D1D  7B1FA3BD8E      0  0  1  0    1  01      00   10
277    38    0C61CB5  3C0399D0  1D0A91A3B  763F477B1C      1  0  1  0    1  00      01   00
278    39    18C396A  780733A0  1A1523477  6C7E8EF639      1  0  1  0    0  10      00   01
279    40    11872D5  700E6741  142A468EF  58FD1DEC72      0  0  0  1    1  11      10   00
280    41    030E5AB  601CCE83  08548D1DF  31FA3BD8E5      0  0  1  1    1  00      11   10
281    42    061CB57  40399D07  10A91A3BF  63F477B1CB      0  0  0  1    1  10      00   11
282    43    0C396AF  00733A0F  01523477E  47E8EF6396      1  0  0  1    0  00      10   00
283    44    1872D5F  00E6741F  02A468EFD  0FD1DEC72C      1  1  0  1    1  00      00   10
```



LifeScan, Inc. Ex. 1019
Page 943 of 1420
IPR2025-00103

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 284 | 45 | 10E5ABE | 01CCE83F | 0548D1DFA | 1FA3BD8E58 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 285 | 46 | 01CB57C | 0399D07F | 0A91A3BF4 | 3F477B1CB0 | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 286 | 47 | 0396AF9 | 0733A0FE | 1523477E9 | 7E8EF63961 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 287 | 48 | 072D5F3 | 0E6741FD | 0A468EFD2 | 7D1DEC72C3 | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 288 | 49 | 0E5ABE7 | 1CCE83FA | 148D1DFA4 | 7A3BD8E587 | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 289 | 50 | 1CB57CE | 399D07F4 | 091A3BF49 | 7477B1CB0F | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 290 | 51 | 196AF9D | 733A0FE9 | 123477E92 | 68EF63961E | 1 | 0 | 0 | 1 | 1 | 00 | 11 | 10 |
| 291 | 52 | 12D5F3B | 66741FD2 | 0468EFD25 | 51DEC72C3C | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 292 | 53 | 05ABE77 | 4CE83FA4 | 08D1DFA4B | 23BD8E5879 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 293 | 54 | 0B57CEE | 19D07F49 | 11A3BF496 | 477B1CB0F2 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 294 | 55 | 16AF9DC | 33A0FE92 | 03477E92C | 0EF63961E4 | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 295 | 56 | 0D5F3B8 | 6741FD25 | 068EFD259 | 1DEC72C3C9 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 296 | 57 | 1ABE771 | 4E83FA4B | 0D1DFA4B3 | 3BD8E58793 | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 297 | 58 | 157CEE2 | 1D07F496 | 1A3BF4967 | 77B1CB0F26 | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 298 | 59 | 0AF9DC5 | 3A0FE92D | 1477E92CE | 6F63961E4D | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 299 | 60 | 15F3B8B | 741FD25A | 08EFD259C | 5EC72C3C9B | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 300 | 61 | 0BE7716 | 683FA4B4 | 11DFA4B39 | 3D8E587937 | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 301 | 62 | 17CEE2D | 507F4968 | 03BF49672 | 7B1CB0F26E | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 302 | 63 | 0F9DC5B | 20FE92D0 | 077E92CE4 | 763961E4DC | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 303 | 64 | 1F3B8B6 | 41FD25A0 | 0EFD259C9 | 6C72C3C9B9 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 304 | 65 | 1E7716D | 03FA4B40 | 1DFA4B393 | 58E5879373 | 1 | 1 | 1 | 1 | 1 | 11 | 01 | 00 |
| 305 | 66 | 1CEE2DB | 07F49680 | 1BF496727 | 31CB0F26E6 | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 306 | 67 | 19DC5B7 | 0FE92D00 | 17E92CE4E | 63961E4DCD | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 307 | 68 | 13B8B6F | 1FD25A00 | 0FD259C9C | 472C3C9B9A | 0 | 1 | 1 | 0 | 0 | 10 | 10 | 11 |
| 308 | 69 | 07716DF | 3FA4B400 | 1FA4B3938 | 0E58793735 | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 309 | 70 | 0EE2DBF | 7F496800 | 1F4967271 | 1CB0F26E6A | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 310 | 71 | 1DC5B7F | 7E92D000 | 1E92CE4E2 | 3961E4DCD4 | 1 | 1 | 1 | 0 | 1 | 11 | 10 | 01 |
| 311 | 72 | 1B8B6FF | 7D25A001 | 1D259C9C4 | 72C3C9B9A9 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 312 | 73 | 1716DFF | 7A4B4002 | 1A4B39389 | 6587937352 | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 313 | 74 | 0E2DBFF | 74968005 | 149672713 | 4B0F26E6A5 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 314 | 75 | 1C5B7FE | 692D000B | 092CE4E26 | 161E4DCD4B | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 315 | 76 | 18B6FFC | 525A0017 | 1259C9C4D | 2C3C9B9A96 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 316 | 77 | 116DFF8 | 24B4002F | 04B39389B | 587937352C | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 317 | 78 | 02DBFF1 | 4968005F | 096727136 | 30F26E6A58 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 318 | 79 | 05B7FE3 | 12D000BF | 12CE4E26C | 61E4DCD4B1 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 319 | 80 | 0B6FFC7 | 25A0017F | 059C9C4D8 | 43C9B9A963 | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 320 | 81 | 16DFF8E | 4B4002FF | 0B39389B1 | 07937352C6 | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 321 | 82 | 0DBFF1C | 168005FF | 167271363 | 0F26E6A58C | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 322 | 83 | 1B7FE38 | 2D000BFF | 0CE4E26C7 | 1E4DCD4B18 | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 323 | 84 | 16FFC70 | 5A0017FF | 19C9C4D8F | 3C9B9A9631 | 0 | 0 | 1 | 1 | 0 | 11 | 10 | 01 |
| 324 | 85 | 0DFF8E1 | 34002FFF | 139389B1E | 7937352C62 | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 325 | 86 | 1BFF1C3 | 68005FFF | 07271363D | 726E6A58C4 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 326 | 87 | 17FE387 | 5000BFFE | 0E4E26C7B | 64DCD4B188 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 327 | 88 | 0FFC70F | 20017FFD | 1C9C4D8F6 | 49B9A96311 | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |
| 328 | 89 | 1FF8E1F | 4002FFFB | 19389B1ED | 137352C623 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 329 | 90 | 1FF1C3F | 0005FFF7 | 1271363DB | 26E6A58C46 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 330 | 91 | 1FE387F | 000BFFEE | 04E26C7B6 | 4DCD4B188C | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 331 | 92 | 1FC70FF | 0017FFDC | 09C4D8F6D | 1B9A963118 | 1 | 0 | 1 | 1 | 1 | 00 | 10 | 00 |
| 332 | 93 | 1F8E1FF | 002FFFB8 | 1389B1EDA | 37352C6231 | 1 | 0 | 0 | 0 | 1 | 01 | 00 | 10 |
| 333 | 94 | 1F1C3FF | 005FFF70 | 071363DB4 | 6E6A58C462 | 1 | 0 | 0 | 0 | 0 | 00 | 01 | 00 |
| 334 | 95 | 1E387FE | 00BFFEE0 | 0E26C7B68 | 5CD4B188C5 | 1 | 1 | 1 | 1 | 0 | 01 | 00 | 01 |
| 335 | 96 | 1C70FFC | 017FFDC1 | 1C4D8F6D1 | 39A963118A | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 336 | 97 | 18E1FF9 | 02FFFB82 | 189B1EDA2 | 7352C62315 | 1 | 1 | 1 | 0 | 0 | 11 | 11 | 01 |
| 337 | 98 | 11C3FF2 | 05FFF705 | 11363DB45 | 66A58C462B | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 338 | 99 | 0387FE4 | 0BFFEE0A | 026C7B68B | 4D4B188C56 | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 339 | 100 | 070FFC9 | 17FFDC15 | 04D8F6D16 | 1A963118AD | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 340 | 101 | 0E1FF92 | 2FFFB82B | 09B1EDA2C | 352C62315A | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |



LifeScan, Inc. Ex. 1019
Page 944 of 1420
IPR2025-00103

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 341 | 102 | 1C3FF24 | 5FFF7057 | 1363DB458 | 6A58C462B4 | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 342 | 103 | 187FE48 | 3FFEE0AE | 06C7B68B0 | 54B188C569 | 1 | 1 | 0 | 1 | 1 | 01 | 10 | 10 |
| 343 | 104 | 10FFC90 | 7FFDC15C | 0D8F6D161 | 2963118AD2 | 0 | 1 | 1 | 0 | 1 | 01 | 01 | 10 |
| 344 | 105 | 01FF920 | 7FFB82B9 | 1B1EDA2C2 | 52C62315A5 | 0 | 1 | 1 | 1 | 0 | 00 | 01 | 01 |
| 345 | 106 | 03FF240 | 7FF70573 | 163DB4584 | 258C462B4B | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 346 | 107 | 07FE481 | 7FEE0AE6 | 0C7B68B08 | 4B188C5696 | 0 | 1 | 1 | 0 | 0 | 00 | 10 | 00 |
| 347 | 108 | 0FFC902 | 7FDC15CD | 18F6D1610 | 163118AD2D | 1 | 1 | 1 | 0 | 1 | 00 | 00 | 10 |
| 348 | 109 | 1FF9204 | 7FB82B9A | 11EDA2C20 | 2C62315A5B | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 349 | 110 | 1FF2408 | 7F705735 | 03DB45841 | 58C462B4B6 | 1 | 0 | 0 | 1 | 1 | 00 | 01 | 00 |
| 350 | 111 | 1FE4810 | 7EE0AE6B | 07B68B082 | 3188C5696C | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 351 | 112 | 1FC9021 | 7DC15CD6 | 0F6D16105 | 63118AD2D8 | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 352 | 113 | 1F92042 | 7B82B9AD | 1EDA2C20B | 462315A5B0 | 1 | 1 | 1 | 0 | 1 | 00 | 00 | 10 |
| 353 | 114 | 1F24084 | 7705735A | 1DB458416 | 0C462B4B61 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 354 | 115 | 1E48108 | 6E0AE6B5 | 1B68B082C | 188C5696C3 | 1 | 0 | 1 | 1 | 0 | 11 | 01 | 00 |
| 355 | 116 | 1C90211 | 5C15CD6A | 16D161059 | 3118AD2D86 | 1 | 0 | 0 | 0 | 0 | 10 | 11 | 01 |
| 356 | 117 | 1920422 | 382B9AD5 | 0DA2C20B3 | 62315A5B0D | 1 | 0 | 1 | 0 | 0 | 10 | 10 | 11 |
| 357 | 118 | 1240845 | 705735AA | 1B4584167 | 4462B4B61A | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 358 | 119 | 048108A | 60AE6B55 | 168B082CF | 08C5696C34 | 0 | 1 | 0 | 1 | 0 | 01 | 10 | 10 |
| 359 | 120 | 0902114 | 415CD6AB | 0D161059E | 118AD2D869 | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 360 | 121 | 1204228 | 02B9AD56 | 1A2C20B3D | 2315A5B0D2 | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 361 | 122 | 0408451 | 05735AAD | 14584167B | 462B4B61A4 | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 362 | 123 | 08108A2 | 0AE6B55B | 08B082CF7 | 0C5696C348 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 363 | 124 | 1021144 | 15CD6AB6 | 1161059EF | 18AD2D8690 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 364 | 125 | 0042289 | 2B9AD56C | 02C20B3DE | 315A5B0D20 | 0 | 1 | 0 | 0 | 1 | 10 | 10 | 10 |



LifeScan, Inc. Ex. 1019
Page 945 of 1420
IPR2025-00103

## 1.5 FOURTH SET OF SAMPLES

```
Initial values for the key, pan address and clock

K'c4[0]   = 21   K'c4[1]   = 87    K'c4[2]   = F0    K'c4[3]  = 4A
K'c4[4]   = BA   K'c4[5]   = 90    K'c4[6]   = 31    K'c4[7]  = D0
K'c4[8]   = 78   K'c4[9]   = 0D    K'c4[10]  = 4C    K'c4[11] = 53
K'c4[12]  = E0   K'c4[13]  = 15    K'c4[14]  = 3A    K'c4[15] = 63

Addr4[0] = 2C   Addr4[1] = 7F    Addr4[2] = 94
Addr4[3] = 56   Addr4[4] = 0F    Addr4[5] = 1B

CL4[0]  = 5F   CL4[1]  = 1A    CL4[2]  = 00    CL4[3]  = 02
```

```
================================================================================================
         Fill LFSRs with initial data
================================================================================================

  t  clk#    LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

  0     0    0000000* 00000000* 000000000* 0000000000*     0  0  0  0    0  00     00   00
  1     1    0000000* 00000001* 000000001* 0000000001*     0  0  0  0    0  00     00   00
  2     2    0000001* 00000002* 000000002* 0000000003*     0  0  0  0    0  00     00   00
  3     3    0000002* 00000004* 000000004* 0000000007*     0  0  0  0    0  00     00   00
  4     4    0000004* 00000009* 000000008* 000000000F*     0  0  0  0    0  00     00   00
  5     5    0000008* 00000013* 000000010* 000000001E*     0  0  0  0    0  00     00   00
  6     6    0000010* 00000027* 000000021* 000000003D*     0  0  0  0    0  00     00   00
  7     7    0000021* 0000004F* 000000043* 000000007A*     0  0  0  0    0  00     00   00
  8     8    0000042* 0000009F* 000000087* 00000000F4*     0  0  0  0    0  00     00   00
  9     9    0000084* 0000013F* 00000010F* 00000001E9*     0  0  0  0    0  00     00   00
 10    10    0000108* 0000027F* 00000021F* 00000003D2*     0  0  0  0    0  00     00   00
 11    11    0000211* 000004FE* 00000043E* 00000007A5*     0  0  0  0    0  00     00   00
 12    12    0000422* 000009FC* 00000087C* 0000000F4A*     0  0  0  0    0  00     00   00
 13    13    0000845* 000013F8* 0000010F8* 0000001E94*     0  0  0  0    0  00     00   00
 14    14    000108B* 000027F0* 0000021F1* 0000003D29*     0  0  0  0    0  00     00   00
 15    15    0002117* 00004FE1* 0000043E3* 0000007A52*     0  0  0  0    0  00     00   00
 16    16    000422E* 00009FC2* 0000087C6* 000000F4A4*     0  0  0  0    0  00     00   00
 17    17    000845D* 00013F84* 000010F8C* 000001E948*     0  0  0  0    0  00     00   00
 18    18    00108BA* 00027F08* 000021F18* 000003D290*     0  0  0  0    0  00     00   00
 19    19    0021174* 0004FE10* 000043E30* 000007A520*     0  0  0  0    0  00     00   00
 20    20    00422E8* 0009FC21* 000087C61* 00000F4A41*     0  0  0  0    0  00     00   00
 21    21    00845D1* 0013F842* 00010F8C3* 00001E9482*     0  0  0  0    0  00     00   00
 22    22    0108BA3* 0027F084* 00021F186* 00003D2905*     0  0  0  0    0  00     00   00
 23    23    0211747* 004FE109* 00043E30C* 00007A520B*     0  0  0  0    0  00     00   00
 24    24    0422E8F* 009FC213* 00087C619* 0000F4A417*     0  1  0  0    1  00     00   00
 25    25    0845D1E* 013F8426* 0010F8C32* 0001E9482F*     1  0  0  0    1  00     00   00
 26    26    108BA3D  027F084D* 0021F1864* 0003D2905E*     0  0  0  0    0  00     00   00
 27    27    011747B  04FE109B* 0043E30C9* 0007A520BC*     0  1  0  0    1  00     00   00
 28    28    022E8F6  09FC2136* 0087C6192* 000F4A4179*     0  1  0  0    1  00     00   00
 29    29    045D1EC  13F8426C* 010F8C325* 001E9482F2*     0  1  0  0    1  00     00   00
 30    30    08BA3D9  27F084D8* 021F1864B* 003D2905E5*     1  1  0  0    0  01     00   00
 31    31    11747B3  4FE109B0* 043E30C97* 007A520BCA*     0  1  0  0    1  00     00   00
 32    32    02E8F67  1FC21360  087C6192E* 00F4A41795*     0  1  1  1    1  01     00   00
 33    33    05D1ECF  3F8426C1  10F8C325C* 01E9482F2B*     0  1  0  1    0  01     00   00
 34    34    0BA3D9F  7F084D82  01F1864B8  03D2905E56*     1  0  0  1    0  01     00   00
```

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 35 | 35 | 1747B3E | 7E109B04 | 03E30C970 | 07A520BCAC* | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 36 | 36 | 0E8F67C | 7C213608 | 07C6192E1 | 0F4A417958* | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 37 | 37 | 1D1ECF8 | 78426C11 | 0F8C325C3 | 1E9482F2B1* | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 38 | 38 | 1A3D9F0 | 7084D822 | 1F1864B86 | 3D2905E563* | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 39 | 39 | 147B3E1 | 6109B044 | 1E30C970C | 7A520BCAC6* | 0 | 0 | 1 | 0 | 1 | 00 | 00 | 00 |

Start clocking Summation Combiner

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40 | 1 | 08F67C2 | 42136088 | 1C6192E18 | 74A417958D | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 41 | 2 | 11ECF84 | 0426C111 | 18C325C30 | 69482F2B1B | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 42 | 3 | 03D9F08 | 084D8222 | 11864B861 | 52905E5637 | 0 | 0 | 0 | 1 | 1 | 11 | 00 | 01 |
| 43 | 4 | 07B3E10 | 109B0444 | 030C970C3 | 2520BCAC6E | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 44 | 5 | 0F67C21 | 21360889 | 06192E186 | 4A417958DC | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 45 | 6 | 1ECF843 | 426C1112 | 0C325C30C | 1482F2B1B8 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 46 | 7 | 1D9F086 | 04D82225 | 1864B8619 | 2905E56370 | 1 | 1 | 1 | 0 | 0 | 01 | 11 | 10 |
| 47 | 8 | 1B3E10D | 09B0444B | 10C970C32 | 520BCAC6E1 | 1 | 1 | 0 | 0 | 1 | 10 | 01 | 11 |
| 48 | 9 | 167C21B | 13608897 | 0192E1865 | 2417958DC3 | 0 | 0 | 0 | 0 | 0 | 00 | 10 | 01 |
| 49 | 10 | 0CF8436 | 26C1112F | 0325C30CB | 482F2B1B87 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 50 | 11 | 19F086D | 4D82225E | 064B86197 | 105E56370F | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 51 | 12 | 13E10DB | 1B0444BC | 0C970C32F | 20BCAC6E1F | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 52 | 13 | 07C21B7 | 36088979 | 192E1865E | 417958DC3F | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 53 | 14 | 0F8436E | 6C1112F2 | 125C30CBD | 02F2B1B87F | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 54 | 15 | 1F086DD | 582225E4 | 04B86197B | 05E56370FF | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 55 | 16 | 1E10DBA | 30444BC9 | 0970C32F7 | 0BCAC6E1FF | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 56 | 17 | 1C21B75 | 60889793 | 12E1865EE | 17958DC3FF | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 57 | 18 | 18436EA | 41112F27 | 05C30CBDD | 2F2B1B87FF | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 58 | 19 | 1086DD4 | 02225E4E | 0B86197BA | 5E56370FFF | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 59 | 20 | 010DBA8 | 0444BC9D | 170C32F74 | 3CAC6E1FFF | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 60 | 21 | 021B750 | 0889793A | 0E1865EE8 | 7958DC3FFF | 0 | 1 | 1 | 0 | 1 | 00 | 01 | 00 |
| 61 | 22 | 0436EA0 | 1112F274 | 1C30CBDD0 | 72B1B87FFE | 0 | 0 | 1 | 1 | 0 | 10 | 00 | 01 |
| 62 | 23 | 086DD40 | 2225E4E9 | 186197BA1 | 656370FFFC | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 63 | 24 | 10DBA81 | 444BC9D3 | 10C32F743 | 4AC6E1FFF8 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 64 | 25 | 01B7502 | 089793A7 | 01865EE86 | 158DC3FFF1 | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 65 | 26 | 036EA05 | 112F274E | 030CBDD0D | 2B1B87FFE3 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 66 | 27 | 06DD40B | 225E4E9C | 06197BA1A | 56370FFFC6 | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 67 | 28 | 0DBA817 | 44BC9D39 | 0C32F7434 | 2C6E1FFF8D | 1 | 1 | 1 | 0 | 1 | 10 | 10 | 11 |
| 68 | 29 | 1B7502E | 09793A72 | 1865EE868 | 58DC3FFF1B | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 69 | 30 | 16EA05D | 12F274E5 | 10CBDD0D0 | 31B87FFE36 | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 70 | 31 | 0DD40BA | 25E4E9CB | 0197BA1A1 | 6370FFFC6D | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 71 | 32 | 1BA8174 | 4BC9D397 | 032F74343 | 46E1FFF8DA | 1 | 1 | 0 | 1 | 0 | 11 | 11 | 01 |
| 72 | 33 | 17502E8 | 1793A72F | 065EE8687 | 0DC3FFF1B4 | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 73 | 34 | 0EA05D0 | 2F274E5E | 0CBDD0D0F | 1B87FFE369 | 1 | 0 | 1 | 1 | 0 | 10 | 11 | 11 |
| 74 | 35 | 1D40BA0 | 5E4E9CBD | 197BA1A1F | 370FFFC6D2 | 1 | 0 | 1 | 0 | 0 | 10 | 10 | 11 |
| 75 | 36 | 1A81741 | 3C9D397B | 12F74343F | 6E1FFF8DA5 | 1 | 1 | 0 | 0 | 0 | 01 | 10 | 10 |
| 76 | 37 | 1502E82 | 793A72F6 | 05EE8687F | 5C3FFF1B4B | 0 | 0 | 0 | 0 | 1 | 00 | 01 | 10 |
| 77 | 38 | 0A05D05 | 7274E5ED | 0BDD0D0FF | 387FFE3696 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 78 | 39 | 140BA0B | 64E9CBDA | 17BA1A1FF | 70FFFC6D2C | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 79 | 40 | 0817416 | 49D397B4 | 0F74343FE | 61FFF8DA59 | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 80 | 41 | 102E82C | 13A72F69 | 1EE8687FD | 43FFF1B4B3 | 0 | 1 | 1 | 1 | 0 | 00 | 11 | 00 |
| 81 | 42 | 005D058 | 274E5ED2 | 1DD0D0FFA | 07FFE36966 | 0 | 0 | 1 | 1 | 0 | 11 | 00 | 11 |
| 82 | 43 | 00BA0B0 | 4E9CBDA5 | 1BA1A1FF5 | 0FFFC6D2CD | 0 | 1 | 1 | 1 | 0 | 00 | 11 | 00 |
| 83 | 44 | 0174160 | 1D397B4A | 174343FEA | 1FFF8DA59B | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 84 | 45 | 02E82C0 | 3A72F695 | 0E8687FD4 | 3FFF1B4B37 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 85 | 46 | 05D0580 | 74E5ED2B | 1D0D0FFA9 | 7FFE36966E | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 86 | 47 | 0BA0B00 | 69CBDA56 | 1A1A1FF53 | 7FFC6D2CDC | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 87 | 48 | 1741600 | 5397B4AC | 14343FEA6 | 7FF8DA59B8 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 88 | 49 | 0E82C01 | 272F6959 | 08687FD4D | 7FF1B4B370 | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 89 | 50 | 1D05802 | 4E5ED2B3 | 10D0FFA9A | 7FE36966E0 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 90 | 51 | 1A0B004 | 1CBDA566 | 01A1FF535 | 7FC6D2CDC0 | 1 | 1 | 0 | 1 | 0 | 11 | 01 | 00 |
| 91 | 52 | 1416009 | 397B4ACC | 0343FEA6B | 7F8DA59B80 | 0 | 0 | 0 | 1 | 0 | 10 | 11 | 01 |
| 92 | 53 | 082C013 | 72F69599 | 0687FD4D7 | 7F1B4B3701 | 1 | 1 | 0 | 0 | 0 | 10 | 10 | 11 |
| 93 | 54 | 1058026 | 65ED2B33 | 0D0FFA9AF | 7E36966E03 | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 94 | 55 | 00B004D | 4BDA5667 | 1A1FF535E | 7C6D2CDC06 | 0 | 1 | 1 | 0 | 1 | 01 | 01 | 10 |
| 95 | 56 | 016009B | 17B4ACCE | 143FEA6BD | 78DA59B80D | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 96 | 57 | 02C0137 | 2F69599D | 087FD4D7B | 71B4B3701A | 0 | 0 | 1 | 1 | 1 | 10 | 11 | 01 |
| 97 | 58 | 058026F | 5ED2B33B | 10FFA9AF6 | 636966E034 | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 98 | 59 | 0B004DF | 3DA56677 | 01FF535ED | 46D2CDC068 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 99 | 60 | 16009BF | 7B4ACCEF | 03FEA6BDB | 0DA59B80D0 | 0 | 0 | 0 | 1 | 1 | 00 | 10 | 01 |
| 100 | 61 | 0C0137F | 769599DF | 07FD4D7B7 | 1B4B3701A1 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 101 | 62 | 18026FE | 6D2B33BE | 0FFA9AF6E | 36966E0342 | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 102 | 63 | 1004DFC | 5A56677D | 1FF535EDD | 6D2CDC0684 | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 103 | 64 | 0009BF9 | 34ACCEFB | 1FEA6BDBB | 5A59B80D09 | 0 | 1 | 1 | 0 | 0 | 10 | 00 | 01 |
| 104 | 65 | 00137F2 | 69599DF7 | 1FD4D7B76 | 34B3701A12 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 105 | 66 | 0026FE5 | 52B33BEF | 1FA9AF6EC | 6966E03424 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 106 | 67 | 004DFCA | 256677DF | 1F535EDD8 | 52CDC06848 | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 107 | 68 | 009BF94 | 4ACCEFBE | 1EA6BDBB0 | 259B80D091 | 0 | 1 | 1 | 1 | 0 | 11 | 01 | 00 |
| 108 | 69 | 0137F29 | 1599DF7C | 1D4D7B760 | 4B3701A123 | 0 | 1 | 1 | 0 | 1 | 10 | 11 | 01 |
| 109 | 70 | 026FE53 | 2B33BEF9 | 1A9AF6EC0 | 166E034246 | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 110 | 71 | 04DFCA7 | 56677DF2 | 1535EDD81 | 2CDC06848D | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 111 | 72 | 09BF94F | 2CCEFBE4 | 0A6BDBB03 | 59B80D091B | 1 | 1 | 1 | 1 | 1 | 00 | 01 | 01 |
| 112 | 73 | 137F29E | 599DF7C9 | 14D7B7607 | 33701A1236 | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 113 | 74 | 06FE53C | 333BEF93 | 09AF6EC0E | 66E034246C | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 114 | 75 | 0DFCA79 | 6677DF26 | 135EDD81D | 4DC06848D8 | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 115 | 76 | 1BF94F2 | 4CEFBE4D | 06BDBB03B | 1B80D091B1 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 116 | 77 | 17F29E5 | 19DF7C9A | 0D7B76077 | 3701A12363 | 0 | 1 | 1 | 0 | 1 | 00 | 11 | 10 |
| 117 | 78 | 0FE53CA | 33BEF934 | 1AF6EC0EF | 6E034246C6 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 118 | 79 | 1FCA794 | 677DF269 | 15EDD81DF | 5C06848D8C | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 119 | 80 | 1F94F29 | 4EFBE4D2 | 0BDBB03BE | 380D091B19 | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 120 | 81 | 1F29E53 | 1DF7C9A5 | 17B76077D | 701A123633 | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 121 | 82 | 1E53CA6 | 3BEF934B | 0F6EC0EFB | 6034246C66 | 1 | 1 | 1 | 0 | 0 | 11 | 11 | 01 |
| 122 | 83 | 1CA794D | 77DF2696 | 1EDD81DF6 | 406848D8CD | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 123 | 84 | 194F29B | 6FBE4D2C | 1DBB03BED | 00D091B19B | 1 | 1 | 1 | 1 | 0 | 11 | 10 | 11 |
| 124 | 85 | 129E536 | 5F7C9A59 | 1B76077DA | 01A1236337 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 125 | 86 | 053CA6C | 3EF934B3 | 16EC0EFB4 | 034246C66E | 0 | 1 | 0 | 0 | 1 | 10 | 00 | 11 |
| 126 | 87 | 0A794D9 | 7DF26967 | 0DD81DF69 | 06848D8CDD | 1 | 1 | 1 | 1 | 0 | 01 | 10 | 00 |
| 127 | 88 | 14F29B3 | 7BE4D2CF | 1BB03BED3 | 0D091B19BB | 0 | 1 | 1 | 0 | 1 | 01 | 01 | 10 |
| 128 | 89 | 09E5366 | 77C9A59F | 176077DA6 | 1A12363377 | 1 | 1 | 0 | 0 | 1 | 11 | 01 | 01 |
| 129 | 90 | 13CA6CD | 6F934B3F | 0EC0EFB4D | 34246C66EF | 0 | 1 | 1 | 0 | 1 | 10 | 11 | 01 |
| 130 | 91 | 0794D9B | 5F26967F | 1D81DF69A | 6848D8CDDF | 0 | 0 | 1 | 0 | 1 | 01 | 10 | 11 |
| 131 | 92 | 0F29B37 | 3E4D2CFE | 1B03BED35 | 5091B19BBE | 1 | 0 | 1 | 1 | 0 | 10 | 01 | 10 |
| 132 | 93 | 1E5366F | 7C9A59FD | 16077DA6B | 212363377C | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 133 | 94 | 1CA6CDF | 7934B3FB | 0C0EFB4D6 | 4246C66EF9 | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 134 | 95 | 194D9BE | 726967F6 | 181DF69AD | 048D8CDDF2 | 1 | 0 | 1 | 1 | 1 | 11 | 00 | 11 |
| 135 | 96 | 129B37D | 64D2CFED | 103BED35B | 091B19BBE5 | 0 | 1 | 0 | 0 | 0 | 01 | 11 | 00 |
| 136 | 97 | 05366FA | 49A59FDA | 0077DA6B7 | 12363377CA | 0 | 1 | 0 | 0 | 0 | 10 | 01 | 11 |
| 137 | 98 | 0A6CDF5 | 134B3FB4 | 00EFB4D6E | 246C66EF95 | 1 | 0 | 0 | 0 | 1 | 00 | 10 | 01 |
| 138 | 99 | 14D9BEA | 26967F69 | 01DF69ADD | 48D8CDDF2B | 0 | 1 | 0 | 1 | 0 | 00 | 00 | 10 |
| 139 | 100 | 09B37D4 | 4D2CFED2 | 03BED35BB | 11B19BBE56 | 1 | 0 | 0 | 1 | 0 | 01 | 00 | 00 |
| 140 | 101 | 1366FA8 | 1A59FDA5 | 077DA6B77 | 2363377CAC | 0 | 0 | 0 | 0 | 1 | 01 | 01 | 00 |
| 141 | 102 | 06CDF51 | 34B3FB4A | 0EFB4D6EF | 46C66EF959 | 0 | 1 | 1 | 1 | 0 | 00 | 01 | 01 |
| 142 | 103 | 0D9BEA2 | 6967F695 | 1DF69ADDF | 0D8CDDF2B2 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 143 | 104 | 1B37D45 | 52CFED2A | 1BED35BBF | 1B19BBE564 | 1 | 1 | 1 | 0 | 1 | 00 | 10 | 00 |
| 144 | 105 | 166FA8A | 259FDA54 | 17DA6B77E | 363377CAC8 | 0 | 1 | 0 | 0 | 1 | 01 | 00 | 10 |
| 145 | 106 | 0CDF515 | 4B3FB4A9 | 0FB4D6EFC | 6C66EF9591 | 1 | 0 | 1 | 0 | 1 | 00 | 01 | 00 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 146 | 107 | 19BEA2B | 167F6952 | 1F69ADDF8 | 58CDDF2B22 | 1 | 0 | 1 | 1 | 1 | 10 | 00 | 01 |
| 147 | 108 | 137D457 | 2CFED2A5 | 1ED35BBF1 | 319BBE5645 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 148 | 109 | 06FA8AF | 59FDA54A | 1DA6B77E2 | 63377CAC8B | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 149 | 110 | 0DF515F | 33FB4A95 | 1B4D6EFC4 | 466EF95916 | 1 | 1 | 1 | 0 | 1 | 01 | 00 | 00 |
| 150 | 111 | 1BEA2BF | 67F6952A | 169ADDF88 | 0CDDF2B22C | 1 | 1 | 0 | 1 | 0 | 11 | 01 | 00 |
| 151 | 112 | 17D457F | 4FED2A55 | 0D35BBF10 | 19BBE56459 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 152 | 113 | 0FA8AFE | 1FDA54AB | 1A6B77E20 | 3377CAC8B3 | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 153 | 114 | 1F515FD | 3FB4A957 | 14D6EFC40 | 66EF959166 | 1 | 1 | 0 | 1 | 1 | 10 | 10 | 11 |
| 154 | 115 | 1EA2BFA | 7F6952AF | 09ADDF880 | 4DDF2B22CC | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 155 | 116 | 1D457F4 | 7ED2A55F | 135BBF100 | 1BBE564598 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 156 | 117 | 1A8AFE8 | 7DA54ABF | 06B77E200 | 377CAC8B31 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 157 | 118 | 1515FD0 | 7B4A957F | 0D6EFC401 | 6EF9591663 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 158 | 119 | 0A2BFA1 | 76952AFE | 1ADDF8803 | 5DF2B22CC7 | 1 | 1 | 1 | 1 | 0 | 00 | 00 | 11 |
| 159 | 120 | 1457F42 | 6D2A55FD | 15BBF1007 | 3BE564598E | 0 | 0 | 0 | 1 | 1 | 00 | 00 | 00 |
| 160 | 121 | 08AFE84 | 5A54ABFB | 0B77E200F | 77CAC8B31C | 1 | 0 | 1 | 1 | 1 | 01 | 00 | 00 |
| 161 | 122 | 115FD09 | 34A957F7 | 16EFC401F | 6F95916639 | 0 | 1 | 0 | 1 | 1 | 00 | 01 | 00 |
| 162 | 123 | 02BFA12 | 6952AFEF | 0DDF8803E | 5F2B22CC73 | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 163 | 124 | 057F424 | 52A55FDF | 1BBF1007D | 3E564598E7 | 0 | 1 | 1 | 0 | 1 | 01 | 11 | 00 |
| 164 | 125 | 0AFE848 | 254ABFBF | 177E200FA | 7CAC8B31CF | 1 | 0 | 0 | 1 | 1 | 10 | 01 | 11 |
| 165 | 126 | 15FD090 | 4A957F7E | 0EFC401F5 | 795916639E | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 166 | 127 | 0BFA121 | 152AFEFD | 1DF8803EA | 72B22CC73C | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 167 | 128 | 17F4243 | 2A55FDFA | 1BF1007D4 | 6564598E78 | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 168 | 129 | 0FE8486 | 54ABFBF4 | 17E200FA8 | 4AC8B31CF0 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 169 | 130 | 1FD090C | 2957F7E8 | 0FC401F51 | 15916639E1 | 1 | 0 | 1 | 1 | 0 | 01 | 11 | 10 |
| 170 | 131 | 1FA1219 | 52AFEFD1 | 1F8803EA3 | 2B22CC73C2 | 1 | 1 | 1 | 0 | 0 | 01 | 01 | 11 |
| 171 | 132 | 1F42432 | 255FDFA2 | 1F1007D47 | 564598E785 | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 172 | 133 | 1E84865 | 4ABFBF44 | 1E200FA8F | 2C8B31CF0B | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 173 | 134 | 1D090CB | 157F7E88 | 1C401F51E | 5916639E17 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 174 | 135 | 1A12196 | 2AFEFD11 | 18803EA3C | 322CC73C2E | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 175 | 136 | 142432C | 55FDFA23 | 11007D479 | 64598E785C | 0 | 1 | 0 | 0 | 1 | 01 | 10 | 11 |
| 176 | 137 | 0848659 | 2BFBF446 | 0200FA8F2 | 48B31CF0B9 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 177 | 138 | 1090CB2 | 57F7E88C | 0401F51E4 | 116639E173 | 0 | 1 | 0 | 0 | 1 | 00 | 10 | 01 |
| 178 | 139 | 0121964 | 2FEFD118 | 0803EA3C8 | 22CC73C2E6 | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 179 | 140 | 02432C9 | 5FDFA230 | 1007D4791 | 4598E785CD | 0 | 1 | 0 | 1 | 0 | 01 | 00 | 00 |
| 180 | 141 | 0486593 | 3FBF4461 | 000FA8F23 | 0B31CF0B9B | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 181 | 142 | 090CB26 | 7F7E88C3 | 001F51E47 | 16639E1736 | 1 | 0 | 0 | 0 | 1 | 11 | 00 | 01 |
| 182 | 143 | 121964D | 7EFD1187 | 003EA3C8F | 2CC73C2E6C | 0 | 1 | 0 | 1 | 1 | 01 | 11 | 00 |
| 183 | 144 | 0432C9B | 7DFA230E | 007D4791E | 598E785CD8 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 184 | 145 | 0865936 | 7BF4461C | 00FA8F23C | 331CF0B9B0 | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 185 | 146 | 10CB26D | 77E88C38 | 01F51E479 | 6639E17361 | 0 | 1 | 0 | 0 | 0 | 00 | 11 | 10 |
| 186 | 147 | 01964DA | 6FD11870 | 03EA3C8F2 | 4C73C2E6C2 | 0 | 1 | 0 | 0 | 1 | 10 | 00 | 11 |
| 187 | 148 | 032C9B4 | 5FA230E1 | 07D4791E4 | 18E785CD84 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 188 | 149 | 0659368 | 3F4461C2 | 0FA8F23C9 | 31CF0B9B09 | 0 | 0 | 1 | 1 | 0 | 00 | 00 | 10 |
| 189 | 150 | 0CB26D0 | 7E88C384 | 1F51E4793 | 639E173612 | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 190 | 151 | 1964DA0 | 7D118709 | 1EA3C8F27 | 473C2E6C24 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 191 | 152 | 12C9B41 | 7A230E12 | 1D4791E4E | 0E785CD848 | 0 | 0 | 1 | 0 | 1 | 01 | 00 | 10 |
| 192 | 153 | 0593683 | 74461C24 | 1A8F23C9C | 1CF0B9B091 | 0 | 0 | 1 | 1 | 1 | 00 | 01 | 00 |
| 193 | 154 | 0B26D06 | 688C3848 | 151E47938 | 39E1736123 | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 194 | 155 | 164DA0D | 51187091 | 0A3C8F271 | 73C2E6C247 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 195 | 156 | 0C9B41A | 2230E123 | 14791E4E3 | 6785CD848F | 1 | 0 | 0 | 1 | 0 | 00 | 00 | 10 |
| 196 | 157 | 1936835 | 4461C247 | 08F23C9C6 | 4F0B9B091E | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 197 | 158 | 126D06A | 08C3848E | 11E47938D | 1E1736123C | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 198 | 159 | 04DA0D5 | 1187091C | 03C8F271B | 3C2E6C2478 | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 199 | 160 | 09B41AA | 230E1238 | 0791E4E37 | 785CD848F1 | 1 | 0 | 0 | 0 | 0 | 01 | 11 | 00 |
| 200 | 161 | 1368354 | 461C2470 | 0F23C9C6F | 70B9B091E3 | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 201 | 162 | 06D06A9 | 0C3848E1 | 1E47938DF | 61736123C6 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 202 | 163 | 0DA0D52 | 187091C3 | 1C8F271BE | 42E6C2478D | 1 | 0 | 1 | 1 | 1 | 00 | 00 | 10 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 203 | 164 | 1B41AA4 | 30E12387 | 191E4E37C | 05CD848F1A | 1 | 1 | 1 | 1 | 0 | 10 | 00 | 00 |
| 204 | 165 | 1683549 | 61C2470F | 123C9C6F9 | 0B9B091E34 | 0 | 1 | 0 | 1 | 0 | 00 | 10 | 00 |
| 205 | 166 | 0D06A92 | 43848E1E | 047938DF3 | 1736123C68 | 1 | 1 | 0 | 0 | 0 | 00 | 00 | 10 |
| 206 | 167 | 1A0D524 | 07091C3C | 08F271BE7 | 2E6C2478D1 | 1 | 0 | 1 | 0 | 0 | 01 | 00 | 00 |
| 207 | 168 | 141AA49 | 0E123879 | 11E4E37CF | 5CD848F1A2 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 208 | 169 | 0835492 | 1C2470F3 | 03C9C6F9F | 39B091E345 | 1 | 0 | 0 | 1 | 0 | 10 | 00 | 01 |
| 209 | 170 | 106A925 | 3848E1E6 | 07938DF3F | 736123C68B | 0 | 0 | 0 | 0 | 0 | 11 | 10 | 00 |
| 210 | 171 | 00D524A | 7091C3CD | 0F271BE7E | 66C2478D16 | 0 | 1 | 1 | 1 | 0 | 01 | 11 | 10 |
| 211 | 172 | 01AA495 | 6123879B | 1E4E37CFD | 4D848F1A2D | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 212 | 173 | 035492A | 42470F36 | 1C9C6F9FB | 1B091E345B | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 213 | 174 | 06A9255 | 048E1E6C | 1938DF3F6 | 36123C68B7 | 0 | 1 | 1 | 0 | 0 | 00 | 00 | 10 |
| 214 | 175 | 0D524AB | 091C3CD8 | 1271BE7EC | 6C2478D16E | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 215 | 176 | 1AA4957 | 123879B1 | 04E37CFD8 | 5848F1A2DD | 1 | 0 | 0 | 0 | 1 | 00 | 00 | 00 |
| 216 | 177 | 15492AF | 2470F363 | 09C6F9FB0 | 3091E345BA | 0 | 0 | 1 | 1 | 0 | 01 | 00 | 00 |
| 217 | 178 | 0A9255E | 48E1E6C7 | 138DF3F61 | 6123C68B75 | 1 | 1 | 0 | 0 | 1 | 00 | 01 | 00 |
| 218 | 179 | 1524ABD | 11C3CD8F | 071BE7EC3 | 42478D16EB | 0 | 1 | 0 | 0 | 1 | 11 | 00 | 01 |
| 219 | 180 | 0A4957B | 23879B1F | 0E37CFD87 | 048F1A2DD6 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 220 | 181 | 1492AF6 | 470F363F | 1C6F9FB0E | 091E345BAD | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |
| 221 | 182 | 09255EC | 0E1E6C7F | 18DF3F61D | 123C68B75B | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 222 | 183 | 124ABD9 | 1C3CD8FF | 11BE7EC3A | 2478D16EB6 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 223 | 184 | 04957B3 | 3879B1FE | 037CFD874 | 48F1A2DD6D | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 224 | 185 | 092AF66 | 70F363FD | 06F9FB0E9 | 11E345BADB | 1 | 1 | 0 | 1 | 1 | 10 | 00 | 01 |
| 225 | 186 | 1255ECD | 61E6C7FA | 0DF3F61D3 | 23C68B75B7 | 0 | 1 | 1 | 1 | 1 | 00 | 10 | 00 |
| 226 | 187 | 04ABD9B | 43CD8FF5 | 1BE7EC3A7 | 478D16EB6E | 0 | 1 | 1 | 1 | 1 | 00 | 00 | 10 |
| 227 | 188 | 0957B37 | 079B1FEA | 17CFD874E | 0F1A2DD6DD | 1 | 1 | 0 | 0 | 0 | 01 | 00 | 00 |
| 228 | 189 | 12AF66F | 0F363FD4 | 0F9FB0E9C | 1E345BADBB | 0 | 0 | 1 | 0 | 0 | 00 | 01 | 00 |
| 229 | 190 | 055ECDE | 1E6C7FA9 | 1F3F61D39 | 3C68B75B76 | 0 | 0 | 1 | 0 | 1 | 11 | 00 | 01 |
| 230 | 191 | 0ABD9BC | 3CD8FF53 | 1E7EC3A73 | 78D16EB6EC | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 231 | 192 | 157B379 | 79B1FEA7 | 1CFD874E6 | 71A2DD6DD9 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 232 | 193 | 0AF66F3 | 7363FD4E | 19FB0E9CD | 6345BADBB2 | 1 | 0 | 1 | 0 | 1 | 01 | 11 | 00 |
| 233 | 194 | 15ECDE6 | 66C7FA9D | 13F61D39A | 468B75B765 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 234 | 195 | 0BD9BCC | 4D8FF53A | 07EC3A735 | 0D16EB6ECA | 1 | 1 | 0 | 0 | 0 | 11 | 10 | 01 |
| 235 | 196 | 17B3799 | 1B1FEA75 | 0FD874E6A | 1A2DD6DD94 | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 236 | 197 | 0F66F33 | 363FD4EA | 1FB0E9CD5 | 345BADBB28 | 1 | 0 | 1 | 0 | 0 | 11 | 00 | 11 |
| 237 | 198 | 1ECDE67 | 6C7FA9D5 | 1F61D39AA | 68B75B7650 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 238 | 199 | 1D9BCCF | 58FF53AB | 1EC3A7354 | 516EB6ECA0 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 239 | 200 | 1B3799E | 31FEA756 | 1D874E6A8 | 22DD6DD940 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |

```
Z[0]  = 3F
Z[1]  = B1
Z[2]  = 67
Z[3]  = D2
Z[4]  = 2F
Z[5]  = A6
Z[6]  = 1F
Z[7]  = B9
Z[8]  = E6
Z[9]  = 84
Z[10] = 43
Z[11] = 07
Z[12] = D8
Z[13] = 1E
Z[14] = E7
Z[15] = C3
```



LifeScan, Inc. Ex. 1019
Page 950 of 1420
IPR2025-00103

*Sample Data*

```
================================================================================================
         Reload this pattern into the LFSRs
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 0E62F3F
LFSR2  <= 6C84A6B1
LFSR3  <= 11E431F67
LFSR4  <= 61E707B9D2
C[t+1] <= 00

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
```

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 240 | 1 | 0E62F3F | 6C84A6B1 | 11E431F67 | 61E707B9D2 | 1 | 1 | 0 | 1 | 0 | 00 | 11 | 00 |
| 241 | 2 | 1CC5E7F | 59094D63 | 03C863ECE | 43CE0F73A5 | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 242 | 3 | 198BCFF | 32129AC6 | 0790C7D9D | 079C1EE74A | 1 | 0 | 0 | 1 | 1 | 01 | 11 | 00 |
| 243 | 4 | 13179FE | 6425358C | 0F218FB3A | 0F383DCE94 | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 244 | 5 | 062F3FD | 484A6B19 | 1E431F675 | 1E707B9D28 | 0 | 0 | 1 | 0 | 1 | 00 | 10 | 01 |
| 245 | 6 | 0C5E7FB | 1094D632 | 1C863ECEB | 3CE0F73A50 | 1 | 1 | 1 | 1 | 0 | 11 | 00 | 10 |
| 246 | 7 | 18BCFF7 | 2129AC64 | 190C7D9D7 | 79C1EE74A1 | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 247 | 8 | 1179FEE | 425358C8 | 1218FB3AE | 7383DCE942 | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 248 | 9 | 02F3FDD | 04A6B190 | 0431F675D | 6707B9D285 | 0 | 1 | 0 | 0 | 1 | 11 | 10 | 00 |
| 249 | 10 | 05E7FBB | 094D6320 | 0863ECEBB | 4E0F73A50B | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 250 | 11 | 0BCFF77 | 129AC640 | 10C7D9D77 | 1C1EE74A16 | 1 | 1 | 0 | 0 | 0 | 11 | 00 | 11 |
| 251 | 12 | 179FEEE | 25358C80 | 018FB3AEE | 383DCE942C | 0 | 0 | 0 | 0 | 1 | 10 | 11 | 00 |
| 252 | 13 | 0F3FDDC | 4A6B1900 | 031F675DD | 707B9D2859 | 1 | 0 | 0 | 0 | 1 | 01 | 10 | 11 |
| 253 | 14 | 1E7FBB8 | 14D63200 | 063ECEBBA | 60F73A50B3 | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 254 | 15 | 1CFF771 | 29AC6401 | 0C7D9D774 | 41EE74A167 | 1 | 1 | 1 | 1 | 0 | 10 | 10 | 01 |
| 255 | 16 | 19FEEE2 | 5358C803 | 18FB3AEE9 | 03DCE942CE | 1 | 0 | 1 | 1 | 1 | 01 | 10 | 10 |
| 256 | 17 | 13FDDC4 | 26B19007 | 11F675DD2 | 07B9D2859C | 0 | 1 | 0 | 1 | 1 | 01 | 01 | 10 |
| 257 | 18 | 07FBB88 | 4D63200E | 03ECEBBA4 | 0F73A50B38 | 0 | 0 | 0 | 0 | 1 | 10 | 01 | 01 |
| 258 | 19 | 0FF7711 | 1AC6401D | 07D9D7748 | 1EE74A1670 | 1 | 1 | 0 | 1 | 1 | 11 | 10 | 01 |
| 259 | 20 | 1FEEE23 | 358C803B | 0FB3AEE91 | 3DCE942CE1 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 260 | 21 | 1FDDC47 | 6B190076 | 1F675DD23 | 7B9D2859C2 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 261 | 22 | 1FBB88F | 563200ED | 1ECEBBA47 | 773A50B385 | 1 | 0 | 1 | 0 | 1 | 11 | 01 | 01 |
| 262 | 23 | 1F7711E | 2C6401DB | 1D9D7748F | 6E74A1670A | 1 | 0 | 1 | 0 | 1 | 10 | 11 | 01 |
| 263 | 24 | 1EEE23D | 58C803B6 | 1B3AEE91E | 5CE942CE15 | 1 | 1 | 1 | 1 | 0 | 11 | 10 | 11 |
| 264 | 25 | 1DDC47A | 3190076C | 1675DD23D | 39D2859C2B | 1 | 1 | 0 | 1 | 0 | 01 | 11 | 10 |
| 265 | 26 | 1BB88F4 | 63200ED9 | 0CEBBA47A | 73A50B3856 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |
| 266 | 27 | 17711E8 | 46401DB2 | 19D7748F5 | 674A1670AD | 0 | 0 | 1 | 0 | 0 | 11 | 01 | 01 |
| 267 | 28 | 0EE23D0 | 0C803B64 | 13AEE91EA | 4E942CE15B | 1 | 1 | 0 | 1 | 0 | 11 | 11 | 01 |
| 268 | 29 | 1DC47A0 | 190076C8 | 075DD23D4 | 1D2859C2B7 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 269 | 30 | 1B88F41 | 3200ED90 | 0EBBA47A9 | 3A50B3856E | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 270 | 31 | 1711E83 | 6401DB20 | 1D7748F53 | 74A1670ADC | 0 | 0 | 1 | 1 | 1 | 11 | 11 | 11 |
| 271 | 32 | 0E23D07 | 4803B641 | 1AEE91EA7 | 6942CE15B8 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 272 | 33 | 1C47A0F | 10076C82 | 15DD23D4F | 52859C2B71 | 1 | 0 | 0 | 1 | 1 | 11 | 11 | 11 |
| 273 | 34 | 188F41E | 200ED905 | 0BBA47A9E | 250B3856E3 | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 274 | 35 | 111E83C | 401DB20A | 17748F53D | 4A1670ADC7 | 0 | 0 | 0 | 0 | 1 | 00 | 11 | 11 |
| 275 | 36 | 023D078 | 003B6414 | 0EE91EA7A | 142CE15B8E | 0 | 0 | 1 | 0 | 1 | 10 | 00 | 11 |
| 276 | 37 | 047A0F0 | 0076C828 | 1DD23D4F5 | 2859C2B71C | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 277 | 38 | 08F41E1 | 00ED9050 | 1BA47A9EA | 50B3856E39 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 278 | 39 | 11E83C2 | 01DB20A0 | 1748F53D5 | 21670ADC72 | 0 | 1 | 0 | 0 | 0 | 10 | 01 | 11 |
| 279 | 40 | 03D0785 | 03B64141 | 0E91EA7AA | 42CE15B8E4 | 0 | 1 | 1 | 1 | 1 | 11 | 10 | 01 |
| 280 | 41 | 07A0F0A | 076C8283 | 1D23D4F54 | 059C2B71C8 | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 281 | 42 | 0F41E14 | 0ED90507 | 1A47A9EA9 | 0B3856E390 | 1 | 1 | 1 | 0 | 1 | 11 | 00 | 11 |
| 282 | 43 | 1E83C29 | 1DB20A0F | 148F53D52 | 1670ADC720 | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 283 | 44 | 1D07853 | 3B64141E | 091EA7AA5 | 2CE15B8E40 | 1 | 0 | 1 | 1 | 0 | 01 | 01 | 11 |



LifeScan, Inc. Ex. 1019
Page 951 of 1420
IPR2025-00103

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 284 | 45 | 1A0F0A6 | 76C8283C | 123D4F54B | 59C2B71C81 | 1 | 1 | 0 | 1 | 0 | 00 | 01 | 01 |
| 285 | 46 | 141E14C | 6D905079 | 047A9EA97 | 33856E3902 | 0 | 1 | 0 | 1 | 0 | 10 | 00 | 01 |
| 286 | 47 | 083C299 | 5B20A0F2 | 08F53D52F | 670ADC7204 | 1 | 0 | 1 | 0 | 0 | 00 | 10 | 00 |
| 287 | 48 | 1078533 | 364141E4 | 11EA7AA5E | 4E15B8E408 | 0 | 0 | 0 | 0 | 0 | 01 | 00 | 10 |
| 288 | 49 | 00F0A67 | 6C8283C8 | 03D4F54BC | 1C2B71C811 | 0 | 1 | 0 | 0 | 0 | 00 | 01 | 00 |
| 289 | 50 | 01E14CE | 59050791 | 07A9EA978 | 3856E39022 | 0 | 0 | 0 | 0 | 0 | 11 | 00 | 01 |
| 290 | 51 | 03C299C | 320A0F23 | 0F53D52F1 | 70ADC72045 | 0 | 0 | 1 | 1 | 1 | 01 | 11 | 00 |
| 291 | 52 | 0785339 | 64141E47 | 1EA7AA5E2 | 615B8E408A | 0 | 0 | 1 | 0 | 0 | 10 | 01 | 11 |
| 292 | 53 | 0F0A673 | 48283C8E | 1D4F54BC4 | 42B71C8115 | 1 | 0 | 1 | 1 | 1 | 11 | 10 | 01 |
| 293 | 54 | 1E14CE6 | 1050791C | 1A9EA9788 | 056E39022B | 1 | 0 | 1 | 0 | 1 | 00 | 11 | 10 |
| 294 | 55 | 1C299CD | 20A0F239 | 153D52F10 | 0ADC720456 | 1 | 1 | 0 | 1 | 1 | 11 | 00 | 11 |
| 295 | 56 | 185339B | 4141E472 | 0A7AA5E20 | 15B8E408AC | 1 | 0 | 1 | 1 | 0 | 00 | 11 | 00 |
| 296 | 57 | 10A6736 | 0283C8E4 | 14F54BC41 | 2B71C81158 | 0 | 1 | 0 | 0 | 1 | 10 | 00 | 11 |
| 297 | 58 | 014CE6C | 050791C9 | 09EA97882 | 56E39022B0 | 0 | 0 | 1 | 1 | 0 | 00 | 10 | 00 |
| 298 | 59 | 0299CD9 | 0A0F2393 | 13D52F104 | 2DC7204561 | 0 | 0 | 0 | 1 | 1 | 01 | 00 | 10 |
| 299 | 60 | 05339B3 | 141E4726 | 07AA5E208 | 5B8E408AC3 | 0 | 0 | 0 | 1 | 0 | 00 | 01 | 00 |
| 300 | 61 | 0A67366 | 283C8E4C | 0F54BC411 | 371C811587 | 1 | 0 | 1 | 0 | 0 | 10 | 00 | 01 |
| 301 | 62 | 14CE6CC | 50791C98 | 1EA978822 | 6E39022B0F | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 302 | 63 | 099CD99 | 20F23930 | 1D52F1045 | 5C7204561E | 1 | 1 | 1 | 0 | 0 | 01 | 11 | 10 |
| 303 | 64 | 1339B33 | 41E47260 | 1AA5E208B | 38E408AC3D | 0 | 1 | 1 | 1 | 0 | 01 | 01 | 11 |
| 304 | 65 | 0673666 | 03C8E4C0 | 154BC4117 | 71C811587A | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |
| 305 | 66 | 0CE6CCC | 0791C980 | 0A978822E | 639022B0F5 | 1 | 1 | 1 | 1 | 1 | 11 | 11 | 01 |
| 306 | 67 | 19CD999 | 0F239301 | 152F1045C | 47204561EB | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 307 | 68 | 139B332 | 1E472603 | 0A5E208B9 | 0E408AC3D6 | 0 | 0 | 1 | 0 | 0 | 11 | 11 | 11 |
| 308 | 69 | 0736664 | 3C8E4C06 | 14BC41172 | 1C811587AD | 0 | 1 | 0 | 1 | 1 | 11 | 11 | 11 |
| 309 | 70 | 0E6CCC8 | 791C980C | 0978822E5 | 39022B0F5A | 1 | 0 | 1 | 0 | 1 | 11 | 11 | 11 |
| 310 | 71 | 1CD9990 | 72393019 | 12F1045CB | 7204561EB4 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 311 | 72 | 19B3320 | 64726033 | 05E208B97 | 6408AC3D69 | 1 | 0 | 0 | 0 | 0 | 11 | 11 | 11 |
| 312 | 73 | 1366640 | 48E4C067 | 0BC41172F | 4811587AD3 | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 313 | 74 | 06CCC81 | 11C980CF | 178822E5E | 1022B0F5A6 | 0 | 1 | 0 | 0 | 0 | 11 | 11 | 11 |
| 314 | 75 | 0D99903 | 2393019E | 0F1045CBC | 204561EB4C | 1 | 1 | 1 | 0 | 0 | 10 | 11 | 11 |
| 315 | 76 | 1B33206 | 4726033D | 1E208B979 | 408AC3D699 | 1 | 0 | 1 | 1 | 1 | 10 | 10 | 11 |
| 316 | 77 | 166640D | 0E4C067B | 1C41172F2 | 011587AD33 | 0 | 0 | 1 | 0 | 1 | 10 | 10 | 10 |
| 317 | 78 | 0CCC81B | 1C980CF6 | 18822E5E5 | 022B0F5A66 | 1 | 1 | 1 | 0 | 1 | 01 | 10 | 10 |
| 318 | 79 | 1999036 | 393019EC | 11045CBCA | 04561EB4CD | 1 | 0 | 0 | 0 | 0 | 01 | 01 | 10 |
| 319 | 80 | 133206C | 726033D9 | 0208B9794 | 08AC3D699B | 0 | 0 | 0 | 1 | 0 | 11 | 01 | 01 |
| 320 | 81 | 06640D9 | 64C067B3 | 041172F29 | 11587AD337 | 0 | 1 | 0 | 0 | 0 | 10 | 11 | 01 |
| 321 | 82 | 0CC81B3 | 4980CF66 | 0822E5E53 | 22B0F5A66F | 1 | 1 | 1 | 1 | 0 | 11 | 10 | 11 |
| 322 | 83 | 1990366 | 13019ECC | 1045CBCA6 | 4561EB4CDF | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 323 | 84 | 13206CC | 26033D98 | 008B9794D | 0AC3D699BE | 0 | 0 | 0 | 1 | 1 | 10 | 00 | 11 |
| 324 | 85 | 0640D98 | 4C067B31 | 01172F29B | 1587AD337C | 0 | 0 | 0 | 1 | 1 | 11 | 10 | 00 |
| 325 | 86 | 0C81B30 | 180CF662 | 022E5E537 | 2B0F5A66F9 | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 326 | 87 | 1903660 | 3019ECC5 | 045CBCA6F | 561EB4CDF3 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 327 | 88 | 1206CC1 | 6033D98A | 08B9794DE | 2C3D699BE6 | 0 | 0 | 1 | 0 | 1 | 11 | 10 | 00 |
| 328 | 89 | 040D983 | 4067B315 | 1172F29BD | 587AD337CC | 0 | 0 | 0 | 0 | 1 | 11 | 11 | 10 |
| 329 | 90 | 081B306 | 00CF662A | 02E5E537A | 30F5A66F98 | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 330 | 91 | 103660C | 019ECC55 | 05CBCA6F4 | 61EB4CDF31 | 0 | 1 | 0 | 1 | 0 | 10 | 10 | 11 |
| 331 | 92 | 006CC19 | 033D98AB | 0B9794DE8 | 43D699BE62 | 0 | 0 | 1 | 1 | 0 | 01 | 10 | 10 |
| 332 | 93 | 00D9833 | 067B3156 | 172F29BD0 | 07AD337CC5 | 0 | 0 | 0 | 1 | 0 | 01 | 01 | 10 |
| 333 | 94 | 01B3066 | 0CF662AC | 0E5E537A0 | 0F5A66F98B | 0 | 1 | 1 | 0 | 1 | 11 | 01 | 01 |
| 334 | 95 | 03660CD | 19ECC559 | 1CBCA6F41 | 1EB4CDF317 | 0 | 1 | 1 | 1 | 0 | 11 | 11 | 01 |
| 335 | 96 | 06CC19B | 33D98AB2 | 19794DE83 | 3D699BE62F | 0 | 1 | 1 | 0 | 1 | 11 | 11 | 11 |
| 336 | 97 | 0D98336 | 67B31565 | 12F29BD06 | 7AD337CC5F | 1 | 1 | 0 | 1 | 0 | 10 | 11 | 11 |
| 337 | 98 | 1B3066D | 4F662ACA | 05E537A0C | 75A66F98BF | 1 | 0 | 0 | 1 | 0 | 10 | 10 | 11 |
| 338 | 99 | 1660CDB | 1ECC5594 | 0BCA6F418 | 6B4CDF317E | 0 | 1 | 1 | 0 | 0 | 01 | 10 | 10 |
| 339 | 100 | 0CC19B7 | 3D98AB29 | 1794DE831 | 5699BE62FC | 1 | 1 | 0 | 1 | 0 | 10 | 01 | 10 |
| 340 | 101 | 198336F | 7B315653 | 0F29BD062 | 2D337CC5F9 | 1 | 0 | 1 | 0 | 0 | 11 | 10 | 01 |

*Sample Data*

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 341 | 102 | 13066DE | 7662ACA7 | 1E537A0C5 | 5A66F98BF2 | 0 | 0 | 1 | 0 | 0 | 00 | 11 | 10 |
| 342 | 103 | 060CDBC | 6CC5594F | 1CA6F418B | 34CDF317E4 | 0 | 1 | 1 | 1 | 1 | 11 | 00 | 11 |
| 343 | 104 | 0C19B78 | 598AB29F | 194DE8317 | 699BE62FC9 | 1 | 1 | 1 | 1 | 1 | 00 | 11 | 00 |
| 344 | 105 | 18336F1 | 3315653F | 129BD062E | 5337CC5F92 | 1 | 0 | 0 | 0 | 1 | 10 | 00 | 11 |
| 345 | 106 | 1066DE2 | 662ACA7E | 0537A0C5C | 266F98BF25 | 0 | 0 | 0 | 0 | 0 | 11 | 10 | 00 |
| 346 | 107 | 00CDBC5 | 4C5594FD | 0A6F418B9 | 4CDF317E4B | 0 | 0 | 1 | 1 | 1 | 00 | 11 | 10 |
| 347 | 108 | 019B78B | 18AB29FA | 14DE83172 | 19BE62FC96 | 0 | 1 | 0 | 1 | 0 | 11 | 00 | 11 |
| 348 | 109 | 0336F16 | 315653F4 | 09BD062E5 | 337CC5F92C | 0 | 0 | 1 | 0 | 0 | 01 | 11 | 00 |
| 349 | 110 | 066DE2D | 62ACA7E8 | 137A0C5CA | 66F98BF258 | 0 | 1 | 0 | 1 | 1 | 10 | 01 | 11 |
| 350 | 111 | 0CDBC5B | 45594FD1 | 06F418B95 | 4DF317E4B1 | 1 | 0 | 0 | 1 | 0 | 11 | 10 | 01 |
| 351 | 112 | 19B78B6 | 0AB29FA2 | 0DE83172B | 1BE62FC962 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 352 | 113 | 136F16C | 15653F45 | 1BD062E57 | 37CC5F92C5 | 0 | 0 | 1 | 1 | 1 | 10 | 01 | 11 |
| 353 | 114 | 06DE2D9 | 2ACA7E8B | 17A0C5CAE | 6F98BF258B | 0 | 1 | 0 | 1 | 0 | 11 | 10 | 01 |
| 354 | 115 | 0DBC5B2 | 5594FD16 | 0F418B95D | 5F317E4B16 | 1 | 1 | 1 | 0 | 0 | 01 | 11 | 10 |
| 355 | 116 | 1B78B64 | 2B29FA2C | 1E83172BB | 3E62FC962C | 1 | 0 | 1 | 0 | 1 | 10 | 01 | 11 |
| 356 | 117 | 16F16C8 | 5653F458 | 1D062E577 | 7CC5F92C58 | 0 | 0 | 1 | 1 | 0 | 11 | 10 | 01 |
| 357 | 118 | 0DE2D91 | 2CA7E8B0 | 1A0C5CAEF | 798BF258B1 | 1 | 1 | 1 | 1 | 1 | 01 | 11 | 10 |
| 358 | 119 | 1BC5B23 | 594FD161 | 1418B95DF | 7317E4B163 | 1 | 0 | 0 | 0 | 0 | 10 | 01 | 11 |
| 359 | 120 | 178B647 | 329FA2C2 | 083172BBF | 662FC962C7 | 0 | 1 | 1 | 0 | 0 | 11 | 10 | 01 |
| 360 | 121 | 0F16C8E | 653F4584 | 1062E577F | 4C5F92C58E | 1 | 0 | 0 | 0 | 0 | 00 | 11 | 10 |
| 361 | 122 | 1E2D91C | 4A7E8B09 | 00C5CAEFE | 18BF258B1C | 1 | 0 | 0 | 1 | 0 | 11 | 00 | 11 |
| 362 | 123 | 1C5B238 | 14FD1613 | 018B95DFC | 317E4B1639 | 1 | 1 | 0 | 0 | 1 | 01 | 11 | 00 |
| 363 | 124 | 18B6471 | 29FA2C27 | 03172BBF9 | 62FC962C72 | 1 | 1 | 0 | 1 | 0 | 01 | 01 | 11 |
| 364 | 125 | 116C8E2 | 53F4584E | 062E577F3 | 45F92C58E4 | 0 | 1 | 0 | 1 | 1 | 11 | 01 | 01 |



LifeScan, Inc. Ex. 1019
Page 953 of 1420
IPR2025-00103

# 2 FREQUENCY HOPPING SAMPLE DATA

The section contains three sets of sample data showing the basic and adapted hopping schemes for different combinations of addresses and initial clock values.



LifeScan, Inc. Ex. 1019
Page 954 of 1420
IPR2025-00103

## 2.1 FIRST SET

```
Hop sequence {k} for PAGE SCAN/INQUIRY SCAN SUBSTATE:
CLKN start:      0x0000000
UAP / LAP:       0x00000000
#ticks:          0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
                 -----------------------------------------------------
0x0000000:          0 |    2 |    4 |    6 |    8 |   10 |   12 |   14 |
0x0008000:         16 |   18 |   20 |   22 |   24 |   26 |   28 |   30 |
0x0010000:         32 |   34 |   36 |   38 |   40 |   42 |   44 |   46 |
0x0018000:         48 |   50 |   52 |   54 |   56 |   58 |   60 |   62 |
0x0020000:          0 |    2 |    4 |    6 |    8 |   10 |   12 |   14 |
0x0028000:         16 |   18 |   20 |   22 |   24 |   26 |   28 |   30 |
0x0030000:         32 |   34 |   36 |   38 |   40 |   42 |   44 |   46 |
0x0038000:         48 |   50 |   52 |   54 |   56 |   58 |   60 |   62 |

Hop sequence {k} for PAGE STATE/INQUIRY SUBSTATE:
CLKE start:      0x0000000
UAP / LAP:       0x00000000
#ticks:          00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
                 -----------------------------------------------------
0x0000000:       48 50 09 13 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0000010:       00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
0x0000020:       48 50 09 13 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0000030:       00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
...
0x0001000:       48 18 09 05 | 20 22 33 37 | 24 26 03 07 | 28 30 35 39 |
0x0001010:       32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
0x0001020:       48 18 09 05 | 20 22 33 37 | 24 26 03 07 | 28 30 35 39 |
0x0001030:       32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
...
0x0002000:       16 18 01 05 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0002010:       00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
0x0002020:       16 18 01 05 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0002030:       00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
...
0x0003000:       48 50 09 13 | 52 22 41 37 | 24 26 03 07 | 28 30 35 39 |
0x0003010:       32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
0x0003020:       48 50 09 13 | 52 22 41 37 | 24 26 03 07 | 28 30 35 39 |
0x0003030:       32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |

Hop sequence {k} for SLAVE PAGE RESPONSE SUBSTATE:
CLKN* =          0x0000010
UAP / LAP:       0x00000000
#ticks:          00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a 1c | 1e
                 ---------------------------------------------------------------
0x0000012:       64 | 02 68 | 04 17 | 06 21 | 08 66 | 10 70 | 12 19 | 14 23 | 16
0x0000032:       01 | 18 05 | 20 33 | 22 37 | 24 03 | 26 07 | 28 35 | 30 39 | 32
0x0000052:       72 | 34 76 | 36 25 | 38 29 | 40 74 | 42 78 | 44 27 | 46 31 | 48
0x0000072:       09 | 50 13 | 52 41 | 54 45 | 56 11 | 58 15 | 60 43 | 62 47 | 00

Hop sequence {k} for MASTER PAGE RESPONSE SUBSTATE:
Offset value:    24
CLKE* =          0x0000012
UAP / LAP:       0x00000000
#ticks:          00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
                 -----------------------------------------------------------------
0x0000014:       02 68 | 04 17 | 06 21 | 08 66 | 10 70 | 12 19 | 14 23 | 16 01 |
0x0000034:       18 05 | 20 33 | 22 37 | 24 03 | 26 07 | 28 35 | 30 39 | 32 72 |
0x0000054:       34 76 | 36 25 | 38 29 | 40 74 | 42 78 | 44 27 | 46 31 | 48 09 |
0x0000074:       50 13 | 52 41 | 54 45 | 56 11 | 58 15 | 60 43 | 62 47 | 00 64 |
```

*Sample Data*

Hop sequence {k} for CONNECTION STATE (Basic channel hopping sequence; ie, non-AFH):
CLK start: 0x0000010
UAP/LAP: 0x00000000

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010: | 08 66 | 10 70 | 12 19 | 14 23 | 16 01 | 18 05 | 20 33 | 22 37 |
| 0x0000030: | 24 03 | 26 07 | 28 35 | 30 39 | 32 72 | 34 76 | 36 25 | 38 29 |
| 0x0000050: | 40 74 | 42 78 | 44 27 | 46 31 | 48 09 | 50 13 | 52 41 | 54 45 |
| 0x0000070: | 56 11 | 58 15 | 60 43 | 62 47 | 32 17 | 36 19 | 34 49 | 38 51 |
| 0x0000090: | 40 21 | 44 23 | 42 53 | 46 55 | 48 33 | 52 35 | 50 65 | 54 67 |
| 0x00000b0: | 56 37 | 60 39 | 58 69 | 62 71 | 64 25 | 68 27 | 66 57 | 70 59 |
| 0x00000d0: | 72 29 | 76 31 | 74 61 | 78 63 | 01 41 | 05 43 | 03 73 | 07 75 |
| 0x00000f0: | 09 45 | 13 47 | 11 77 | 15 00 | 64 49 | 66 53 | 68 02 | 70 06 |
| 0x0000110: | 01 51 | 03 55 | 05 04 | 07 08 | 72 57 | 74 61 | 76 10 | 78 14 |
| 0x0000130: | 09 59 | 11 63 | 13 12 | 15 16 | 17 65 | 19 69 | 21 18 | 23 22 |
| 0x0000150: | 33 67 | 35 71 | 37 20 | 39 24 | 25 73 | 27 77 | 29 26 | 31 30 |
| 0x0000170: | 41 75 | 43 00 | 45 28 | 47 32 | 17 02 | 21 04 | 19 34 | 23 36 |
| 0x0000190: | 33 06 | 37 08 | 35 38 | 39 40 | 25 10 | 29 12 | 27 42 | 31 44 |
| 0x00001b0: | 41 14 | 45 16 | 43 46 | 47 48 | 49 18 | 53 20 | 51 50 | 55 52 |
| 0x00001d0: | 65 22 | 69 24 | 67 54 | 71 56 | 57 26 | 61 28 | 59 58 | 63 60 |
| 0x00001f0: | 73 30 | 77 32 | 75 62 | 00 64 | 49 34 | 51 42 | 57 66 | 59 74 |
| 0x0000210: | 53 36 | 55 44 | 61 68 | 63 76 | 65 50 | 67 58 | 73 03 | 75 11 |
| 0x0000230: | 69 52 | 71 60 | 77 05 | 00 13 | 02 38 | 04 46 | 10 70 | 12 78 |
| 0x0000250: | 06 40 | 08 48 | 14 72 | 16 01 | 18 54 | 20 62 | 26 07 | 28 15 |
| 0x0000270: | 22 56 | 24 64 | 30 09 | 32 17 | 02 66 | 06 74 | 10 19 | 14 27 |
| 0x0000290: | 04 70 | 08 78 | 12 23 | 16 31 | 18 03 | 22 11 | 26 35 | 30 43 |
| 0x00002b0: | 20 07 | 24 15 | 28 39 | 32 47 | 34 68 | 38 76 | 42 21 | 46 29 |
| 0x00002d0: | 36 72 | 40 01 | 44 25 | 48 33 | 50 05 | 54 13 | 58 37 | 62 45 |
| 0x00002f0: | 52 09 | 56 17 | 60 41 | 64 49 | 34 19 | 36 35 | 50 51 | 52 67 |
| 0x0000310: | 38 21 | 40 37 | 54 53 | 56 69 | 42 27 | 44 43 | 58 59 | 60 75 |
| 0x0000330: | 46 29 | 48 45 | 62 61 | 64 77 | 66 23 | 68 39 | 03 55 | 05 71 |
| 0x0000350: | 70 25 | 72 41 | 07 57 | 09 73 | 74 31 | 76 47 | 11 63 | 13 00 |
| 0x0000370: | 78 33 | 01 49 | 15 65 | 17 02 | 66 51 | 70 67 | 03 04 | 07 20 |
| 0x0000390: | 68 55 | 72 71 | 05 08 | 09 24 | 74 59 | 78 75 | 11 12 | 15 28 |
| 0x00003b0: | 76 63 | 01 00 | 13 16 | 17 32 | 19 53 | 23 69 | 35 06 | 39 22 |
| 0x00003d0: | 21 57 | 25 73 | 37 10 | 41 26 | 27 61 | 31 77 | 43 14 | 47 30 |
| 0x00003f0: | 29 65 | 33 02 | 45 18 | 49 34 | 19 04 | 21 08 | 23 20 | 25 24 |

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with all channel used; ie, AFH(79)):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x7fffffffffffffffff

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 08 08 | 10 10 | 12 12 | 14 14 | 16 16 | 18 18 | 20 20 | 22 22 |
| 0x0000030 | 24 24 | 26 26 | 28 28 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 |
| 0x0000050 | 40 40 | 42 42 | 44 44 | 46 46 | 48 48 | 50 50 | 52 52 | 54 54 |
| 0x0000070 | 56 56 | 58 58 | 60 60 | 62 62 | 32 32 | 36 36 | 34 34 | 38 38 |
| 0x0000090 | 40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
| 0x00000b0 | 56 56 | 60 60 | 58 58 | 62 62 | 64 64 | 68 68 | 66 66 | 70 70 |
| 0x00000d0 | 72 72 | 76 76 | 74 74 | 78 78 | 01 01 | 05 05 | 03 03 | 07 07 |
| 0x00000f0 | 09 09 | 13 13 | 11 11 | 15 15 | 64 64 | 66 66 | 68 68 | 70 70 |
| 0x0000110 | 01 01 | 03 03 | 05 05 | 07 07 | 72 72 | 74 74 | 76 76 | 78 78 |
| 0x0000130 | 09 09 | 11 11 | 13 13 | 15 15 | 17 17 | 19 19 | 21 21 | 23 23 |
| 0x0000150 | 33 33 | 35 35 | 37 37 | 39 39 | 25 25 | 27 27 | 29 29 | 31 31 |
| 0x0000170 | 41 41 | 43 43 | 45 45 | 47 47 | 17 17 | 21 21 | 19 19 | 23 23 |
| 0x0000190 | 33 33 | 37 37 | 35 35 | 39 39 | 25 25 | 29 29 | 27 27 | 31 31 |
| 0x00001b0 | 41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
| 0x00001d0 | 65 65 | 69 69 | 67 67 | 71 71 | 57 57 | 61 61 | 59 59 | 63 63 |
| 0x00001f0 | 73 73 | 77 77 | 75 75 | 00 00 | 49 49 | 51 51 | 57 57 | 59 59 |
| 0x0000210 | 53 53 | 55 55 | 61 61 | 63 63 | 65 65 | 67 67 | 73 73 | 75 75 |
| 0x0000230 | 69 69 | 71 71 | 77 77 | 00 00 | 02 02 | 04 04 | 10 10 | 12 12 |
| 0x0000250 | 06 06 | 08 08 | 14 14 | 16 16 | 18 18 | 20 20 | 26 26 | 28 28 |
| 0x0000270 | 22 22 | 24 24 | 30 30 | 32 32 | 02 02 | 06 06 | 10 10 | 14 14 |
| 0x0000290 | 04 04 | 08 08 | 12 12 | 16 16 | 18 18 | 22 22 | 26 26 | 30 30 |
| 0x00002b0 | 20 20 | 24 24 | 28 28 | 32 32 | 34 34 | 38 38 | 42 42 | 46 46 |
| 0x00002d0 | 36 36 | 40 40 | 44 44 | 48 48 | 50 50 | 54 54 | 58 58 | 62 62 |
| 0x00002f0 | 52 52 | 56 56 | 60 60 | 64 64 | 34 34 | 36 36 | 50 50 | 52 52 |
| 0x0000310 | 38 38 | 40 40 | 54 54 | 56 56 | 42 42 | 44 44 | 58 58 | 60 60 |
| 0x0000330 | 46 46 | 48 48 | 62 62 | 64 64 | 66 66 | 68 68 | 03 03 | 05 05 |
| 0x0000350 | 70 70 | 72 72 | 07 07 | 09 09 | 74 74 | 76 76 | 11 11 | 13 13 |
| 0x0000370 | 78 78 | 01 01 | 15 15 | 17 17 | 66 66 | 70 70 | 03 03 | 07 07 |
| 0x0000390 | 68 68 | 72 72 | 05 05 | 09 09 | 74 74 | 78 78 | 11 11 | 15 15 |
| 0x00003b0 | 76 76 | 01 01 | 13 13 | 17 17 | 19 19 | 23 23 | 35 35 | 39 39 |
| 0x00003d0 | 21 21 | 25 25 | 37 37 | 41 41 | 27 27 | 31 31 | 43 43 | 47 47 |
| 0x00003f0 | 29 29 | 33 33 | 45 45 | 49 49 | 19 19 | 21 21 | 23 23 | 25 25 |



LifeScan, Inc. Ex. 1019
Page 957 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with channels 0 to 21 unused):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x7fffffffffffc00000

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 | 40 40 | 42 42 | 22 22 |
| 0x0000030 | 24 24 | 26 26 | 28 28 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 |
| 0x0000050 | 40 40 | 42 42 | 44 44 | 46 46 | 48 48 | 50 50 | 52 52 | 54 54 |
| 0x0000070 | 56 56 | 58 58 | 60 60 | 62 62 | 32 32 | 36 36 | 34 34 | 38 38 |
| 0x0000090 | 40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
| 0x00000b0 | 56 56 | 60 60 | 58 58 | 62 62 | 64 64 | 68 68 | 66 66 | 70 70 |
| 0x00000d0 | 72 72 | 76 76 | 74 74 | 78 78 | 45 45 | 49 49 | 47 47 | 51 51 |
| 0x00000f0 | 53 53 | 57 57 | 55 55 | 59 59 | 64 64 | 66 66 | 68 68 | 70 70 |
| 0x0000110 | 45 45 | 47 47 | 49 49 | 51 51 | 72 72 | 74 74 | 76 76 | 78 78 |
| 0x0000130 | 53 53 | 55 55 | 57 57 | 59 59 | 61 61 | 63 63 | 65 65 | 23 23 |
| 0x0000150 | 33 33 | 35 35 | 37 37 | 39 39 | 25 25 | 27 27 | 29 29 | 31 31 |
| 0x0000170 | 41 41 | 43 43 | 45 45 | 47 47 | 61 61 | 65 65 | 63 63 | 23 23 |
| 0x0000190 | 33 33 | 37 37 | 35 35 | 39 39 | 25 25 | 29 29 | 27 27 | 31 31 |
| 0x00001b0 | 41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
| 0x00001d0 | 65 65 | 69 69 | 67 67 | 71 71 | 57 57 | 61 61 | 59 59 | 63 63 |
| 0x00001f0 | 73 73 | 77 77 | 75 75 | 66 66 | 49 49 | 51 51 | 57 57 | 59 59 |
| 0x0000210 | 53 53 | 55 55 | 61 61 | 63 63 | 65 65 | 67 67 | 73 73 | 75 75 |
| 0x0000230 | 69 69 | 71 71 | 77 77 | 66 66 | 68 68 | 70 70 | 76 76 | 78 78 |
| 0x0000250 | 72 72 | 74 74 | 23 23 | 25 25 | 27 27 | 29 29 | 26 26 | 28 28 |
| 0x0000270 | 22 22 | 24 24 | 30 30 | 32 32 | 68 68 | 72 72 | 76 76 | 23 23 |
| 0x0000290 | 70 70 | 74 74 | 78 78 | 25 25 | 27 27 | 22 22 | 26 26 | 30 30 |
| 0x00002b0 | 29 29 | 24 24 | 28 28 | 32 32 | 34 34 | 38 38 | 42 42 | 46 46 |
| 0x00002d0 | 36 36 | 40 40 | 44 44 | 48 48 | 50 50 | 54 54 | 58 58 | 62 62 |
| 0x00002f0 | 52 52 | 56 56 | 60 60 | 64 64 | 34 34 | 36 36 | 50 50 | 52 52 |
| 0x0000310 | 38 38 | 40 40 | 54 54 | 56 56 | 42 42 | 44 44 | 58 58 | 60 60 |
| 0x0000330 | 46 46 | 48 48 | 62 62 | 64 64 | 66 66 | 68 68 | 34 34 | 36 36 |
| 0x0000350 | 70 70 | 72 72 | 38 38 | 40 40 | 74 74 | 76 76 | 42 42 | 44 44 |
| 0x0000370 | 78 78 | 32 32 | 46 46 | 48 48 | 66 66 | 70 70 | 34 34 | 38 38 |
| 0x0000390 | 68 68 | 72 72 | 36 36 | 40 40 | 74 74 | 78 78 | 42 42 | 46 46 |
| 0x00003b0 | 76 76 | 32 32 | 44 44 | 48 48 | 50 50 | 23 23 | 35 35 | 39 39 |
| 0x00003d0 | 52 52 | 25 25 | 37 37 | 41 41 | 27 27 | 31 31 | 43 43 | 47 47 |
| 0x00003f0 | 29 29 | 33 33 | 45 45 | 49 49 | 50 50 | 52 52 | 23 23 | 25 25 |

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with even channels used):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x5555555555555555555

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 08 08 | 10 10 | 12 12 | 14 14 | 16 16 | 18 18 | 20 20 | 22 22 |
| 0x0000030 | 24 24 | 26 26 | 28 28 | 30 30 | 32 32 | 34 34 | 36 36 | 38 38 |
| 0x0000050 | 40 40 | 42 42 | 44 44 | 46 46 | 48 48 | 50 50 | 52 52 | 54 54 |
| 0x0000070 | 56 56 | 58 58 | 60 60 | 62 62 | 32 32 | 36 36 | 34 34 | 38 38 |
| 0x0000090 | 40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
| 0x00000b0 | 56 56 | 60 60 | 58 58 | 62 62 | 64 64 | 68 68 | 66 66 | 70 70 |
| 0x00000d0 | 72 72 | 76 76 | 74 74 | 78 78 | 00 00 | 04 04 | 02 02 | 06 06 |
| 0x00000f0 | 08 08 | 12 12 | 10 10 | 14 14 | 64 64 | 66 66 | 68 68 | 70 70 |
| 0x0000110 | 00 00 | 02 02 | 04 04 | 06 06 | 72 72 | 74 74 | 76 76 | 78 78 |
| 0x0000130 | 08 08 | 10 10 | 12 12 | 14 14 | 16 16 | 18 18 | 20 20 | 22 22 |
| 0x0000150 | 32 32 | 34 34 | 36 36 | 38 38 | 24 24 | 26 26 | 28 28 | 30 30 |
| 0x0000170 | 40 40 | 42 42 | 44 44 | 46 46 | 16 16 | 20 20 | 18 18 | 22 22 |
| 0x0000190 | 32 32 | 36 36 | 34 34 | 38 38 | 24 24 | 28 28 | 26 26 | 30 30 |
| 0x00001b0 | 40 40 | 44 44 | 42 42 | 46 46 | 48 48 | 52 52 | 50 50 | 54 54 |
| 0x00001d0 | 64 64 | 68 68 | 66 66 | 70 70 | 56 56 | 60 60 | 58 58 | 62 62 |
| 0x00001f0 | 72 72 | 76 76 | 74 74 | 00 00 | 48 48 | 50 50 | 56 56 | 58 58 |
| 0x0000210 | 52 52 | 54 54 | 60 60 | 62 62 | 64 64 | 66 66 | 72 72 | 74 74 |
| 0x0000230 | 68 68 | 70 70 | 76 76 | 00 00 | 02 02 | 04 04 | 10 10 | 12 12 |
| 0x0000250 | 06 06 | 08 08 | 14 14 | 16 16 | 18 18 | 20 20 | 26 26 | 28 28 |
| 0x0000270 | 22 22 | 24 24 | 30 30 | 32 32 | 02 02 | 06 06 | 10 10 | 14 14 |
| 0x0000290 | 04 04 | 08 08 | 12 12 | 16 16 | 18 18 | 22 22 | 26 26 | 30 30 |
| 0x00002b0 | 20 20 | 24 24 | 28 28 | 32 32 | 34 34 | 38 38 | 42 42 | 46 46 |
| 0x00002d0 | 36 36 | 40 40 | 44 44 | 48 48 | 50 50 | 54 54 | 58 58 | 62 62 |
| 0x00002f0 | 52 52 | 56 56 | 60 60 | 64 64 | 34 34 | 36 36 | 50 50 | 52 52 |
| 0x0000310 | 38 38 | 40 40 | 54 54 | 56 56 | 42 42 | 44 44 | 58 58 | 60 60 |
| 0x0000330 | 46 46 | 48 48 | 62 62 | 64 64 | 66 66 | 68 68 | 00 00 | 02 02 |
| 0x0000350 | 70 70 | 72 72 | 04 04 | 06 06 | 74 74 | 76 76 | 08 08 | 10 10 |
| 0x0000370 | 78 78 | 78 78 | 12 12 | 14 14 | 66 66 | 70 70 | 00 00 | 04 04 |
| 0x0000390 | 68 68 | 72 72 | 02 02 | 06 06 | 74 74 | 78 78 | 08 08 | 12 12 |
| 0x00003b0 | 76 76 | 78 78 | 10 10 | 14 14 | 16 16 | 20 20 | 32 32 | 36 36 |
| 0x00003d0 | 18 18 | 22 22 | 34 34 | 38 38 | 24 24 | 28 28 | 40 40 | 44 44 |
| 0x00003f0 | 26 26 | 30 30 | 42 42 | 46 46 | 16 16 | 18 18 | 20 20 | 22 22 |



LifeScan, Inc. Ex. 1019
Page 959 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with odd channels used):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x2aaaaaaaaaaaaaaaaaaa

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 09 09 | 11 11 | 13 13 | 15 15 | 17 17 | 19 19 | 21 21 | 23 23 |
| 0x0000030 | 25 25 | 27 27 | 29 29 | 31 31 | 33 33 | 35 35 | 37 37 | 39 39 |
| 0x0000050 | 41 41 | 43 43 | 45 45 | 47 47 | 49 49 | 51 51 | 53 53 | 55 55 |
| 0x0000070 | 57 57 | 59 59 | 61 61 | 63 63 | 33 33 | 37 37 | 35 35 | 39 39 |
| 0x0000090 | 41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
| 0x00000b0 | 57 57 | 61 61 | 59 59 | 63 63 | 65 65 | 69 69 | 67 67 | 71 71 |
| 0x00000d0 | 73 73 | 77 77 | 75 75 | 01 01 | 01 01 | 05 05 | 03 03 | 07 07 |
| 0x00000f0 | 09 09 | 13 13 | 11 11 | 15 15 | 65 65 | 67 67 | 69 69 | 71 71 |
| 0x0000110 | 01 01 | 03 03 | 05 05 | 07 07 | 73 73 | 75 75 | 77 77 | 01 01 |
| 0x0000130 | 09 09 | 11 11 | 13 13 | 15 15 | 17 17 | 19 19 | 21 21 | 23 23 |
| 0x0000150 | 33 33 | 35 35 | 37 37 | 39 39 | 25 25 | 27 27 | 29 29 | 31 31 |
| 0x0000170 | 41 41 | 43 43 | 45 45 | 47 47 | 17 17 | 21 21 | 19 19 | 23 23 |
| 0x0000190 | 33 33 | 37 37 | 35 35 | 39 39 | 25 25 | 29 29 | 27 27 | 31 31 |
| 0x00001b0 | 41 41 | 45 45 | 43 43 | 47 47 | 49 49 | 53 53 | 51 51 | 55 55 |
| 0x00001d0 | 65 65 | 69 69 | 67 67 | 71 71 | 57 57 | 61 61 | 59 59 | 63 63 |
| 0x00001f0 | 73 73 | 77 77 | 75 75 | 03 03 | 49 49 | 51 51 | 57 57 | 59 59 |
| 0x0000210 | 53 53 | 55 55 | 61 61 | 63 63 | 65 65 | 67 67 | 73 73 | 75 75 |
| 0x0000230 | 69 69 | 71 71 | 77 77 | 03 03 | 05 05 | 07 07 | 13 13 | 15 15 |
| 0x0000250 | 09 09 | 11 11 | 17 17 | 19 19 | 21 21 | 23 23 | 29 29 | 31 31 |
| 0x0000270 | 25 25 | 27 27 | 33 33 | 35 35 | 05 05 | 09 09 | 13 13 | 17 17 |
| 0x0000290 | 07 07 | 11 11 | 15 15 | 19 19 | 21 21 | 25 25 | 29 29 | 33 33 |
| 0x00002b0 | 23 23 | 27 27 | 31 31 | 35 35 | 37 37 | 41 41 | 45 45 | 49 49 |
| 0x00002d0 | 39 39 | 43 43 | 47 47 | 51 51 | 53 53 | 57 57 | 61 61 | 65 65 |
| 0x00002f0 | 55 55 | 59 59 | 63 63 | 67 67 | 37 37 | 39 39 | 53 53 | 55 55 |
| 0x0000310 | 41 41 | 43 43 | 57 57 | 59 59 | 45 45 | 47 47 | 61 61 | 63 63 |
| 0x0000330 | 49 49 | 51 51 | 65 65 | 67 67 | 69 69 | 71 71 | 03 03 | 05 05 |
| 0x0000350 | 73 73 | 75 75 | 07 07 | 09 09 | 77 77 | 01 01 | 11 11 | 13 13 |
| 0x0000370 | 03 03 | 01 01 | 15 15 | 17 17 | 69 69 | 73 73 | 03 03 | 07 07 |
| 0x0000390 | 71 71 | 75 75 | 05 05 | 09 09 | 77 77 | 03 03 | 11 11 | 15 15 |
| 0x00003b0 | 01 01 | 01 01 | 13 13 | 17 17 | 19 19 | 23 23 | 35 35 | 39 39 |
| 0x00003d0 | 21 21 | 25 25 | 37 37 | 41 41 | 27 27 | 31 31 | 43 43 | 47 47 |
| 0x00003f0 | 29 29 | 33 33 | 45 45 | 49 49 | 19 19 | 21 21 | 23 23 | 25 25 |



LifeScan, Inc. Ex. 1019
Page 960 of 1420
IPR2025-00103

## 2.2 SECOND SET

```
Hop sequence {k} for PAGE SCAN/INQUIRY SCAN SUBSTATE:
CLKN start:      0x0000000
ULAP:            0x2a96ef25
#ticks:          0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
                 ---------------------------------------------------------
0x0000000:         49 |   13 |   17 |   51 |   55 |   19 |   23 |   53 |
0x0008000:         57 |   21 |   25 |   27 |   31 |   74 |   78 |   29 |
0x0010000:         33 |   76 |    1 |   35 |   39 |    3 |    7 |   37 |
0x0018000:         41 |    5 |    9 |   43 |   47 |   11 |   15 |   45 |
0x0020000:         49 |   13 |   17 |   51 |   55 |   19 |   23 |   53 |
0x0028000:         57 |   21 |   25 |   27 |   31 |   74 |   78 |   29 |
0x0030000:         33 |   76 |    1 |   35 |   39 |    3 |    7 |   37 |
0x0038000:         41 |    5 |    9 |   43 |   47 |   11 |   15 |   45 |

Hop sequence {k} for PAGE STATE/INQUIRY SUBSTATE:
CLKE start:      0x0000000
ULAP:            0x2a96ef25
#ticks:          00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
                 -----------------------------------------------------------
0x0000000:       41 05 10 04 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0000010:       49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
0x0000020:       41 05 10 04 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0000030:       49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
...
0x0001000:       41 21 10 36 | 25 27 38 63 | 31 74 65 59 | 78 29 61 00 |
0x0001010:       33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |
0x0001020:       41 21 10 36 | 25 27 38 63 | 31 74 65 59 | 78 29 61 00 |
0x0001030:       33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |
...
0x0002000:       57 21 42 36 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0002010:       49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
0x0002020:       57 21 42 36 | 09 43 06 16 | 47 11 18 12 | 15 45 14 32 |
0x0002030:       49 13 34 28 | 17 51 30 24 | 55 19 26 20 | 23 53 22 40 |
...
0x0003000:       41 05 10 04 | 09 27 06 63 | 31 74 65 59 | 78 29 61 00 |
0x0003010:       33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |
0x0003020:       41 05 10 04 | 09 27 06 63 | 31 74 65 59 | 78 29 61 00 |
0x0003030:       33 76 02 75 | 01 35 77 71 | 39 03 73 67 | 07 37 69 08 |

Hop sequence {k} for SLAVE PAGE RESPONSE SUBSTATE:
CLKN* =          0x0000010
ULAP:            0x2a96ef25
#ticks:          00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a
                 -----------------------------------------------------------
0x0000012:       34 | 13 28 | 17 30 | 51 24 | 55 26 | 19 20 | 23 22 | 53
0x0000032:       42 | 21 36 | 25 38 | 27 63 | 31 65 | 74 59 | 78 61 | 29
0x0000052:       02 | 76 75 | 01 77 | 35 71 | 39 73 | 03 67 | 07 69 | 37
0x0000072:       10 | 05 04 | 09 06 | 43 16 | 47 18 | 11 12 | 15 14 | 45

Hop sequence {k} for MASTER PAGE RESPONSE SUBSTATE:
Offset value:    24
CLKE* =          0x0000012
ULAP:            0x2a96ef25
#ticks:          00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a |
                 -----------------------------------------------------------
0x0000014:       13 28 | 17 30 | 51 24 | 55 26 | 19 20 | 23 22 | 53 40 |
0x0000034:       21 36 | 25 38 | 27 63 | 31 65 | 74 59 | 78 61 | 29 00 |
0x0000054:       76 75 | 01 77 | 35 71 | 39 73 | 03 67 | 07 69 | 37 08 |
0x0000074:       05 04 | 09 06 | 43 16 | 47 18 | 11 12 | 15 14 | 45 32 |
```



LifeScan, Inc. Ex. 1019
Page 961 of 1420
IPR2025-00103

*Sample Data*

Hop sequence {k} for CONNECTION STATE (Basic channel hopping sequence; ie, non-AFH):
CLK start: 0x0000010
ULAP: 0x2a96ef25

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010: | 55 26 | 19 20 | 23 22 | 53 40 | 57 42 | 21 36 | 25 38 | 27 63 |
| 0x0000030: | 31 65 | 74 59 | 78 61 | 29 00 | 33 02 | 76 75 | 01 77 | 35 71 |
| 0x0000050: | 39 73 | 03 67 | 07 69 | 37 08 | 41 10 | 05 04 | 09 06 | 43 16 |
| 0x0000070: | 47 18 | 11 12 | 15 14 | 45 32 | 02 66 | 47 60 | 49 64 | 04 54 |
| 0x0000090: | 06 58 | 51 52 | 53 56 | 08 70 | 10 74 | 55 68 | 57 72 | 59 14 |
| 0x00000b0: | 61 18 | 27 12 | 29 16 | 63 30 | 65 34 | 31 28 | 33 32 | 67 22 |
| 0x00000d0: | 69 26 | 35 20 | 37 24 | 71 38 | 73 42 | 39 36 | 41 40 | 75 46 |
| 0x00000f0: | 77 50 | 43 44 | 45 48 | 00 62 | 26 11 | 69 05 | 73 07 | 36 17 |
| 0x0000110: | 40 19 | 04 13 | 08 15 | 38 25 | 42 27 | 06 21 | 10 23 | 12 48 |
| 0x0000130: | 16 50 | 59 44 | 63 46 | 14 56 | 18 58 | 61 52 | 65 54 | 28 64 |
| 0x0000150: | 32 66 | 75 60 | 00 62 | 30 72 | 34 74 | 77 68 | 02 70 | 20 01 |
| 0x0000170: | 24 03 | 67 76 | 71 78 | 22 09 | 58 43 | 24 37 | 26 41 | 68 47 |
| 0x0000190: | 70 51 | 36 45 | 38 49 | 72 55 | 74 59 | 40 53 | 42 57 | 44 78 |
| 0x00001b0: | 46 03 | 12 76 | 14 01 | 48 07 | 50 11 | 16 05 | 18 09 | 60 15 |
| 0x00001d0: | 62 19 | 28 13 | 30 17 | 64 23 | 66 27 | 32 21 | 34 25 | 52 31 |
| 0x00001f0: | 54 35 | 20 29 | 22 33 | 56 39 | 19 04 | 62 63 | 66 00 | 07 73 |
| 0x0000210: | 11 10 | 54 69 | 58 06 | 23 75 | 27 12 | 70 71 | 74 08 | 76 33 |
| 0x0000230: | 01 49 | 44 29 | 48 45 | 13 35 | 17 51 | 60 31 | 64 47 | 05 41 |
| 0x0000250: | 09 57 | 52 37 | 56 53 | 21 43 | 25 59 | 68 39 | 72 55 | 78 65 |
| 0x0000270: | 03 02 | 46 61 | 50 77 | 15 67 | 51 36 | 17 18 | 19 34 | 41 24 |
| 0x0000290: | 43 40 | 09 22 | 11 38 | 57 28 | 59 44 | 25 26 | 27 42 | 29 63 |
| 0x00002b0: | 31 00 | 76 61 | 78 77 | 45 67 | 47 04 | 13 65 | 15 02 | 37 71 |
| 0x00002d0: | 39 08 | 05 69 | 07 06 | 53 75 | 55 12 | 21 73 | 23 10 | 33 16 |
| 0x00002f0: | 35 32 | 01 14 | 03 30 | 49 20 | 75 60 | 39 48 | 43 56 | 00 66 |
| 0x0000310: | 04 74 | 47 62 | 51 70 | 08 68 | 12 76 | 55 64 | 59 72 | 61 18 |
| 0x0000330: | 65 26 | 29 14 | 33 22 | 69 20 | 73 28 | 37 16 | 41 24 | 77 34 |
| 0x0000350: | 02 42 | 45 30 | 49 38 | 06 36 | 10 44 | 53 32 | 57 40 | 63 50 |
| 0x0000370: | 67 58 | 31 46 | 35 54 | 71 52 | 28 13 | 73 03 | 75 11 | 34 17 |
| 0x0000390: | 36 25 | 02 15 | 04 23 | 42 21 | 44 29 | 10 19 | 12 27 | 14 48 |
| 0x00003b0: | 16 56 | 61 46 | 63 54 | 22 52 | 24 60 | 69 50 | 71 58 | 30 64 |
| 0x00003d0: | 32 72 | 77 62 | 00 70 | 38 68 | 40 76 | 06 66 | 08 74 | 18 01 |
| 0x00003f0: | 20 09 | 65 78 | 67 07 | 26 05 | 44 29 | 32 23 | 36 25 | 70 43 |

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with all channel used; ie, AFH(79)):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x7fffffffffffffffff

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 55 55 | 19 19 | 23 23 | 53 53 | 57 57 | 21 21 | 25 25 | 27 27 |
| 0x0000030 | 31 31 | 74 74 | 78 78 | 29 29 | 33 33 | 76 76 | 01 01 | 35 35 |
| 0x0000050 | 39 39 | 03 03 | 07 07 | 37 37 | 41 41 | 05 05 | 09 09 | 43 43 |
| 0x0000070 | 47 47 | 11 11 | 15 15 | 45 45 | 02 02 | 47 47 | 49 49 | 04 04 |
| 0x0000090 | 06 06 | 51 51 | 53 53 | 08 08 | 10 10 | 55 55 | 57 57 | 59 59 |
| 0x00000b0 | 61 61 | 27 27 | 29 29 | 63 63 | 65 65 | 31 31 | 33 33 | 67 67 |
| 0x00000d0 | 69 69 | 35 35 | 37 37 | 71 71 | 73 73 | 39 39 | 41 41 | 75 75 |
| 0x00000f0 | 77 77 | 43 43 | 45 45 | 00 00 | 26 26 | 69 69 | 73 73 | 36 36 |
| 0x0000110 | 40 40 | 04 04 | 08 08 | 38 38 | 42 42 | 06 06 | 10 10 | 12 12 |
| 0x0000130 | 16 16 | 59 59 | 63 63 | 14 14 | 18 18 | 61 61 | 65 65 | 28 28 |
| 0x0000150 | 32 32 | 75 75 | 00 00 | 30 30 | 34 34 | 77 77 | 02 02 | 20 20 |
| 0x0000170 | 24 24 | 67 67 | 71 71 | 22 22 | 58 58 | 24 24 | 26 26 | 68 68 |
| 0x0000190 | 70 70 | 36 36 | 38 38 | 72 72 | 74 74 | 40 40 | 42 42 | 44 44 |
| 0x00001b0 | 46 46 | 12 12 | 14 14 | 48 48 | 50 50 | 16 16 | 18 18 | 60 60 |
| 0x00001d0 | 62 62 | 28 28 | 30 30 | 64 64 | 66 66 | 32 32 | 34 34 | 52 52 |
| 0x00001f0 | 54 54 | 20 20 | 22 22 | 56 56 | 19 19 | 62 62 | 66 66 | 07 07 |
| 0x0000210 | 11 11 | 54 54 | 58 58 | 23 23 | 27 27 | 70 70 | 74 74 | 76 76 |
| 0x0000230 | 01 01 | 44 44 | 48 48 | 13 13 | 17 17 | 60 60 | 64 64 | 05 05 |
| 0x0000250 | 09 09 | 52 52 | 56 56 | 21 21 | 25 25 | 68 68 | 72 72 | 78 78 |
| 0x0000270 | 03 03 | 46 46 | 50 50 | 15 15 | 51 51 | 17 17 | 19 19 | 41 41 |
| 0x0000290 | 43 43 | 09 09 | 11 11 | 57 57 | 59 59 | 25 25 | 27 27 | 29 29 |
| 0x00002b0 | 31 31 | 76 76 | 78 78 | 45 45 | 47 47 | 13 13 | 15 15 | 37 37 |
| 0x00002d0 | 39 39 | 05 05 | 07 07 | 53 53 | 55 55 | 21 21 | 23 23 | 33 33 |
| 0x00002f0 | 35 35 | 01 01 | 03 03 | 49 49 | 75 75 | 39 39 | 43 43 | 00 00 |
| 0x0000310 | 04 04 | 47 47 | 51 51 | 08 08 | 12 12 | 55 55 | 59 59 | 61 61 |
| 0x0000330 | 65 65 | 29 29 | 33 33 | 69 69 | 73 73 | 37 37 | 41 41 | 77 77 |
| 0x0000350 | 02 02 | 45 45 | 49 49 | 06 06 | 10 10 | 53 53 | 57 57 | 63 63 |
| 0x0000370 | 67 67 | 31 31 | 35 35 | 71 71 | 28 28 | 73 73 | 75 75 | 34 34 |
| 0x0000390 | 36 36 | 02 02 | 04 04 | 42 42 | 44 44 | 10 10 | 12 12 | 14 14 |
| 0x00003b0 | 16 16 | 61 61 | 63 63 | 22 22 | 24 24 | 69 69 | 71 71 | 30 30 |
| 0x00003d0 | 32 32 | 77 77 | 00 00 | 38 38 | 40 40 | 06 06 | 08 08 | 18 18 |
| 0x00003f0 | 20 20 | 65 65 | 67 67 | 26 26 | 44 44 | 32 32 | 36 36 | 70 70 |



LifeScan, Inc. Ex. 1019
Page 963 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with channels 0 to 21 unused):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x7fffffffffffc00000

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 55 55 | 50 50 | 23 23 | 53 53 | 57 57 | 52 52 | 25 25 | 27 27 |
| 0x0000030 | 31 31 | 74 74 | 78 78 | 29 29 | 33 33 | 76 76 | 32 32 | 35 35 |
| 0x0000050 | 39 39 | 34 34 | 38 38 | 37 37 | 41 41 | 36 36 | 40 40 | 43 43 |
| 0x0000070 | 47 47 | 42 42 | 46 46 | 45 45 | 55 55 | 47 47 | 49 49 | 57 57 |
| 0x0000090 | 59 59 | 51 51 | 53 53 | 61 61 | 63 63 | 55 55 | 57 57 | 59 59 |
| 0x00000b0 | 61 61 | 27 27 | 29 29 | 63 63 | 65 65 | 31 31 | 33 33 | 67 67 |
| 0x00000d0 | 69 69 | 35 35 | 37 37 | 71 71 | 73 73 | 39 39 | 41 41 | 75 75 |
| 0x00000f0 | 77 77 | 43 43 | 45 45 | 53 53 | 26 26 | 69 69 | 73 73 | 36 36 |
| 0x0000110 | 40 40 | 57 57 | 61 61 | 38 38 | 42 42 | 59 59 | 63 63 | 65 65 |
| 0x0000130 | 69 69 | 59 59 | 63 63 | 67 67 | 71 71 | 61 61 | 65 65 | 28 28 |
| 0x0000150 | 32 32 | 75 75 | 53 53 | 30 30 | 34 34 | 77 77 | 55 55 | 73 73 |
| 0x0000170 | 24 24 | 67 67 | 71 71 | 22 22 | 58 58 | 24 24 | 26 26 | 68 68 |
| 0x0000190 | 70 70 | 36 36 | 38 38 | 72 72 | 74 74 | 40 40 | 42 42 | 44 44 |
| 0x00001b0 | 46 46 | 65 65 | 67 67 | 48 48 | 50 50 | 69 69 | 71 71 | 60 60 |
| 0x00001d0 | 62 62 | 28 28 | 30 30 | 64 64 | 66 66 | 32 32 | 34 34 | 52 52 |
| 0x00001f0 | 54 54 | 73 73 | 22 22 | 56 56 | 37 37 | 62 62 | 66 66 | 25 25 |
| 0x0000210 | 29 29 | 54 54 | 58 58 | 23 23 | 27 27 | 70 70 | 74 74 | 76 76 |
| 0x0000230 | 76 76 | 44 44 | 48 48 | 31 31 | 35 35 | 60 60 | 64 64 | 23 23 |
| 0x0000250 | 27 27 | 52 52 | 56 56 | 39 39 | 25 25 | 68 68 | 72 72 | 78 78 |
| 0x0000270 | 78 78 | 46 46 | 50 50 | 33 33 | 51 51 | 35 35 | 37 37 | 41 41 |
| 0x0000290 | 43 43 | 27 27 | 29 29 | 57 57 | 59 59 | 25 25 | 27 27 | 29 29 |
| 0x00002b0 | 31 31 | 76 76 | 78 78 | 45 45 | 47 47 | 31 31 | 33 33 | 37 37 |
| 0x00002d0 | 39 39 | 23 23 | 25 25 | 53 53 | 55 55 | 39 39 | 23 23 | 33 33 |
| 0x00002f0 | 35 35 | 76 76 | 78 78 | 49 49 | 75 75 | 39 39 | 43 43 | 40 40 |
| 0x0000310 | 44 44 | 47 47 | 51 51 | 48 48 | 52 52 | 55 55 | 59 59 | 61 61 |
| 0x0000330 | 65 65 | 29 29 | 33 33 | 69 69 | 73 73 | 37 37 | 41 41 | 77 77 |
| 0x0000350 | 42 42 | 45 45 | 49 49 | 46 46 | 50 50 | 53 53 | 57 57 | 63 63 |
| 0x0000370 | 67 67 | 31 31 | 35 35 | 71 71 | 28 28 | 73 73 | 75 75 | 34 34 |
| 0x0000390 | 36 36 | 42 42 | 44 44 | 42 42 | 44 44 | 50 50 | 52 52 | 54 54 |
| 0x00003b0 | 56 56 | 61 61 | 63 63 | 22 22 | 24 24 | 69 69 | 71 71 | 30 30 |
| 0x00003d0 | 32 32 | 77 77 | 40 40 | 38 38 | 40 40 | 46 46 | 48 48 | 58 58 |
| 0x00003f0 | 60 60 | 65 65 | 67 67 | 26 26 | 44 44 | 32 32 | 36 36 | 70 70 |



LifeScan, Inc. Ex. 1019
Page 964 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with even channels used):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x5555555555555555555

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 52 52 | 16 16 | 20 20 | 50 50 | 54 54 | 18 18 | 22 22 | 24 24 |
| 0x0000030 | 28 28 | 74 74 | 78 78 | 26 26 | 30 30 | 76 76 | 78 78 | 32 32 |
| 0x0000050 | 36 36 | 00 00 | 04 04 | 34 34 | 38 38 | 02 02 | 06 06 | 40 40 |
| 0x0000070 | 44 44 | 08 08 | 12 12 | 42 42 | 02 02 | 44 44 | 46 46 | 04 04 |
| 0x0000090 | 06 06 | 48 48 | 50 50 | 08 08 | 10 10 | 52 52 | 54 54 | 56 56 |
| 0x00000b0 | 58 58 | 24 24 | 26 26 | 60 60 | 62 62 | 28 28 | 30 30 | 64 64 |
| 0x00000d0 | 66 66 | 32 32 | 34 34 | 68 68 | 70 70 | 36 36 | 38 38 | 72 72 |
| 0x00000f0 | 74 74 | 40 40 | 42 42 | 00 00 | 26 26 | 66 66 | 70 70 | 36 36 |
| 0x0000110 | 40 40 | 04 04 | 08 08 | 38 38 | 42 42 | 06 06 | 10 10 | 12 12 |
| 0x0000130 | 16 16 | 56 56 | 60 60 | 14 14 | 18 18 | 58 58 | 62 62 | 28 28 |
| 0x0000150 | 32 32 | 72 72 | 00 00 | 30 30 | 34 34 | 74 74 | 02 02 | 20 20 |
| 0x0000170 | 24 24 | 64 64 | 68 68 | 22 22 | 58 58 | 24 24 | 26 26 | 68 68 |
| 0x0000190 | 70 70 | 36 36 | 38 38 | 72 72 | 74 74 | 40 40 | 42 42 | 44 44 |
| 0x00001b0 | 46 46 | 12 12 | 14 14 | 48 48 | 50 50 | 16 16 | 18 18 | 60 60 |
| 0x00001d0 | 62 62 | 28 28 | 30 30 | 64 64 | 66 66 | 32 32 | 34 34 | 52 52 |
| 0x00001f0 | 54 54 | 20 20 | 22 22 | 56 56 | 14 14 | 62 62 | 66 66 | 02 02 |
| 0x0000210 | 06 06 | 54 54 | 58 58 | 18 18 | 22 22 | 70 70 | 74 74 | 76 76 |
| 0x0000230 | 76 76 | 44 44 | 48 48 | 08 08 | 12 12 | 60 60 | 64 64 | 00 00 |
| 0x0000250 | 04 04 | 52 52 | 56 56 | 16 16 | 20 20 | 68 68 | 72 72 | 78 78 |
| 0x0000270 | 78 78 | 46 46 | 50 50 | 10 10 | 46 46 | 12 12 | 14 14 | 36 36 |
| 0x0000290 | 38 38 | 04 04 | 06 06 | 52 52 | 54 54 | 20 20 | 22 22 | 24 24 |
| 0x00002b0 | 26 26 | 76 76 | 78 78 | 40 40 | 42 42 | 08 08 | 10 10 | 32 32 |
| 0x00002d0 | 34 34 | 00 00 | 02 02 | 48 48 | 50 50 | 16 16 | 18 18 | 28 28 |
| 0x00002f0 | 30 30 | 76 76 | 78 78 | 44 44 | 70 70 | 34 34 | 38 38 | 00 00 |
| 0x0000310 | 04 04 | 42 42 | 46 46 | 08 08 | 12 12 | 50 50 | 54 54 | 56 56 |
| 0x0000330 | 60 60 | 24 24 | 28 28 | 64 64 | 68 68 | 32 32 | 36 36 | 72 72 |
| 0x0000350 | 02 02 | 40 40 | 44 44 | 06 06 | 10 10 | 48 48 | 52 52 | 58 58 |
| 0x0000370 | 62 62 | 26 26 | 30 30 | 66 66 | 28 28 | 68 68 | 70 70 | 34 34 |
| 0x0000390 | 36 36 | 02 02 | 04 04 | 42 42 | 44 44 | 10 10 | 12 12 | 14 14 |
| 0x00003b0 | 16 16 | 56 56 | 58 58 | 22 22 | 24 24 | 64 64 | 66 66 | 30 30 |
| 0x00003d0 | 32 32 | 72 72 | 00 00 | 38 38 | 40 40 | 06 06 | 08 08 | 18 18 |
| 0x00003f0 | 20 20 | 60 60 | 62 62 | 26 26 | 44 44 | 32 32 | 36 36 | 70 70 |



LifeScan, Inc. Ex. 1019
Page 965 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with odd channels used):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x2aaaaaaaaaaaaaaaaaaa

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 55 55 | 19 19 | 23 23 | 53 53 | 57 57 | 21 21 | 25 25 | 27 27 |
| 0x0000030 | 31 31 | 77 77 | 03 03 | 29 29 | 33 33 | 01 01 | 01 01 | 35 35 |
| 0x0000050 | 39 39 | 03 03 | 07 07 | 37 37 | 41 41 | 05 05 | 09 09 | 43 43 |
| 0x0000070 | 47 47 | 11 11 | 15 15 | 45 45 | 07 07 | 47 47 | 49 49 | 09 09 |
| 0x0000090 | 11 11 | 51 51 | 53 53 | 13 13 | 15 15 | 55 55 | 57 57 | 59 59 |
| 0x00000b0 | 61 61 | 27 27 | 29 29 | 63 63 | 65 65 | 31 31 | 33 33 | 67 67 |
| 0x00000d0 | 69 69 | 35 35 | 37 37 | 71 71 | 73 73 | 39 39 | 41 41 | 75 75 |
| 0x00000f0 | 77 77 | 43 43 | 45 45 | 05 05 | 31 31 | 69 69 | 73 73 | 41 41 |
| 0x0000110 | 45 45 | 09 09 | 13 13 | 43 43 | 47 47 | 11 11 | 15 15 | 17 17 |
| 0x0000130 | 21 21 | 59 59 | 63 63 | 19 19 | 23 23 | 61 61 | 65 65 | 33 33 |
| 0x0000150 | 37 37 | 75 75 | 05 05 | 35 35 | 39 39 | 77 77 | 07 07 | 25 25 |
| 0x0000170 | 29 29 | 67 67 | 71 71 | 27 27 | 63 63 | 29 29 | 31 31 | 73 73 |
| 0x0000190 | 75 75 | 41 41 | 43 43 | 77 77 | 01 01 | 45 45 | 47 47 | 49 49 |
| 0x00001b0 | 51 51 | 17 17 | 19 19 | 53 53 | 55 55 | 21 21 | 23 23 | 65 65 |
| 0x00001d0 | 67 67 | 33 33 | 35 35 | 69 69 | 71 71 | 37 37 | 39 39 | 57 57 |
| 0x00001f0 | 59 59 | 25 25 | 27 27 | 61 61 | 19 19 | 67 67 | 71 71 | 07 07 |
| 0x0000210 | 11 11 | 59 59 | 63 63 | 23 23 | 27 27 | 75 75 | 01 01 | 03 03 |
| 0x0000230 | 01 01 | 49 49 | 53 53 | 13 13 | 17 17 | 65 65 | 69 69 | 05 05 |
| 0x0000250 | 09 09 | 57 57 | 61 61 | 21 21 | 25 25 | 73 73 | 77 77 | 05 05 |
| 0x0000270 | 03 03 | 51 51 | 55 55 | 15 15 | 51 51 | 17 17 | 19 19 | 41 41 |
| 0x0000290 | 43 43 | 09 09 | 11 11 | 57 57 | 59 59 | 25 25 | 27 27 | 29 29 |
| 0x00002b0 | 31 31 | 03 03 | 05 05 | 45 45 | 47 47 | 13 13 | 15 15 | 37 37 |
| 0x00002d0 | 39 39 | 05 05 | 07 07 | 53 53 | 55 55 | 21 21 | 23 23 | 33 33 |
| 0x00002f0 | 35 35 | 01 01 | 03 03 | 49 49 | 75 75 | 39 39 | 43 43 | 07 07 |
| 0x0000310 | 11 11 | 47 47 | 51 51 | 15 15 | 19 19 | 55 55 | 59 59 | 61 61 |
| 0x0000330 | 65 65 | 29 29 | 33 33 | 69 69 | 73 73 | 37 37 | 41 41 | 77 77 |
| 0x0000350 | 09 09 | 45 45 | 49 49 | 13 13 | 17 17 | 53 53 | 57 57 | 63 63 |
| 0x0000370 | 67 67 | 31 31 | 35 35 | 71 71 | 35 35 | 73 73 | 75 75 | 41 41 |
| 0x0000390 | 43 43 | 09 09 | 11 11 | 49 49 | 51 51 | 17 17 | 19 19 | 21 21 |
| 0x00003b0 | 23 23 | 61 61 | 63 63 | 29 29 | 31 31 | 69 69 | 71 71 | 37 37 |
| 0x00003d0 | 39 39 | 77 77 | 07 07 | 45 45 | 47 47 | 13 13 | 15 15 | 25 25 |
| 0x00003f0 | 27 27 | 65 65 | 67 67 | 33 33 | 51 51 | 39 39 | 43 43 | 77 77 |



LifeScan, Inc. Ex. 1019
Page 966 of 1420
IPR2025-00103

*Sample Data*

## 2.3 THIRD SET

```
Hop sequence {k} for PAGE SCAN/INQUIRY SCAN SUBSTATE:
CLKN start:      0x0000000
ULAP:            0x6587cba9
#ticks:          0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
                 -----------------------------------------------------
0x0000000:         16 |   65 |   67 |   18 |   20 |   53 |   55 |    6 |
0x0008000:          8 |   57 |   59 |   10 |   12 |   69 |   71 |   22 |
0x0010000:         24 |   73 |   75 |   26 |   28 |   45 |   47 |   77 |
0x0018000:          0 |   49 |   51 |    2 |    4 |   61 |   63 |   14 |
0x0020000:         16 |   65 |   67 |   18 |   20 |   53 |   55 |    6 |
0x0028000:          8 |   57 |   59 |   10 |   12 |   69 |   71 |   22 |
0x0030000:         24 |   73 |   75 |   26 |   28 |   45 |   47 |   77 |
0x0038000:          0 |   49 |   51 |    2 |    4 |   61 |   63 |   14 |

Hop sequence {k} for PAGE STATE/INQUIRY SUBSTATE:
CLKE start:      0x0000000
ULAP:            0x6587cba9
#ticks:          00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
                 -----------------------------------------------------
0x0000000:       00 49 36 38 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0000010:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
0x0000020:       00 49 36 38 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0000030:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
...
0x0001000:       00 57 36 70 | 59 10 74 72 | 12 69 76 78 | 71 22 03 01 |
0x0001010:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
0x0001020:       00 57 36 70 | 59 10 74 72 | 12 69 76 78 | 71 22 03 01 |
0x0001030:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
...
0x0002000:       08 57 68 70 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0002010:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
0x0002020:       08 57 68 70 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0002030:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
...
0x0003000:       00 49 36 38 | 51 10 42 72 | 12 69 76 78 | 71 22 03 01 |
0x0003010:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
0x0003020:       00 49 36 38 | 51 10 42 72 | 12 69 76 78 | 71 22 03 01 |
0x0003030:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |

Hop sequence {k} for SLAVE PAGE RESPONSE SUBSTATE:
CLKN* =          0x0000010
ULAP:            0x6587cba9
#ticks:          00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a 1c | 1e
                 ---------------------------------------------------------------
0x0000012:       52 | 65 54 | 67 58 | 18 56 | 20 60 | 53 62 | 55 66 | 06 64 | 08
0x0000032:       68 | 57 70 | 59 74 | 10 72 | 12 76 | 69 78 | 71 03 | 22 01 | 24
0x0000052:       05 | 73 07 | 75 11 | 26 09 | 28 13 | 45 30 | 47 34 | 77 32 | 00
0x0000072:       36 | 49 38 | 51 42 | 02 40 | 04 44 | 61 46 | 63 50 | 14 48 | 16

Hop sequence {k} for MASTER PAGE RESPONSE SUBSTATE:
Offset value:    24
CLKE* =          0x0000012
ULAP:            0x6587cba9
#ticks:          00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
                 ---------------------------------------------------------------
0x0000014:       65 54 | 67 58 | 18 56 | 20 60 | 53 62 | 55 66 | 06 64 | 08 68 |
0x0000034:       57 70 | 59 74 | 10 72 | 12 76 | 69 78 | 71 03 | 22 01 | 24 05 |
0x0000054:       73 07 | 75 11 | 26 09 | 28 13 | 45 30 | 47 34 | 77 32 | 00 36 |
0x0000074:       49 38 | 51 42 | 02 40 | 04 44 | 61 46 | 63 50 | 14 48 | 16 52 |
                                                                    Page 1
```



LifeScan, Inc. Ex. 1019
Page 967 of 1420
IPR2025-00103

*Sample Data*

Hop sequence {k} for CONNECTION STATE (Basic channel hopping sequence; ie, non-AFH):
CLK start: 0x0000010
ULAP: 0x6587cba9

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010: | 20 60 | 53 62 | 55 66 | 06 64 | 08 68 | 57 70 | 59 74 | 10 72 |
| 0x0000030: | 12 76 | 69 78 | 71 03 | 22 01 | 24 05 | 73 07 | 75 11 | 26 09 |
| 0x0000050: | 28 13 | 45 30 | 47 34 | 77 32 | 00 36 | 49 38 | 51 42 | 02 40 |
| 0x0000070: | 04 44 | 61 46 | 63 50 | 14 48 | 50 05 | 16 07 | 20 09 | 48 11 |
| 0x0000090: | 52 13 | 06 15 | 10 17 | 38 19 | 42 21 | 08 23 | 12 25 | 40 27 |
| 0x00000b0: | 44 29 | 22 31 | 26 33 | 54 35 | 58 37 | 24 39 | 28 41 | 56 43 |
| 0x00000d0: | 60 45 | 77 62 | 02 64 | 30 66 | 34 68 | 00 70 | 04 72 | 32 74 |
| 0x00000f0: | 36 76 | 14 78 | 18 01 | 46 03 | 72 29 | 42 39 | 44 43 | 74 41 |
| 0x0000110: | 76 45 | 46 47 | 48 51 | 78 49 | 01 53 | 50 63 | 52 67 | 03 65 |
| 0x0000130: | 05 69 | 54 55 | 56 59 | 07 57 | 09 61 | 58 71 | 60 75 | 11 73 |
| 0x0000150: | 13 77 | 30 15 | 32 19 | 62 17 | 64 21 | 34 31 | 36 35 | 66 33 |
| 0x0000170: | 68 37 | 38 23 | 40 27 | 70 25 | 27 61 | 72 71 | 76 73 | 25 75 |
| 0x0000190: | 29 77 | 78 00 | 03 02 | 31 04 | 35 06 | 01 16 | 05 18 | 33 20 |
| 0x00001b0: | 37 22 | 07 08 | 11 10 | 39 12 | 43 14 | 09 24 | 13 26 | 41 28 |
| 0x00001d0: | 45 30 | 62 47 | 66 49 | 15 51 | 19 53 | 64 63 | 68 65 | 17 67 |
| 0x00001f0: | 21 69 | 70 55 | 74 57 | 23 59 | 53 22 | 35 12 | 37 28 | 67 14 |
| 0x0000210: | 69 30 | 23 32 | 25 48 | 55 34 | 57 50 | 39 40 | 41 56 | 71 42 |
| 0x0000230: | 73 58 | 27 36 | 29 52 | 59 38 | 61 54 | 43 44 | 45 60 | 75 46 |
| 0x0000250: | 77 62 | 15 00 | 17 16 | 47 02 | 49 18 | 31 08 | 33 24 | 63 10 |
| 0x0000270: | 65 26 | 19 04 | 21 20 | 51 06 | 06 54 | 65 42 | 69 58 | 18 46 |
| 0x0000290: | 22 62 | 55 64 | 59 01 | 08 68 | 12 05 | 71 72 | 75 09 | 24 76 |
| 0x00002b0: | 28 13 | 57 66 | 61 03 | 10 70 | 14 07 | 73 74 | 77 11 | 26 78 |
| 0x00002d0: | 30 15 | 47 32 | 51 48 | 00 36 | 04 52 | 63 40 | 67 56 | 16 44 |
| 0x00002f0: | 20 60 | 49 34 | 53 50 | 02 38 | 38 78 | 12 05 | 14 13 | 44 07 |
| 0x0000310: | 46 15 | 16 17 | 18 25 | 48 19 | 50 27 | 24 33 | 26 41 | 56 35 |
| 0x0000330: | 58 43 | 20 21 | 22 29 | 52 23 | 54 31 | 28 37 | 30 45 | 60 39 |
| 0x0000350: | 62 47 | 00 64 | 02 72 | 32 66 | 34 74 | 08 01 | 10 09 | 40 03 |
| 0x0000370: | 42 11 | 04 68 | 06 76 | 36 70 | 70 31 | 42 35 | 46 43 | 74 39 |
| 0x0000390: | 78 47 | 48 49 | 52 57 | 01 53 | 05 61 | 56 65 | 60 73 | 09 69 |
| 0x00003b0: | 13 77 | 50 51 | 54 59 | 03 55 | 07 63 | 58 67 | 62 75 | 11 71 |
| 0x00003d0: | 15 00 | 32 17 | 36 25 | 64 21 | 68 29 | 40 33 | 44 41 | 72 37 |
| 0x00003f0: | 76 45 | 34 19 | 38 27 | 66 23 | 11 71 | 05 18 | 07 22 | 13 20 |



LifeScan, Inc. Ex. 1019
Page 968 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with all channel used; ie, AFH(79)):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x7fffffffffffffffff

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 20 20 | 53 53 | 55 55 | 06 06 | 08 08 | 57 57 | 59 59 | 10 10 |
| 0x0000030 | 12 12 | 69 69 | 71 71 | 22 22 | 24 24 | 73 73 | 75 75 | 26 26 |
| 0x0000050 | 28 28 | 45 45 | 47 47 | 77 77 | 00 00 | 49 49 | 51 51 | 02 02 |
| 0x0000070 | 04 04 | 61 61 | 63 63 | 14 14 | 50 50 | 16 16 | 20 20 | 48 48 |
| 0x0000090 | 52 52 | 06 06 | 10 10 | 38 38 | 42 42 | 08 08 | 12 12 | 40 40 |
| 0x00000b0 | 44 44 | 22 22 | 26 26 | 54 54 | 58 58 | 24 24 | 28 28 | 56 56 |
| 0x00000d0 | 60 60 | 77 77 | 02 02 | 30 30 | 34 34 | 00 00 | 04 04 | 32 32 |
| 0x00000f0 | 36 36 | 14 14 | 18 18 | 46 46 | 72 72 | 42 42 | 44 44 | 74 74 |
| 0x0000110 | 76 76 | 46 46 | 48 48 | 78 78 | 01 01 | 50 50 | 52 52 | 03 03 |
| 0x0000130 | 05 05 | 54 54 | 56 56 | 07 07 | 09 09 | 58 58 | 60 60 | 11 11 |
| 0x0000150 | 13 13 | 30 30 | 32 32 | 62 62 | 64 64 | 34 34 | 36 36 | 66 66 |
| 0x0000170 | 68 68 | 38 38 | 40 40 | 70 70 | 27 27 | 72 72 | 76 76 | 25 25 |
| 0x0000190 | 29 29 | 78 78 | 03 03 | 31 31 | 35 35 | 01 01 | 05 05 | 33 33 |
| 0x00001b0 | 37 37 | 07 07 | 11 11 | 39 39 | 43 43 | 09 09 | 13 13 | 41 41 |
| 0x00001d0 | 45 45 | 62 62 | 66 66 | 15 15 | 19 19 | 64 64 | 68 68 | 17 17 |
| 0x00001f0 | 21 21 | 70 70 | 74 74 | 23 23 | 53 53 | 35 35 | 37 37 | 67 67 |
| 0x0000210 | 69 69 | 23 23 | 25 25 | 55 55 | 57 57 | 39 39 | 41 41 | 71 71 |
| 0x0000230 | 73 73 | 27 27 | 29 29 | 59 59 | 61 61 | 43 43 | 45 45 | 75 75 |
| 0x0000250 | 77 77 | 15 15 | 17 17 | 47 47 | 49 49 | 31 31 | 33 33 | 63 63 |
| 0x0000270 | 65 65 | 19 19 | 21 21 | 51 51 | 06 06 | 65 65 | 69 69 | 18 18 |
| 0x0000290 | 22 22 | 55 55 | 59 59 | 08 08 | 12 12 | 71 71 | 75 75 | 24 24 |
| 0x00002b0 | 28 28 | 57 57 | 61 61 | 10 10 | 14 14 | 73 73 | 77 77 | 26 26 |
| 0x00002d0 | 30 30 | 47 47 | 51 51 | 00 00 | 04 04 | 63 63 | 67 67 | 16 16 |
| 0x00002f0 | 20 20 | 49 49 | 53 53 | 02 02 | 38 38 | 12 12 | 14 14 | 44 44 |
| 0x0000310 | 46 46 | 16 16 | 18 18 | 48 48 | 50 50 | 24 24 | 26 26 | 56 56 |
| 0x0000330 | 58 58 | 20 20 | 22 22 | 52 52 | 54 54 | 28 28 | 30 30 | 60 60 |
| 0x0000350 | 62 62 | 00 00 | 02 02 | 32 32 | 34 34 | 08 08 | 10 10 | 40 40 |
| 0x0000370 | 42 42 | 04 04 | 06 06 | 36 36 | 70 70 | 42 42 | 46 46 | 74 74 |
| 0x0000390 | 78 78 | 48 48 | 52 52 | 01 01 | 05 05 | 56 56 | 60 60 | 09 09 |
| 0x00003b0 | 13 13 | 50 50 | 54 54 | 03 03 | 07 07 | 58 58 | 62 62 | 11 11 |
| 0x00003d0 | 15 15 | 32 32 | 36 36 | 64 64 | 68 68 | 40 40 | 44 44 | 72 72 |
| 0x00003f0 | 76 76 | 34 34 | 38 38 | 66 66 | 11 11 | 05 05 | 07 07 | 13 13 |



LifeScan, Inc. Ex. 1019
Page 969 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with channels 0 to 21 unused):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x7fffffffffffc00000

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 29 29 | 53 53 | 55 55 | 72 72 | 74 74 | 57 57 | 59 59 | 76 76 |
| 0x0000030 | 78 78 | 69 69 | 71 71 | 22 22 | 24 24 | 73 73 | 75 75 | 26 26 |
| 0x0000050 | 28 28 | 45 45 | 47 47 | 77 77 | 66 66 | 49 49 | 51 51 | 68 68 |
| 0x0000070 | 70 70 | 61 61 | 63 63 | 23 23 | 50 50 | 25 25 | 29 29 | 48 48 |
| 0x0000090 | 52 52 | 72 72 | 76 76 | 38 38 | 42 42 | 74 74 | 78 78 | 40 40 |
| 0x00000b0 | 44 44 | 22 22 | 26 26 | 54 54 | 58 58 | 24 24 | 28 28 | 56 56 |
| 0x00000d0 | 60 60 | 77 77 | 68 68 | 30 30 | 34 34 | 66 66 | 70 70 | 32 32 |
| 0x00000f0 | 36 36 | 23 23 | 27 27 | 46 46 | 72 72 | 42 42 | 44 44 | 74 74 |
| 0x0000110 | 76 76 | 46 46 | 48 48 | 78 78 | 32 32 | 50 50 | 52 52 | 34 34 |
| 0x0000130 | 36 36 | 54 54 | 56 56 | 38 38 | 40 40 | 58 58 | 60 60 | 42 42 |
| 0x0000150 | 44 44 | 30 30 | 32 32 | 62 62 | 64 64 | 34 34 | 36 36 | 66 66 |
| 0x0000170 | 68 68 | 38 38 | 40 40 | 70 70 | 27 27 | 72 72 | 76 76 | 25 25 |
| 0x0000190 | 29 29 | 78 78 | 34 34 | 31 31 | 35 35 | 32 32 | 36 36 | 33 33 |
| 0x00001b0 | 37 37 | 38 38 | 42 42 | 39 39 | 43 43 | 40 40 | 44 44 | 41 41 |
| 0x00001d0 | 45 45 | 62 62 | 66 66 | 46 46 | 50 50 | 64 64 | 68 68 | 48 48 |
| 0x00001f0 | 52 52 | 70 70 | 74 74 | 23 23 | 53 53 | 35 35 | 37 37 | 67 67 |
| 0x0000210 | 69 69 | 23 23 | 25 25 | 55 55 | 57 57 | 39 39 | 41 41 | 71 71 |
| 0x0000230 | 73 73 | 27 27 | 29 29 | 59 59 | 61 61 | 43 43 | 45 45 | 75 75 |
| 0x0000250 | 77 77 | 46 46 | 48 48 | 47 47 | 49 49 | 31 31 | 33 33 | 63 63 |
| 0x0000270 | 65 65 | 50 50 | 52 52 | 51 51 | 59 59 | 65 65 | 69 69 | 71 71 |
| 0x0000290 | 22 22 | 55 55 | 59 59 | 61 61 | 65 65 | 71 71 | 75 75 | 24 24 |
| 0x00002b0 | 28 28 | 57 57 | 61 61 | 63 63 | 67 67 | 73 73 | 77 77 | 26 26 |
| 0x00002d0 | 30 30 | 47 47 | 51 51 | 53 53 | 57 57 | 63 63 | 67 67 | 69 69 |
| 0x00002f0 | 73 73 | 49 49 | 53 53 | 55 55 | 38 38 | 65 65 | 67 67 | 44 44 |
| 0x0000310 | 46 46 | 69 69 | 71 71 | 48 48 | 50 50 | 24 24 | 26 26 | 56 56 |
| 0x0000330 | 58 58 | 73 73 | 22 22 | 52 52 | 54 54 | 28 28 | 30 30 | 60 60 |
| 0x0000350 | 62 62 | 53 53 | 55 55 | 32 32 | 34 34 | 61 61 | 63 63 | 40 40 |
| 0x0000370 | 42 42 | 57 57 | 59 59 | 36 36 | 70 70 | 42 42 | 46 46 | 74 74 |
| 0x0000390 | 78 78 | 48 48 | 52 52 | 76 76 | 23 23 | 56 56 | 60 60 | 27 27 |
| 0x00003b0 | 31 31 | 50 50 | 54 54 | 78 78 | 25 25 | 58 58 | 62 62 | 29 29 |
| 0x00003d0 | 33 33 | 32 32 | 36 36 | 64 64 | 68 68 | 40 40 | 44 44 | 72 72 |
| 0x00003f0 | 76 76 | 34 34 | 38 38 | 66 66 | 29 29 | 23 23 | 25 25 | 31 31 |

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with even channels used):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x5555555555555555555

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 20 20 | 52 52 | 54 54 | 06 06 | 08 08 | 56 56 | 58 58 | 10 10 |
| 0x0000030 | 12 12 | 68 68 | 70 70 | 22 22 | 24 24 | 72 72 | 74 74 | 26 26 |
| 0x0000050 | 28 28 | 44 44 | 46 46 | 76 76 | 00 00 | 48 48 | 50 50 | 02 02 |
| 0x0000070 | 04 04 | 60 60 | 62 62 | 14 14 | 50 50 | 16 16 | 20 20 | 48 48 |
| 0x0000090 | 52 52 | 06 06 | 10 10 | 38 38 | 42 42 | 08 08 | 12 12 | 40 40 |
| 0x00000b0 | 44 44 | 22 22 | 26 26 | 54 54 | 58 58 | 24 24 | 28 28 | 56 56 |
| 0x00000d0 | 60 60 | 76 76 | 02 02 | 30 30 | 34 34 | 00 00 | 04 04 | 32 32 |
| 0x00000f0 | 36 36 | 14 14 | 18 18 | 46 46 | 72 72 | 42 42 | 44 44 | 74 74 |
| 0x0000110 | 76 76 | 46 46 | 48 48 | 78 78 | 78 78 | 50 50 | 52 52 | 00 00 |
| 0x0000130 | 02 02 | 54 54 | 56 56 | 04 04 | 06 06 | 58 58 | 60 60 | 08 08 |
| 0x0000150 | 10 10 | 30 30 | 32 32 | 62 62 | 64 64 | 34 34 | 36 36 | 66 66 |
| 0x0000170 | 68 68 | 38 38 | 40 40 | 70 70 | 24 24 | 72 72 | 76 76 | 22 22 |
| 0x0000190 | 26 26 | 78 78 | 00 00 | 28 28 | 32 32 | 78 78 | 02 02 | 30 30 |
| 0x00001b0 | 34 34 | 04 04 | 08 08 | 36 36 | 40 40 | 06 06 | 10 10 | 38 38 |
| 0x00001d0 | 42 42 | 62 62 | 66 66 | 12 12 | 16 16 | 64 64 | 68 68 | 14 14 |
| 0x00001f0 | 18 18 | 70 70 | 74 74 | 20 20 | 50 50 | 32 32 | 34 34 | 64 64 |
| 0x0000210 | 66 66 | 20 20 | 22 22 | 52 52 | 54 54 | 36 36 | 38 38 | 68 68 |
| 0x0000230 | 70 70 | 24 24 | 26 26 | 56 56 | 58 58 | 40 40 | 42 42 | 72 72 |
| 0x0000250 | 74 74 | 12 12 | 14 14 | 44 44 | 46 46 | 28 28 | 30 30 | 60 60 |
| 0x0000270 | 62 62 | 16 16 | 18 18 | 48 48 | 06 06 | 62 62 | 66 66 | 18 18 |
| 0x0000290 | 22 22 | 52 52 | 56 56 | 08 08 | 12 12 | 68 68 | 72 72 | 24 24 |
| 0x00002b0 | 28 28 | 54 54 | 58 58 | 10 10 | 14 14 | 70 70 | 74 74 | 26 26 |
| 0x00002d0 | 30 30 | 44 44 | 48 48 | 00 00 | 04 04 | 60 60 | 64 64 | 16 16 |
| 0x00002f0 | 20 20 | 46 46 | 50 50 | 02 02 | 38 38 | 12 12 | 14 14 | 44 44 |
| 0x0000310 | 46 46 | 16 16 | 18 18 | 48 48 | 50 50 | 24 24 | 26 26 | 56 56 |
| 0x0000330 | 58 58 | 20 20 | 22 22 | 52 52 | 54 54 | 28 28 | 30 30 | 60 60 |
| 0x0000350 | 62 62 | 00 00 | 02 02 | 32 32 | 34 34 | 08 08 | 10 10 | 40 40 |
| 0x0000370 | 42 42 | 04 04 | 06 06 | 36 36 | 70 70 | 42 42 | 46 46 | 74 74 |
| 0x0000390 | 78 78 | 48 48 | 52 52 | 76 76 | 00 00 | 56 56 | 60 60 | 04 04 |
| 0x00003b0 | 08 08 | 50 50 | 54 54 | 78 78 | 02 02 | 58 58 | 62 62 | 06 06 |
| 0x00003d0 | 10 10 | 32 32 | 36 36 | 64 64 | 68 68 | 40 40 | 44 44 | 72 72 |
| 0x00003f0 | 76 76 | 34 34 | 38 38 | 66 66 | 06 06 | 00 00 | 02 02 | 08 08 |



LifeScan, Inc. Ex. 1019
Page 971 of 1420
IPR2025-00103

*Sample Data*

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with odd channels used):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x2aaaaaaaaaaaaaaaaaaa

| #ticks: | 00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
|---|---|---|---|---|---|---|---|---|
| 0x0000010 | 23 23 | 53 53 | 55 55 | 09 09 | 11 11 | 57 57 | 59 59 | 13 13 |
| 0x0000030 | 15 15 | 69 69 | 71 71 | 25 25 | 27 27 | 73 73 | 75 75 | 29 29 |
| 0x0000050 | 31 31 | 45 45 | 47 47 | 77 77 | 03 03 | 49 49 | 51 51 | 05 05 |
| 0x0000070 | 07 07 | 61 61 | 63 63 | 17 17 | 53 53 | 19 19 | 23 23 | 51 51 |
| 0x0000090 | 55 55 | 09 09 | 13 13 | 41 41 | 45 45 | 11 11 | 15 15 | 43 43 |
| 0x00000b0 | 47 47 | 25 25 | 29 29 | 57 57 | 61 61 | 27 27 | 31 31 | 59 59 |
| 0x00000d0 | 63 63 | 77 77 | 05 05 | 33 33 | 37 37 | 03 03 | 07 07 | 35 35 |
| 0x00000f0 | 39 39 | 17 17 | 21 21 | 49 49 | 75 75 | 45 45 | 47 47 | 77 77 |
| 0x0000110 | 01 01 | 49 49 | 51 51 | 03 03 | 01 01 | 53 53 | 55 55 | 03 03 |
| 0x0000130 | 05 05 | 57 57 | 59 59 | 07 07 | 09 09 | 61 61 | 63 63 | 11 11 |
| 0x0000150 | 13 13 | 33 33 | 35 35 | 65 65 | 67 67 | 37 37 | 39 39 | 69 69 |
| 0x0000170 | 71 71 | 41 41 | 43 43 | 73 73 | 27 27 | 75 75 | 01 01 | 25 25 |
| 0x0000190 | 29 29 | 03 03 | 03 03 | 31 31 | 35 35 | 01 01 | 05 05 | 33 33 |
| 0x00001b0 | 37 37 | 07 07 | 11 11 | 39 39 | 43 43 | 09 09 | 13 13 | 41 41 |
| 0x00001d0 | 45 45 | 65 65 | 69 69 | 15 15 | 19 19 | 67 67 | 71 71 | 17 17 |
| 0x00001f0 | 21 21 | 73 73 | 77 77 | 23 23 | 53 53 | 35 35 | 37 37 | 67 67 |
| 0x0000210 | 69 69 | 23 23 | 25 25 | 55 55 | 57 57 | 39 39 | 41 41 | 71 71 |
| 0x0000230 | 73 73 | 27 27 | 29 29 | 59 59 | 61 61 | 43 43 | 45 45 | 75 75 |
| 0x0000250 | 77 77 | 15 15 | 17 17 | 47 47 | 49 49 | 31 31 | 33 33 | 63 63 |
| 0x0000270 | 65 65 | 19 19 | 21 21 | 51 51 | 11 11 | 65 65 | 69 69 | 23 23 |
| 0x0000290 | 27 27 | 55 55 | 59 59 | 13 13 | 17 17 | 71 71 | 75 75 | 29 29 |
| 0x00002b0 | 33 33 | 57 57 | 61 61 | 15 15 | 19 19 | 73 73 | 77 77 | 31 31 |
| 0x00002d0 | 35 35 | 47 47 | 51 51 | 05 05 | 09 09 | 63 63 | 67 67 | 21 21 |
| 0x00002f0 | 25 25 | 49 49 | 53 53 | 07 07 | 43 43 | 17 17 | 19 19 | 49 49 |
| 0x0000310 | 51 51 | 21 21 | 23 23 | 53 53 | 55 55 | 29 29 | 31 31 | 61 61 |
| 0x0000330 | 63 63 | 25 25 | 27 27 | 57 57 | 59 59 | 33 33 | 35 35 | 65 65 |
| 0x0000350 | 67 67 | 05 05 | 07 07 | 37 37 | 39 39 | 13 13 | 15 15 | 45 45 |
| 0x0000370 | 47 47 | 09 09 | 11 11 | 41 41 | 75 75 | 47 47 | 51 51 | 01 01 |
| 0x0000390 | 05 05 | 53 53 | 57 57 | 01 01 | 05 05 | 61 61 | 65 65 | 09 09 |
| 0x00003b0 | 13 13 | 55 55 | 59 59 | 03 03 | 07 07 | 63 63 | 67 67 | 11 11 |
| 0x00003d0 | 15 15 | 37 37 | 41 41 | 69 69 | 73 73 | 45 45 | 49 49 | 77 77 |
| 0x00003f0 | 03 03 | 39 39 | 43 43 | 71 71 | 11 11 | 05 05 | 07 07 | 13 13 |



LifeScan, Inc. Ex. 1019
Page 972 of 1420
IPR2025-00103

# 3 ACCESS CODE SAMPLE DATA

## Different access codes (GIAC, DIACs, others...)

LAP with LSB as rightmost bit.

| | Bit transmit order on air | | |
|---|---|---|---|
| | ----------> | | |
| LAP: | Preamble: | Sync word: | Trailer: |
| 000000 | 5 | 7e7041e3 4000000d | 5 |
| ffffff | a | e758b522 7ffffff2 | a |
| 9e8b33 | 5 | 475c58cc 73345e72 | a |
| 9e8b34 | 5 | 28ed3c34 cb345e72 | a |
| 9e8b36 | 5 | 62337b64 1b345e72 | a |
| 9e8b39 | a | c05747b9 e7345e72 | a |
| 9e8b3d | 5 | 7084eab0 2f345e72 | a |
| 9e8b42 | 5 | 64c86d2b 90b45e72 | a |
| 9e8b48 | a | e3c3725e 04b45e72 | a |
| 9e8b4f | a | 8c7216a6 bcb45e72 | a |
| 9e8b57 | a | b2f16c30 fab45e72 | a |
| 9e8b60 | 5 | 57bd3b22 c1b45e72 | a |
| 9e8b6a | a | d0b62457 55b45e72 | a |
| 9e8b75 | a | 81843a39 abb45e72 | a |
| 9e8b81 | 5 | 0ca96681 e0745e72 | a |
| 9e8b8e | a | aecd5a5c 1c745e72 | a |
| 9e8b9c | 5 | 17453fbf ce745e72 | a |
| 9e8bab | a | f20968ad f5745e72 | a |
| 9e8bbb | 5 | 015f4a1e f7745e72 | a |
| 9e8bcc | a | d8c695a0 0cf45e72 | a |
| 9e8bde | 5 | 614ef043 def45e72 | a |
| 9e8bf1 | a | ba81ddc7 a3f45e72 | a |
| 9e8c05 | 5 | 64a7dc4f 680c5e72 | a |
| 9e8c1a | 5 | 3595c221 960c5e72 | a |
| 9e8c30 | a | cb35cc0d 830c5e72 | a |
| 9e8c47 | 5 | 12ac13b3 788c5e72 | a |
| 9e8c5f | 5 | 2c2f6925 3e8c5e72 | a |
| 9e8c78 | 5 | 3a351c84 078c5e72 | a |
| 9e8c92 | 5 | 7396d0f3 124c5e72 | a |
| 9e8cad | 5 | 5b0fdfc4 6d4c5e72 | a |
| 9e8cc9 | a | aea2eb38 e4cc5e72 | a |
| 9e8ce6 | 5 | 756dc6bc 99cc5e72 | a |
| 9e8d04 | 5 | 214cf934 882c5e72 | a |
| 9e8d23 | 5 | 37568c95 b12c5e72 | a |
| 9e8d43 | 5 | 72281560 f0ac5e72 | a |
| 9e8d64 | 5 | 643260c1 c9ac5e72 | a |
| 9e8d86 | a | e044f493 986c5e72 | a |
| 9e8da9 | 5 | 3b8bd917 e56c5e72 | a |
| 9e8dcd | a | ce26edeb 6cec5e72 | a |
| 9e8df2 | a | e6bfe2dc 13ec5e72 | a |
| 9e8e18 | a | 82dcde3d c61c5e72 | a |
| 9e8e3f | a | 94c6ab9c ff1c5e72 | a |

*Sample Data*

| | | | |
|---|---|---|---|
| 9e8e67 | a | 969059a6 799c5e72 | a |
| 9e8e90 | a | c4dfccef 425c5e72 | a |
| 9e8eba | 5 | 3a7fc2c3 575c5e72 | a |
| 9e8ee5 | 5 | 57985401 69dc5e72 | a |
| 9e8f11 | 5 | 0ae2a363 623c5e72 | a |
| 9e8f3e | a | d12d8ee7 1f3c5e72 | a |
| 9e8f6c | 5 | 547063a8 0dbc5e72 | a |
| 9e8f9b | 5 | 063ff6e1 367c5e72 | a |
| 9e8fcb | a | c9bc5cfe f4fc5e72 | a |
| 9e8ffc | 5 | 2cf00bec cffc5e72 | a |
| 9e902e | a | 8ec5052f 5d025e72 | a |
| 9e9061 | 5 | 1074b15e 61825e72 | a |
| 9e9095 | a | 9d59ede6 2a425e72 | a |
| 9e90ca | a | f0be7b24 14c25e72 | a |
| 9e9100 | 5 | 10e10dd0 c0225e72 | a |
| 9e9137 | a | f5ad5ac2 fb225e72 | a |
| 9e916f | a | f7fba8f8 7da25e72 | a |
| 9e91a8 | 5 | 2f490e5b c5625e72 | a |
| 9e91e2 | a | 94979982 91e25e72 | a |
| 9e921d | 5 | 26cda478 2e125e72 | a |
| 9e9259 | a | aacb81dd 26925e72 | a |
| 9e9296 | a | bfac7f5b da525e72 | a |
| 9e92d4 | a | c9a7b0a7 cad25e72 | a |
| 9e9313 | a | c142bdde 32325e72 | a |
| 616cec | 5 | 586a491f 0dcda18d | 5 |
| 616ceb | 5 | 37db2de7 b5cda18d | 5 |
| 616ce9 | 5 | 7d056ab7 65cda18d | 5 |
| 616ce6 | a | df61566a 99cda18d | 5 |
| 616ce2 | 5 | 6fb2fb63 51cda18d | 5 |
| 616cdd | 5 | 472bf454 2ecda18d | 5 |
| 616cd7 | a | c020eb21 bacda18d | 5 |
| 616cd0 | a | af918fd9 02cda18d | 5 |
| 616cc8 | a | 9112f54f 44cda18d | 5 |
| 616cbf | 5 | 488b2af1 bf4da18d | 5 |
| 616cb5 | a | cf803584 2b4da18d | 5 |
| 616caa | a | 9eb22bea d54da18d | 5 |
| 616c9e | a | a49cb509 9e4da18d | 5 |
| 616c91 | 5 | 06f889d4 624da18d | 5 |
| 616c83 | a | bf70ec37 b04da18d | 5 |
| 616c74 | a | ed3f797e 8b8da18d | 5 |
| 616c64 | 5 | 1e695bcd 898da18d | 5 |
| 616c53 | a | fb250cdf b28da18d | 5 |
| 616c41 | 5 | 42ad693c 608da18d | 5 |
| 616c2e | a | a5b7cc14 dd0da18d | 5 |
| 616c1a | a | 9f9952f7 960da18d | 5 |
| 616c05 | a | ceab4c99 680da18d | 5 |
| 616bef | a | d403ddde fdf5a18d | 5 |
| 616bd8 | 5 | 314f8acc c6f5a18d | 5 |
| 616bc0 | 5 | 0fccf05a 80f5a18d | 5 |
| 616ba7 | 5 | 25030d57 7975a18d | 5 |
| 616b8d | a | dba3037b 6c75a18d | 5 |
| 616b72 | 5 | 4439ce17 13b5a18d | 5 |

*Sample Data*

| | | | |
|---|---|---|---|
| 616b56 | a | 8d417247 5ab5a18d | 5 |
| 616b39 | 5 | 6a5bd76f e735a18d | 5 |
| 616b1b | 5 | 592e8166 b635a18d | 5 |
| 616afc | 5 | 28609d46 cfd5a18d | 5 |
| 616adc | 5 | 51cb8c1f 4ed5a18d | 5 |
| 616abb | 5 | 7b047112 b755a18d | 5 |
| 616a99 | 5 | 4871271b e655a18d | 5 |
| 616a76 | 5 | 24bdc8c4 9b95a18d | 5 |
| 616a52 | a | edc57494 d295a18d | 5 |
| 616a2d | a | f989f30f 6d15a18d | 5 |
| 616a07 | 5 | 0729fd23 7815a18d | 5 |
| 6169e0 | a | 8bf0ba4f 81e5a18d | 5 |
| 6169b8 | a | 89a64875 0765a18d | 5 |
| 61698f | 5 | 6cea1f67 3c65a18d | 5 |
| 616965 | 5 | 2549d310 29a5a18d | 5 |
| 61693a | 5 | 48ae45d2 1725a18d | 5 |
| 61690e | 5 | 7280db31 5c25a18d | 5 |
| 6168e1 | a | ce1b9f34 61c5a18d | 5 |
| 6168b3 | 5 | 4b46727b 7345a18d | 5 |
| 616884 | a | ae0a2569 4845a18d | 5 |
| 616854 | a | ea5fc581 4a85a18d | 5 |
| 616823 | 5 | 33c61a3f b105a18d | 5 |
| 6167f1 | a | c49fb8c5 63f9a18d | 5 |
| 6167be | 5 | 5a2e0cb4 5f79a18d | 5 |
| 61678a | 5 | 60009257 1479a18d | 5 |
| 616755 | a | 86314e62 eab9a18d | 5 |
| 61671f | 5 | 3defd9bb be39a18d | 5 |
| 6166e8 | a | bff7e728 c5d9a18d | 5 |
| 6166b0 | a | bda11512 4359a18d | 5 |
| 616677 | 5 | 6513b3b1 fb99a18d | 5 |
| 61663d | a | decd2468 af19a18d | 5 |
| 616602 | a | f6542b5f d019a18d | 5 |
| 6165c6 | a | dc44b49b d8e9a18d | 5 |
| 616589 | 5 | 42f500ea e469a18d | 5 |
| 61654b | a | bf2885e1 34a9a18d | 5 |
| 61650c | a | ec4c69b5 4c29a18d | 5 |



LifeScan, Inc. Ex. 1019
Page 975 of 1420
IPR2025-00103

# 4 HEC AND PACKET HEADER SAMPLE DATA

This section contains examples of HECs computed for sample UAP and packet header contents (Data). The resulting 54 bit packet headers are shown in the rightmost column. Note that the UAP, Data and HEC values are in hexadecimal notation, while the header is in octal notation. The header is transmitted from left to right over the air.

| UAP | Data | HEC | Header (octal) |
|---|---|---|---|
| 00 | 123 | e1 | 770007 007070 000777 |
| 47 | 123 | 06 | 770007 007007 700000 |
| 00 | 124 | 32 | 007007 007007 007700 |
| 47 | 124 | d5 | 007007 007070 707077 |
| 00 | 125 | 5a | 707007 007007 077070 |
| 47 | 125 | bd | 707007 007070 777707 |
| 00 | 126 | e2 | 077007 007007 000777 |
| 47 | 126 | 05 | 077007 007070 700000 |
| 00 | 127 | 8a | 777007 007007 070007 |
| 47 | 127 | 6d | 777007 007070 770770 |
| 00 | 11b | 9e | 770770 007007 777007 |
| 47 | 11b | 79 | 770770 007070 077770 |
| 00 | 11c | 4d | 007770 007070 770070 |
| 47 | 11c | aa | 007770 007007 070707 |
| 00 | 11d | 25 | 707770 007070 700700 |
| 47 | 11d | c2 | 707770 007007 000077 |
| 00 | 11e | 9d | 077770 007070 777007 |
| 47 | 11e | 7a | 077770 007007 077770 |
| 00 | 11f | f5 | 777770 007070 707777 |
| 47 | 11f | 12 | 777770 007007 007000 |



LifeScan, Inc. Ex. 1019
Page 976 of 1420
IPR2025-00103

# 5 CRC SAMPLE DATA

This section shows the CRC computed for a sample 10 byte payload and a UAP of 0x47.

Data:
-----

data[0] = 0x4e
data[1] = 0x01
data[2] = 0x02
data[3] = 0x03
data[4] = 0x04
data[5] = 0x05
data[6] = 0x06
data[7] = 0x07
data[8] = 0x08
data[9] = 0x09

UAP = 0x47

==> CRC = 6d d2

Codeword (hexadecimal notation):
--------------------------------

4e 01 02 03 04 05 06 07 08 09 6d d2

NB: Over the air each byte in the codeword
    is sent with the LSB first.



LifeScan, Inc. Ex. 1019
Page 977 of 1420
IPR2025-00103

# 6 COMPLETE SAMPLE PACKETS

## 6.1 EXAMPLE OF DH1 PACKET

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 011
TYPE = 0100 (DH1)
FLOW = 0
ARQN = 1
SEQN = 0

Payload: (MSB...LSB)
--------------------
payload length: 5 bytes
logical channel = 10 (UA/I, Start L2CAP message)
flow = 1
data byte 1 = 00000001
data byte 2 = 00000010
data byte 3 = 00000011
data byte 4 = 00000100
data byte 5 = 00000101

HEC and CRC initialization: (MSB...LSB)
---------------------------------------
uap = 01000111

NO WHITENING USED




AIR DATA (LSB...MSB)

Packet header (incl HEC):
-------------------------
111111000
000000111000
000111000
000111111000000000000000

Payload (incl payload header and CRC):
--------------------------------------------------------------
01110100
10000000
01000000
11000000
00100000
10100000
1110110000110110
```

## 6.2 EXAMPLE OF DM1 PACKET

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 011
TYPE = 0011 (DM1)
FLOW = 0
ARQN = 1
SEQN = 0

Payload: (MSB...LSB)
--------------------
payload length: 5 bytes
logical channel = 10 (UA/I, Start L2CAP message)
flow = 1
data byte 1 = 00000001
data byte 2 = 00000010
data byte 3 = 00000011
data byte 4 = 00000100
data byte 5 = 00000101

HEC and CRC initialization: (MSB...LSB)
---------------------------------------
uap = 01000111

NO WHITENING USED




AIR DATA (LSB...MSB)

Packet header (incl HEC):
-------------------------
111111000
111111000000
000111000
111000000111111111111000

Payload (incl payload header, FEC23, CRC and 6 padded zeros):
-------------------------------------------------------------
0111010010 11001
0000000100 01011
0000110000 11110
0000100000 00111
1010000011 01100
1011000011 00010
0110000000 10001
```



LifeScan, Inc. Ex. 1019
Page 979 of 1420
IPR2025-00103

# 7 SIMPLE PAIRING SAMPLE DATA

This section provides sample data for the Simple Pairing cryptographic functions (f1, f2, f3, g and the ECDH calculations).

## 7.1 P-192 SAMPLE DATA

In each data set, the bytes are ordered from least significant on the right to most significant on the left.

### 7.1.1 Data Set 1

```
Private A:   07915f86918ddc27005df1d6cf0c142b625ed2eff4a518ff
Private B:   1e636ca790b50f68f15d8dbe86244e309211d635de00e16d
Public A(x): 15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
Public A(y): b09d42b81bc5bd009f79e4b59dbbaa857fca856fb9f7ea25
DHKey:       fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
```

### 7.1.2 Data Set 2

```
Private A:   52ec1ca6e0ec973c29065c3ca10be80057243002f09bb43e
Private B:   57231203533e9efe18cc622fd0e34c6a29c6e0fa3ab3bc53
Public A(x): 45571f027e0d690795d61560804da5de789a48f94ab4b07e
Public A(y): 0220016e8a6bce74b45ffec1e664aaa0273b7cbd907a8e2b
DHKey:       a20a34b5497332aa7a76ab135cc0c168333be309d463c0c0
```

### 7.1.3 Data Set 3

```
Private A:   00a0df08eaf51e6e7be519d67c6749ea3f4517cdd2e9e821
Private B:   2bf5e0d1699d50ca5025e8e2d9b13244b4d322a328be1821
Public A(x): 2ed35b430fa45f9d329186d754eeeb0495f0f653127f613d
Public A(y): 27e08db74e424395052ddae7e3d5a8fecb52a8039b735b73
DHKey:       3b3986ba70790762f282a12a6d3bcae7a2ca01e25b87724e
```

### 7.1.4 Data Set 4

```
Private A:   030a4af66e1a4d590a83e0284fca5cdf83292b84f4c71168
Private B:   12448b5c69ecd10c0471060f2bf86345c5e83c03d16bae2c
Public A(x): f24a6899218fa912e7e4a8ba9357cb8182958f9fa42c968c
Public A(y): 7c0b8a9ebe6ea92e968c3a65f9f1a9716fe826ad88c97032
DHKey:       4a78f83fba757c35f94abea43e92effdd2bc700723c61939
```

### 7.1.5 Data Set 5

```
Private A:   604df406c649cb460be16244589a40895c0db7367dc11a2f
Private B:   526c2327303cd505b9cf0c012471902bb9e842ce32b0addc
Public A(x): cbe3c629aceb41b73d475a79fbfe8c08cdc80ceec00ee7c9
```



LifeScan, Inc. Ex. 1019
Page 980 of 1420
IPR2025-00103

```
Public A(y): f9f70f7ae42abda4f33af56f7f6aa383354e453fa1a2bd18
DHKey:       64d4fe35567e6ea0ca31f947e1533a635436d4870ce88c45
```

### 7.1.6 Data Set 6

```
Private A:   1a2c582a09852979eb2cee18fb0befb9a55a6d06f6a8fad3
Private B:   243778916920d68df535955bc1a3cccd5811133a8205ae41
Public A(x): eca2d8d30bbef3ba8b7d591fdb98064a6c7b870cdcebe67c
Public A(y): 2e4163a44f3ae26e70dae86f1bf786e1a5db5562a8ed9fee
DHKey:       6433b36a7e9341940e78a63e31b3cf023282f7f1e3bf83bd
```

### 7.1.7 Data Set 7

```
Private A:   0f494dd08b493edb07228058a9f30797ff147a5a2adef9b3
Private B:   2da4cd46d9e06e81b1542503f2da89372e927877bececlbe
Public A(x): 9f56a8aa27346d66652a546abacc7d69c17fd66e0853989f
Public A(y): d7234c1464882250df7bbe67e0fa22aae475dc58af0c4210
DHKey:       c67beda9baf3c96a30616bf87a7d0ae704bc969e5cad354b
```

### 7.1.8 Data Set 8

```
Private A:   7381d2bc6ddecb65126564cb1af6ca1985d19fb57f0fff16
Private B:   18e276beff75adc3d520badb3806822e1c820f1064447848
Public A(x): 61c7f3c6f9e09f41423dce889de1973d346f2505a5a3b19b
Public A(y): 919972ff4cd6aed8a4821e3adc358b41f7be07ede20137df
DHKey:       6931496eef2fcfb03e0b1eef515dd4e1b0115b8b241b0b84
```

### 7.1.9 Data Set 9

```
Private A:   41c7b484ddc37ef6b7952c379f87593789dac6e4f3d8d8e6
Private B:   33e4eaa77f78216e0e99a9b200f81d2ca20dc74ad62d9b78
Public A(x): 9f09c773adb8e7b66b5d986cd15b143341a66d824113c15f
Public A(y): d2000a91738217ab8070a76c5f96c03de317dfab774f4837
DHKey:       a518f3826bb5fa3d5bc37da4217296d5b6af51e5445c6625
```

### 7.1.10 Data Set 10

```
Private A:   703cf5ee9c075f7726d0bb36d131c664f5534a6e6305d631
Private B:   757291c620a0e7e9dd13ce09ceb729c0ce1980e64d569b5f
Public A(x): fa2b96d382cf894aeeb0bd985f3891e655a6315cd5060d03
Public A(y): f7e8206d05c7255300cc56c88448158c497f2df596add7a2
DHKey:       12a3343bb453bb5408da42d20c2d0fcc18ff078f56d9c68c
```

## 7.2 HASH FUNCTIONS SAMPLE DATA

In each data set, the bytes are ordered from least significant on the right to most significant on the left.

### 7.2.1 f1()

```
Set 1a
          U:       15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          V:       356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          X:       d5cb8454d177733effffb2ec712baeab
          Z:       00
          output:  1bdc955a9d542ffc9f9e670cdf665010

Set 1b
          U:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          X:      d5cb8454d177733effffb2ec712baeab
          Z:      80
          output: 611325ebcb6e5269b868113306095fa6

Set 1c
          U:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          V:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          X:      d5cb8454d177733effffb2ec712baeab
          Z:      81
          output: b68df39fd8a406b06a6c517d3666cf91

Set 2a (swapped U and V inputs compared with set 1)
          U:       356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          V:       15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          X:       d5cb8454d177733effffb2ec712baeab
          Z:       00
          output:  f4e1ec4b88f305e81477627b1643a927

Set 2b (swapped U and V inputs compared with set 1)
          U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          V:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          X:      d5cb8454d177733effffb2ec712baeab
          Z:      80
          output: ac6aa7cfa96ae99dd3a74225adb068ae

Set 2c (swapped U and V inputs compared with set 1)
          U:      356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          V:      15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          X:      d5cb8454d177733effffb2ec712baeab
          Z:      81
          output: 5ad4721258aa1fa06082edad980d0cc5

Set 3a (U and V set to same value as U in set 1)
          U:       15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          V:       15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          X:       d5cb8454d177733effffb2ec712baeab
          Z:       00
          output:  49125fc1e8cdc615826c15e5d23ede41
```

```
Set 3b (U and V set to same value as V in set 1)
          U:       356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          V:       356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          X:       d5cb8454d177733effffb2ec712baeab
          Z:       80
          output:  159f204c520565175c2b9c523acad2eb

Set 3c (U and V set to same value as V in set 1)
          U:       356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          V:       356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          X:       d5cb8454d177733effffb2ec712baeab
          Z:       81
          output:  9a162ff9a8235e5b12539ba0ff9179da
```

### 7.2.2 g()

```
Set 1
          U:15207009984421a6586f9fc3fe7e4329d2809ea51125f8ed
          V:356b31938421fbbf2fb331c89fd588a69367e9a833f56812
          X:        d5cb8454d177733effffb2ec712baeab
          Y:        a6e8e7cc25a75f6e216583f7ff3dc4cf
          output:             52146a1e
          output (decimal):   1377069598
          6 digits (decimal): 069598
```

### 7.2.3 f2()

```
Set 1
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          keyID:  62746c6b
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: c234c1198f3b520186ab92a2f874934e
```

### 7.2.4 f3()

```
Set 1
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000000
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 5e6a346b8add7ee80e7ec0c2461b1509
```

```
Set 2
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000001
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 7840e5445a13e3ce6e48a2decbe51482

Set 3
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000002
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: da9afb5c6c9dbe0af4722b532520c4b3


Set 4
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000003
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 2c0f220c50075285852e01bcee4b5f90

Set 5
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000100
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 0a096af0fa61dce0933987febe95fc7d

Set 6
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000101
          A1:     56123737bfce
```

```
          A2:     a713702dcfc1
          output: 49b8d74007888e770e1a49d6810069b9

Set 7
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000102
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 309cd0327dec2514894a0c88b101a436

Set 8
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000103
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 4512274ba875b156c2187e2061b90434

Set 9 (same as set 1 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000000
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 8d56dc59e70855f563b5e85e42d5964e

Set 10 (same as set 2 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000001
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: c92fdacbf0ce931e9c4087a9dfb7bc0b

Set 11 (same as set 3 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000002
```

```
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 52ac910200dc34285bbbf2144883c498


Set 12 (same as set 4 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000003
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: c419d677e0d426e6bb36de5fa54c5041

Set 13 (same as set 5 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000100
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: fb0e1f9f7c623c1bf2675fcff1551137

Set 14 (same as set 6 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000101
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 16c7be68184f1170fbbb2bef5a9c515d

Set 15 (same as set 7 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000102
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 24849f33d3ac05fef9034c18d9adb310

Set 16 (same as set 8 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
```

```
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  000103
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: e0f484bb0b071483285903e85094046b


Set 17
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010000
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 4bf22677415ed90aceb21873c71c1884

Set 18
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010001
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 0d4b97992eb570efb369cfe45e1681b5

Set 19
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010002
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 0f0906bbfa75e95c471e97c4211b2741


Set 20
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010003
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 88f1f60ce1ff4bf8aa08a170dd061d4e

Set 21
```

```
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010100
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 940f88f25317c358d9bd2f146778e36b

Set 22
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010101
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: 591396355ac4dc72be05a15e718ec945

Set 23
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010102
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: a3dc055f656abb1d6e11d3f37340189a

Set 24
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     d5cb8454d177733effffb2ec712baeab
          N2:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010103
          A1:     56123737bfce
          A2:     a713702dcfc1
          output: fd5412a22ba5dd893852608f8ab0c934

Set 25 (same as set 1 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010000
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 2a742039c5fd6c6faafce17b619ac56f
```

*Sample Data*

```
Set 26 (same as set 2 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010001
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: a60d89efb52db7905179a6c63b8f212a

Set 27 (same as set 3 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010002
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: cb02f803d755fd936f0a832f4ee9fd4a


Set 28 (same as set 4 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010003
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 00786c04a24561485aaf22808871b874

Set 29 (same as set 5 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010100
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 2a58ef2d99281d472a88027423f8215f

Set 30 (same as set 6 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010101
          A1:     a713702dcfc1
          A2:     56123737bfce
```

```
          output: ff7ab3752a144232f2cbbcbf979f5517

Set 31 (same as set 7 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010102
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 9d7fccef23625b1cc684fbf3f8b0e182

Set 32 (same as set 8 with N1 and N2 swapped and A1 and A2 swapped)
          W:      fb3ba2012c7e62466e486e229290175b4afebc13fdccee46
          N1:     a6e8e7cc25a75f6e216583f7ff3dc4cf
          N2:     d5cb8454d177733effffb2ec712baeab
          R:      12a3343bb453bb5408da42d20c2d0fc8
          IOcap:  010103
          A1:     a713702dcfc1
          A2:     56123737bfce
          output: 864f87e26dece4dfdde30ade1e463d4f
```

# 8 WHITENING SEQUENCE SAMPLE DATA

This section shows the output of the whitening sequence generator.

| Whitening Sequence (=D7) | Whitening LFSR D7.....D0 |
|---|---|
| 1 | 1111111 |
| 1 | 1101111 |
| 1 | 1001111 |
| 0 | 0001111 |
| 0 | 0011110 |
| 0 | 0111100 |
| 1 | 1111000 |
| 1 | 1100001 |
| 1 | 1010011 |
| 0 | 0110111 |
| 1 | 1101110 |
| 1 | 1001101 |
| 0 | 0001011 |
| 0 | 0010110 |
| 0 | 0101100 |
| 1 | 1011000 |
| 0 | 0100001 |
| 1 | 1000010 |
| 0 | 0010101 |
| 0 | 0101010 |
| 1 | 1010100 |
| 0 | 0111001 |
| 1 | 1110010 |
| 1 | 1110101 |
| 1 | 1111011 |
| 1 | 1100111 |
| 1 | 1011111 |
| 0 | 0101111 |
| 1 | 1011110 |
| 0 | 0101101 |
| 1 | 1011010 |
| 0 | 0100101 |
| 1 | 1001010 |
| 0 | 0000101 |
| 0 | 0001010 |
| 0 | 0010100 |
| 0 | 0101000 |
| 1 | 1010000 |
| 0 | 0110001 |
| 1 | 1100010 |
| 1 | 1010101 |
| 0 | 0111011 |
| 1 | 1110110 |

*Sample Data*

| | |
|---|---|
| 1 | 1111101 |
| 1 | 1101011 |
| 1 | 1000111 |
| 0 | 0011111 |
| 0 | 0111110 |
| 1 | 1111100 |
| 1 | 1101001 |
| 1 | 1000011 |
| 0 | 0010111 |
| 0 | 0101110 |
| 1 | 1011100 |
| 0 | 0101001 |
| 1 | 1010010 |
| 0 | 0110101 |
| 1 | 1101010 |
| 1 | 1000101 |
| 0 | 0011011 |
| 0 | 0110110 |
| 1 | 1101100 |
| 1 | 1001001 |
| 0 | 0000011 |
| 0 | 0000110 |
| 0 | 0001100 |
| 0 | 0011000 |
| 0 | 0110000 |
| 1 | 1100000 |
| 1 | 1010001 |
| 0 | 0110011 |
| 1 | 1100110 |
| 1 | 1011101 |
| 0 | 0101011 |
| 1 | 1010110 |
| 0 | 0111101 |
| 1 | 1111010 |
| 1 | 1100101 |
| 1 | 1011011 |
| 0 | 0100111 |
| 1 | 1001110 |
| 0 | 0001101 |
| 0 | 0011010 |
| 0 | 0110100 |
| 1 | 1101000 |
| 1 | 1000001 |
| 0 | 0010011 |
| 0 | 0100110 |
| 1 | 1001100 |
| 0 | 0001001 |
| 0 | 0010010 |
| 0 | 0100100 |
| 1 | 1001000 |
| 0 | 0000001 |
| 0 | 0000010 |

*Sample Data*

| | |
|---|---|
| 0 | 0000100 |
| 0 | 0001000 |
| 0 | 0010000 |
| 0 | 0100000 |
| 1 | 1000000 |
| 0 | 0010001 |
| 0 | 0100010 |
| 1 | 1000100 |
| 0 | 0011001 |
| 0 | 0110010 |
| 1 | 1100100 |
| 1 | 1011001 |
| 0 | 0100011 |
| 1 | 1000110 |
| 0 | 0011101 |
| 0 | 0111010 |
| 1 | 1110100 |
| 1 | 1111001 |
| 1 | 1100011 |
| 1 | 1010111 |
| 0 | 0111111 |
| 1 | 1111110 |
| 1 | 1101101 |
| 1 | 1001011 |
| 0 | 0000111 |
| 0 | 0001110 |
| 0 | 0011100 |
| 0 | 0111000 |
| 1 | 1110000 |
| 1 | 1110001 |
| 1 | 1110011 |
| 1 | 1110111 |
| 1 | 1111111 |

# 9 FEC SAMPLE DATA

```
================================================
 Rate 2/3 FEC -- (15,10) Shortened Hamming Code
================================================

Data is in hexadecimal notation, the codewords are in binary notation. The codeword bits
are sent from left to right over the air interface. The space in the codeword indicates
the start of parity bits.

Data:           Codeword:

0x001       1000000000 11010
0x002       0100000000 01101
0x004       0010000000 11100
0x008       0001000000 01110
0x010       0000100000 00111
0x020       0000010000 11001
0x040       0000001000 10110
0x080       0000000100 01011
0x100       0000000010 11111
0x200       0000000001 10101
```



LifeScan, Inc. Ex. 1019
Page 994 of 1420
IPR2025-00103

# 10 ENCRYPTION KEY SAMPLE DATA

Explanation:

For the sections 9.1 - 9.5, the hexadecimal sample data is written with the least significant byte at the leftmost position and the most significant byte at the rightmost position. Within each byte, the *least significant bit* (LSB) is at the rightmost position and the *most significant bit* (MSB) is at the leftmost position Thus, a line reading:

aco: 48afcdd4bd40fef76693b113

means aco[0]=0x48, ac[1]=0xaf, ..., aco[11]=0x13. The LSB of aco[11] is '1' and the MSB of aco[11] is '0'.

Key [ i]: denotes the ith sub-key in Ar or A'r;
round r: denotes the input to the rth round;
added ->: denotes the input to round 3 in A'r after adding original input (of round 1).

## 10.1 FOUR TESTS OF E1

```
rand    :00000000000000000000000000000000
address :000000000000
key     :00000000000000000000000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000000
Key [ 2]:4697b1baa3b7100ac537b3c95a28ac64
round  2:78d19f9307d2476a523ec7a8a026042a
Key [ 3]:ecabaac66795580df89af66e66dc053d
Key [ 4]:8ac3d8896ae9364943bfebd4969b68a0
round  3:600265247668dda0e81c07bbb30ed503
Key [ 5]:5d57921fd5715cbb22c1be7bbc996394
Key [ 6]:2a61b8343219fdfb1740e6511d41448f
round  4:d7552ef7cc9dbde568d80c2215bc4277
Key [ 7]:dd0480dee731d67f01a2f739da6f23ca
Key [ 8]:3ad01cd1303e12a1cd0fe0a8af82592c
round  5:fb06bef32b52ab8f2a4f2b6ef7f6d0cd
Key [ 9]:7dadb2efc287ce75061302904f2e7233
Key [10]:c08dcfa981e2c4272f6c7a9f52e11538
round  6:b46b711ebb3cf69e847a75f0ab884bdd
Key [11]:fc2042c708e409555e8c147660ffdfd7
Key [12]:fa0b21001af9a6b9e89e624cd99150d2
round  7:c585f308ff19404294f06b292e978994
Key [13]:18b40784ea5ba4c80ecb48694b4e9c35
Key [14]:454d54e5253c0c4a8b3fcca7db6baef4
round  8:2665fadb13acf952bf74b4ab12264b9f
Key [15]:2df37c6d9db52674f29353b0f011ed83
Key [16]:b60316733b1e8e70bd861b477e2456f1
Key [17]:884697b1baa3b7100ac537b3c95a28ac
```



LifeScan, Inc. Ex. 1019
Page 995 of 1420
IPR2025-00103

*Sample Data*

```
round  1:158ffe43352085e8a5ec7a88e1ff2ba8
Key [ 1]:e9e5dfc1b3a79583e9e5dfc1b3a79583
Key [ 2]:7595bf57e0632c59f435c16697d4c864
round  2:0b5cc75febcdf7827ca29ec0901b6b5b
Key [ 3]:e31b96afcc75d286ef0ae257cbbc05b7
Key [ 4]:0d2a27b471bc0108c6263aff9d9b3b6b
round  3:e4278526c8429211f7f2f0016220aef4
added ->:f1b68365fd6217f952de6a89831fd95c
Key [ 5]:98d1eb5773cf59d75d3b17b3bc37c191
Key [ 6]:fd2b79282408ddd4ea0aa7511133336f
round  4:d0304ad18337f86040145d27aa5c8153
Key [ 7]:331227756638a41d57b0f7e071ee2a98
Key [ 8]:aa0dd8cc68b406533d0f1d64aabacf20
round  5:84db909d213bb0172b8b6aaf71bf1472
Key [ 9]:669291b0752e63f806fce76f10e119c8
Key [10]:ef8bdd46be8ee0277e9b78adef1ec154
round  6:f835f52921e903dfa762f1df5abd7f95
Key [11]:f3902eb06dc409cfd78384624964bf51
Key [12]:7d72702b21f97984a721c99b0498239d
round  7:ae6c0b4bb09f25c6a5d9788a31b605d1
Key [13]:532e60bceaf902c52a06c2c283ecfa32
Key [14]:181715e5192efb2a64129668cf5d9dd4
round  8:744a6235b86cc0b853cc9f74f6b65311
Key [15]:83017c1434342d4290e961578790f451
Key [16]:2603532f365604646ff65803795ccce5
Key [17]:882f7c907b565ea58dae1c928a0dcf41
sres    :056c0fe6
aco     :48afcdd4bd40fef76693b113
----------------------------------------
rand    :bc3f30689647c8d7c5a03ca80a91eceb
address :7ca89b233c2d
key     :159dd9f43fc3d328efba0cd8a861fa57
round  1:bc3f30689647c8d7c5a03ca80a91eceb
Key [ 1]:159dd9f43fc3d328efba0cd8a861fa57
Key [ 2]:326558b3c15551899a97790e65ff669e
round  2:3e950edf197615638cc19c09f8fedc9b
Key [ 3]:62e879b65b9f53bbfbd020c624b1d682
Key [ 4]:73415f30bac8ab61f410adc9442992db
round  3:6a7640791cb536678936c5ecd4ae5a73
Key [ 5]:5093cfa1d31c1c48acd76df030ea3c31
Key [ 6]:0b4acc2b8f1f694fc7bd91f4a70f3009
round  4:fca2c022a577e2ffb2aa007589693ec7
Key [ 7]:2ca43fc817947804ecff148d50d6f6c6
Key [ 8]:3fcd73524b533e00b7f7825bea2040a4
round  5:e97f8ea4ed1a6f4a36ffc179dc6bb563
Key [ 9]:6c67bec76ae8c8cc4d289f69436d3506
Key [10]:95ed95ee8cb97e61d75848464bffb379
round  6:38b07261d7340d028749de1773a415c7
Key [11]:ff566c1fc6b9da9ac502514550f3e9d2
Key [12]:ab5ce3f5c887d0f49b87e0d380e12f47
round  7:58241f1aed7c1c3e047d724331a0b774
Key [13]:a2cab6f95eac7d655dbe84a6cd4c47f5
```

*Sample Data*

```
Key [14]:f5caff88af0af8c42a20b5bbd2c8b460
round  8:3d1aaeff53c0910de63b9788b13c490f
Key [15]:185099c1131cf97001e2f36fda415025
Key [16]:a0ebb82676bc75e8378b189eff3f6b1d
Key [17]:cf5b348aaee27ae332b4f1bfa10289a6
round  1:2e4b417b9a2a9cfd7d8417d9a6a556eb
Key [ 1]:fe78b835f26468ab069fd3991b086fda
Key [ 2]:095c5a51c6fa6d3ac1d57fa19aa382bd
round  2:b8bca81d6bb45af9d92beadd9300f5ed
Key [ 3]:1af866df817fd9f4ec00bc704192cffc
Key [ 4]:f4a8a059c1f575f076f5fbb24bf16590
round  3:351aa16dec2c3a4787080249ed323eae
added ->:1b65e2167656d6bafa8c19904bd79445
Key [ 5]:8c9d18d9356a9954d341b4286e88ea1f
Key [ 6]:5c958d370102c9881bf753e69c7da029
round  4:2ce8fef47dda6a5bee74372e33e478a2
Key [ 7]:7eb2985c3697429fbe0da334bb51f795
Key [ 8]:af900f4b63a1138e2874bfb7c628b7b8
round  5:572787f563e1643c1c862b7555637fb4
Key [ 9]:834c8588dd8f3d4f31117a488420d69b
Key [10]:bc2b9b81c15d9a80262f3f48e9045895
round  6:16b4968c5d02853c3a43aa4cdb5f26ac
Key [11]:f08608c9e39ad3147cba61327919c958
Key [12]:2d4131decf4fa3a959084714a9e85c11
round  7:10e4120c7cccef9dd4ba4e6da8571b01
Key [13]:c934fd319c4a2b5361fa8eef05ae9572
Key [14]:4904c17aa47868e40471007cde3a97c0
round  8:f9081772498fed41b6ffd72b71fcf6c6
Key [15]:ea5e28687e97fa3f833401c86e6053ef
Key [16]:1168f58252c4ecfccafbdb3af857b9f2
Key [17]:b3440f69ef951b78b5cbd6866275301b
sres    :8d5205c5
aco     :3ed75df4abd9af638d144e94
----------------------------------------
rand    :0891caee063f5da1809577ff94ccdcfb
address :c62f19f6ce98
key     :45298d06e46bac21421ddfbed94c032b
round  1:0891caee063f5da1809577ff94ccdcfb
Key [ 1]:45298d06e46bac21421ddfbed94c032b
Key [ 2]:8f03e1e1fe1c191cad35a897bc400597
round  2:1c6ca013480a685c1b28e0317f7167e1
Key [ 3]:4f2ce3a092dde854ef496c8126a69e8e
Key [ 4]:968caee2ac6d7008c07283daec67f2f2
round  3:06b4915f5fcc1fc551a52048f0af8a26
Key [ 5]:ab0d5c31f94259a6bf85ee2d22edf56c
Key [ 6]:dfb74855c0085ce73dc17b84bfd50a92
round  4:077a92b040acc86e6e0a877db197a167
Key [ 7]:8f888952662b3db00d4e904e7ea53b5d
Key [ 8]:5e18bfcc07799b0132db88cd6042f599
round  5:7204881fb300914825fdc863e8ceadf3
Key [ 9]:bfca91ad9bd3d1a06c582b1d5512dddf
Key [10]:a88bc477e3fa1d5a59b5e6cf793c7a41
```

*Sample Data*

```
round  6:27031131d86cea2d747deb4f756143aa
Key [11]:f3cfb8dac8aea2a6a8ef95af3a2a2767
Key [12]:77beb90670c5300b03aa2b2232d3d40c
round  7:fc8c13d49149b1ce8d86f96e44a00065
Key [13]:b578373650af36a06e19fe335d726d32
Key [14]:6bcee918c7d0d24dfdf42237fcf99d53
round  8:04ef5f5a7ddf846cda0a07782fc23866
Key [15]:399f158241eb3e079f45d7b96490e7ea
Key [16]:1bcfbe98ecde2add52aa63ea79fb917a
Key [17]:ee8bc03ec08722bc2b075492873374af
round  1:d989d7a40cde7032d17b52f8117b69d5
Key [ 1]:2ecc6cc797cc41a2ab02007f6af396ae
Key [ 2]:acfaef7609c12567d537ae1cf9dc2198
round  2:8e76eb9a29b2ad5eea790db97aee37c1
Key [ 3]:079c8ff9b73d428df879906a0b87a6c8
Key [ 4]:19f2710baf403a494193d201f3a8c439
round  3:346bb7c35b2539676375aafe3af69089
added ->:edf48e675703a955b2f0fc062b71f95c
Key [ 5]:d623a6498f915cb2c8002765247b2f5a
Key [ 6]:900109093319bc30108b3d9434a77a72
round  4:fafb6c1f3ebbd2477be2da49dd923f69
Key [ 7]:e28e2ee6e72e7f4e5b5c11f10d204228
Key [ 8]:8e455cd11f8b9073a2dfa5413c7a4bc5
round  5:7c72230df588060a3cf920f9b0a08f06
Key [ 9]:28afb26e2c7a64238c41cefc16c53e74
Key [10]:d08dcafc2096395ba0d2dddd0e471f4d
round  6:55991df991db26ff00073a12baa3031d
Key [11]:fcffdcc3ad8faae091a7055b934f87c1
Key [12]:f8df082d77060252c02d91e55bd6a7d6
round  7:70ec682ff864375f63701fa4f6be5377
Key [13]:bef3706e523d708e8a44147d7508bc35
Key [14]:3e98ab283ca2422d56a56cf8b06caeb3
round  8:172f12ec933da85504b4ea5c90f8f0ea
Key [15]:87ad9625d06645d22598dd5ef811ea2c
Key [16]:8bd3db0cc8168009e5da90877e13a36f
Key [17]:0e74631d813a8351ac7039b348c41b42
sres    :00507e5f
aco     :2a5f19fbf60907e69f39ca9f
----------------------------------------
rand    :0ecd61782b4128480c05dc45542b1b8c
address :f428f0e624b3
key     :35949a914225fabad91995d226de1d92
round  1:0ecd61782b4128480c05dc45542b1b8c
Key [ 1]:35949a914225fabad91995d226de1d92
Key [ 2]:ea6b3dcccc8ee5d88de349fa5010404f
round  2:8935e2e263fbc4b9302cabdfc06bce3e
Key [ 3]:920f3a0f2543ce535d4e7f25ad80648a
Key [ 4]:ad47227edf9c6874e80ba80ebb95d2c9
round  3:b4c8b878675f184a0c72f3dab51f8f05
Key [ 5]:81a941ca7202b5e884ae8fa493ecac3d
Key [ 6]:bcde1520bee3660e86ce2f0fb78b9157
round  4:77ced9f2fc42bdd5c6312b87fc2377c5
```

*Sample Data*

```
Key [ 7]:c8eee7423d7c6efa75ecec0d2cd969d3
Key [ 8]:910b3f838a02ed441fbe863a02b4a1d0
round  5:fe28e8056f3004d60bb207e628b39cf2
Key [ 9]:56c647c1e865eb078348962ae070972d
Key [10]:883965da77ca5812d8104e2b640aec0d
round  6:1f2ba92259d9e88101518f145a33840f
Key [11]:61d4cb7e4f8868a283327806a9bd8d4d
Key [12]:9f57de3a3ff310e21dc1e696ce060304
round  7:cc9b5d0218d29037e88475152ebebb2f
Key [13]:7aa1d8adc1aeed7127ef9a18f6eb2d8e
Key [14]:b4db9da3bf865912acd14904c7f7785d
round  8:b04d352bedc02682e4a7f59d7cda1dba
Key [15]:a13d7141ef1f6c7d867e3d175467381b
Key [16]:08b2bc058e50d6141cdd566a307e1acc
Key [17]:057b2b4b4be5dc0ac49e50489b8006c9
round  1:5cfacc773bae995cd7f1b81e7c9ec7df
Key [ 1]:1e717950f5828f3930fe4a9395858815
Key [ 2]:d1623369b733d98bbc894f75866c544c
round  2:d571ffa21d9daa797b1a0a3c962fc64c
Key [ 3]:4abf27664ae364cc8a7e5bcf88214cc4
Key [ 4]:2aaedda8dc4933dd6aeaf6e5c0d5a482
round  3:e17c8e498a00f125bf654c938c23f36d
added ->:bd765a3eb1ae8a796856048df0c1bab2
Key [ 5]:bc7f8ab2d86000f47b1946cc8d7a7a2b
Key [ 6]:6b28544cb13ec6c5d98470df2cf900b7
round  4:a9727c26f2f06bd9920e83c8605dcd76
Key [ 7]:1be840d9107f2c9523f66bb19f5464a1
Key [ 8]:61d6fb1aa2f0c2b26fb2a3d6de8c177c
round  5:aeff751f146eab7e4626b2e2c9e2fb39
Key [ 9]:adabfc82570c568a233173099f23f4c2
Key [10]:b7df6b55ad266c0f1ff7452101f59101
round  6:cf412b95f454d5185e67ca671892e5bd
Key [11]:8e04a7282a2950dcbaea28f300e22de3
Key [12]:21362c114433e29bda3e4d51f803b0cf
round  7:16165722fe4e07ef88f8056b17d89567
Key [13]:710c8fd5bb3cbb5f132a7061de518bd9
Key [14]:0791de7334f4c87285809343f3ead3bd
round  8:28854cd6ad4a3c572b15490d4b81bc3f
Key [15]:4f47f0e5629a674bfcd13770eb3a3bd9
Key [16]:58a6d9a16a284cc0aead2126c79608a1
Key [17]:a564082a0a98399f43f535fd5cefad34
sres    :80e5629c
aco     :a6fe4dcde3924611d3cc6ba1


========================================
```

## 10.2 FOUR TESTS OF E21

```
rand    :00000000000000000000000000000000
address :000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000006
Key [ 2]:4697b1baa3b7100ac537b3c95a28dc94
round  2:98611307ab76bbde9a86af1ce8cad412
Key [ 3]:ecabaac66795580df89af66e665d863d
Key [ 4]:8ac3d8896ae9364943bfebd4a2a768a0
round  3:820999ad2e6618f4b578974beeedf9e7
added ->:820999ad2e6618f4b578974beeedf9e7
Key [ 5]:5d57921fd5715cbb22c1bedb1c996394
Key [ 6]:2a61b8343219fdfb1740e9541d41448f
round  4:acd6edec87581ac22dbdc64ea4ced3a2
Key [ 7]:dd0480dee731d67f01ba0f39da6f23ca
Key [ 8]:3ad01cd1303e12a18dcfe0a8af82592c
round  5:1c7798732f09fbfe25795a4a2fbc93c2
Key [ 9]:7dadb2efc287ce7b0c1302904f2e7233
Key [10]:c08dcfa981e2f4572f6c7a9f52e11538
round  6:c05b88b56aa70e9c40c79bb81cd911bd
Key [11]:fc2042c708658a555e8c147660ffdfd7
Key [12]:fa0b21002605a6b9e89e624cd99150d2
round  7:abacc71b481c84c798d1bdf3d62f7e20
Key [13]:18b407e44a5ba4c80ecb48694b4e9c35
Key [14]:454d57e8253c0c4a8b3fcca7db6baef4
round  8:e8204e1183ae85cf19edb2c86215b700
Key [15]:2d0b946d9db52674f29353b0f011ed83
Key [16]:76c316733b1e8e70bd861b477e2456f1
Key [17]:8e4697b1baa3b7100ac537b3c95a28ac
Ka      :d14ca028545ec262cee700e39b5c39ee
----------------------------------------
rand    :2dd9a550343191304013b2d7e1189d09
address :cac4364303b6
round  1:cac4364303b6cac4364303b6cac43643
Key [ 1]:2dd9a550343191304013b2d7e1189d0f
Key [ 2]:14c4335b2c43910c5dcc71d81a14242b
round  2:e169f788aad45a9011f11db5270b1277
Key [ 3]:55bfb712cba168d1a48f6e74cd9f4388
Key [ 4]:2a2b3aacca695caef2821b0fb48cc253
round  3:540f9c76652e92c44987c617035037bf
added ->:9ed3d23566e45c007fcac9a1c9146dfc
Key [ 5]:a06aab22d9a287384042976b4b6b00ee
Key [ 6]:c229d054bb72e8eb230e6dcdb32d16b7
round  4:83659a41675f7171ea57909dc5a79ab4
Key [ 7]:23c4812ab1905ddf77dedaed4105649a
Key [ 8]:40d87e272a7a1554ae2e85e3638cdf52
round  5:0b9382d0ed4f2fccdbb69d0db7b130a4
Key [ 9]:bdc064c6a39f6b84fe40db359f62a3c4
Key [10]:58228db841ce3cee983aa721f36aa1b9
round  6:c6ebda0f8f489792f09c189568226c1f
Key [11]:a815bacd6fa747a0d4f52883ac63ebe7
```

*Sample Data*

```
Key [12]:a9ce513b38ea006c333ecaaefcf1d0f8
round  7:75a8aba07e69c9065bcd831c40115116
Key [13]:3635e074792d4122130e5b824e52cd60
Key [14]:511bdb61bb28de72a5d794bffbf407df
round  8:57a6e279dcb764cf7dd6a749dd60c735
Key [15]:a32f5f21044b6744b6d913b13cdb4c0a
Key [16]:9722bbaeef281496ef8c23a9d41e92f4
Key [17]:807370560ad7e8a13a054a65a03b4049
Ka      :e62f8bac609139b3999aedbc9d228042
----------------------------------------
rand    :dab3cffe9d5739d1b7bf4a667ae5ee24
address :02f8fd4cd661
round  1:02f8fd4cd66102f8fd4cd66102f8fd4c
Key [ 1]:dab3cffe9d5739d1b7bf4a667ae5ee22
Key [ 2]:e315a8a65d809ec7c289e69c899fbdcc
round  2:ef85ff081b8709405e19f3e275cec7dc
Key [ 3]:df6a119bb50945fc8a3394e7216448f3
Key [ 4]:87fe86fb0d58b5dd0fb3b6b1dab51d07
round  3:aa25c21bf577d92dd97381e3e9edcc54
added ->:a81dbf5723d8dbd524bf5782ebe5c918
Key [ 5]:36cc253c506c0021c91fac9d8c469e90
Key [ 6]:d5fda00f113e303809b7f7d78a1a2b0e
round  4:9e69ce9b53caec3990894d2baed41e0d
Key [ 7]:c14b5edc10cabf16bc2a2ba4a8ae1e40
Key [ 8]:74c6131afc8dce7e11b03b1ea8610c16
round  5:a5460fa8cedca48a14fd02209e01f02e
Key [ 9]:346cfc553c6cbc9713edb55f4dcbc96c
Key [10]:bddf027cb059d58f0509f8963e9bdec6
round  6:92b33f11eadcacc5a43dd05f13d334dd
Key [11]:8eb9e040c36c4c0b4a7fd3dd354d53c4
Key [12]:c6ffecdd5e135b20879b9dfa4b34bf51
round  7:fb0541aa5e5df1a61c51aef606eb5a41
Key [13]:bf12f5a6ba08dfc4fda4bdfc68c997d9
Key [14]:37c4656b9215f3c959ea688fb64ad327
round  8:f0bbd2b94ae374346730581fc77a9c98
Key [15]:e87bb0d86bf421ea4f779a8eee3a866c
Key [16]:faa471e934fd415ae4c0113ec7f0a5ad
Key [17]:95204a80b8400e49db7cf6fd2fd40d9a
Ka      :b0376d0a9b338c2e133c32b69cb816b3
----------------------------------------
rand    :13ecad08ad63c37f8a54dc56e82f4dc1
address :9846c5ead4d9
round  1:9846c5ead4d99846c5ead4d99846c5ea
Key [ 1]:13ecad08ad63c37f8a54dc56e82f4dc7
Key [ 2]:ad04f127bed50b5e671d6510d392eaed
round  2:97374e18cdd0a6f7a5aa49d1ac875c84
Key [ 3]:57ad159e5774fa222f2f3039b9cd5101
Key [ 4]:9a1e9e1068fede02ef90496e25fd8e79
round  3:9dd3260373edd9d5f4e774826b88fd2d
added ->:0519ebe9a7c6719331d1485bf3cec2c7
Key [ 5]:378dce167db62920b0b392f7cfca316e
Key [ 6]:db4277795c87286faee6c9e9a6b71a93
```

```
round  4:40ec6563450299ac4e120d88672504d6
Key [ 7]:ec01aa2f5a8a793b36c1bb858d254380
Key [ 8]:2921a66cfa5bf74ac535424564830e98
round  5:57287bbb041bd6a56c2bd931ed410cd4
Key [ 9]:07018e45aab61b3c3726ee3d57dbd5f6
Key [10]:627381f0fa4c02b0c7d3e7dfbffc3333
round  6:66affa66a8dcd36e36bf6c3f1c6a276e
Key [11]:33b57c925bd5551999f716e138efbe79
Key [12]:a6dc7f9aa95bcc9243aebd12608f657a
round  7:450e65184fd8c72c578d5cdecb286743
Key [13]:a6a6db00fd8c72a28ea57ea542f6e102
Key [14]:dcf3377daeb2e24e61f0ad6620951c1f
round  8:e5eb180b519a4e673f21b7c4f4573f3d
Key [15]:621240b9506b462a7fa250da41844626
Key [16]:ae297810f01f43dc35756cd119ee73d6
Key [17]:b959835ec2501ad3894f8b8f1f4257f9
Ka      :5b61e83ad04d23e9d1c698851fa30447
========================================
```

## 10.3 THREE TESTS OF E22

(for K_master and overlay generation)

```
rand    :001de169248850245a5f7cc7f0d6d633
PIN     :d5a51083a04a1971f18649ea8b79311a
round  1:001de169248850245a5f7cc7f0d6d623
Key [ 1]:d5a51083a04a1971f18649ea8b79311a
Key [ 2]:7317cdbff57f9b99f9810a2525b17cc7
round  2:5f05c143347b59acae3cb002db23830f
Key [ 3]:f08bd258adf1d4ae4a54d8ccb26220b2
Key [ 4]:91046cbb4ccc43db18d6dd36ca7313eb
round  3:c8f3e3300541a25b6ac5a80c3105f3c4
added ->:c810c45921c9f27f302424cbc1dbc9e7
Key [ 5]:67fb2336f4d9f069da58d11c82f6bd95
Key [ 6]:4fed702c75bd72c0d3d8f38707134c50
round  4:bd5e0c3a97fa55b91a3bbbf306ebb978
Key [ 7]:41c947f80cdc0464c50aa89070af314c
Key [ 8]:680eecfa8daf41c7109c9a5cb1f26d75
round  5:21c1a762c3cc33e75ce8976a73983087
Key [ 9]:6e33fbd94d00ff8f72e8a7a0d2cebc4c
Key [10]:f4d726054c6b948add99fabb5733ddc3
round  6:56d0df484345582f6b574a449ba155eb
Key [11]:4eda2425546a24cac790f49ef2453b53
Key [12]:cf2213624ed1510408a5a3e00b7333df
round  7:120cf9963fe9ff22993f7fdf9600d9b8
Key [13]:d04b1a25b0b8fec946d5ecfa626d04c9
Key [14]:01e5611b0f0e140bdb64585fd3ae5269
round  8:a6337400ad8cb47fefb91332f5cb2713
Key [15]:f15b2dc433f534f61bf718770a3698b1
Key [16]:f990d0273d8ea2b9e0b45917a781c720
Key [17]:f41b3cc13d4301297bb6bdfcb3e5a1dd
Ka      :539e4f2732e5ae2de1e0401f0813bd0d
```



LifeScan, Inc. Ex. 1019
Page 1002 of 1420
IPR2025-00103

*Sample Data*

```
----------------------------------------
rand    :67ed56bfcf99825f0c6b349369da30ab
PIN     :7885b515e84b1f082cc499976f1725ce
round  1:67ed56bfcf99825f0c6b349369da30bb
Key [ 1]:7885b515e84b1f082cc499976f1725ce
Key [ 2]:72445901fdaf506beb036f4412512248
round  2:6b160b66a1f6c26c1f3432f463ef5aa1
Key [ 3]:59f0e4982e97633e5e7fd133af8f2c5b
Key [ 4]:b4946ec77a41bf7c729d191e33d458ab
round  3:3f22046c964c3e5ca2a26ec9a76a9f67
added ->:580f5ad359e5c003ae0da25ace44cfdc
Key [ 5]:eb0b839f97bdf534183210678520bbef
Key [ 6]:cff0bc4a94e5c8b2a2d24d9f59031e19
round  4:87aa61fc0ff88e744c195249b9a33632
Key [ 7]:592430f14d8f93db95dd691af045776d
Key [ 8]:3b55b404222bf445a6a2ef5865247695
round  5:83dcf592a854226c4dcd94e1ecf1bc75
Key [ 9]:a9714b86319ef343a28b87456416bd52
Key [10]:e6598b24390b3a0bf2982747993b0d78
round  6:dee0d13a52e96bcf7c72045a21609fc6
Key [11]:62051d8c51973073bff959b032c6e1e2
Key [12]:29e94f4ab73296c453c833e217a1a85b
round  7:08488005761e6c7c4dbb203ae453fe3a
Key [13]:0e255970b3e2fc235f59fc5acb10e8ce
Key [14]:d0dfbb3361fee6d4ffe45babf1cd7abf
round  8:0d81e89bddde7a7065316c47574feb8f
Key [15]:c12eee4eb38b7a171f0f736003774b40
Key [16]:8f962523f1c0abd9a087a0dfb11643d3
Key [17]:24be1c66cf8b022f12f1fb4c60c93fd1
Ka      :04435771e03a9daceb8bb1a493ee9bd8
----------------------------------------
rand    :40a94509238664f244ff8e3d13b119d3
PIN     :1ce44839badde30396d03c4c36f23006
round  1:40a94509238664f244ff8e3d13b119c3
Key [ 1]:1ce44839badde30396d03c4c36f23006
Key [ 2]:6dd97a8f91d628be4b18157af1a9dcba
round  2:0eac5288057d9947a24eabc1744c4582
Key [ 3]:fef9583d5f55fd4107ad832a725db744
Key [ 4]:fc3893507016d7c1db2bd034a230a069
round  3:60b424f1082b0cc3bd61be7b4c0155f0
added ->:205d69f82bb17031f9604c465fb26e33
Key [ 5]:0834d04f3e7e1f7f85f0c1db685ab118
Key [ 6]:1852397f9a3723169058e9b62bb3682b
round  4:2c6b65a49d66af6566675afdd6fa7d7d
Key [ 7]:6c10da21d762ae4ac1ba22a96d9007b4
Key [ 8]:9aa23658b90470a78d686344b8a9b0e7
round  5:a2c537899665113a42f1ac24773bdc31
Key [ 9]:137dee3bf879fe7bd02fe6d888e84f16
Key [10]:466e315a1863f47d0f93bc6827cf3450
round  6:e26982980d79b21ed3e20f8c3e71ba96
Key [11]:0b33cf831465bb5c979e6224d7f79f7c
Key [12]:92770660268ede827810d707a0977d73
```

```
round  7:e7b063c4e2e3110b89b7e1631c762dd5
Key [13]:7be30ae4961cf24ca17625a77bb7a9f8
Key [14]:be65574a33ae30e6e82dbd2826d3cc1a
round  8:7a963e37b2c2e76b489cfe40a2cf00e5
Key [15]:ed0ba7dd30d60a5e69225f0a33011e5b
Key [16]:765c990f4445e52b39e6ed6105ad1c4f
Key [17]:52627bf9f35d94f30d5b07ef15901adc
Ka      :9cde4b60f9b5861ed9df80858bac6f7f


========================================
```

## 10.4 TESTS OF E22 WITH PIN AUGMENTING

for PIN lengths 1,...,16 bytes

```
rand    :24b101fd56117d42c0545a4247357048
PIN length =16 octets
PIN     :fd397c7f5c1f937cdf82d8816cc377e2
round  1:24b101fd56117d42c0545a4247357058
Key [ 1]:fd397c7f5c1f937cdf82d8816cc377e2
Key [ 2]:0f7aac9c9b53f308d9fdbf2c78e3c30e
round  2:838edfe1226266953ccba8379d873107
Key [ 3]:0b8ac18d4bb44fad2efa115e43945abc
Key [ 4]:887b16b062a83bfa469772c25b456312
round  3:8cd0c9283120aba89a7f9d635dd4fe3f
added ->:a881cad5673128ea5ad3f7211a096e67
Key [ 5]:2248cbe6d299e9d3e8fd35a91178f65b
Key [ 6]:b92af6237385bd31f8fb57fb1bdd824e
round  4:2648d9c618a622b10ef80c4dbaf68b99
Key [ 7]:2bf5ffe84a37878ede2d4c30be60203b
Key [ 8]:c9cb6cec60cb8a8f29b99fcf3e71e40f
round  5:b5a7d9e96f68b14ccebf361de3914d0f
Key [ 9]:5c2f8a702e4a45575b103b0cce8a91c6
Key [10]:d453db0c9f9ddbd11e355d9a34d9b11b
round  6:632a091e7eefe1336857ddafd1ff3265
Key [11]:32805db7e59c5ed4acabf38d27e3fece
Key [12]:fde3a8eedfa3a12be09c1a8a00890fd7
round  7:048531e9fd3efa95910540150f8b137b
Key [13]:def07eb23f3a378f059039a2124bc4c2
Key [14]:2608c58f23d84a09b9ce95e5caac1ab4
round  8:461814ec7439d412d0732f0a6f799a6a
Key [15]:0a7ed16481a623e56ee1442ffa74f334
Key [16]:12add59aca0d19532f1516979954e369
Key [17]:dd43d02d39ffd6a386a4b98b4ac6eb23
Ka      :a5f2adf328e4e6a2b42f19c8b74ba884
----------------------------------------
rand    :321964061ac49a436f9fb9824ac63f8b
PIN length =15 octets
PIN     :ad955d58b6b8857820ac1262d617a6
address :0314c0642543
round  1:321964061ac49a436f9fb9824ac63f9b
```

*Sample Data*

```
Key [ 1]:ad955d58b6b8857820ac1262d617a603
Key [ 2]:f281736f68e3d30b2ac7c67f125dc416
round  2:7c4a4ece1398681f4bafd309328b7770
Key [ 3]:43c157f4c8b360387c32ab330f9c9aa8
Key [ 4]:3a3049945a298f6d076c19219c47c3cb
round  3:9672b00738bdfaf9bd92a855bc6f3afb
added ->:a48b1401228194bad23161d7f6357960
Key [ 5]:c8e2eaa6d73b7de18f3228ab2173bc69
Key [ 6]:8623f44488222e66a293677cf30bf2bb
round  4:9b30247aad3bf133712d034b46d21c68
Key [ 7]:f3e500902fba31db9bae50ef30e484a4
Key [ 8]:49d4b1137c18f4752dd9955a5a8d2f43
round  5:4492c25fda08083a768b4b5588966b23
Key [ 9]:9d59c451989e74785cc097eda7e42ab8
Key [10]:251de25f3917dcd99c18646107a641fb
round  6:21ae346635714d2623041f269978c0ee
Key [11]:80b8f78cb1a49ec0c3e32a238e60fddf
Key [12]:beb84f4d20a501e4a24ecfbde481902b
round  7:9b56a3d0f8932f20c6a77a229514fb00
Key [13]:852571b44f35fd9d9336d3c1d2506656
Key [14]:d0a0d510fb06ba76e69b8ee3ebc1b725
round  8:6cd8492b2fd31a86978bcdf644eb08a8
Key [15]:c7ffd523f32a874ed4a93430a25976de
Key [16]:16cdcb25e62964876d951fdcc07030d3
Key [17]:def32c0e12596f9582e5e3c52b303f52
Ka      :c0ec1a5694e2b48d54297911e6c98b8f
----------------------------------------
rand    :d4ae20c80094547d7051931b5cc2a8d6
PIN length =14 octets
PIN     :e1232e2c5f3b833b3309088a87b6
address :fabecc58e609
round  1:d4ae20c80094547d7051931b5cc2a8c6
Key [ 1]:e1232e2c5f3b833b3309088a87b6fabe
Key [ 2]:5f0812b47cd3e9a30d7707050fffa1f2
round  2:1f45f16be89794bef33e4547c9c0916a
Key [ 3]:77b681944763244ffa3cd71b248b79b5
Key [ 4]:e2814e90e04f485958ce58c9133e2be6
round  3:b10d2f4ac941035263cee3552d774d2f
added ->:65bb4f82c9d5572f131f764e7139f5e9
Key [ 5]:520acad20801dc639a2c6d66d9b79576
Key [ 6]:c72255cdb61d42be72bd45390dd25ba5
round  4:ead4dc34207b6ea721c62166e155aaad
Key [ 7]:ebf04c02075bf459ec9c3ec06627d347
Key [ 8]:a1363dd2812ee800a4491c0c74074493
round  5:f507944f3018e20586d81d7f326aae9d
Key [ 9]:b0b6ba79493dc833d7f425be7b8dadb6
Key [10]:08cd23e536b9b9b53e85eb004cba3111
round  6:fff450f4302a2b3571e8405e148346da
Key [11]:fec22374c6937dcd26171f4d2edfada3
Key [12]:0f1a8ef5979c69ff44f620c2e007b6e4
round  7:de558779589897f3402a90ee78c3f921
Key [13]:901fb66f0779d6aad0c0fba1fe812cb5
```



LifeScan, Inc. Ex. 1019
Page 1005 of 1420
IPR2025-00103

*Sample Data*

```
Key [14]:a0cab3cd15cd23603adc8d4474efb239
round  8:b2df0aa0c9f07fbbaa02f510a29cf540
Key [15]:18edc3f4296dd6f1dea13f7c143117a1
Key [16]:8d3d52d700a379d72ded81687f7546c7
Key [17]:5927badfe602f29345f840bb53e1dea6
Ka      :d7b39be13e3692c65b4a9e17a9c55e17
----------------------------------------
rand    :272b73a2e40db52a6a61c6520549794a
PIN length =13 octets
PIN     :549f2694f353f5145772d8ae1e
address :20487681eb9f
round  1:272b73a2e40db52a6a61c6520549795a
Key [ 1]:549f2694f353f5145772d8ae1e204876
Key [ 2]:42c855593d66b0c458fd28b95b6a5fbf
round  2:d7276dc8073f7677c31f855bde9501e2
Key [ 3]:75d0a69ae49a2da92e457d767879df52
Key [ 4]:b3aa7e7492971afaa0fb2b64827110df
round  3:71aae503831133d19bc452da4d0e409b
added ->:56d558a1671ee8fbf12518884857b9c1
Key [ 5]:9c8cf1604a98e9a503c342e272de5cf6
Key [ 6]:d35bc2df6b85540a27642106471057d9
round  4:f41a709c89ea80481aa3d2b9b2a9f8ca
Key [ 7]:b454dda74aeb4eff227ba48a58077599
Key [ 8]:bcba6aec050116aa9b7c6a9b7314d796
round  5:20fdda20f4a26b1bd38eb7f355a7be87
Key [ 9]:d41f8a9de0a716eb7167a1b6e321c528
Key [10]:5353449982247782d168ab43f17bc4d8
round  6:a70e316997eeed49a5a9ef9ba5e913b5
Key [11]:32cbc9cf1a81e36a45153972347ce4ac
Key [12]:5747619006cf4ef834c749f2c4b9feb6
round  7:e66f2317a825f589f76b47b6aa6e73fb
Key [13]:f9b68beba0a09d2a570a7dc88cc3c3c2
Key [14]:55718f9a4f0b1f9484e8c6b186a41a4b
round  8:5f68f940440a9798e074776019804ada
Key [15]:4ecc29be1b4d78433f6aa30db974a7fb
Key [16]:8470a066ffb00cda7b08059599f919f5
Key [17]:f39a36d74e960a051e1ca98b777848f4
Ka      :9ac64309a37c25c3b4a584fc002a1618
----------------------------------------
rand    :7edb65f01a2f45a2bc9b24fb3390667e
PIN length =12 octets
PIN     :2e5a42797958557b23447ca8
address :04f0d2737f02
round  1:7edb65f01a2f45a2bc9b24fb3390666e
Key [ 1]:2e5a42797958557b23447ca804f0d273
Key [ 2]:18a97c856561eb23e71af8e9e1be4799
round  2:3436e12db8ffdc1265cb5a86da2fed0b
Key [ 3]:7c0908dcbc73201e17c4f7aa1ab8aec8
Key [ 4]:7cb58833602fbe4194c7cc797ce8c454
round  3:caed6af4226f67e4ad1914620803ef2a
added ->:b4c8cf04389eac4611b438993b935544
Key [ 5]:f4dce7d607b5234562d0ebb2267b08b8
```

*Sample Data*

```
Key [ 6]:560b75c5545751fd8fa99fa4346e654b
round  4:ee67c87d6f74bb75db98f68bff0192c1
Key [ 7]:32f10cefd8d3e6424c6f91f1437808af
Key [ 8]:a934a46045be30fb3be3a5f3f7b18837
round  5:792398dcbeb8d10bdb07ae3c819e943c
Key [ 9]:a0f12e97c677a0e8ac415cd2c8a7ca88
Key [10]:e27014c908785f5ca03e8c6a1da3bf13
round  6:e778b6e0c3e8e7edf90861c7916d97a8
Key [11]:1b4a4303bcc0b2e0f41c72d47654bd9f
Key [12]:4b1302a50046026d6c9054fc8387965a
round  7:1fafddc7efa5f04c1dec1869d3f2d9bb
Key [13]:58c334bb543d49eca562cdbe0280e0fc
Key [14]:bdb60d383c692d06476b76646c8dec48
round  8:3d7c326d074bd6aa222ea050f04a3c7f
Key [15]:78c0162506be0b5953e8403c01028f93
Key [16]:24d7dbbe834dbd7b67f57fcf0d39d60f
Key [17]:2e74f1f3331c0f6585e87b2f715e187e
Ka      :d3af4c81e3f482f062999dee7882a73b
----------------------------------------
rand    :26a92358294dce97b1d79ec32a67e81a
PIN length =11 octets
PIN     :05fbad03f52fa9324f7732
address :b9ac071f9d70
round  1:26a92358294dce97b1d79ec32a67e80a
Key [ 1]:05fbad03f52fa9324f7732b9ac071f9d
Key [ 2]:2504c9691c04a18480c8802e922098c0
round  2:0be20e3d76888e57b6bf77f97a8714fb
Key [ 3]:576b2791d1212bea8408212f2d43e77e
Key [ 4]:90ae36dcce8724adb618f912d1b27297
round  3:1969667060764453257d906b7e58bd5b
added ->:3f12892849c312c494542ea854bfa551
Key [ 5]:bc492c42c9e87f56ec31af5474e9226e
Key [ 6]:c135d1dbed32d9519acfb4169f3e1a10
round  4:ac404205118fe771e54aa6f392da1153
Key [ 7]:83ccbdbbaf17889b7d18254dc9252fa1
Key [ 8]:80b90a1767d3f2848080802764e21711
round  5:41795e89ae9a0cf776ffece76f47fd7a
Key [ 9]:cc24e4a86e8eed129118fd3d5223a1dc
Key [10]:7b1e9c0eb9dab083574be7b7015a62c9
round  6:29ca9e2f87ca00370ef1633505bfba4b
Key [11]:888e6d88cf4beb965cf7d4f32b696baa
Key [12]:6d642f3e5510b0b043a44daa2cf5eec0
round  7:81fc891c3c6fd99acc00028a387e2366
Key [13]:e224f85da2ab63a23e2a3a036e421358
Key [14]:c8dc22aaa739e2cb85d6a0c08226c7d0
round  8:e30b537e7a000e3d2424a9c0f04c4042
Key [15]:a969aa818c6b324bae391bedcdd9d335
Key [16]:6974b6f2f07e4c55f2cc0435c45bebd1
Key [17]:134b925ebd98e6b93c14aee582062fcb
Ka      :be87b44d079d45a08a71d15208c5cb50
----------------------------------------
rand    :0edef05327eab5262430f21fc91ce682
```



LifeScan, Inc. Ex. 1019
Page 1007 of 1420
IPR2025-00103

*Sample Data*

```
PIN length =10 octets
PIN     :8210e47390f3f48c32b3
address :7a3cdfe377d1
round  1:0edef05327eab5262430f21fc91ce692
Key [ 1]:8210e47390f3f48c32b37a3cdfe377d1
Key [ 2]:c6be4c3e425e749b620a94c779e33a7e
round  2:07ca3c7a7a6bcbc31d79a856d9cffc0e
Key [ 3]:2587cec2a4b8e4f996a9ed664350d5dd
Key [ 4]:70e4bf72834d9d3dbb7eb2c239216dc0
round  3:792ad2ac4e4559d1463714d2f161b6f4
added ->:7708c2ff692f0ef7626706cd387d9c66
Key [ 5]:6696e1e7f8ac037e1fff3598f0c164e2
Key [ 6]:23dbfe4d0b561bea08fbcef25e49b648
round  4:7d8c71a9d7fbdcbd851bdf074550b100
Key [ 7]:b03648acd021550edee904431a02f00c
Key [ 8]:cb169220b7398e8f077730aa4bf06baa
round  5:b6fcaa45064ffd557e4b7b30cfbb83e0
Key [ 9]:af602c2ba16a454649951274c2be6527
Key [10]:5d60b0a7a09d524143eca13ad680bc9c
round  6:b3416d391a0c26c558843debd0601e9e
Key [11]:9a2f39bfe558d9f562c5f09a5c3c0263
Key [12]:72cae8eebd7fabd9b1848333c2aab439
round  7:abe4b498d9c36ea97b8fd27d7f813913
Key [13]:15f27ea11e83a51645d487b81371d7dc
Key [14]:36083c8666447e03d33846edf444eb12
round  8:8032104338a945ba044d102eabda3b22
Key [15]:0a3a8977dd48f3b6c1668578befadd02
Key [16]:f06b6675d78ca0ee5b1761bdcdab516d
Key [17]:cbc8a7952d33aa0496f7ea2d05390b23
Ka      :bf0706d76ec3b11cce724b311bf71ff5
----------------------------------------
rand    :86290e2892f278ff6c3fb917b020576a
PIN length = 9 octets
PIN     :3dcdffcfd086802107
address :791a6a2c5cc3
round  1:86290e2892f278ff6c3fb917b0205765
Key [ 1]:3dcdffcfd086802107791a6a2c5cc33d
Key [ 2]:b4962f40d7bb19429007062a3c469521
round  2:1ec59ffd3065f19991872a7863b0ef02
Key [ 3]:eb9ede6787dd196b7e340185562bf28c
Key [ 4]:2964e58aacf7287d1717a35b100ae23b
round  3:f817406f1423fc2fe33e25152679eaaf
added ->:7e404e47861574d08f7dde02969941ca
Key [ 5]:6abf9a314508fd61e486fa4e376c3f93
Key [ 6]:6da148b7ee2632114521842cbb274376
round  4:e9c2a8fac22b8c7cf0c619e2b3f890ed
Key [ 7]:df889cc34fda86f01096d52d116e620d
Key [ 8]:5eb04b147dc39d1974058761ae7b73fc
round  5:444a8aac0efee1c02f8d38f8274b7b28
Key [ 9]:8426cc59eee391b2bd50cf8f1efef8b3
Key [10]:8b5d220a6300ade418da791dd8151941
round  6:9185f983db150b1bccab1e5c12eb63a1
```

*Sample Data*

```
Key [11]:82ba4ddef833f6a4d18b07aa011f2798
Key [12]:ce63d98794682054e73d0359dad35ec4
round  7:5eded2668f5916dfd036c09e87902886
Key [13]:da794357652e80c70ad8b0715dbe33d6
Key [14]:732ef2c0c3220b31f3820c375e27bb29
round  8:88a5291b4acbba009a85b7dd6a834b3b
Key [15]:3ce75a61d4b465b70c95d7ccd5799633
Key [16]:5df9bd2c3a17a840cdaafb76c171db7c
Key [17]:3f8364b089733d902bccb0cd3386846f
Ka      :cdb0cc68f6f6fbd70b46652de3ef3ffb
----------------------------------------
rand    :3ab52a65bb3b24a08eb6cd284b4b9d4b
PIN length = 8 octets
PIN     :d0fb9b6838d464d8
address :25a868db91ab
round  1:3ab52a65bb3b24a08eb6cd284b4b9d45
Key [ 1]:d0fb9b6838d464d825a868db91abd0fb
Key [ 2]:2573f47b49dad6330a7a9155b7ae8ba1
round  2:ad2ffdff408fcfab44941016a9199251
Key [ 3]:d2c5b8fb80cba13712905a589adaee71
Key [ 4]:5a3381511b338719fae242758dea0997
round  3:2ddc17e570d7931a2b1d13f6ace928a5
added ->:17914180cb12b7baa5d3e0dee734c5e0
Key [ 5]:e0a4d8ac27fbe2783b7bcb3a36a6224d
Key [ 6]:949324c6864deac3eca8e324853e11c3
round  4:62c1db5cf31590d331ec40ad692e8df5
Key [ 7]:6e67148088a01c2d4491957cc9ddc4aa
Key [ 8]:557431deab7087bb4c03fa27228f60c6
round  5:9c8933bc361f4bde4d1bda2b5f8bb235
Key [ 9]:a2551aca53329e70ade3fd2bb7664697
Key [10]:05d0ad35de68a364b54b56e2138738fe
round  6:9156db34136aa06655bf28a05be0596a
Key [11]:1616a6b13ce2f2895c722e8495181520
Key [12]:b12e78a1114847b01f6ed2f5a1429a23
round  7:84dcc292ed836c1c2d523f2a899a2ad5
Key [13]:316e144364686381944e95afd8a026bb
Key [14]:1ab551b88d39d97ea7a9fe136dbfe2e1
round  8:87bdcac878d777877f4eccf042cfee5e
Key [15]:70e21ab08c23c7544524b64492b25cc9
Key [16]:35f730f2ae2b950a49a1bf5c8b9f8866
Key [17]:2f16924c22db8b74e2eadf1ba4ebd37c
Ka      :983218718ca9aa97892e312d86dd9516
----------------------------------------
rand    :a6dc447ff08d4b366ff96e6cf207e179
PIN length = 7 octets
PIN     :9c57e10b4766cc
address :54ebd9328cb6
round  1:a6dc447ff08d4b366ff96e6cf207e174
Key [ 1]:9c57e10b4766cc54ebd9328cb69c57e1
Key [ 2]:00a609f4d61db26993c8177e3ee2bba8
round  2:1ed26b96a306d7014f4e5c9ee523b73d
Key [ 3]:646d7b5f9aaa528384bda3953b542764
```

*Sample Data*

```
Key [ 4]:a051a42212c0e9ad5c2c248259aca14e
round  3:a53f526db18e3d7d53edbfc9711041ed
added ->:031b9612411b884b3ce62da583172299
Key [ 5]:d1bd5e64930e7f838d8a33994462d8b2
Key [ 6]:5dc7e2291e32435665ebd6956bec3414
round  4:9438be308ec83f35c560e2796f4e0559
Key [ 7]:10552f45af63b0f15e2919ab37f64fe7
Key [ 8]:c44d5717c114a58b09207392ebe341f8
round  5:b79a7b14386066d339f799c40479cb3d
Key [ 9]:6886e47b782325568eaf59715a75d8ff
Key [10]:8e1e335e659cd36b132689f78c147bda
round  6:ef232462228aa166438d10c34e17424b
Key [11]:8843efeedd5c2b7c3304d647f932f4d1
Key [12]:13785aaedd0adf67abb4f01872392785
round  7:02d133fe40d15f1073673b36bba35abd
Key [13]:837d7ca2722419e6be3fae35900c3958
Key [14]:93f8442973e7fccf2e7232d1d057c73a
round  8:275506a3d08c84e94cc58ed60054505e
Key [15]:8a7a9edffa3c52918bc6a45f57d91f5d
Key [16]:f214a95d777f763c56109882c4b52c84
Key [17]:10e2ee92c5ea1ddc5eb010e55510c403
Ka      :9cd6650ead86323e87cafb1ff516d1e0
----------------------------------------
rand    :3348470a7ea6cc6eb81b40472133262c
PIN length = 6 octets
PIN     :fcad169d7295
address :430d572f8842
round  1:3348470a7ea6cc6eb81b404721332620
Key [ 1]:fcad169d7295430d572f8842fcad169d
Key [ 2]:b3479d4d4fd178c43e7bc5b0c7d8983c
round  2:af976da9225066d563e10ab955e6fc32
Key [ 3]:7112462b37d82dd81a2a35d9eb43cb7c
Key [ 4]:c5a7030f8497945ac7b84600d1d161fb
round  3:d08f826ebd55a0bd7591c19a89ed9bde
added ->:e3d7c964c3fb6cd3cdac01dda820c1fe
Key [ 5]:84b0c6ef4a63e4dff19b1f546d683df5
Key [ 6]:f4023edfc95d1e79ed4bb4de9b174f5d
round  4:6cd952785630dfc7cf81eea625e42c5c
Key [ 7]:ea38dd9a093ac9355918632c90c79993
Key [ 8]:dbba01e278ddc76380727f5d7135a7de
round  5:93573b2971515495978264b88f330f7f
Key [ 9]:d4dc3a31be34e412210fafa6eca00776
Key [10]:39d1e190ee92b0ff16d92a8be58d2fa0
round  6:b3f01d5e7fe1ce6da7b46d8c389baf47
Key [11]:1eb081328d4bcf94c9117b12c5cf22ac
Key [12]:7e047c2c552f9f1414d946775fabfe30
round  7:0b833bff6106d5bae033b4ce5af5a924
Key [13]:e78e685d9b2a7e29e7f2a19d1bc38ebd
Key [14]:1b582272a3121718c4096d2d8602f215
round  8:23de0bbdc70850a7803f4f10c63b2c0f
Key [15]:8569e860530d9c3d48a0870dac33f676
Key [16]:6966b528fdd1dc222527052c8f6cf5a6
```

*Sample Data*

```
Key [17]:a34244c757154c53171c663b0b56d5c2
Ka      :98f1543ab4d87bd5ef5296fb5e3d3a21
----------------------------------------
rand    :0f5bb150b4371ae4e5785293d22b7b0c
PIN length = 5 octets
PIN     :b10d068bca
address :b44775199f29
round  1:0f5bb150b4371ae4e5785293d22b7b07
Key [ 1]:b10d068bcab44775199f29b10d068bca
Key [ 2]:aec70d1048f1bbd2c18040318a8402ad
round  2:342d2b79d7fb7cd110379742b9842c79
Key [ 3]:6d8d5cf338f29ef4420639ef488e4fa9
Key [ 4]:a1584117541b759ba6d9f7eb2bedcbba
round  3:9407e8e3e810603921bf81cfda62770a
added ->:9b6299b35c477addc437d35c088df20d
Key [ 5]:09a20676666aeed6f22176274eb433f4
Key [ 6]:840472c001add5811a054be5f5c74754
round  4:9a3ba953225a7862c0a842ed3d0b2679
Key [ 7]:fad9e45c8bf70a972fcd9bff0e8751f5
Key [ 8]:e8f30ff666dfd212263416496ff3b2c2
round  5:2c573b6480852e875df34b28a5c44509
Key [ 9]:964cdba0cf8d593f2fc40f96daf8267a
Key [10]:bcd65c11b13e1a70bcd4aafba8864fe3
round  6:21b0cc49e880c5811d24dee0194e6e9e
Key [11]:468c8548ea9653c1a10df6288dd03c1d
Key [12]:5d252d17af4b09d3f4b5f7b5677b8211
round  7:e6d6bdcd63e1d37d9883543ba86392fd
Key [13]:e814bf307c767428c67793dda2df95c7
Key [14]:4812b979fdc20f0ff0996f61673a42cc
round  8:e3dde7ce6bd7d8a34599aa04d6a760ab
Key [15]:5b1e2033d1cd549fc4b028146eb5b3b7
Key [16]:0f284c14fb8fe706a5343e3aa35af7b1
Key [17]:b1f7a4b7456d6b577fded6dc7a672e37
Ka      :c55070b72bc982adb972ed05d1a74ddb
----------------------------------------
rand    :148662a4baa73cfadb55489159e476e1
PIN length = 4 octets
PIN     :fb20f177
address :a683bd0b1896
round  1:148662a4baa73cfadb55489159e476eb
Key [ 1]:fb20f177a683bd0b1896fb20f177a683
Key [ 2]:47266cefbfa468ca7916b458155dc825
round  2:3a942eb6271c3f4e433838a5d3fcbd27
Key [ 3]:688853a6d6575eb2f6a2724b0fbc133b
Key [ 4]:7810df048019634083a2d9219d0b5fe0
round  3:9c835b98a063701c0887943596780769
added ->:8809bd3c1a0aace6d3dcdca4cf5c7d82
Key [ 5]:c78f6dcf56da1bbd413828b33f5865b3
Key [ 6]:eb3f3d407d160df3d293a76d1a513c4a
round  4:7e68c4bafa020a4a59b5a1968105bab5
Key [ 7]:d330e038d6b19d5c9bb0d7285a360064
Key [ 8]:9bd3ee50347c00753d165faced702d9c
```



LifeScan, Inc. Ex. 1019
Page 1011 of 1420
IPR2025-00103

*Sample Data*

```
round  5:227bad0cf0838bdb15b3b3872c24f592
Key [ 9]:9543ad0fb3fe74f83e0e2281c6d4f5f0
Key [10]:746cd0383c17e0e80e6d095a87fd0290
round  6:e026e98c71121a0cb739ef6f59e14d26
Key [11]:fa28bea4b1c417536608f11f406ea1dd
Key [12]:3aee0f4d21699df9cb8caf5354a780ff
round  7:cd6a6d8137d55140046f8991da1fa40a
Key [13]:372b71bc6d1aa6e785358044fbcf05f4
Key [14]:00a01501224c0405de00aa2ce7b6ab04
round  8:52cd7257fe8d0c782c259bcb6c9f5942
Key [15]:c7015c5c1d7c030e00897f104a006d4a
Key [16]:260a9577790c62e074e71e19fd2894df
Key [17]:c041b7a231493acd15ddcdaee94b9f52
Ka      :7ec864df2f1637c7e81f2319ae8f4671
----------------------------------------
rand    :193a1b84376c88882c8d3b4ee93ba8d5
PIN length = 3 octets
PIN     :a123b9
address :4459a44610f6
round  1:193a1b84376c88882c8d3b4ee93ba8dc
Key [ 1]:a123b94459a44610f6a123b94459a446
Key [ 2]:5f64d384c8e990c1d25080eb244dde9b
round  2:3badbd58f100831d781ddd3ccedefd3f
Key [ 3]:5abc00eff8991575c00807c48f6dbea5
Key [ 4]:127521158ad6798fb6479d1d2268abe6
round  3:0b53075a49c6bf2df2421c655fdedf68
added ->:128d22de7e3247a5decf572bb61987b4
Key [ 5]:f2a1f620448b8e56665608df2ab3952f
Key [ 6]:7c84c0af02aad91dc39209c4edd220b1
round  4:793f4484fb592e7a78756fd4662f990d
Key [ 7]:f6445b647317e7e493bb92bf6655342f
Key [ 8]:3cae503567c63d3595eb140ce60a84c0
round  5:9e46a8df925916a342f299a8306220a0
Key [ 9]:734ed5a806e072bbecb4254993871679
Key [10]:cda69ccb4b07f65e3c8547c11c0647b8
round  6:6bf9cd82c9e1be13fc58eae0b936c75a
Key [11]:c48e531d3175c2bd26fa25cc8990e394
Key [12]:6d93d349a6c6e9ff5b26149565b13d15
round  7:e96a9871471240f198811d4b8311e9a6
Key [13]:5c4951e85875d663526092cd4cbdb667
Key [14]:f19f7758f5cde86c3791efaf563b3fd0
round  8:e94ca67d3721d5fb08ec069191801a46
Key [15]:bf0c17f3299b37d984ac938b769dd394
Key [16]:7edf4ad772a6b9048588f97be25bde1c
Key [17]:6ee7ba6afefc5b561abbd8d6829e8150
Ka      :ac0daabf17732f632e34ef193658bf5d
----------------------------------------
rand    :1453db4d057654e8eb62d7d62ec3608c
PIN length = 2 octets
PIN     :3eaf
address :411fbbb51d1e
round  1:1453db4d057654e8eb62d7d62ec36084
```

*Sample Data*

```
Key [ 1]:3eaf411fbbb51d1e3eaf411fbbb51d1e
Key [ 2]:c3a1a997509f00fb4241aba607109c64
round  2:0b78276c1ebc65707d38c9c5fa1372bd
Key [ 3]:3c729833ae1ce7f84861e4dbad6305cc
Key [ 4]:c83a43c3a66595cb8136560ed29be4ff
round  3:23f3f0f6441563d4c202cee0e5cb2335
added ->:3746cbbb418bb73c2964a536cb8e83b1
Key [ 5]:18b26300b86b70acdd1c8f5cbc7c5da8
Key [ 6]:04efc75309b98cd8f1cef5513c18e41e
round  4:c61afa90d3c14bdf588320e857afdc00
Key [ 7]:517c789cecadc455751af73198749fb8
Key [ 8]:fd9711f913b5c844900fa79dd765d0e2
round  5:a8a0e02ceb556af8bfa321789801183a
Key [ 9]:bb5cf30e7d3ceb930651b1d16ee92750
Key [10]:3d97c7862ecab42720e984972f8efd28
round  6:0b58e922438d224db34b68fca9a5ea12
Key [11]:4ce730344f6b09e449dcdb64cd466666
Key [12]:38828c3a56f922186adcd9b713cdcc31
round  7:b90664c4ac29a8b4bb26debec9ffc5f2
Key [13]:d30fd865ea3e9edcff86a33a2c319649
Key [14]:1fdb63e54413acd968195ab6fa424e83
round  8:6934de3067817cefd811abc5736c163b
Key [15]:a16b7c655bbaa262c807cba8ae166971
Key [16]:7903dd68630105266049e23ca607cda7
Key [17]:888446f2d95e6c2d2803e6f4e815ddc9
Ka      :1674f9dc2063cc2b83d3ef8ba692ebef
----------------------------------------
rand    :1313f7115a9db842fcedc4b10088b48d
PIN length = 1 octets
PIN     :6d
address :008aa9be62d5
round  1:1313f7115a9db842fcedc4b10088b48a
Key [ 1]:6d008aa9be62d56d008aa9be62d56d00
Key [ 2]:46ebfeafb6657b0a1984a8dc0893accf
round  2:839b23b83b5701ab095bafd162ec0ac7
Key [ 3]:8e15595edcf058af62498ee3c1dc6098
Key [ 4]:dd409c3444e94b9cc08396ae967542a0
round  3:c0a2010cc44f2139427f093f4f97ae68
added ->:d3b5f81d9eecd97bbe6ccd8e4f1f62e2
Key [ 5]:487deff5d519f6a6481e947b926f633c
Key [ 6]:5b4b6e3477ed5c2c01f6e607d3418963
round  4:1a5517a0efad3575931d8ea3bee8bd07
Key [ 7]:34b980088d2b5fd6b6a2aceeda99c9c4
Key [ 8]:e7d06d06078acc4ecdbc8da800b73078
round  5:d3ce1fdfe716d72c1075ff37a8a2093f
Key [ 9]:7d375bad245c3b757380021af8ecd408
Key [10]:14dac4bc2f4dc4929a6cceec47f4c3a3
round  6:47e90cb55be6e8dd0f583623c2f2257b
Key [11]:66cfda3c63e464b05e2e7e25f8743ad7
Key [12]:77cfccda1ad380b9fdf1df10846b50e7
round  7:f866ae6624f7abd4a4f5bd24b04b6d43
Key [13]:3e11dd84c031a470a8b66ec6214e44cf
```

```
Key [14]:2f03549bdb3c511eea70b65ddbb08253
round  8:02e8e17cf8be4837c9c40706b613dfa8
Key [15]:e2f331229ddfcc6e7bea08b01ab7e70c
Key [16]:b6b0c3738c5365bc77331b98b3fba2ab
Key [17]:f5b3973b636119e577c5c15c87bcfd19
Ka      :38ec0258134ec3f08461ae5c328968a1

========================================
```

## 10.5 FOUR TESTS OF E3

```
rand    :00000000000000000000000000000000
aco     :48afcdd4bd40fef76693b113
key     :00000000000000000000000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000000
Key [ 2]:4697b1baa3b7100ac537b3c95a28ac64
round  2:78d19f9307d2476a523ec7a8a026042a
Key [ 3]:ecabaac66795580df89af66e66dc053d
Key [ 4]:8ac3d8896ae9364943bfebd4969b68a0
round  3:600265247668dda0e81c07bbb30ed503
Key [ 5]:5d57921fd5715cbb22c1be7bbc996394
Key [ 6]:2a61b8343219fdfb1740e6511d41448f
round  4:d7552ef7cc9dbde568d80c2215bc4277
Key [ 7]:dd0480dee731d67f01a2f739da6f23ca
Key [ 8]:3ad01cd1303e12a1cd0fe0a8af82592c
round  5:fb06bef32b52ab8f2a4f2b6ef7f6d0cd
Key [ 9]:7dadb2efc287ce75061302904f2e7233
Key [10]:c08dcfa981e2c4272f6c7a9f52e11538
round  6:b46b711ebb3cf69e847a75f0ab884bdd
Key [11]:fc2042c708e409555e8c147660ffdfd7
Key [12]:fa0b21001af9a6b9e89e624cd99150d2
round  7:c585f308ff19404294f06b292e978994
Key [13]:18b40784ea5ba4c80ecb48694b4e9c35
Key [14]:454d54e5253c0c4a8b3fcca7db6baef4
round  8:2665fadb13acf952bf74b4ab12264b9f
Key [15]:2df37c6d9db52674f29353b0f011ed83
Key [16]:b60316733b1e8e70bd861b477e2456f1
Key [17]:884697b1baa3b7100ac537b3c95a28ac
round  1:5d3ecb17f26083df0b7f2b9b29aef87c
Key [ 1]:e9e5dfc1b3a79583e9e5dfc1b3a79583
Key [ 2]:7595bf57e0632c59f435c16697d4c864
round  2:de6fe85c5827233fe22514a16f321bd8
Key [ 3]:e31b96afcc75d286ef0ae257cbbc05b7
Key [ 4]:0d2a27b471bc0108c6263aff9d9b3b6b
round  3:7cd335b50d09d139ea6702623af85edb
added ->:211100a2ff6954e6e1e62df913a656a7
Key [ 5]:98d1eb5773cf59d75d3b17b3bc37c191
Key [ 6]:fd2b79282408ddd4ea0aa7511133336f
round  4:991dccb3201b5b1c4ceff65a3711e1e9
Key [ 7]:331227756638a41d57b0f7e071ee2a98
```

*Sample Data*

```
Key [ 8]:aa0dd8cc68b406533d0f1d64aabacf20
round  5:18768c7964818805fe4c6ecae8a38599
Key [ 9]:669291b0752e63f806fce76f10e119c8
Key [10]:ef8bdd46be8ee0277e9b78adef1ec154
round  6:82f9aa127a72632af43d1a17e7bd3a09
Key [11]:f3902eb06dc409cfd78384624964bf51
Key [12]:7d72702b21f97984a721c99b0498239d
round  7:1543d7870bf2d6d6efab3cbf62dca97d
Key [13]:532e60bceaf902c52a06c2c283ecfa32
Key [14]:181715e5192efb2a64129668cf5d9dd4
round  8:eee3e8744a5f8896de95831ed837ffd5
Key [15]:83017c1434342d4290e961578790f451
Key [16]:2603532f365604646ff65803795ccce5
Key [17]:882f7c907b565ea58dae1c928a0dcf41
kc      :cc802aecc7312285912e90af6a1e1154
----------------------------------------
rand    :950e604e655ea3800fe3eb4a28918087
aco     :68f4f472b5586ac5850f5f74
key     :34e86915d20c485090a6977931f96df5
round  1:950e604e655ea3800fe3eb4a28918087
Key [ 1]:34e86915d20c485090a6977931f96df5
Key [ 2]:8de2595003f9928efaf37e5229935bdb
round  2:d46f5a04c967f55840f83d1cdb5f9afc
Key [ 3]:46f05ec979a97cb6ddf842ecc159c04a
Key [ 4]:b468f0190a0a83783521deae8178d071
round  3:e16edede9cb6297f32e1203e442ac73a
Key [ 5]:8a171624dedbd552356094daaadcf12a
Key [ 6]:3085e07c85e4b99313f6e0c837b5f819
round  4:805144e55e1ece96683d23366fc7d24b
Key [ 7]:fe45c27845169a66b679b2097d147715
Key [ 8]:44e2f0c35f64514e8bec66c5dc24b3ad
round  5:edbaf77af070bd22e9304398471042f1
Key [ 9]:0d534968f3803b6af447eaf964007e7b
Key [10]:f5499a32504d739ed0b3c547e84157ba
round  6:0dab1a4c846aef0b65b1498812a73b50
Key [11]:e17e8e456361c46298e6592a6311f3fb
Key [12]:ec6d14da05d60e8abac807646931711f
round  7:1e7793cac7f55a8ab48bd33bc9c649e0
Key [13]:2b53dde3d89e325e5ff808ed505706ae
Key [14]:41034e5c3fb0c0d4f445f0cf23be79b0
round  8:3723768baa78b6a23ade095d995404da
Key [15]:e2ca373d405a7abf22b494f28a6fd247
Key [16]:74e09c9068c0e8f1c6902d1b70537c30
Key [17]:767a7f1acf75c3585a55dd4a428b2119
round  1:39809afb773efd1b7510cd4cb7c49f34
Key [ 1]:1d0d48d485abddd3798b483a82a0f878
Key [ 2]:aed957e600a5aed5217984dd5fef6fd8
round  2:6436ddbabe92655c87a7d0c12ae5e5f6
Key [ 3]:fee00bb0de89b6ef0a289696a4faa884
Key [ 4]:33ce2f4411db4dd9b7c42cc586b8a2ba
round  3:cec690f7e0aa5f063062301e049a5cc5
added ->:f7462a0c97e85c1d4572fd52b35efbf1
```

*Sample Data*

```
Key [ 5]:b5116f5c6c29e05e4acb4d02a46a3318
Key [ 6]:ff4fa1f0f73d1a3c67bc2298abc768f9
round  4:dcdfe942e9f0163fc24a4718844b417d
Key [ 7]:5453650c0819e001e48331ad0e9076e0
Key [ 8]:b4ff8dda778e26c0dce08349b81c09a1
round  5:265a16b2f766afae396e7a98c189fda9
Key [ 9]:f638fa294427c6ed94300fd823b31d10
Key [10]:1ccfa0bd86a9879b17d4bc457e3e03d6
round  6:628576b5291d53d1eb8611c8624e863e
Key [11]:0eaee2ef4602ac9ca19e49d74a76d335
Key [12]:6e1062f10a16e0d378476da3943842e9
round  7:d7b9c2e9b2d5ea5c27019324cae882b3
Key [13]:40be960bd22c744c5b23024688e554b9
Key [14]:95c9902cb3c230b44d14ba909730d211
round  8:97fb6065498385e47eb3df6e2ca439dd
Key [15]:10d4b6e1d1d6798aa00aa2951e32d58d
Key [16]:c5d4b91444b83ee578004ab8876ba605
Key [17]:1663a4f98e2862eddd3ec2fb03dcc8a4
kc      :c1beafea6e747e304cf0bd7734b0a9e2
----------------------------------------
rand    :6a8ebcf5e6e471505be68d5eb8a3200c
aco     :658d791a9554b77c0b2f7b9f
key     :35cf77b333c294671d426fa79993a133
round  1:6a8ebcf5e6e471505be68d5eb8a3200c
Key [ 1]:35cf77b333c294671d426fa79993a133
Key [ 2]:c4524e53b95b4bf2d7b2f095f63545fd
round  2:ade94ec585db0d27e17474b58192c87a
Key [ 3]:c99776768c6e9f9dd3835c52cea8d18a
Key [ 4]:f1295db23823ba2792f21217fc01d23f
round  3:da8dc1a10241ef9e6e069267cd2c6825
Key [ 5]:9083db95a6955235bbfad8aeefec5f0b
Key [ 6]:8bab6bc253d0d0c7e0107feab728ff68
round  4:e6665ca0772ceecbc21222ff7be074f8
Key [ 7]:2fa1f4e7a4cf3ccd876ec30d194cf196
Key [ 8]:267364be247184d5337586a19df8bf84
round  5:a857a9326c9ae908f53fee511c5f4242
Key [ 9]:9aef21965b1a6fa83948d107026134c7
Key [10]:d2080c751def5dc0d8ea353cebf7b973
round  6:6678748a1b5f21ac05cf1b117a7c342f
Key [11]:d709a8ab70b0d5a2516900421024b81e
Key [12]:493e4843805f1058d605c8d1025f8a56
round  7:766c66fe9c460bb2ae39ec01e435f725
Key [13]:b1ed21b71daea03f49fe74b2c11fc02b
Key [14]:0e1ded7ebf23c72324a0165a698c65c7
round  8:396e0ff7b2b9b7a3b35c9810882c7596
Key [15]:b3bf4841dc92f440fde5f024f9ce8be9
Key [16]:1c69bc6c2994f4c84f72be8f6b188963
Key [17]:bb7b66286dd679a471e2792270f3bb4d
round  1:45654f2f26549675287200f07cb10ec9
Key [ 1]:1e2a5672e66529e4f427b0682a3a34b6
Key [ 2]:974944f1ce0037b1febcf61a2bc961a2
round  2:990cd869c534e76ed4f4af7b3bfbc6c8
```

*Sample Data*

```
Key [ 3]:8147631fb1ce95d624b480fc7389f6c4
Key [ 4]:6e90a2db33d284aa13135f3c032aa4f4
round  3:ceb662f875aa6b94e8192b5989abf975
added ->:8b1bb1d753fe01e1c08b2ba9f55c07bc
Key [ 5]:cbad246d24e36741c46401e6387a05f9
Key [ 6]:dcf52aaec5713110345a41342c566fc8
round  4:d4e000be5de78c0f56ff218f3c1df61b
Key [ 7]:8197537aa9d27e67d17c16b182c8ec65
Key [ 8]:d66e00e73d835927a307a3ed79d035d8
round  5:9a4603bdef954cfaade2052604bed4e4
Key [ 9]:71d46257ecc1022bcd312ce6c114d75c
Key [10]:f91212fa528379651fbd2c32890c5e5f
round  6:09a0fd197ab81eb933eece2fe0132dbb
Key [11]:283acc551591fadce821b02fb9491814
Key [12]:ca5f95688788e20d94822f162b5a3920
round  7:494f455a2e7a5db861ece816d4e363e4
Key [13]:ba574aef663c462d35399efb999d0e40
Key [14]:6267afc834513783fef1601955fe0628
round  8:37a819f91c8380fb7880e640e99ca947
Key [15]:fdcd9be5450eef0f8737e6838cd38e2b
Key [16]:8cfbd9b8056c6a1ce222b92b94319b38
Key [17]:4f64c1072c891c39eeb95e63318462e0
kc      :a3032b4df1cceba8adc1a04427224299
----------------------------------------
rand    :5ecd6d75db322c75b6afbd799cb18668
aco     :63f701c7013238bbf88714ee
key     :b9f90c53206792b1826838b435b87d4d
round  1:5ecd6d75db322c75b6afbd799cb18668
Key [ 1]:b9f90c53206792b1826838b435b87d4d
Key [ 2]:15f74bbbde4b9d1e08f858721f131669
round  2:72abb85fc80c15ec2b00d72873ef9ad4
Key [ 3]:ef7fb29f0b01f82706c7439cc52f2dab
Key [ 4]:3003a6aecdee06b9ac295cce30dcdb93
round  3:2f10bab93a0f73742183c68f712dfa24
Key [ 5]:5fcdbb3afdf7df06754c954fc6340254
Key [ 6]:ddaa90756635579573fe8ca1f93d4a38
round  4:183b145312fd99d5ad08e7ca4a52f04e
Key [ 7]:27ca8a7fc703aa61f6d7791fc19f704a
Key [ 8]:702029d8c6e42950762317e730ec5d18
round  5:cbad52d3a026b2e38b9ae6fefffecc32
Key [ 9]:ff15eaa3f73f4bc2a6ccfb9ca24ed9c5
Key [10]:034e745246cd2e2cfc3bda39531ca9c5
round  6:ce5f159d0a1acaacd9fb4643272033a7
Key [11]:0a4d8ff5673731c3dc8fe87e39a34b77
Key [12]:637592fab43a19ac0044a21afef455a2
round  7:8a49424a10c0bea5aba52dbbffcbcce8
Key [13]:6b3fde58f4f6438843cdbe92667622b8
Key [14]:a10bfa35013812f39bf2157f1c9fca4e
round  8:f5e12da0e93e26a5850251697ec0b917
Key [15]:2228fe5384e573f48fdd19ba91f1bf57
Key [16]:5f174db2bc88925c0fbc6b5485bafc08
Key [17]:28ff90bd0dc31ea2bb479feb7d8fe029
```

*Sample Data*

```
round  1:0c75eed2b54c1cfb9ff522daef94ed4d
Key [ 1]:a21ceb92d3c027326b4de775865fe8d0
Key [ 2]:26f64558a9f0a1652f765efd546f3208
round  2:48d537ac209a6aa07b70000016c602e8
Key [ 3]:e64f9ef630213260f1f79745a0102ae5
Key [ 4]:af6a59d7cebfd0182dcca9a537c4add8
round  3:8b6d517ac893743a401b3fb7911b64e1
added ->:87e23fa87ddf90c1df10616d7eaf51ac
Key [ 5]:9a6304428b45da128ab64c8805c32452
Key [ 6]:8af4d1e9d80cb73ec6b44e9b6e4f39d8
round  4:9f0512260a2f7a5067efc35bf1706831
Key [ 7]:79cc2d138606f0fca4e549c34a1e6d19
Key [ 8]:803dc5cdde0efdbee7a1342b2cd4d344
round  5:0cfd7856edfafac51f29e86365de6f57
Key [ 9]:e8fa996448e6b6459ab51e7be101325a
Key [10]:2acc7add7b294acb444cd933f0e74ec9
round  6:2f1fa34bf352dc77c0983a01e8b7d622
Key [11]:f57de39e42182efd6586b86a90c86bb1
Key [12]:e418dfd1bb22ebf1bfc309cd27f5266c
round  7:ee4f7a53849bf73a747065d35f3752b1
Key [13]:80a9959133856586370854db6e0470b3
Key [14]:f4c1bc2f764a0193749f5fc09011a1ae
round  8:8fec6f7249760ebf69e370e9a4b80a92
Key [15]:d036cef70d6470c3f52f1b5d25b0c29d
Key [16]:d0956af6b8700888a1cc88f07ad226dc
Key [17]:1ce8b39c4c7677373c30849a3ee08794
kc      :ea520cfc546b00eb7c3a6cea3ecb39ed

========================================
```



LifeScan, Inc. Ex. 1019
Page 1018 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1019 of 1420
IPR2025-00103





LifeScan, Inc. Ex. 1019
Page 1020 of 1420
IPR2025-00103

# SECURITY SPECIFICATION

***This document describes the specification of the security system which 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.***



LifeScan, Inc. Ex. 1019
Page 1021 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1022 of 1420
IPR2025-00103

# CONTENTS

**Contents** ..... **853**

**1 Security Overview** ..... **855**
1.1 Pausing Encryption and Role Switch ..... 856
1.2 Change Connection Link Keys ..... 856
1.3 Periodically Refreshing Encryption Keys ..... 856

**2 Random Number Generation** ..... **857**

**3 Key Management** ..... **859**
3.1 Key Types ..... 859
3.2 Key Generation and Initialization ..... 861
3.2.1 Generation of initialization key, ..... 862
3.2.2 Authentication ..... 862
3.2.3 Generation of a unit key ..... 862
3.2.4 Generation of a combination key ..... 863
3.2.5 Generating the encryption key ..... 864
3.2.6 Point-to-multipoint configuration ..... 865
3.2.7 Modifying the link keys ..... 865
3.2.8 Generating a master key ..... 866

**4 Encryption** ..... **868**
4.1 Encryption Key Size Negotiation ..... 869
4.2 Encryption of Broadcast Messages ..... 869
4.3 Encryption Concept ..... 870
4.4 Encryption Algorithm ..... 871
4.4.1 The operation of the cipher ..... 873
4.5 LFSR Initialization ..... 874
4.6 Key Stream Sequence ..... 877

**5 Authentication** ..... **878**
5.1 Repeated Attempts ..... 880

**6 The Authentication And Key-Generating Functions** ..... **881**
6.1 The Authentication Function E1 ..... 881
6.2 The Functions Ar and A'r ..... 883
6.2.1 The round computations ..... 883
6.2.2 The substitution boxes "e" and "l" ..... 883
6.2.3 Key scheduling ..... 884
6.3 E2-Key Generation Function for Authentication ..... 885
6.4 E3-Key Generation Function for Encryption ..... 887

**7 Secure Simple Pairing** ..... **888**
7.1 Phase 1: Public Key Exchange ..... 889



LifeScan, Inc. Ex. 1019
Page 1023 of 1420
IPR2025-00103

7.2 Phase 2: Authentication Stage 1 ..... 890
7.2.1 Authentication Stage 1: Numeric Comparison Protocol .... 890
7.2.2 Authentication Stage 1: Out of Band Protocol ..... 892
7.2.3 Authentication Stage 1: Passkey Entry Protocol ..... 894
7.3 Phase 3: Authentication Stage 2 ..... 896
7.4 Phase 4: Link Key Calculation ..... 897
7.5 Phase 5: LMP Authentication and Encryption ..... 897
7.6 Elliptic Curve Definition ..... 897
7.7 Cryptographic Function Definitions ..... 898
7.7.1 The Simple Pairing Commitment Function f1 ..... 898
7.7.2 The Simple Pairing Numeric Verification Function g ... 899
7.7.3 The Simple Pairing Key Derivation Function f2 ..... 899
7.7.4 The Simple Pairing Check Function f3 ..... 900

**8 List of Figures ..... 902**

**9 List of Tables ..... 903**



LifeScan, Inc. Ex. 1019
Page 1024 of 1420
IPR2025-00103

# 1 SECURITY OVERVIEW

Bluetooth 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: a Bluetooth device address, 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 1.1.

| Entity | Size |
|---|---|
| BD_ADDR | 48 bits |
| Private user key, authentication | 128 bits |
| Private user key, encryption configurable length (byte-wise) | 8-128 bits |
| RAND | 128 bits |

*Table 1.1: Entities used in authentication and encryption procedures.*

The Bluetooth device address (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 towards 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 Section 6.4 on page 887). 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 established, the particular application running on the device decides when, or if, to change it. To underline the fundamental impor-



LifeScan, Inc. Ex. 1019
Page 1025 of 1420
IPR2025-00103

tance of the authentication key to a specific link, it is often referred to as the link key.

The RAND is a pseudo-random number which can be derived from a random or pseudo-random process in the device. This is not a static parameter and will change frequently.

In the remainder of this chapter, the terms user and application are used interchangeably to designate the entity that is at either side.

## 1.1 PAUSING ENCRYPTION AND ROLE SWITCH

To perform a role switch on a connection encryption must be disabled or paused as the encryption is based off the master's clock and Bluetooth address information. Unfortunately, if the role switch is required, and encryption is turned off, the device on the other end of the link will not be aware of the reason for disabling encryption, and can therefore take two possible actions: send clear text data, or disconnect. Neither of these possible actions is desirable. When performing a role switch on an encrypted link, the role switch shall be performed as a single operation where possible. If this is not possible because the other device does not support the encryption pause feature, then when a device wishes to have an encrypted link, and encryption is disabled, then the device should not send any user data, and should not disconnect. If both devices support the encryption pause feature, then this procedure shall be used.

## 1.2 CHANGE CONNECTION LINK KEYS

It is possible to perform a change of connection link keys while a link is encrypted, but it is not possible to use the new link keys until encryption has been stopped and then restarted. As with role switches, disabling encryption and then re-enabling it again can cause user data to be sent in the clear or a disconnection to occur. The use of encryption pausing prevents this problem from occurring. On devices that do not support the encryption pause feature, when a device wishes to have an encrypted link, and encryption is disabled, then the device should not send any data, and should not disconnect. If both devices support the encryption pause feature, then this procedure shall be used.

## 1.3 PERIODICALLY REFRESHING ENCRYPTION KEYS

If both devices support the encryption pause feature, then the encryption keys shall be refreshed by the Link Manager at least once every $2^{28}$ Bluetooth Clocks (about 23.3 hours) if it is not refreshed by the Host or the remote Link Manager. To refresh an encryption key, the Host maybe use the Change Connection Link Key procedure or request an encryption key refresh. If the encryption key has not been refreshed before going stale, a device may disconnect the link.



LifeScan, Inc. Ex. 1019
Page 1026 of 1420
IPR2025-00103

# 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, nonces used in Simple Pairing, for Passkeys used in authentication.

*1/2*$^{L}$ A device shall use a pseudo random number generator compliant with [FIPS PUB 140-2] (http://csrc.nist.gov/publications/fips/fips140-2/fips1402annexc.pdf)

An example of a possible random generator is provided in [FIPS PUB 186-2] Appendix 3.1 (http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf), which can be used by replacing SHA-1 function with the SHA-256 function

The device shall use a seed with at least the minimum entropy required by the pseudo random number generator.

The random number generator shall be tested against the [FIPS SP800-22] (http://csrc.nist.gov/rng/SP800-22b.pdf). This encompasses the verification of the following statistical tests performed on the output of the PRNG as specified by the [FIPS SP800-22]:

1. The Frequency (Monobit) Test
2. Frequency Test within a Block
3. The Runs Test
4. Test for the Longest-Run-of-Ones in a Block
5. The Binary Matrix Rank Test
6. The Discrete Fourier Transform (Spectral) Test
7. The Non-overlapping Template Matching Test
8. The Overlapping Template Matching Test
9. Maurer's "Universal Statistical" Test
10. The Lempel-Ziv Compression Test
11. The Linear Complexity Test
12. The Serial Test
13. The Approximate Entropy Test
14. The Cumulative Sums (Cusums) Test
15. The Random Excursions Test
16. The Random Excursions Variant Test

These tests are part of standard statistical mathematical packages. Some test suites, like the Diehard test suite can be used to verify the compliance. Alternatively, other tools, such as the DieHarder (http://www.phy.duke.edu/~rgb/General/rand_rate.php) or the available NIST tools (http://csrc.nist.gov/

CryptoToolkit/tkrng.html, http://csrc.nist.gov/rng) and the corresponding recommendations (http://csrc.nist.gov/rng/SP800-22b.pdf) may also be used.



LifeScan, Inc. Ex. 1019
Page 1028 of 1420
IPR2025-00103

# 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 over-riding the permitted key size, the Bluetooth baseband 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 Section 6.4 on page 887.

Changing a link key shall also be done through the defined baseband procedures. Depending on what kind of link key it is, different approaches are required. The details are found in Section 3.2.7 on page 865.

## 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 non-volatile 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 Section 3.2.7 on page 865.

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 Section 3.2.6 on page 865.

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 on-going connection (session).



LifeScan, Inc. Ex. 1019
Page 1029 of 1420
IPR2025-00103

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 which 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 only have to store 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 only be used during the current session. It shall only replace the original link key temporarily. For example, this may be utilized when a master wants to reach more than two devices simultaneously using the same encryption key, see Section 3.2.6 on page 865.

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 only be used during initialization.



LifeScan, Inc. Ex. 1019
Page 1030 of 1420
IPR2025-00103

The PIN may be a fixed number provided with the device (for example when there is no user interface as in a 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, for example 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 re-initialization by users who once obtained 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.

## 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 Bluetooth 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.



LifeScan, Inc. Ex. 1019
Page 1031 of 1420
IPR2025-00103

### 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 6.4 on page 887. 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 Section 5.1 on page 880).

The details of the $E_{22}$ algorithm can be found in Section 6.3 on page 885.

### 3.2.2 Authentication

The authentication procedure shall be carried out as described in Section 5 on page 878. During each authentication, a new $AU\_RAND_A$ 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 Section 3.2.5 on page 864.

The claimant/verifier status is determined by the LM.

### 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 Section 6.3 on page 885. Once created, the unit key should be stored in non-volatile memory and very rarely changed. If after initial-

ization 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 only has to remember 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 3.1 on page 863, 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, for example with device C, device A will reuse its unit key $K_A$, whereas device C stores it as $K_{CA}$.

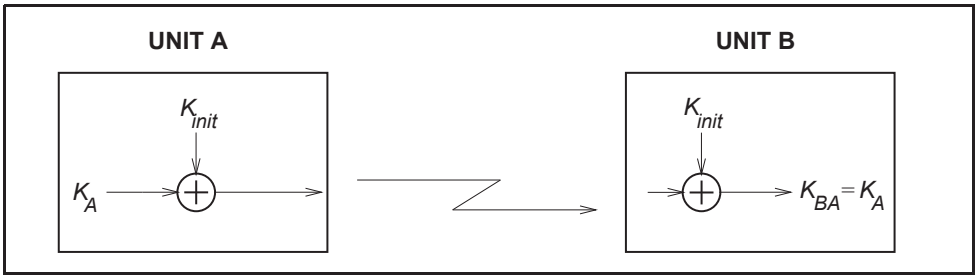



*Figure 3.1: Generation of unit key. When the unit key has been exchanged, the initialization key is discarded in both devices.*

### 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 device 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

$$LK\_K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A) \quad \text{(EQ 1)}$$

and

$$LK\_K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B), \quad \text{(EQ 2)}$$

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 Bluetooth



LifeScan, Inc. Ex. 1019
Page 1033 of 1420
IPR2025-00103

device address of the other device. Thus, device A shall calculate (EQ 2) on page 863 and device B shall calculate (EQ 1) on page 863. 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 is depicted in Figure 3.2 on page 864.

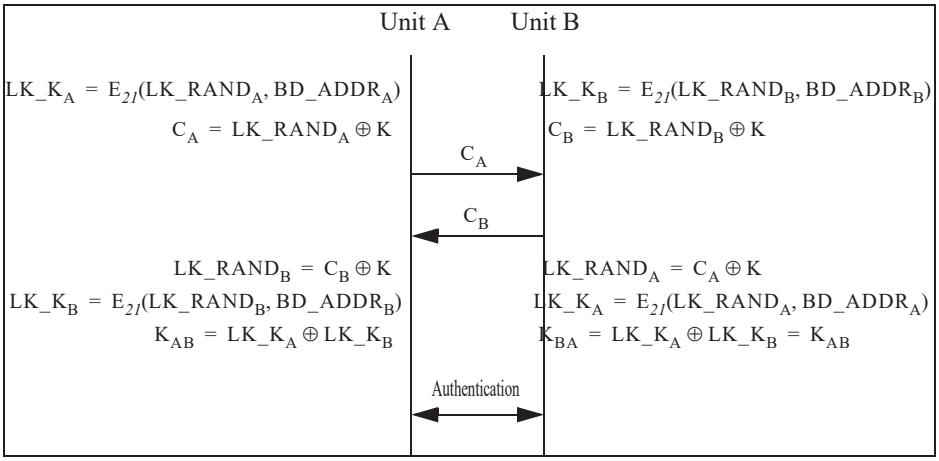



*Figure 3.2: Generating a combination key. The old link key (K) is discarded after the exchange of a new combination key has succeeded*

### 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 number (COF), 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:[1]

$$\text{COF} = \begin{cases} \text{BD\_ADDR} \cup \text{BD\_ADDR}, & \text{if link key is a master key} \\ \text{ACO}, & \text{otherwise.} \end{cases} \quad \text{(EQ 3)}$$

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

1. $x \cup y$ denotes the concatenation of $x$ and $y$.



LifeScan, Inc. Ex. 1019
Page 1034 of 1420
IPR2025-00103

encryption mode. The details of the key generating function $E_3$ can be found in Section 6.4 on page 887.

### 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 Section 3.2.8 on page 866. 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.

### 3.2.7 Modifying the 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 re-initialization of all devices connecting. Changing the unit key can be justified in some circumstances, e.g. to deny access to all previously allowed devices.



LifeScan, Inc. Ex. 1019
Page 1035 of 1420
IPR2025-00103

If the key change concerns combination keys, then the procedure is straightforward. The change procedure is identical to the procedure described in Figure 3.2 on page 864, 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.

### 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 Section 3.2.6 on page 865), 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

$$K_{master} = E_{22}(RAND1, RAND2, 16). \quad \text{(EQ 4)}$$

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 (non-temporary) 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:

$$K_C = E_3(K_{master}, EN\_RAND, COF) \quad \text{(EQ 5)}$$

where the value of COF shall be derived from the master's BD_ADDR as specified by equation (EQ 3) on page 864. The details of the encryption key gener-

ating function are described in Section 6.4 on page 887. The message flow between the master and the slave when generating the master key is depicted in Figure 3.3. Note that in this case the ACO produced during the authentication is not used when computing the ciphering key.

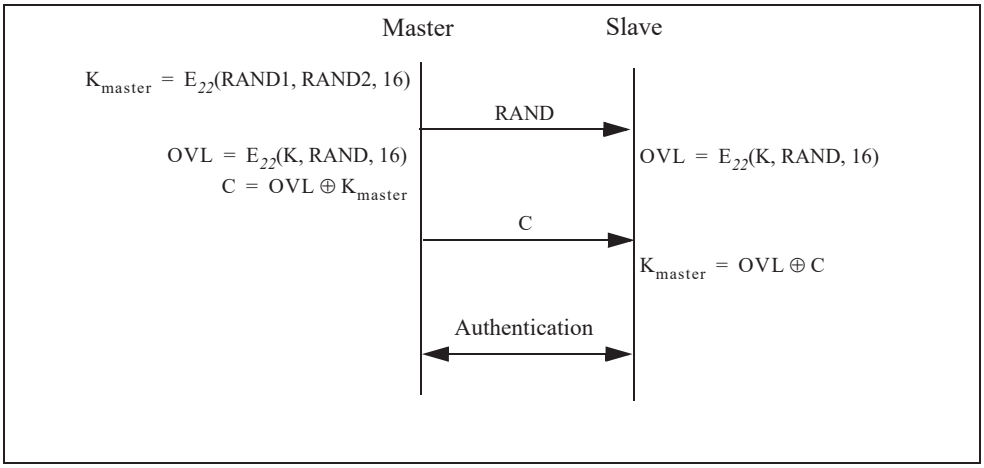



*Figure 3.3: Master link key distribution and computation of the corresponding encryption key.*



LifeScan, Inc. Ex. 1019
Page 1037 of 1420
IPR2025-00103

# 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 re-synchronized for every payload. The overall principle is shown in Figure 4.1 on page 868.

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 and 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 re-synchronization frequency will disrupt such attacks.

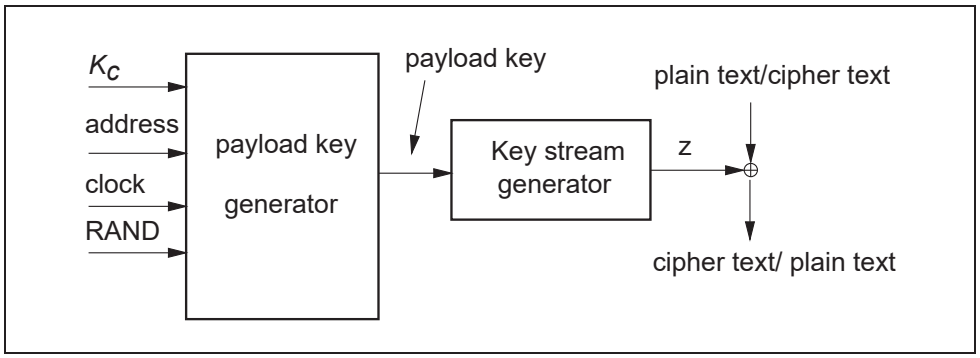



*Figure 4.1: Stream ciphering for Bluetooth with $E_0$.*



LifeScan, Inc. Ex. 1019
Page 1038 of 1420
IPR2025-00103

## 4.1 ENCRYPTION KEY SIZE NEGOTIATION

Each device implementing the baseband specification shall have a parameter defining the maximal allowed key length, $L_{max}$, $1 \leq L_{max} \leq 16$ (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 $L_{min}^{(S)} \leq L_{sug}^{(M)}$, 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 case of an abort link encryption can not 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.

## 4.2 ENCRYPTION OF BROADCAST MESSAGES

There may be three settings for the baseband regarding encryption:

1. No encryption.
   This is the default setting. No messages are encrypted
2. 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.
3. 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.



LifeScan, Inc. Ex. 1019
Page 1039 of 1420
IPR2025-00103

## 4.3 ENCRYPTION CONCEPT

| Broadcast traffic | Individually addressed traffic |
|---|---|
| No encryption | No encryption |
| No encryption | Encryption, $K_{master}$ |
| Encryption, $K_{master}$ | Encryption, $K_{master}$ |

*Table 4.1: Possible encryption modes for a slave in possession of a master key.*

For the encryption routine, a stream cipher algorithm is used in which ciphering bits are bit-wise 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 Bluetooth device address (BD_ADDR), 26 bits of the master real-time clock ($CLK_{26\text{-}1}$) and the encryption key $K_C$ as input, see Figure 4.2 on page 871 (where it is assumed that device A is the master).

The encryption key $K_C$ is derived from the current link key, COF, and a random number, $EN\_RAND_A$ (see Section 6.4 on page 887). 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 Section 4.5 on page 874.

The real-time clock is incremented for each slot. The $E_0$ algorithm shall be re-initialized at the start of each new packet (i.e. for Master-to-Slave as well as for Slave-to-Master transmission). By using $CLK_{26\text{-}1}$ at least one bit is changed between two transmissions. Thus, a new keystream is generated after each re-initialization. For packets covering more than a single slot, the Bluetooth clock 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.



LifeScan, Inc. Ex. 1019
Page 1040 of 1420
IPR2025-00103

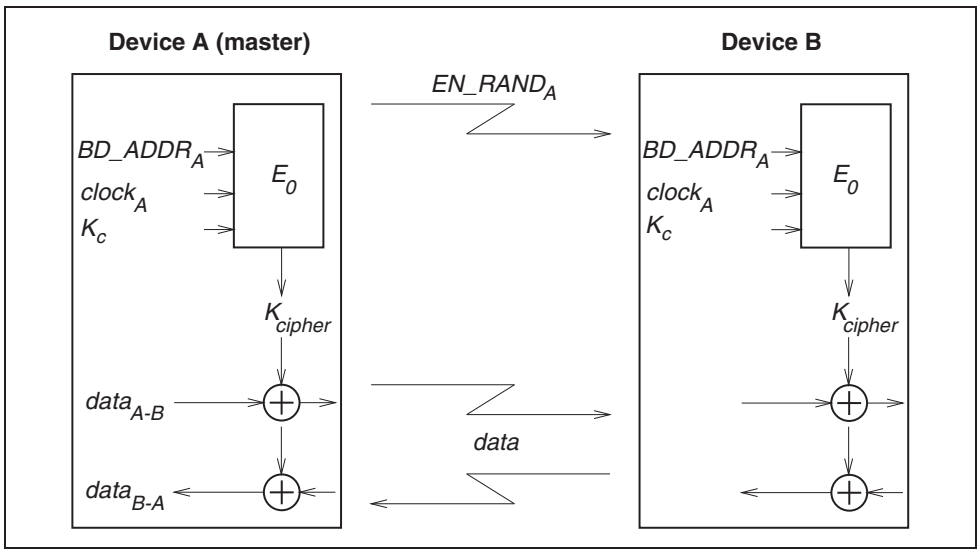



*Figure 4.2: Functional description of the encryption procedure*

## 4.4 ENCRYPTION ALGORITHM

The system uses linear feedback shift registers (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 Bluetooth address, the master clock bits $CLK_{26-1}$, and a 128-bit RAND value. Figure 4.3 on page 872 shows the setup.

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 4.2 on page 872. 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.



LifeScan, Inc. Ex. 1019
Page 1041 of 1420
IPR2025-00103

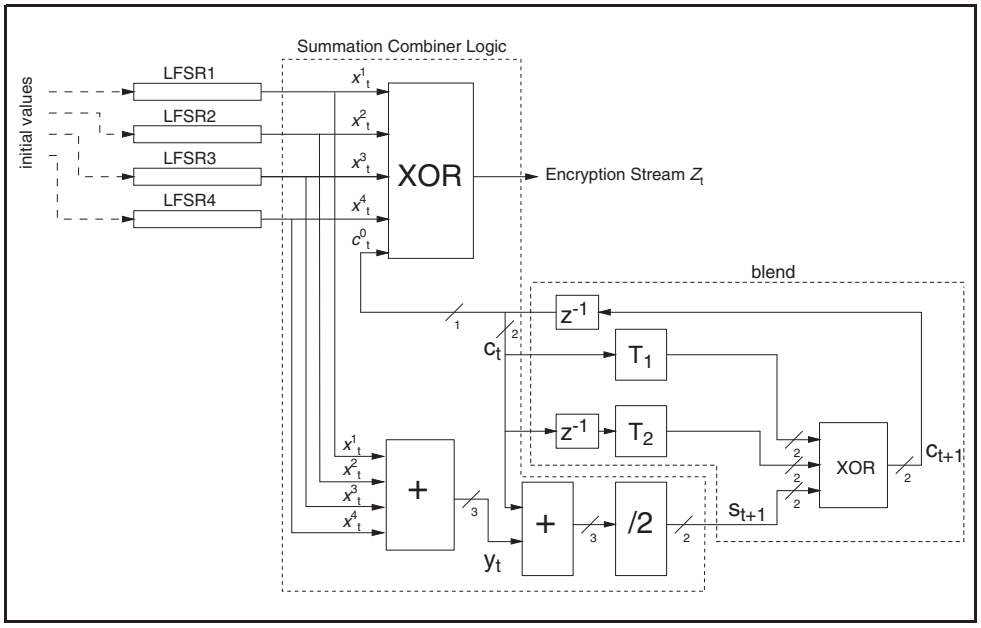



*Figure 4.3: Concept of the encryption engine.*

| 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 |

*Table 4.2: The four primitive feedback polynomials.*

Let $x_t^i$ denote the $t^{th}$ symbol of $LFSR_i$. The value $y_t$ is derived from the four-tuple $x_t^1, \ldots, x_t^4$ using the following equation:

$$y_t = \sum_{i=1}^{4} x_t^i, \qquad \text{(EQ 6)}$$

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 the following equations:

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\}, \qquad \text{(EQ 7)}$$

$$s_{t+1} = (s^{1}_{t+1}, s^{0}_{t+1}) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\}, \quad \text{(EQ 8)}$$

$$c_{t+1} = (c^{1}_{t+1}, c^{0}_{t+1}) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \quad \text{(EQ 9)}$$

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 $\alpha$ be a zero of this polynomial in GF(4). The mappings $T_1$ and $T_2$ are now defined as:

$$T_1: GF(4) \rightarrow GF(4)$$
$$x \mapsto x$$
$$T_2: GF(4) \rightarrow GF(4)$$
$$x \mapsto (\alpha + 1)x.$$

The elements of GF(4) can be written as binary vectors. This is summarized in Table 4.3.

| x | $T_1[x]$ | $T_2[x]$ |
|---|---|---|
| 00 | 00 | 00 |
| 01 | 01 | 11 |
| 10 | 10 | 01 |
| 11 | 11 | 10 |

*Table 4.3: The mappings $T_1$ and $T_2$.*

Since the mappings are linear, they can be implemented using XOR gates; i.e.

$$T_1: \quad (x_1, x_0) \mapsto (x_1, x_0),$$
$$T_2: \quad (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0).$$

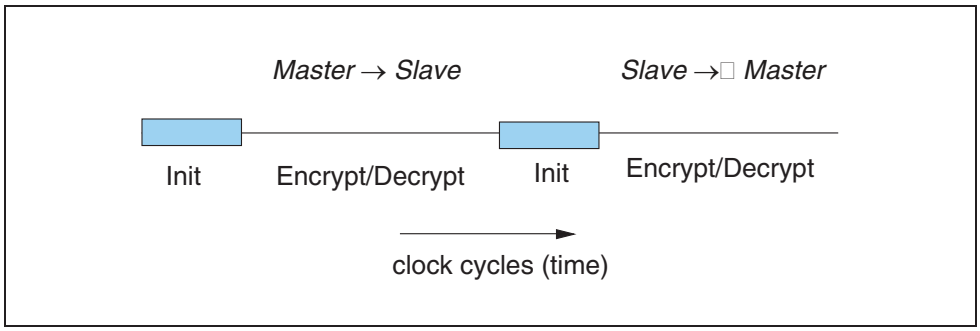
### 4.4.1 The operation of the cipher

Figure 4.4 on page 873 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 4.4: Overview of the operation of the encryption engine. Between each start of a packet (TX or RX), the LFSRs are re-initialized.*



LifeScan, Inc. Ex. 1019
Page 1043 of 1420
IPR2025-00103




## 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 Bluetooth device address, and the 26 master clock bits $CLK_{26\text{-}1}$.

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 (EQ 10) on page 875.

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 which 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 Bluetooth device address is the LSB of BD_ADDR[3].

The details of the initialization shall be as follows:

1. 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



LifeScan, Inc. Ex. 1019
Page 1044 of 1420
IPR2025-00103

effective key length in number of octets. The resulting encryption key is $K'_C$:

$$K'_C(x) = g_2^{(L)}(x)(K_C(x) \bmod g_1^{(L)}(x)), \quad \text{(EQ 10)}$$

where $\deg(g_1^{(L)}(x)) = 8L$ and $\deg(g_2^{(L)}(x)) \leq 128 - 8L$. The polynomials are defined in Table 4.4.

2. Shift the 3 inputs $K'_C$, the Bluetooth device address, the clock, and the six-bit constant 111001 into the LFSRs. In total, 208 bits are shifted in:

   a) Open all switches shown in Figure 4.5 on page 876;

   b) Arrange inputs bits as shown in Figure 4.5; Set the content of all shift register elements to zero. Set $t = 0$.

   c) Start shifting bits into the LFSRs. The rightmost bit at each level of Figure 4.5 is the first bit to enter the corresponding LFSR.

   d) When the first input bit at level i reaches the rightmost position of $LFSR_i$, close the switch of this LFSR.

   e) 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.

   f) 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.

3. To mix initial data, continue to clock until 200 symbols have been produced with all switches closed ($t = 239$);

4. 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 4.6 at $t = 240$;

After the parallel load in item 4, the blend register contents shall be updated for each subsequent clock.

| 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 |

*Table 4.4: Polynomials used when creating $K'_c$.* [1]



LifeScan, Inc. Ex. 1019
Page 1045 of 1420
IPR2025-00103

| L | deg | $g_1^{(L)}$ | deg | $g_2^{(L)}$ |
|---|---|---|---|---|
| 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 |
| 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 |

*Table 4.4: Polynomials used when creating $K'_c$. .* [1]

1. All polynomials are in hexadecimal notation. The LSB is in the rightmost position.

In Figure 4.5, all bits are shifted into the LFSRs, starting with the least significant bit (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 $CLK_{26}$.

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).

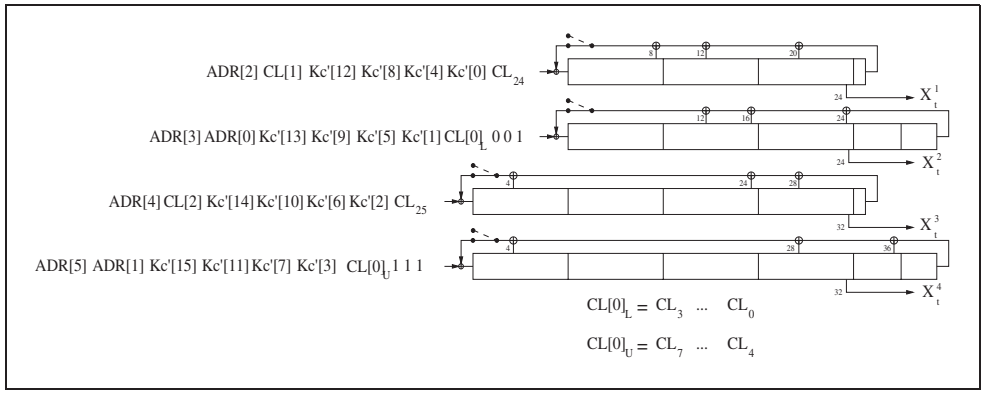



*Figure 4.5: Arranging the input to the LFSRs.*



LifeScan, Inc. Ex. 1019
Page 1046 of 1420
IPR2025-00103

In Figure 4.6, 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 $LFSR_4$.

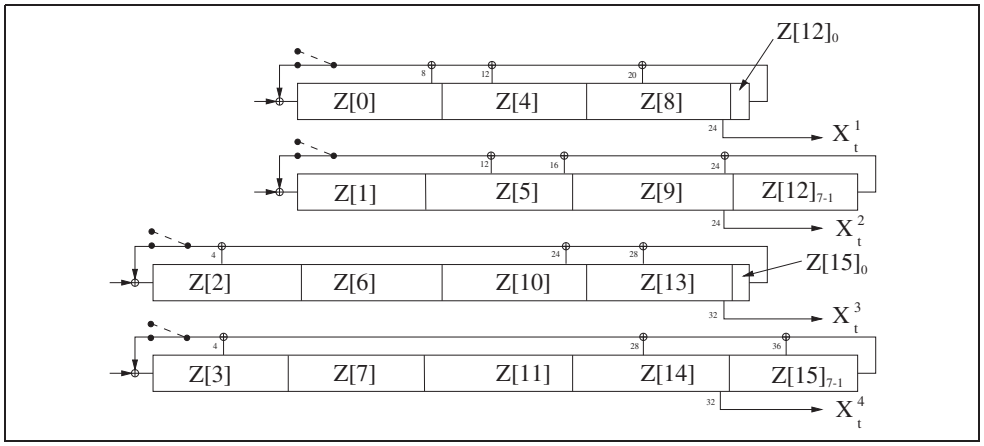



*Figure 4.6: Distribution of the 128 last generated output symbols within the LFSRs.*

## 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 [Part G] Section 1 on page 749. A necessary, but not sufficient, condition for all Bluetooth compliant implementations of encryption is to produce these encryption streams for identical initialization values.



LifeScan, Inc. Ex. 1019
Page 1047 of 1420
IPR2025-00103

# 5 AUTHENTICATION

Authentication uses a challenge-response scheme in which a claimant's knowledge of a secret key is checked through a 2-move protocol using symmetric secret keys. The latter implies that a correct claimant/verifier pair share the same secret key, for example 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 SRES to the verifier, see Figure 5.1 on page 878. This figure also shows that the input to $E_1$ consists of the tuple $AU\_RAND_A$ and the Bluetooth device address (BD_ADDR) of the claimant. The use of this address prevents a simple reflection attack[1]. The secret K shared by devices A and B is the current link key.

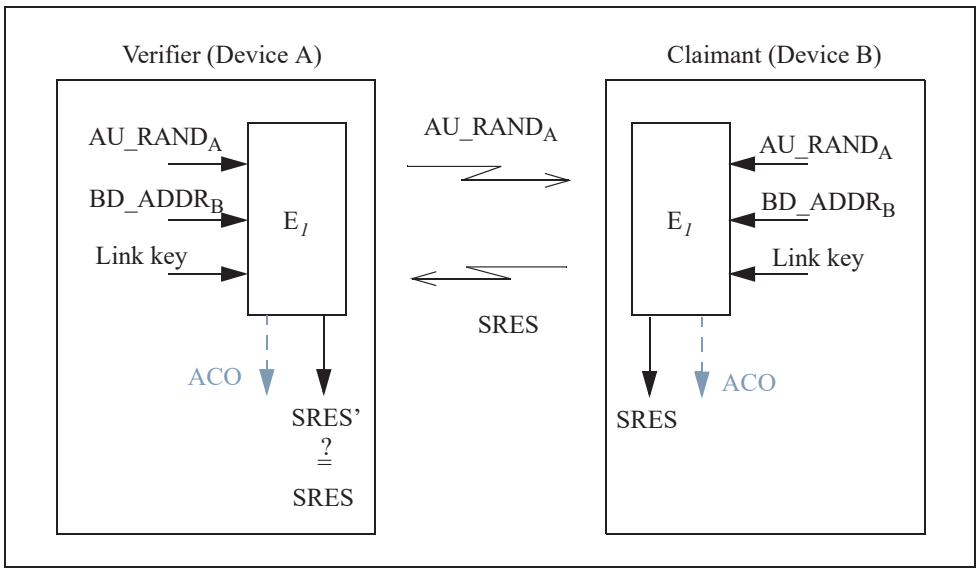



*Figure 5.1: Challenge-response for the Bluetooth.*

The challenge-response scheme for symmetric keys is depicted in Figure 5.2 on page 879.

1. The reflection attack actually forms no threat because all service requests are dealt with on a FIFO bases. When preemption is introduced, this attack is potentially dangerous.



LifeScan, Inc. Ex. 1019
Page 1048 of 1420
IPR2025-00103

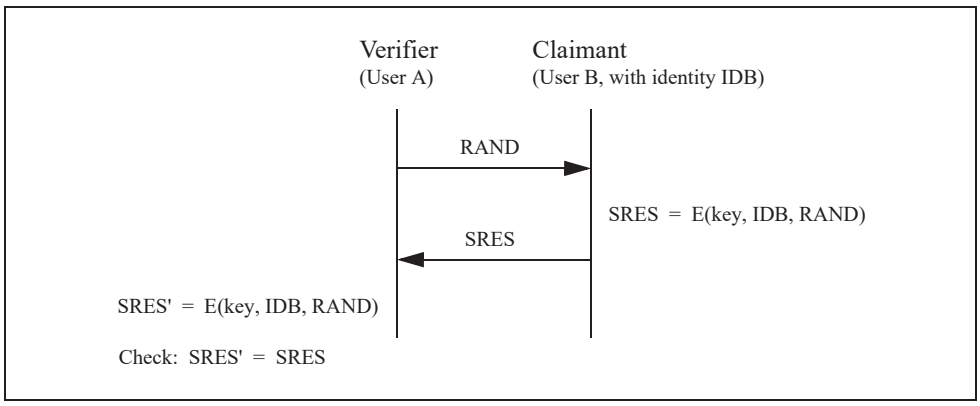



*Figure 5.2: Challenge-response for symmetric key systems.*

The verifier is not required to be the master. The application indicates which device has to be authenticated. Some applications only require 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 5.1 on page 878, 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.

## 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, the waiting interval shall be increased exponentially. That is, 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[1]. 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 protect a device's private key, a device should implement a method to prevent an attacker from retrieving useful information about the device's private key using invalid public keys. For this purpose, a device can use one of the following methods:

- Change its private key after three failed attempts from any BD_ADDR and after 10 successful pairings from any BD_ADDR; or after a combination of these such that 3 successful pairings count as one failed pairing; or
- Verify that the received public keys from any BD_ADDR are on the correct curve; or
- Implement elliptic curve point addition and doubling using formulas that are valid only on the correct curve.

1. Another appropriate value larger than 1 may be used.



LifeScan, Inc. Ex. 1019
Page 1050 of 1420
IPR2025-00103

# 6 THE AUTHENTICATION AND KEY-GENERATING FUNCTIONS

This section describes the algorithms used for authentication and key generation.

## 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.

$$\begin{aligned} A_r:\ \{0,1\}^{128} \times \{0,1\}^{128} &\rightarrow \{0,1\}^{128} \\ (k \times x) &\mapsto t. \end{aligned} \qquad \text{(EQ 11)}$$

The details of $A_r$ are given in the next section. The function $E_1$ is constructed using $A_r$ as follows

$$\begin{aligned} E_1:\ \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{48} &\rightarrow \{0,1\}^{32} \times \{0,1\}^{96} \\ (K, RAND, address) &\mapsto (SRES, ACO), \end{aligned} \qquad \text{(EQ 12)}$$

where $SRES = Hash(K, RAND, address, 6)[0, \ldots, 3]$, where $Hash$ is a keyed hash function defined as[1],

$$\begin{aligned} &Hash: \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{8 \times L} \times \{6,12\} \rightarrow \{0,1\}^{128} \\ &(K, I_1, I_2, L) \mapsto A'_r([\tilde{K}], [E(I_2, L) +_{16} (A_r(K, I_1) \oplus_{16} I_1)]), \end{aligned} \qquad \text{(EQ 13)}$$

and where

$$\begin{aligned} &E: \{0,1\}^{8 \times L} \times \{6,12\} \rightarrow \{0,1\}^{8 \times 16} \\ &(X[0, \ldots, L-1], L) \mapsto (X[i \pmod L] \ \text{ for } i = 0 \ldots 15), \end{aligned} \qquad \text{(EQ 14)}$$

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 $\tilde{K}$ for the second use of $A_r$ (actually $A'_r$) is offset from $K$ as follows[2]

1. The operator $+_{16}$ denotes bytewise addition mod 256 of the 16 octets, and the operator $\oplus_{16}$ denotes bytewise XORing of the 16 octets.
2. The constants are the largest primes below 257 for which 10 is a primitive root.



LifeScan, Inc. Ex. 1019
Page 1051 of 1420
IPR2025-00103

$$\begin{aligned}
\tilde{K}[0] &= (K[0] + 233) \mod 256, & \tilde{K}[1] &= K[1] \oplus 229, \\
\tilde{K}[2] &= (K[2] + 223) \mod 256, & \tilde{K}[3] &= K[3] \oplus 193, \\
\tilde{K}[4] &= (K[4] + 179) \mod 256, & \tilde{K}[5] &= K[5] \oplus 167, \\
\tilde{K}[6] &= (K[6] + 149) \mod 256, & \tilde{K}[7] &= K[7] \oplus 131, \\
\tilde{K}\{8\} &= K[8] \oplus 233, & \tilde{K}[9] &= (K[9] + 229) \mod 256, \\
\tilde{K}[10] &= K[10] \oplus 223, & \tilde{K}[11] &= (K[11] + 193) \mod 256, \\
\tilde{K}[12] &= K[12] \oplus 179, & \tilde{K}[13] &= (K[13] + 167) \mod 256, \\
\tilde{K}[14] &= K[14] \oplus 149, & \tilde{K}[15] &= (K[15] + 131) \mod 256.
\end{aligned} \qquad \text{(EQ 15)}$$

A data flowchart of the computation of $E_1$ is shown in Figure 6.1 on page 882. $E_1$ is also used to deliver the parameter ACO (Authenticated Ciphering Offset) that is used in the generation of the ciphering key by $E_3$, see equations (EQ 3) on page 864 and (EQ 23) on page 887. The value of ACO is formed by the octets 4 through 15 of the output of the hash function defined in (EQ 13) on page 881:

$$ACO = \text{Hash}(K, RAND, \text{address}, 6)[4, \ldots, 15]. \qquad \text{(EQ 16)}$$

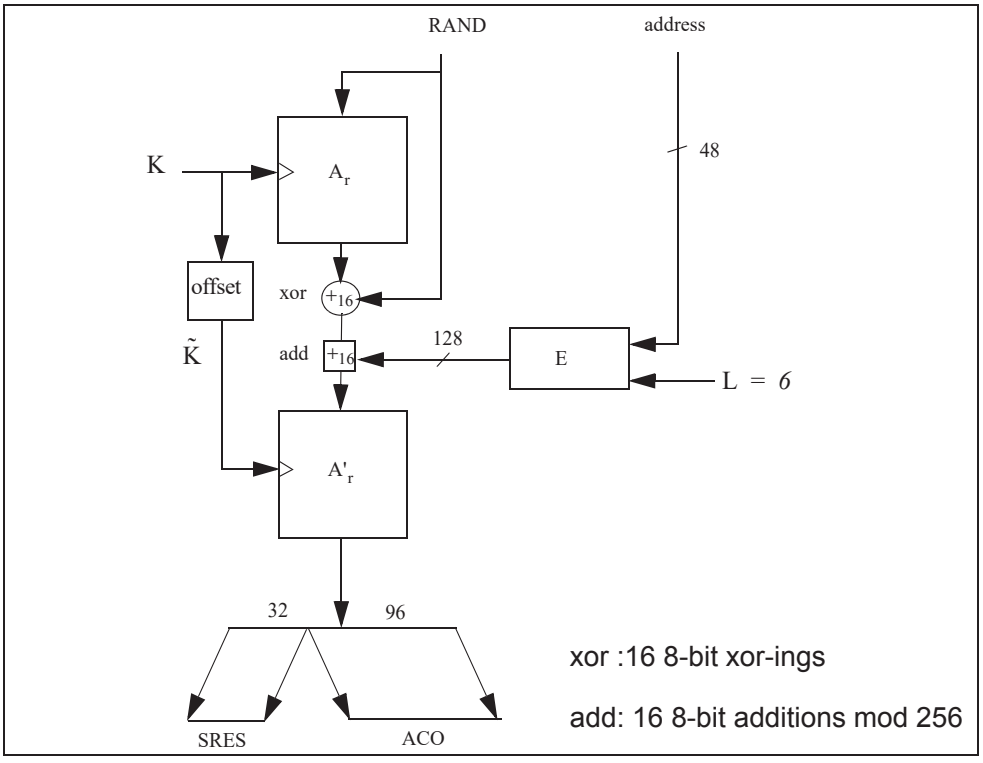



*Figure 6.1: Flow of data for the computation of $E_1$.*

## 6.2 THE FUNCTIONS $A_r$ AND $A'_r$

The function $A_r$ is identical to SAFER+. It consists of a set of 8 layers, (each layer is called a round) and a parallel mechanism for generating the sub keys $K_p[j]$, $p = 1, 2, \ldots, 17$, 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 non-invertible and prevents the use of $A'_r$ (especially in $E_{2x}$) as an encryption function. See Figure 6.2 on page 884 for details.

### 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 6.2 on page 884. The sub keys 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.

### 6.2.2 The substitution boxes “e” and “l”

In Figure 6.2 on page 884 two boxes are shown, marked “e” and “l”. These boxes implement the same substitutions as are used in SAFER+; i.e. they implement

$$
\begin{aligned}
e, l &: \{0, \ldots, 255\} \rightarrow \{0, \ldots, 255\}, \\
e &: i \mapsto (45^i \pmod{257}) \pmod{256}, \\
l &: i \mapsto j \text{ s.t. } i = e(j).
\end{aligned}
$$

Their role, as in the SAFER+ algorithm, is to introduce non-linearity.



LifeScan, Inc. Ex. 1019
Page 1053 of 1420
IPR2025-00103

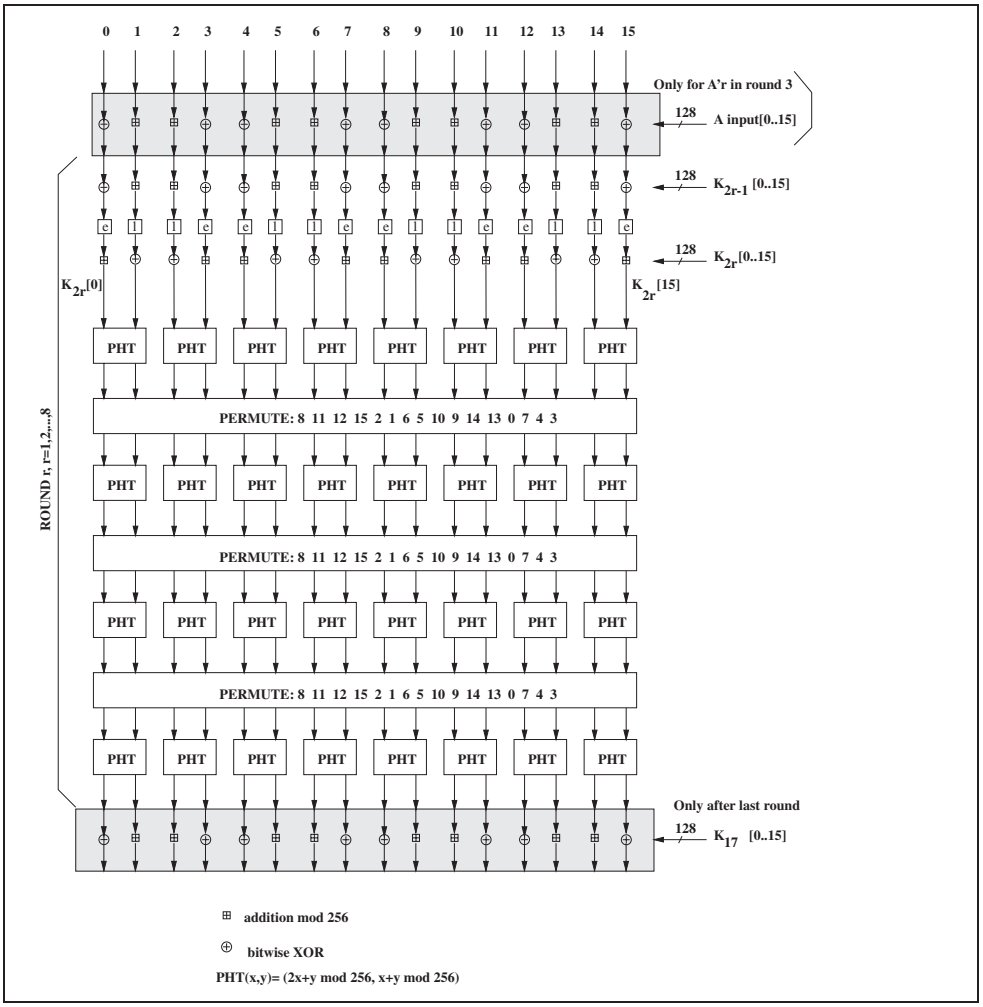



*Figure 6.2: One round in $A_r$ and $A'_r$. The permutation boxes show how input byte indices are mapped onto output byte indices. Thus, position 8 is mapped on position 0 (leftmost), position 11 is mapped on position 1, etc.*

### 6.2.3 Key scheduling

In each round, 2 batches of 16 octet-wide keys are needed. These round keys are derived as specified by the key scheduling in SAFER+. Figure 6.3 on page 885 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 following equation:

$$B_p[i] = ((45^{(45^{17p+i+1} \bmod 257)} \bmod 257) \bmod 256), \text{ for } i = 0, \ldots, 15. \qquad \text{(EQ 17)}$$

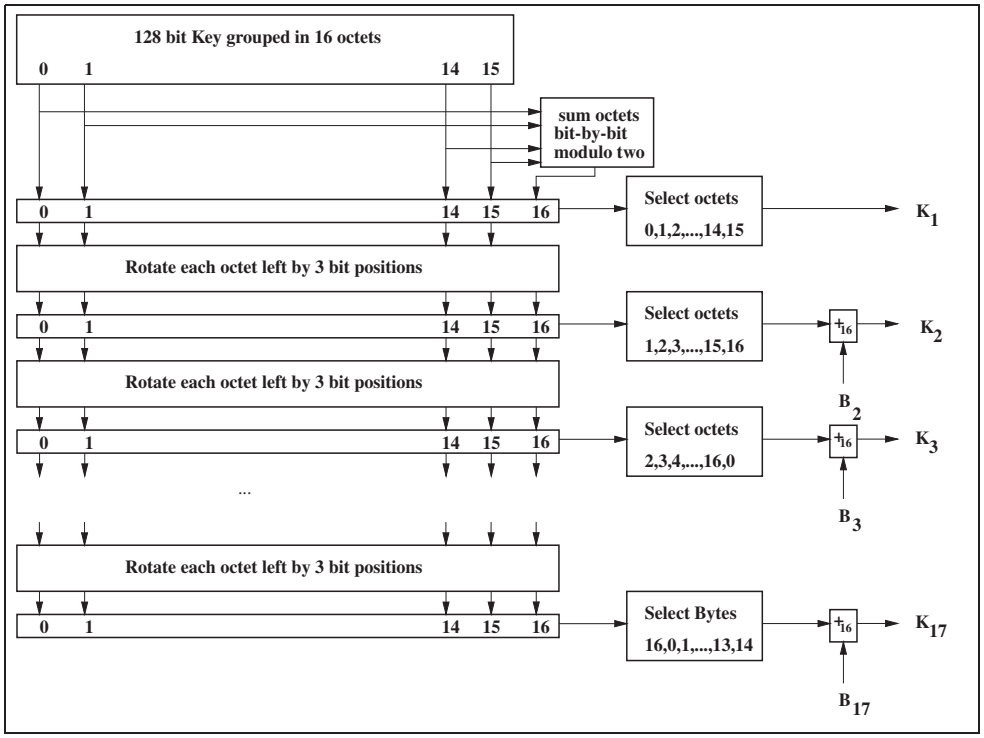



*Figure 6.3: Key scheduling in* $A_r$.

## 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 6.4 on page 887. The figure shows two modes of operation for the algorithm. In the first mode, $E_{21}$ produces a 128-bit kink 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 Section 3.2.1 on page 862 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 following equations:

$$E_{21}: \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{128}$$
$$(\text{RAND}, \text{address}) \mapsto A'_r(X, Y) \qquad \text{(EQ 18)}$$

where (for mode 1)

$$\begin{cases} X = \text{RAND}[0 \ldots 14] \cup (\text{RAND}[15] \oplus 6) \\ Y = \bigcup_{i=0}^{15} \text{address}[i \pmod 6] \end{cases} \qquad \text{(EQ 19)}$$

Let $L$ be the number of octets in the user PIN. The augmenting is defined by

$$\text{PIN}' = \begin{cases} \text{PIN}[0 \ldots L-1] \cup \text{BD\_ADDR}[0 \ldots \min\{5, 15-L\}], & L < 16, \\ \text{PIN}[0 \ldots L-1], & L = 16, \end{cases} \qquad \text{(EQ 20)}$$

Then, in mode 2, $E_2$ (denoted $E_{22}$) is

$$E_{22}: \{0,1\}^{8L'} \times \{0,1\}^{128} \times \{1, 2, \ldots, 16\} \rightarrow \{0,1\}^{128}$$
$$(\text{PIN}', \text{RAND}, L') \mapsto A'_r(X, Y) \qquad \text{(EQ 21)}$$

where

$$\begin{cases} X = \bigcup_{i=0}^{15} \text{PIN}'[i \pmod{L'}], \\ Y = \text{RAND}[0 \ldots 14] \cup (\text{RAND}[15] \oplus L'), \end{cases} \qquad \text{(EQ 22)}$$

and $L' = \min\{16, L+6\}$ is the number of octets in PIN'.



LifeScan, Inc. Ex. 1019
Page 1056 of 1420
IPR2025-00103

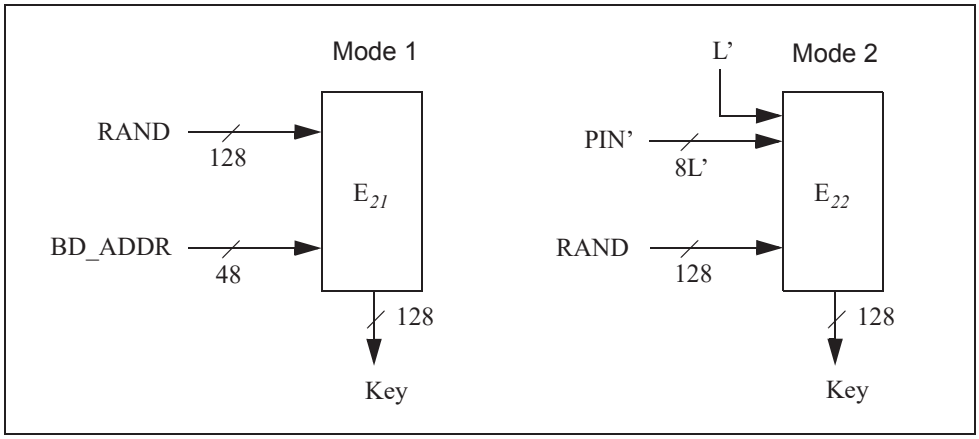



*Figure 6.4: Key generating algorithm $E_2$ and its two modes. Mode 1 is used for unit and combination keys, while mode 2 is used for $K_{init}$ and $K_{master}$.*

## 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

$$E_3: \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{96} \rightarrow \{0,1\}^{128}$$
$$(K, RAND, COF) \mapsto Hash(K, RAND, COF, 12) \quad \text{(EQ 23)}$$

where Hash is the hash function as defined by (EQ 13) on page 881. 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 Section 4.5 on page 874. A block scheme of $E_3$ is depicted in Figure 6.5.

The value of COF is determined as specified by equation (EQ 3) on page 864.

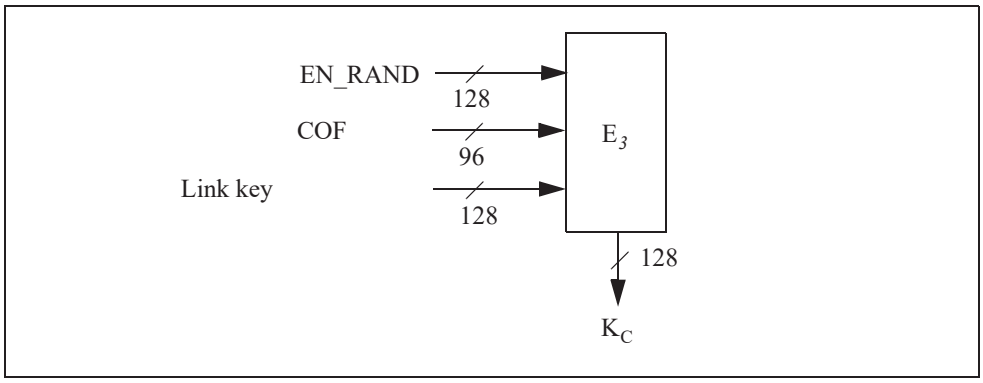



*Figure 6.5: Generation of the encryption key*



LifeScan, Inc. Ex. 1019
Page 1057 of 1420
IPR2025-00103

# 7 SECURE SIMPLE PAIRING

The Secure Simple Pairing security functions and procedures are described in this section. In addition, a cryptographic analysis of each procedure is provided.

There are five phases of Secure Simple Pairing:

- Phase 1: Public key exchange
- Phase 2: Authentication Stage 1
- Phase 3: Authentication Stage 2
- Phase 4: Link key calculation
- Phase 5: LMP Authentication and Encryption

Phases 1, 3, 4 and 5 are the same for all protocols whereas phase 2 (Authentication Stage 1) is different depending on the protocol used. Distributed through these five phases are 13 steps.

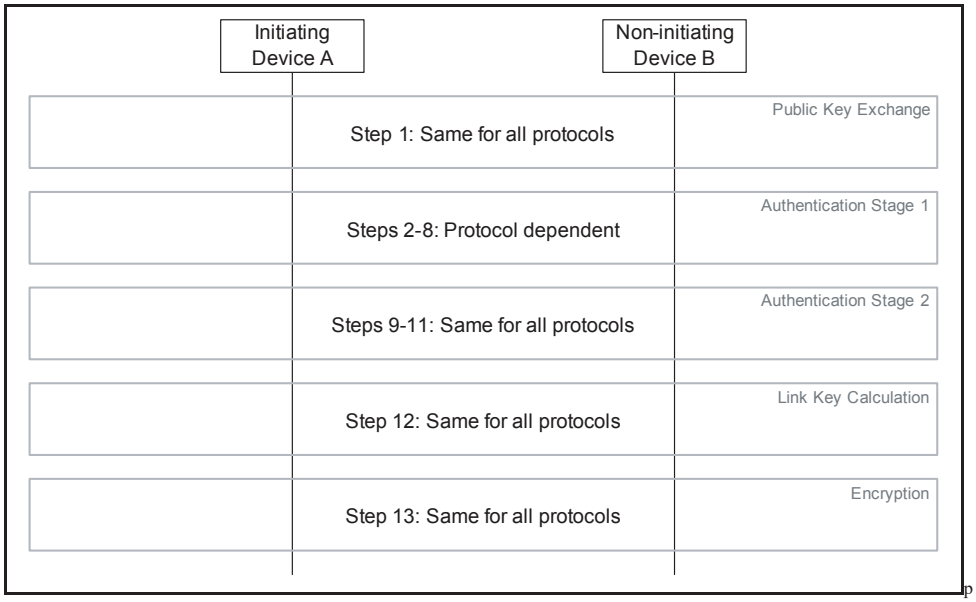



*Figure 7.1: Simple Pairing Security Phases*

The terminology used in the security sections is defined in Table 7.1:

| Term | Definition |
|---|---|
| Cx | Commitment value from device X |
| Cxi | $i^{th}$ commitment value from device X. Only used in the passkey entry protocol |

*Table 7.1: Terminology*



LifeScan, Inc. Ex. 1019
Page 1058 of 1420
IPR2025-00103

| Term | Definition |
|---|---|
| DHKey | Diffie Hellman key |
| Ex | Check value from device X |
| f1( ) | Used to generate the 128-bit commitment values Ca and Cb |
| f2( ) | Used to compute the link key and possible other keys from the DHKey and random nonces |
| f3( ) | Used to compute check values Ea and Eb in Authentication Stage 2 |
| g( ) | Used to compute numeric check values |
| IOcapA | IO capabilities of device A |
| IOcapB | IO capabilities of device B |
| LK | Link Key |
| Nx | Nonce (unique random value) from device X |
| Nxi | $i^{th}$ nonce (unique random value) from device X. Only used in the passkey entry protocol |
| PKx | Public Key of device X |
| rx | Random value generated by device X |
| rxi | Bit i of the random value rx. Only used in the passkey entry protocol |
| SKx | Secret (Private) Key of device X |
| Vx | Confirmation value on device X. Only used in the numeric compare protocol. |
| X | BD_ADDR of device X |

*Table 7.1: Terminology*

## 7.1 PHASE 1: PUBLIC KEY EXCHANGE

Initially, each device generates its own Elliptic Curve Diffie-Hellman (ECDH) public-private key pair (step 1). This key pair needs to be generated only once per device and may be computed in advance of pairing. A device may, at any time, choose to discard its public-private key pair and generate a new one, although there is not a requirement to do so.

Pairing is initiated by the initiating device sending its public key to the receiving device (step 1a). The responding device replies with its own public key (step 1b) These public keys are not regarded as secret although they may identify the devices. Note that steps 1a and 1b are the same in all three protocols.



LifeScan, Inc. Ex. 1019
Page 1059 of 1420
IPR2025-00103

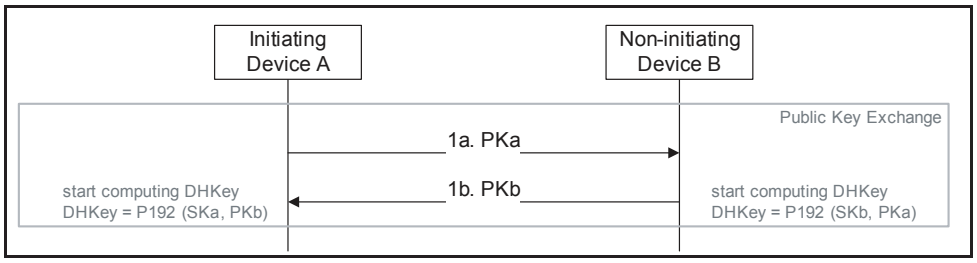



*Figure 7.2: Public Key Exchange Details*

## 7.2 PHASE 2: AUTHENTICATION STAGE 1

Authentication Stage 1 has three different protocols: Numeric Comparison, Out-of-Band, and Passkey Entry. Note that there is not a separate protocol for the Just Works association model. This association model shares the Numeric Comparison protocol.

The protocol is chosen based on the IO capabilities of the two devices.

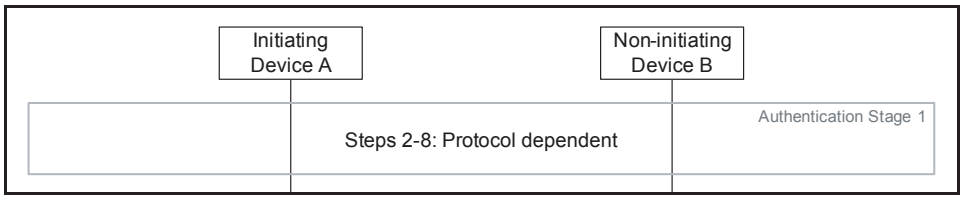



*Figure 7.3: Authentication Stage 1 (High Level)*

The three protocols are described in the following sections.

### 7.2.1 Authentication Stage 1: Numeric Comparison Protocol

The Numeric Comparison protocol provides limited protection against active "man-in-the-middle" (MITM) attacks as an active man-in-the-middle will succeed with a probability of 0.000001 on each iteration of the protocol. Provided that there is no MITM at the time the pairing is performed, the shared Link Key that is generated is computationally secure from even a passive eavesdropper that may have been present during the pairing.

The sequence diagram of Authentication Stage 1 for the Numeric Comparison protocol from the cryptographic point of view is shown below:

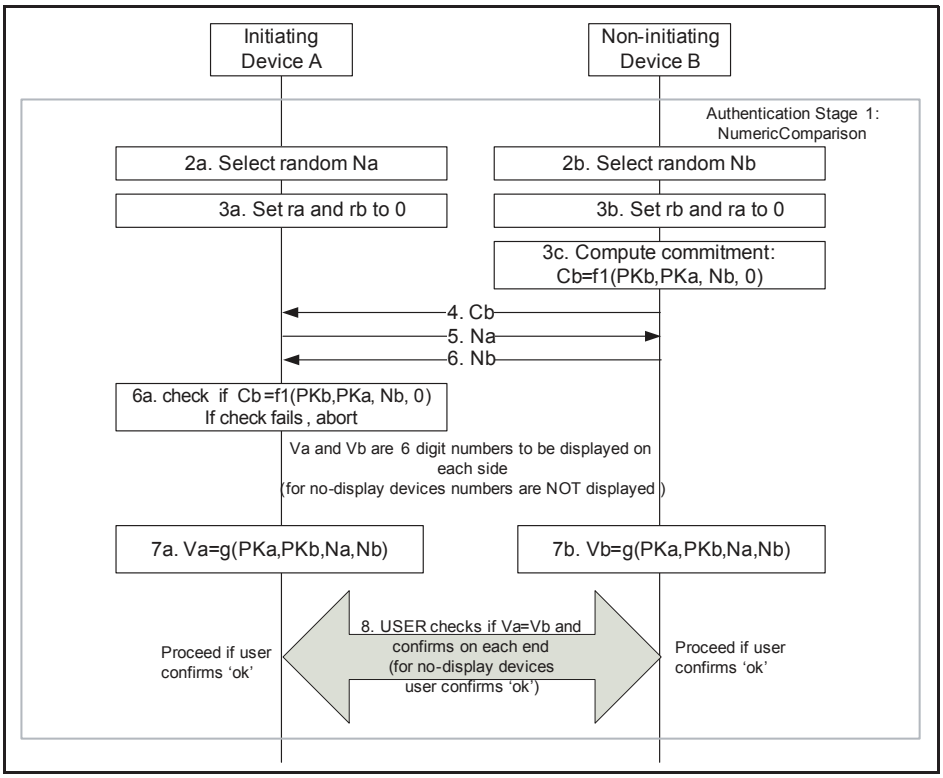



*Figure 7.4: Authentication Stage 1: Numeric Comparison Protocol Details*

After the public keys have been exchanged, each device selects a pseudo-random 128-bit nonce (step 2). This value is used to prevent replay attacks and must be freshly generated with each instantiation of the pairing protocol. This value should be generated directly from a physical source of randomness or with a good pseudo-random generator seeded with a random value from a physical source.

Following this the responding device then computes a commitment to the two public keys that have been exchanged and its own nonce value (step 3c). This commitment is computed as a one-way function of these values and is transmitted to the initiating device (step 4). The commitment prevents an attacker from changing these values at a later time.

The initiating and responding devices then exchange their respective nonce values (steps 5 and 6) and the initiating device confirms the commitment (step 6a). A failure at this point indicates the presence of an attacker or other transmission error and causes the protocol to abort. The protocol may be repeated with or without the generation of new public-private key pairs, but new nonces must be generated if the protocol is repeated.

Assuming that the commitment check succeeds, the two devices each compute 6-digit confirmation values that are displayed to the user on their respective devices (steps 7a, 7b, and 8). The user is expected to check that these 6-

digit values match and to confirm if there is a match. If there is no match, the protocol aborts and, as before, new nonces must be generated if the protocol is to be repeated.

An active MITM must inject its own key material into this process to have any effect other than denial-of-service. A simple MITM attack will result in the two 6-digit display values being different with probability 0.999999. A more sophisticated attack may attempt to engineer the display values to match, but this is thwarted by the commitment sequence. If the attacker first exchanges nonces with the responding device, it must commit to the nonce that it will use with the initiating device before it sees the nonce from the initiating device. If the attacker first exchanges nonces with the initiating device, it must send a nonce to the responding device before seeing the nonce from the responding device. In each case, the attacker must commit to at least the second of its nonces before knowing the second nonce from the legitimate devices. It therefore cannot choose its own nonces in such a way as to cause the display values to match.

### 7.2.2 Authentication Stage 1: Out of Band Protocol

The Out-of-Band protocol is used when authentication information has been received by at least one of the devices and indicated in the OOB Authentication Data Present parameter in the LMP IO capability exchange sequence. The mode in which the discovery of the peer device is first done in-band and then followed by the transmission of authentication parameters through OOB interface is not supported. The sequence diagram for Authentication Stage 1 for Out of Band from the cryptographic point of view is shown below:



LifeScan, Inc. Ex. 1019
Page 1062 of 1420
IPR2025-00103

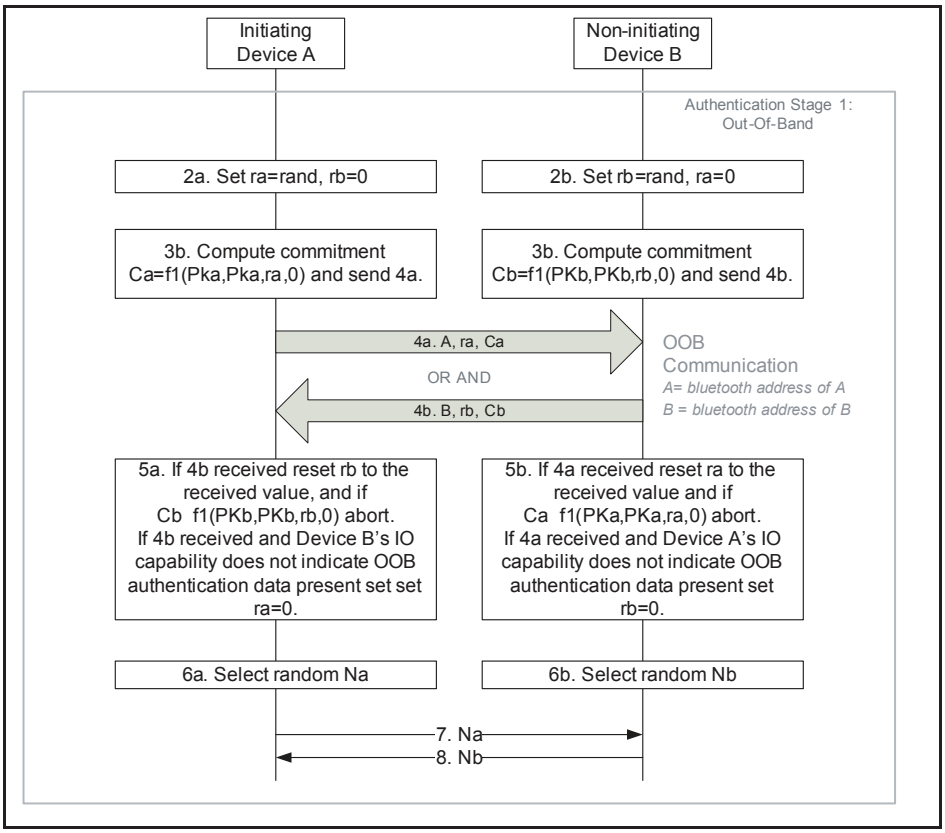



*Figure 7.5: Authentication Stage 1: Out of Band Details*

**Principle of operation**: If both devices can transmit and/or receive data over an out-of-band channel, then mutual authentication will be based on the commitments of the public keys (Ca and Cb) exchanged OOB in Authentication stage 1. If OOB communication is possible only in one direction (e.g., one device can only be read as e.g. is the case of a device equipped with a passive NFC tag), then authentication of the device receiving the OOB communication will be based on that device knowing a random number r sent via OOB. In this case, r must be secret: r can be created afresh every time, or access to the device sending r must be restricted. If r is not sent by a device, it is assumed to be 0 by the device receiving the OOB information in step 4a or 4b.

If the OOB communication cannot be tampered with (i.e. immune to a MITM attack), this association method is also not susceptible to MiTM attack. Also, in the Out of Band protocol the size of authentication parameters (Ca, Cb and ra, rb) need not be restricted by what the user can comfortably read or type. For that reason the Out of Band protocol can be more secure than using the Numeric Comparison or Passkey Entry protocols. However, both devices need to have matching OOB interfaces.

**Roles of A and B**: The OOB Authentication Stage 1 protocol is symmetric with respect to the roles of A and B. It does not require that device A always will ini-

tiate pairing and it automatically resolves asymmetry in the OOB communication, e.g. in the case when one of the devices has an NFC tag and can only transmit OOB.

**Order of steps**: The public key exchange must happen before the verification step 5. In the diagram the in-band public key exchange between the devices (step 1) is done before the OOB communication (step 4). But when the pairing is initiated by an OOB interface, public key exchange will happen after the OOB communication (step 1 will be between steps 4 and 5).

Values of ra and rb: Since the direction of the peer's OOB interface cannot be verified before the OOB communication takes place, a device should always generate and if possible transmit through its OOB interface a random number r to the peer. Each device applies the following rules locally to set the values of its own r and the value of the peer's r:

1. Initially, r of the device is set to a random number and r of the peer is set to 0 (step 2).
2. If a device has received OOB, it sets the peer's r value to what was sent by the peer (Step 5).
3. If the remote device's OOB Authentication Data parameter sent in the LMP IO capabilities exchange sequence is set to No OOB Data Received, it sets its own r value to 0 (Step 5)

These rules ensure that when entering Authentication Stage 2, both A and B have the same values for ra and rb if OOB communication took place.

### 7.2.3 Authentication Stage 1: Passkey Entry Protocol

The Passkey Entry protocol is used when LMP IO capability exchange sequence indicates that Passkey Entry shall be used.

The sequence diagram for Authentication Stage 1 for Passkey Entry from the cryptographic point of view is shown in Figure 7.6.



LifeScan, Inc. Ex. 1019
Page 1064 of 1420
IPR2025-00103

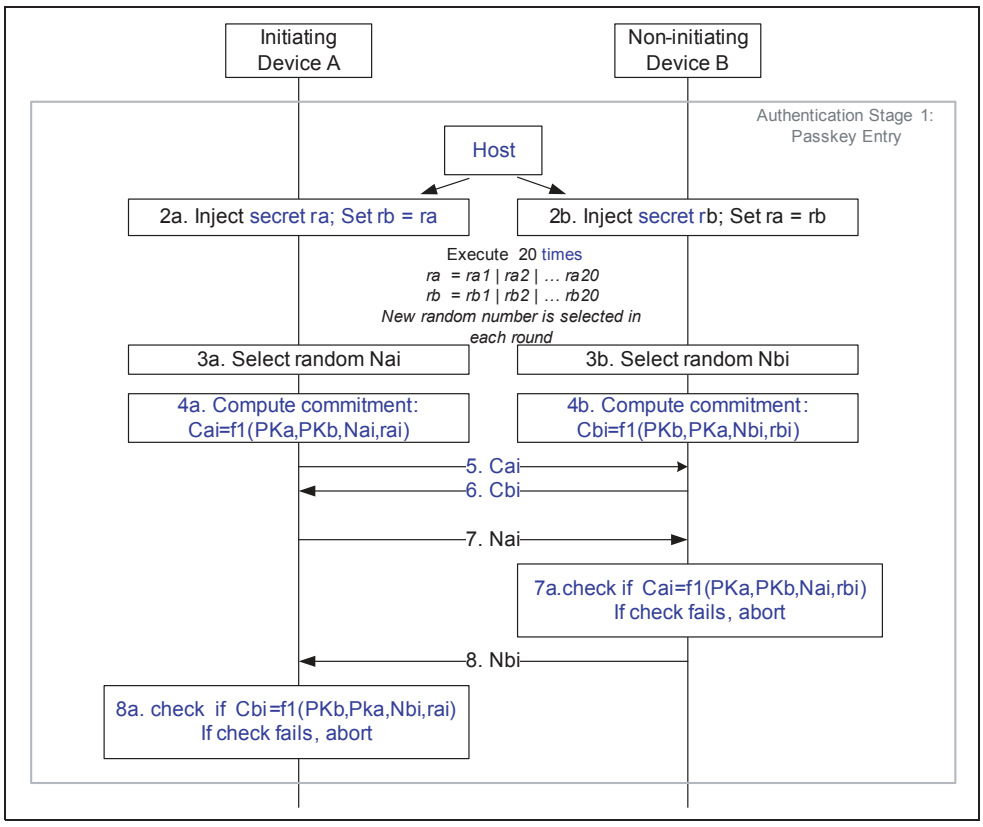



*Figure 7.6: Authentication Stage 1: Passkey Entry Details*

The user inputs an identical Passkey into both devices. Alternately, the Passkey may be generated and displayed on one device, and the user then inputs it into the other (step 2). This short shared key will be the basis of the mutual authentication of the devices. Steps 3 through 8 are repeated k times for a k-bit Passkey --e.g., k=20 for a 6-digit Passkey (999999=0xF423F).

In Steps 3-8, each side commits to each bit of the Passkey, using a long nonce (128 bits), and sending the hash of the nonce, the bit of the Passkey, and both public keys to the other party. The parties then take turns revealing their commitments until the entire Passkey has been mutually disclosed. The first party to reveal a commitment for a given bit of the Passkey effectively reveals that bit of the Passkey in the process, but the other party then has to reveal the corresponding commitment to show the same bit value for that bit of the Passkey, or else the first party will then abort the protocol, after which no more bits of the Passkey are revealed.

This "gradual disclosure" prevents leakage of more than 1 bit of un-guessed Passkey information in the case of a MITM attack. A MITM attacker with only partial knowledge of the Passkey will only receive one incorrectly-guessed bit of the Passkey before the protocol fails. Hence, a MitM attacker who engages

first one side, then the other will only gain an advantage of at most two bits over a simple brute-force guesser which succeeds with probability 0.000001.

The long nonce is included in the commitment hash to make it difficult to brute-force even after the protocol has failed. The public Diffie-Hellman values are included to tie the Passkey protocol to the original ECDH key exchange, to prevent a MitM from substituting the attacker's public key on both sides of the ECDH exchange in standard MitM fashion.

At the end of this stage, Na is set to $Na_{20}$ and Nb is set to $Nb_{20}$ for use in Authentication Stage 2.

## 7.3 PHASE 3: AUTHENTICATION STAGE 2

The second stage of authentication then confirms that both devices have successfully completed the exchange. This stage is identical in all three protocols.

Each device computes a new confirmation value that includes the previously exchanged values and the newly derived shared key (step 9). The initiating device then transmits its confirmation value which is checked by the responding device (step 10). If this check fails, it indicates that the initiating device has not confirmed the pairing, and the protocol must be aborted. The responding device then transmits its confirmation value which is checked by the initiating device (step 11). A failure indicates that the responding device has not confirmed the pairing and the protocol should abort.

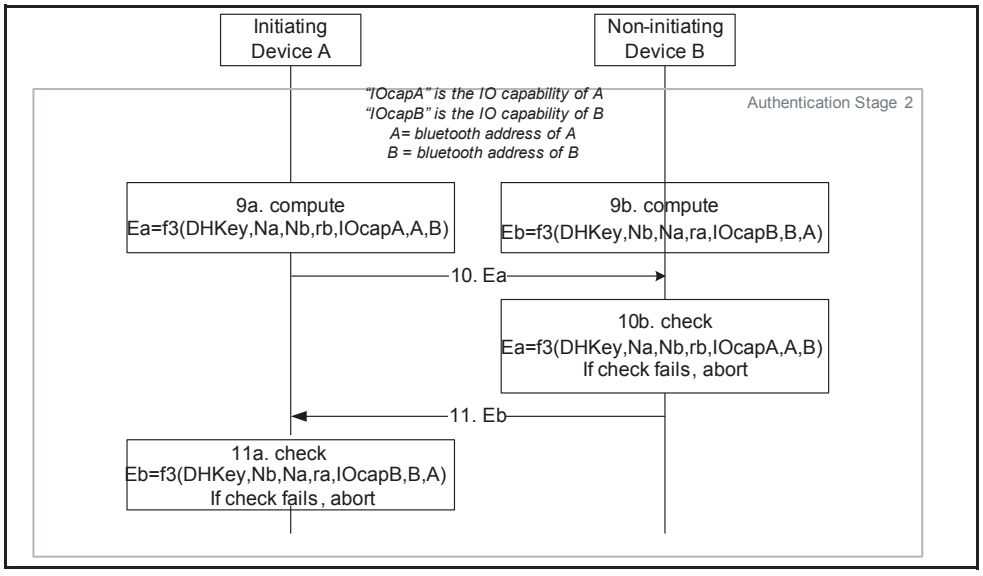



*Figure 7.7: Authentication Stage 2*



LifeScan, Inc. Ex. 1019
Page 1066 of 1420
IPR2025-00103

## 7.4 PHASE 4: LINK KEY CALCULATION

Once both sides have confirmed the pairing, a link key is computed from the derived shared key and the publicly exchanged data (step 12). The nonces ensure the freshness of the key even if long-term ECDH values are used by both sides. This link key is used to maintain the pairing.

When computing the link key both parties shall input the parameters in the same order to ensure that both devices compute the same key: device A's parameters are those of the piconet master and device B's parameters are those of the piconet slave.

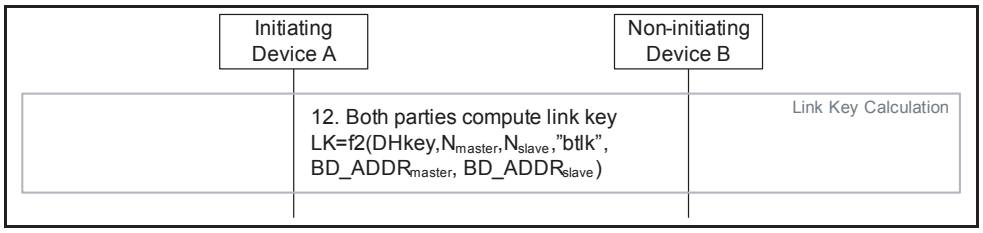



*Figure 7.8: Link Key Calculation*

## 7.5 PHASE 5: LMP AUTHENTICATION AND ENCRYPTION

The final phase in Simple Pairing is generating the encryption keys. This is the same as the final steps in legacy pairing.

## 7.6 ELLIPTIC CURVE DEFINITION

Simple Pairing uses the FIPS P-192 curve defined in [13]. Elliptic curves are specified by p, a, b and ares of the form

$$E: y^2 = x^3 + ax + b \pmod p$$

For each value of b a unique curve can be developed. In NIST P-192:

a = mod (-3, p)

b is defined and its method of generation can be verified by using SHA-1 (with a given seed s and using $b^2c = -27 \pmod p$)

The following parameters are given:

- The prime modulus p, order r, base point x-coordinate Gx, base point y-coordinate Gy.
- The integers p and r are given in decimal form; bit strings and field elements are given in hex.

```
p = 6277101735386680763835789423207666416083908700390324961279
r = 6277101735386680763835789423176059013767194773182842284081
b = 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
```



LifeScan, Inc. Ex. 1019
Page 1067 of 1420
IPR2025-00103

```
Gx = 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
Gy = 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
```

The function P192 is defined as follows. Given an integer u, 0 < u < r, and a point V on the curve E, the value P192(u,V) is computed as the x-coordinate of the $u^{th}$ multiple uV of the point V.

The private keys shall be between 1 and r/2, where r is the Order of the Abelian Group on the elliptic curve (e.g. between 1 and $2^{192}/2$).

## 7.7 CRYPTOGRAPHIC FUNCTION DEFINITIONS

In addition to computing the Elliptic Curve Diffie Hellman key, the Numeric Comparison, Out-of-Band and Passkey Entry protocols require four cryptographic functions. These functions are known as *f1*, *g*, *f2* and *f3*.

*f1* is used to generate the 128-bit commitment values Ca and Cb

*g* is used to compute the 6-digit numeric check values

*f2* is used to compute the link key and possible other keys from the DHKey and random nonces

*f3* is used to compute check values Ea and Eb in Authentication Stage 2.

The basic building block for these functions is based on as it already exists in the key derivation function E3 (which is the same as E1 but with a larger input).

### 7.7.1 The Simple Pairing Commitment Function f1

The commitments are computed with function f1. The definition of the Simple Pairing commitment function makes use of the MAC function HMAC based on $SHA\text{-}256_x$, which is denoted as $HMAC\text{-}SHA\text{-}256_X$ with 128-bit key X.

The inputs to the Simple Pairing function f1 are:

U is 192 bits

V is 192 bits

X is 128 bits

Z is 8 bits

Z is zero (i.e. 8 bits of zeros) for Numeric Comparison and OOB protocol. In the Passkey protocol the most significant bit of Z is set equal to one and the least significant bit is made up from one bit of the passkey e.g. if passkey is '1' then Z = 0x81 and if the passkey is '0' then Z = 0x80.

The output of the Simple Pairing f1 function is:

$$f1(U, V, X, Z) = HMAC\text{-}SHA\text{-}256_X (U \,||\, V \,||\, Z) / 2^{128}$$

The inputs to *f1* are different depending on the different protocols.

| Numeric Comparison | Out-Of-Band | Passkey Entry |
|---|---|---|
| Ca = f1(PKax, PKbx, Na, 0) | Ca = f1(PKax, PKax, Ra, 0) | Cai = f1(PKax, PKbx, Nai, rai) |
| Cb = f1(PKbx, PKax, Nb, 0) | Cb = f1(PKbx, PKbx, Rb, 0) | Cbi = f1(PKbx, PKax, Nbi, rbi) |

*Table 7.2: Inputs to f1 for the different protocols*

where PKax denotes the x-coordinate of the public key PKa of A. Similarly, PKbx denotes the x-coordinate of the public key PKb of B. Nai is the nonce value of $i^{th}$ round. For each round Nai value is a new 128 bit number. Similarly, rai is one bit value of the passkey expanded to 8 bits (either 0x80 or 0x81).

### 7.7.2 The Simple Pairing Numeric Verification Function *g*

The Simple Pairing *g* function is defined as follows:

The inputs to the Simple Pairing function *g* are:

U is 192 bits

V is 192 bits

X is 128 bits

Y is 128 bits

The output of the Simple Pairing g function is:

$$g(U, V, X, Y) = \text{SHA-256}(U \,||\, V \,||\, X \,||\, Y) \bmod 2^{32}$$

The numeric verification value is taken as six least significant digits of the 32-bit integer *g*(PKax, PKbx, Na, Nb) where PKax denotes the x-coordinate of the public key PKa of A and PKbx denotes the x-coordinate of the public key PKb of B.

Output of SHA-256 is truncated to 32 bits by taking the least significant 32 bits of the output of SHA-256. This value is converted to decimal numeric value. The checksum used for numeric comparison is the least significant six digits.

$$\text{Compare Value} = g(U, V, X, Y) \bmod 10^6$$

For example, if output = 0x 01 2e b7 2a then decimal value = 19838762 and the checksum used for numeric comparison is 838762.

### 7.7.3 The Simple Pairing Key Derivation Function f2

The definition of the Simple Pairing key derivation function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-$256_W$ with 192-bit key W.

The inputs to the Simple Pairing function *f2* are:



LifeScan, Inc. Ex. 1019
Page 1069 of 1420
IPR2025-00103

W is 192 bits

$N_1$ is 128 bits

$N_2$ is 128 bits

keyID is 32 bits

$A_1$ is 48 bits

$A_2$ is 48 bits

The string "btlk" is mapped into keyID using extended ASCII[1] as follows:

keyID[0] = `0110 1011 (lsb)`
keyID[1] = `0110 1100`
keyID[2] = `0111 0100`
keyID[3] = `0110 0010`

KeyID = `0x62746c6b`

The output of the Simple Pairing f2 function is:

f2(W, N1, N2, KeyID, A1, A2) = HMAC-SHA-256W (N1 || N2|| KeyID || A1|| A2) / $2^{128}$

The output of *f2* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256.

The link key is then calculated as:

LK = f2(DHKey, N_master, N_slave, "btlk", BD_ADDR_master, BD_ADDR_slave)

### 7.7.4 The Simple Pairing Check Function f3

The definition of the Simple Pairing f3 check function makes use of the MAC function HMAC based on SHA-256, which is denoted as HMAC-SHA-$256_W$ with 192-bit key W.

The inputs to the Simple Pairing function f3 are

W is 192 bits

$N_1$ is 128 bits

$N_2$ is 128 bits

R is 128 bits

IOcap is 24 bits

$A_1$ is 48 bits

1. http://en.wikipedia.org/wiki/ISO_8859-1



LifeScan, Inc. Ex. 1019
Page 1070 of 1420
IPR2025-00103

$A_2$ is 48 bits

IOcap is three octets with the most significant octet as the Authentication Requirements parameter, the middle octet as the LMP Out-of-Band Authentication Data parameter, and the least significant octet as the LMP IO capability parameter.

The output of the Simple Pairing *f3* function is:

$$f3(W, N_1, N_2, R, IOcap, A_1, A_2) = HMAC\text{-}SHA\text{-}256W\ (N_1 \,||\, N_2 \,||\, R \,||\, IOcap \,||\, A_1 \,||\, A_2) / 2^{128}$$

The output of *f3* is taken as the 128 most significant (leftmost) bits of the output of HMAC-SHA-256. The check values are computed with function f3. The inputs to f3 are different depending on the different protocols:

| Numeric Comparison | Out-Of-Band | Passkey Entry |
|---|---|---|
| Ea = *f3*(DHKey, Na, Nb, 0, IOcapA, A, B) | Ea = *f3*(DHKey, Na, Nb, rb, IOcapA, A, B) | Ea = *f3*(DHKey, Na20, Nb20, rb, IOcapA, A, B) |
| Eb = *f3*(DHKey, Nb, Na, 0, IOcapB, B, A) | Eb = *f3*(DHKey, Nb, Na, ra, IOcapB, B, A) | Eb = *f3*(DHKey, Nb20, Na20, ra, IOcapB, B, A) |

*Table 7.3: Inputs to f3 for the different protocols*

DHKey denotes the shared secret Diffie-Hellman Key computed as P192(SKa,PKb) by A and as P192(SKb,PKa) by B. Adata denotes the capability data of A and Bdata denotes the capability data of B. In Passkey Entry, the data ra and rb are 6-digit passkey values which are expressed as a 128-bit integer. For instance,

if the 6-digit value of ra is 131313

then R = 0x 00 00 00 00 00 00 00 00 00 00 00 00 00 02 00 f1.

The input A is the BD_ADDR of device A and the input B is the BD_ADDR of device B.



LifeScan, Inc. Ex. 1019
Page 1071 of 1420
IPR2025-00103

# 8 LIST OF FIGURES

Figure 3.1: Generation of unit key. When the unit key has been exchanged, the initialization key is discarded in both devices. ..... 863
Figure 3.2: Generating a combination key. The old link key (K) is discarded after the exchange of a new combination key has succeeded 864
Figure 3.3: Master link key distribution and computation of the corresponding encryption key. ..... 867
Figure 4.1: Stream ciphering for Bluetooth with E0. ..... 868
Figure 4.2: Functional description of the encryption procedure ..... 871
Figure 4.3: Concept of the encryption engine. ..... 872
Figure 4.4: Overview of the operation of the encryption engine. Between each start of a packet (TX or RX), the LFSRs are re-initialized. ..... 873
Figure 4.5: Arranging the input to the LFSRs. ..... 876
Figure 4.6: Distribution of the 128 last generated output symbols within the LFSRs. ..... 877
Figure 5.1: Challenge-response for the Bluetooth. ..... 878
Figure 5.2: Challenge-response for symmetric key systems. ..... 879
Figure 6.1: Flow of data for the computation of . ..... 882
Figure 6.2: One round in and . The permutation boxes 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. ..... 884
Figure 6.3: Key scheduling in . ..... 885
Figure 6.4: Key generating algorithm and its two modes. Mode 1 is used for unit and combination keys, while mode 2 is used for and . ... 887
Figure 6.5: Generation of the encryption key ..... 887
Figure 7.1: Simple Pairing Security Phases ..... 888
Figure 7.2: Public Key Exchange Details ..... 890
Figure 7.3: Authentication Stage 1 (High Level) ..... 890
Figure 7.4: Authentication Stage 1: Numeric Comparison Protocol Details ..... 891
Figure 7.5: Authentication Stage 1: Out of Band Details ..... 893
Figure 7.6: Authentication Stage 1: Passkey Entry Details ..... 895
Figure 7.7: Authentication Stage 2 ..... 896
Figure 7.8: Link Key Calculation ..... 897



LifeScan, Inc. Ex. 1019
Page 1072 of 1420
IPR2025-00103

# 9 LIST OF TABLES

Table 1.1: Entities used in authentication and encryption procedures.....855
Table 4.1: Possible encryption modes for a slave in possession of a master key.....870
Table 4.2: The four primitive feedback polynomials.....872
Table 4.3: The mappings *T1* and *T2*. .....873
Table 4.4: Polynomials used when creating *K'c*. . .....875
Table 7.1: Terminology.....888
Table 7.2: Inputs to f1 for the different protocols.....899
Table 7.3: Inputs to f3 for the different protocols.....901



LifeScan, Inc. Ex. 1019
Page 1073 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1074 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1075 of 1420
IPR2025-00103





LifeScan, Inc. Ex. 1019
Page 1076 of 1420
IPR2025-00103

Specification Volume 3

# SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Wireless Connections Made Easy



# Core System Package

## [Host volume]

Covered Core Package version:
2.1 + EDR

Bluetooth®

1077

LifeScan, Inc. Ex. 1019
Page 1077 of 1420
IPR2025-00103

## Revision History

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

## Contributors

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

## Web Site

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

## Disclaimer and Copyright Notice

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

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

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.



LifeScan, Inc. Ex. 1019
Page 1078 of 1420
IPR2025-00103

THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

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

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

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

Copyright © 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007

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

*Third-party brands and names are the property of their respective owners.

LifeScan, Inc. Ex. 1019
Page 1080 of 1420
IPR2025-00103

# TABLE OF CONTENTS

**Part A**
**LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION**

**1 Introduction ..... 19**
1.1 L2CAP Features ..... 19
1.2 Assumptions ..... 23
1.3 Scope ..... 23
1.4 Terminology ..... 24

**2 General Operation ..... 27**
2.1 Channel Identifiers ..... 27
2.2 Operation Between Devices ..... 27
2.3 Operation Between Layers ..... 29
2.4 Modes of Operation ..... 29

**3 Data Packet Format ..... 31**
3.1 Connection-oriented Channel in Basic L2CAP Mode ..... 31
3.2 Connectionless Data Channel in Basic L2CAP Mode ..... 32
3.3 Connection-oriented Channel in Retransmission/Flow Control Modes 33
3.3.1 L2CAP header fields ..... 33
3.3.2 Control field (2 octets) ..... 34
3.3.3 L2CAP SDU length field (2 octets) ..... 36
3.3.4 Information payload field (0 to 65531 octets) ..... 36
3.3.5 Frame check sequence (2 octets) ..... 37
3.3.6 Invalid frame detection ..... 38

**4 Signaling Packet Formats ..... 39**
4.1 Command Reject (code 0x01) ..... 41
4.2 Connection Request (code 0x02) ..... 42
4.3 Connection Response (code 0x03) ..... 44
4.4 Configuration Request (code 0x04) ..... 45
4.5 Configuration Response (code 0x05) ..... 48
4.6 Disconnection Request (code 0x06) ..... 50
4.7 Disconnection Response (code 0x07) ..... 51
4.8 Echo Request (code 0x08) ..... 51
4.9 Echo Response (code 0x09) ..... 52
4.10 Information Request (code 0x0A) ..... 52
4.11 Information Response (code 0x0B) ..... 53
4.12 Extended Feature Mask ..... 54

**5 Configuration Parameter Options ..... 55**



LifeScan, Inc. Ex. 1019
Page 1081 of 1420
IPR2025-00103

5.1 Maximum Transmission Unit (MTU) ..... 55
5.2 Flush Timeout Option ..... 57
5.3 Quality of Service (QoS) Option ..... 58
5.4 Retransmission and Flow Control Option ..... 62

**6 State Machine ..... 65**
6.1 General rules for the state machine: ..... 65
6.1.1 CLOSED state ..... 66
6.1.2 WAIT_CONNECT_RSP state ..... 67
6.1.3 WAIT_CONNECT state ..... 67
6.1.4 CONFIG state ..... 68
6.1.5 OPEN state ..... 71
6.1.6 WAIT_DISCONNECT state ..... 71
6.2 Timers events ..... 73
6.2.1 RTX ..... 73
6.2.2 ERTX ..... 74

**7 General Procedures ..... 77**
7.1 Configuration Process ..... 77
7.1.1 Request path ..... 78
7.1.2 Response path ..... 78
7.2 Fragmentation and Recombination ..... 79
7.2.1 Fragmentation of L2CAP PDUs ..... 79
7.2.2 Recombination of L2CAP PDUs ..... 80
7.3 Encapsulation of SDUs ..... 81
7.3.1 Segmentation of L2CAP SDUs ..... 81
7.3.2 Reassembly of L2CAP SDUs ..... 82
7.3.3 Segmentation and fragmentation ..... 82
7.4 Delivery of Erroneous L2CAP SDUs ..... 83
7.5 Operation with Flushing ..... 83
7.6 Connectionless Data Channel ..... 84

**8 Procedures for Flow Control and Retransmission ..... 85**
8.1 Information Retrieval ..... 85
8.2 Function of PDU Types for Flow Control and Retransmission ..... 85
8.2.1 Information frame (I-frame) ..... 85
8.2.2 Supervisory Frame (S-frame) ..... 85
8.3 Variables and Sequence Numbers ..... 87
8.3.1 Sending peer ..... 87
8.3.2 Receiving peer ..... 89
8.4 Retransmission Mode ..... 91
8.4.1 Transmitting frames ..... 91



LifeScan, Inc. Ex. 1019
Page 1082 of 1420
IPR2025-00103

8.4.2 Receiving I-frames ....93
8.4.3 I-frames pulled by the SDU reassembly function ....93
8.4.4 Sending and receiving acknowledgements ....94
8.4.5 Receiving REJ frames ....95
8.4.6 Waiting acknowledgements ....95
8.4.7 Exception conditions ....95
8.5 Flow Control Mode ....97
8.5.1 Transmitting I-frames ....97
8.5.2 Receiving I-frames ....98
8.5.3 I-frames pulled by the SDU reassembly function ....98
8.5.4 Sending and receiving acknowledgements ....98
8.5.5 Waiting acknowledgements ....99
8.5.6 Exception conditions ....99
**9 List of Figures ....101**
**10 List of Tables ....103**

**Part B**
**SERVICE DISCOVERY PROTOCOL (SDP)**

**1 Introduction ....113**
1.1 General Description ....113
1.2 Motivation ....113
1.3 Requirements ....113
1.4 Non-requirements and Deferred Requirements ....114
1.5 Conventions ....114
1.5.1 Bit And Byte Ordering Conventions ....114
**2 Overview ....115**
2.1 SDP Client-Server Interaction ....115
2.2 Service Record ....116
2.3 Service Attribute ....117
2.3.1 Attribute ID ....118
2.3.2 Attribute Value ....119
2.4 Service Class ....119
2.4.1 A Printer Service Class Example ....119
2.5 Searching for Services ....121
2.5.1 UUID ....121
2.5.2 Service Search Patterns ....122
2.6 Browsing for Services ....122
2.6.1 Example Service Browsing Hierarchy ....123



LifeScan, Inc. Ex. 1019
Page 1083 of 1420
IPR2025-00103

**3 Data Representation ..... 125**
3.1 Data Element ..... 125
3.2 Data Element Type Descriptor ..... 125
3.3 Data Element Size Descriptor ..... 126
3.4 Data Element Examples ..... 127
**4 Protocol Description ..... 128**
4.1 Transfer Byte Order ..... 128
4.2 Protocol Data Unit Format ..... 128
4.3 Partial Responses and Continuation State ..... 130
4.4 Error Handling ..... 130
4.4.1 SDP_ErrorResponse PDU ..... 131
4.5 ServiceSearch Transaction ..... 131
4.5.1 SDP_ServiceSearchRequest PDU ..... 131
4.5.2 SDP_ServiceSearchResponse PDU ..... 133
4.6 ServiceAttribute Transaction ..... 135
4.6.1 SDP_ServiceAttributeRequest PDU ..... 135
4.6.2 SDP_ServiceAttributeResponse PDU ..... 137
4.7 ServiceSearchAttribute Transaction ..... 138
4.7.1 SDP_ServiceSearchAttributeRequest PDU ..... 138
4.7.2 SDP_ServiceSearchAttributeResponse PDU ..... 140
**5 Service Attribute Definitions ..... 142**
5.1 Universal Attribute Definitions ..... 142
5.1.1 ServiceRecordHandle Attribute ..... 142
5.1.2 ServiceClassIDList Attribute ..... 143
5.1.3 ServiceRecordState Attribute ..... 143
5.1.4 ServiceID Attribute ..... 143
5.1.5 ProtocolDescriptorList Attribute ..... 144
5.1.6 AdditionalProtocolDescriptorList Attribute ..... 145
5.1.7 BrowseGroupList Attribute ..... 146
5.1.8 LanguageBaseAttributeIDList Attribute ..... 146
5.1.9 ServiceInfoTimeToLive Attribute ..... 147
5.1.10 ServiceAvailability Attribute ..... 148
5.1.11 BluetoothProfileDescriptorList Attribute ..... 148
5.1.12 DocumentationURL Attribute ..... 149
5.1.13 ClientExecutableURL Attribute ..... 149
5.1.14 IconURL Attribute ..... 150
5.1.15 ServiceName Attribute ..... 150
5.1.16 ServiceDescription Attribute ..... 151
5.1.17 ProviderName Attribute ..... 151



LifeScan, Inc. Ex. 1019
Page 1084 of 1420
IPR2025-00103

5.1.18 Reserved Universal Attribute IDs ....151
5.2 ServiceDiscoveryServer Service Class Attribute Definitions....152
5.2.1 ServiceRecordHandle Attribute ....152
5.2.2 ServiceClassIDList Attribute ....152
5.2.3 VersionNumberList Attribute ....152
5.2.4 ServiceDatabaseState Attribute ....153
5.2.5 Reserved Attribute IDs ....153
5.3 BrowseGroupDescriptor Service Class Attribute Definitions....153
5.3.1 ServiceClassIDList Attribute ....153
5.3.2 GroupID Attribute ....154
5.3.3 Reserved Attribute IDs ....154
**6 Security ....155**
**7 Appendix ....156**

**Part C**
**GENERIC ACCESS PROFILE**

**1 Introduction ....176**
1.1 Scope ....176
1.2 Symbols and conventions ....176
1.2.1 Requirement status symbols ....176
1.2.2 Signaling diagram conventions ....177
1.2.3 Notation for timers and counters ....177
**2 Profile overview ....178**
2.1 Profile stack ....178
2.2 Configurations and roles ....178
2.3 User requirements and scenarios ....179
2.4 Profile fundamentals ....180
2.5 Conformance ....180
**3 User interface aspects ....181**
3.1 The user interface level ....181
3.2 Representation of Bluetooth parameters ....181
3.2.1 Bluetooth device address (BD_ADDR) ....181
3.2.2 Bluetooth device name (the user-friendly name) ....181
3.2.3 Bluetooth passkey (Bluetooth PIN) ....182
3.2.4 Class of Device ....183
3.3 Pairing ....184
**4 Modes ....185**
4.1 Discoverability modes ....185
4.1.1 Non-discoverable mode ....186



LifeScan, Inc. Ex. 1019
Page 1085 of 1420
IPR2025-00103

4.1.2 Limited discoverable mode ..... 186
4.1.3 General discoverable mode ..... 187
4.2 Connectability modes ..... 189
4.2.1 Non-connectable mode ..... 189
4.2.2 Connectable mode ..... 189
4.3 Bonding modes ..... 191
4.3.1 Non-bondable mode ..... 191
4.3.2 Bondable mode ..... 191
**5 Security Aspects ..... 193**
5.1 Authentication ..... 193
5.1.1 Purpose ..... 193
5.1.2 Term on UI level ..... 193
5.1.3 Procedure ..... 194
5.1.4 Conditions ..... 194
5.2 Security modes ..... 194
5.2.1 Legacy Security Modes ..... 196
5.2.2 Security mode 4 (service level enforced security) ..... 198
**6 Idle mode procedures ..... 213**
6.1 General inquiry ..... 213
6.1.1 Purpose ..... 213
6.1.2 Term on UI level ..... 213
6.1.3 Description ..... 214
6.1.4 Conditions ..... 214
6.2 Limited inquiry ..... 214
6.2.1 Purpose ..... 214
6.2.2 Term on UI level ..... 215
6.2.3 Description ..... 215
6.2.4 Conditions ..... 215
6.3 Name discovery ..... 216
6.3.1 Purpose ..... 216
6.3.2 Term on UI level ..... 216
6.3.3 Description ..... 217
6.3.4 Conditions ..... 218
6.4 Device discovery ..... 218
6.4.1 Purpose ..... 218
6.4.2 Term on UI level ..... 218
6.4.3 Description ..... 219
6.4.4 Conditions ..... 219
6.5 Bonding ..... 219



LifeScan, Inc. Ex. 1019
Page 1086 of 1420
IPR2025-00103

6.5.1 Purpose ....219
6.5.2 Term on UI level ....219
6.5.3 Description ....220
6.5.4 Conditions ....221
**7 Establishment procedures ....223**
7.1 Link establishment ....223
7.1.1 Purpose ....223
7.1.2 Term on UI level ....223
7.1.3 Description ....224
7.1.4 Conditions ....225
7.2 Channel establishment....226
7.2.1 Purpose ....226
7.2.2 Term on UI level ....226
7.2.3 Description ....226
7.2.4 Conditions ....227
7.3 Connection establishment....228
7.3.1 Purpose ....228
7.3.2 Term on UI level ....228
7.3.3 Description ....228
7.3.4 Conditions ....229
7.4 Establishment of additional connection....229
**8 Extended Inquiry Response Data Format ....230**
8.1 EIR Data Type Definitions ....231
8.1.1 Service Class UUIDs....231
8.1.2 Local Name ....232
8.1.3 Flags....232
8.1.4 Manufacturer Specific Data ....233
8.1.5 TX Power Level....233
8.1.6 Secure Simple Pairing Out of Band (OOB) ....233
8.2 Example Extended Inquiry Response....234
**9 Definitions....236**
9.1 General definitions ....236
9.2 Connection-related definitions....236
9.3 Device-related definitions....237
9.4 Procedure-related definitions ....238
9.5 Security-related definitions....238
**10 Appendix A (Normative): Timers and constants ....240**
**11 Appendix B (Informative): Information Flows of Related Procedures ....241**



LifeScan, Inc. Ex. 1019
Page 1087 of 1420
IPR2025-00103

11.1 LMP-authentication ..... 241
11.2 LMP-pairing ..... 242
11.3 Service discovery ..... 243
**12 References ..... 244**

**Part D**
**TEST SUPPORT**

**1 Test Methodology ..... 249**
1.1 Test Scenarios ..... 249
1.1.1 Test setup ..... 249
1.1.2 Transmitter Test ..... 250
1.1.3 LoopBack test ..... 255
1.1.4 Pause test ..... 258
1.2 References ..... 258
**2 Test Control Interface (TCI) ..... 259**
2.1 Introduction ..... 259
2.1.1 Terms used ..... 259
2.1.2 Usage of the interface ..... 259
2.2 TCI Configurations ..... 260
2.2.1 Bluetooth RF requirements ..... 260
2.2.2 Bluetooth protocol requirements ..... 261
2.2.3 Bluetooth profile requirements ..... 262
2.3 TCI Configuration and Usage ..... 263
2.3.1 Transport layers ..... 263
2.3.2 Baseband and link manager qualification ..... 264
2.3.3 HCI qualification ..... 266



LifeScan, Inc. Ex. 1019
Page 1088 of 1420
IPR2025-00103

# LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION

***The Bluetooth logical link control and adaptation protocol (L2CAP) supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. The protocol state machine, packet format, and composition are described in this document.***



LifeScan, Inc. Ex. 1019
Page 1089 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1090 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 19**
1.1 L2CAP Features ..... 19
1.2 Assumptions ..... 23
1.3 Scope ..... 23
1.4 Terminology ..... 24
**2 General Operation ..... 27**
2.1 Channel Identifiers ..... 27
2.2 Operation Between Devices ..... 27
2.3 Operation Between Layers ..... 29
2.4 Modes of Operation ..... 29
**3 Data Packet Format ..... 31**
3.1 Connection-oriented Channel in Basic L2CAP Mode ..... 31
3.2 Connectionless Data Channel in Basic L2CAP Mode ..... 32
3.3 Connection-oriented Channel in Retransmission/Flow Control Modes ..... 33
3.3.1 L2CAP header fields ..... 33
3.3.2 Control field (2 octets) ..... 34
3.3.3 L2CAP SDU length field (2 octets) ..... 34
3.3.4 Information payload field (0 to 65531 octets) ..... 36
3.3.5 Frame check sequence (2 octets) ..... 37
3.3.6 Invalid frame detection ..... 38
**4 Signaling Packet Formats ..... 39**
4.1 Command Reject (code 0x01) ..... 41
4.2 Connection Request (code 0x02) ..... 43
4.3 Connection Response (code 0x03) ..... 44
4.4 Configuration Request (code 0x04) ..... 45
4.5 Configuration Response (code 0x05) ..... 48
4.6 Disconnection Request (code 0x06) ..... 50
4.7 Disconnection Response (code 0x07) ..... 51
4.8 Echo Request (code 0x08) ..... 51
4.9 Echo Response (code 0x09) ..... 52
4.10 Information Request (code 0x0A) ..... 52
4.11 Information Response (code 0x0B) ..... 53
4.12 Extended Feature Mask ..... 54
**5 Configuration Parameter Options ..... 55**
5.1 Maximum Transmission Unit (MTU) ..... 55
5.2 Flush Timeout Option ..... 57
5.3 Quality of Service (QoS) Option ..... 58



LifeScan, Inc. Ex. 1019
Page 1091 of 1420
IPR2025-00103

5.4 Retransmission and Flow Control Option ..... 62

**6 State Machine ..... 65**

6.1 General rules for the state machine: ..... 65

6.1.1 CLOSED state ..... 66

6.1.2 WAIT_CONNECT_RSP state ..... 67

6.1.3 WAIT_CONNECT state ..... 67

6.1.4 CONFIG state ..... 68

6.1.5 OPEN state ..... 71

6.1.6 WAIT_DISCONNECT state ..... 71

6.2 Timers events ..... 73

6.2.1 RTX ..... 73

6.2.2 ERTX ..... 74

**7 General Procedures ..... 77**

7.1 Configuration Process ..... 77

7.1.1 Request path ..... 78

7.1.2 Response path ..... 78

7.2 Fragmentation and Recombination ..... 79

7.2.1 Fragmentation of L2CAP PDUs ..... 79

7.2.2 Recombination of L2CAP PDUs ..... 80

7.3 Encapsulation of SDUs ..... 81

7.3.1 Segmentation of L2CAP SDUs ..... 81

7.3.2 Reassembly of L2CAP SDUs ..... 82

7.3.3 Segmentation and fragmentation ..... 82

7.4 Delivery of Erroneous L2CAP SDUs ..... 83

7.5 Operation with Flushing ..... 83

7.6 Connectionless Data Channel ..... 84

**8 Procedures for Flow Control and Retransmission ..... 85**

8.1 Information Retrieval ..... 85

8.2 Function of PDU Types for Flow Control and Retransmission ..... 85

8.2.1 Information frame (I-frame) ..... 85

8.2.2 Supervisory Frame (S-frame) ..... 85

8.3 Variables and Sequence Numbers ..... 87

8.3.1 Sending peer ..... 87

8.3.2 Receiving peer ..... 89

8.4 Retransmission Mode ..... 91

8.4.1 Transmitting frames ..... 91

8.4.2 Receiving I-frames ..... 93

8.4.3 I-frames pulled by the SDU reassembly function ..... 93

8.4.4 Sending and receiving acknowledgements ..... 94



LifeScan, Inc. Ex. 1019
Page 1092 of 1420
IPR2025-00103

8.4.5 Receiving REJ frames....95
8.4.6 Waiting acknowledgements....95
8.4.7 Exception conditions....95
8.5 Flow Control Mode....97
8.5.1 Transmitting I-frames....97
8.5.2 Receiving I-frames....98
8.5.3 I-frames pulled by the SDU reassembly function....98
8.5.4 Sending and receiving acknowledgements....98
8.5.5 Waiting acknowledgements....99
8.5.6 Exception conditions....99
**9 List of Figures....101**
**10 List of Tables....103**



LifeScan, Inc. Ex. 1019
Page 1093 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1094 of 1420
IPR2025-00103

# 1 INTRODUCTION

This section of the Bluetooth Specification defines the Logical Link Control and Adaptation Layer Protocol, referred to as L2CAP. L2CAP is layered over the Link Controller Protocol and resides in the data link layer as shown in Figure 1.1. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive upper layer data packets (L2CAP Service Data Units, SDU) up to 64 kilobytes in length. L2CAP also permits per-channel flow control and retransmission via the Flow Control and Retransmission Modes.

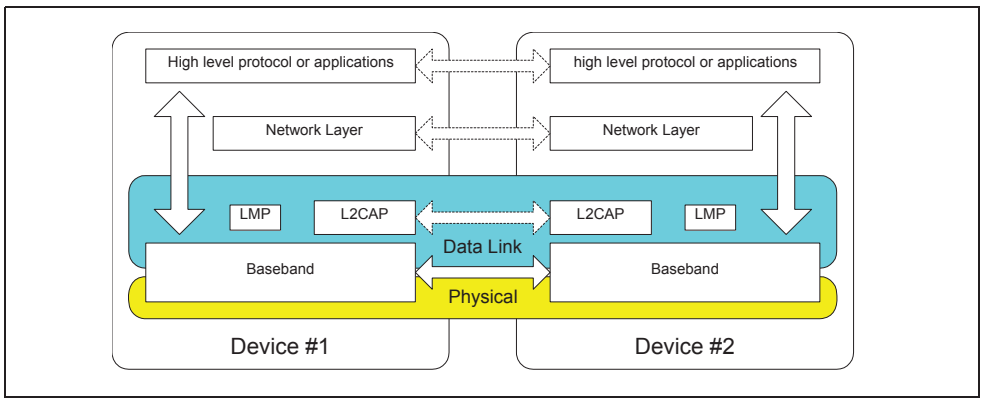



*Figure 1.1: L2CAP within protocol layers*

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

## 1.1 L2CAP FEATURES

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

Figure 1.2 on page 20 breaks down L2CAP into its architectural components. The Channel Manager provides the control plane functionality and is responsible for all internal signaling, L2CAP peer-to-peer signaling and signaling with higher and lower layers. It performs the state machine functionality described in Section 6 on page 65 and uses message formats described in Section 4 on page 39, Section 5 on page 55. The Retransmission and Flow Control block provides per-channel flow control and optional retransmission for applications that require it. The Resource Manager is responsible for providing a frame relay service to the Channel Manager, the Retransmission and Flow Control



LifeScan, Inc. Ex. 1019
Page 1095 of 1420
IPR2025-00103

block and those application data streams that do not require Retransmission and Flow Control services. It is responsible for coordinating the transmission and reception of packets related to multiple L2CAP channels over the facilities offered at the lower layer interface.

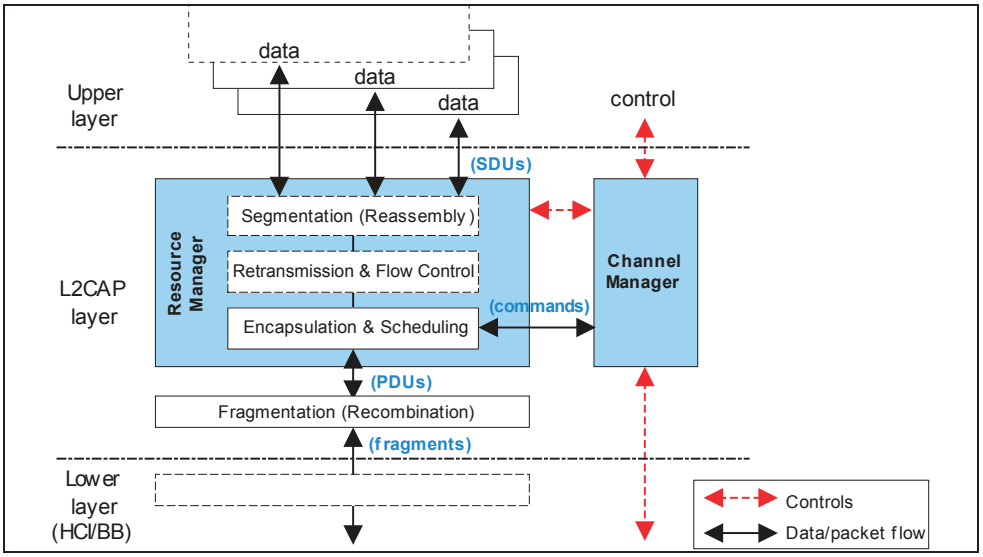



*Figure 1.2: L2CAP architectural blocks*



LifeScan, Inc. Ex. 1019
Page 1096 of 1420
IPR2025-00103

- *Protocol/channel multiplexing*

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

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

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

- *Segmentation and reassembly*

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

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

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

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

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

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

- *Flow control per L2CAP channel*

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

- *Error control and retransmissions*

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



LifeScan, Inc. Ex. 1019
Page 1097 of 1420
IPR2025-00103

- *Fragmentation and Recombination*

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

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

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

- *Quality of Service*

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

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



LifeScan, Inc. Ex. 1019
Page 1098 of 1420
IPR2025-00103

## 1.2 ASSUMPTIONS

The protocol is designed based on the following assumptions:

1. The ACL logical transport and L2CAP logical link between two devices is set up using the Link Manager Protocol. The baseband provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than 1 unicast ACL logical transport exists between any two devices.
2. The baseband always provides the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bi-directional. Multicasts and unidirectional traffic (e.g., video) do not require duplex channels.
3. The L2CAP layer provides a channel with a degree of reliability based on the mechanisms available at the baseband layer and with optional additional packet segmentation and error detection that can be enabled in the enhanced L2CAP layer. The baseband performs data integrity checks and resends data until it has been successfully acknowledged or a timeout occurs. Because acknowledgements may be lost, timeouts may occur even after the data has been successfully sent. The link controller protocol uses a 1-bit sequence number. Note that the use of baseband broadcast packets is prohibited if reliability is required, since all broadcasts start the first segment of an L2CAP packet with the same sequence bit.
4. Some applications will expect independent flow control, independence from the effects of other traffic and, in some cases, better error control than the baseband provides. The Flow and Error Control block provides two modes. Retransmission Mode offers segmentation, flow control and L2CAP PDU retransmissions. Flow control mode offers just the segmentation and flow control functions. If Basic L2CAP mode is chosen, the Flow and Error Control block is not used.

## 1.3 SCOPE

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

- L2CAP does not transport audio or transparent synchronous data designated for SCO or eSCO logical transports.
- L2CAP does not support a reliable multicast channel. See Section 3.2 on page 32.
- L2CAP does not support the concept of a global group name.



LifeScan, Inc. Ex. 1019
Page 1099 of 1420
IPR2025-00103

## 1.4 TERMINOLOGY

The following formal definitions apply:

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

*Table 1.1: Terminology*



LifeScan, Inc. Ex. 1019
Page 1100 of 1420
IPR2025-00103

| Term | Description |
|---|---|
| Basic L2CAP header | Minimum L2CAP protocol 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 B-frame is a PDU used in the Basic L2CAP mode for L2CAP data packets. It contains a complete SDU as its payload, encapsulated by a basic L2CAP header. |
| Information frame (I-frame) | An I-frame is 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) | An S-frame is 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 C-frame is a PDU that contains L2CAP signaling messages exchanged between the peer L2CAP entities. C-frames are exclusively used on the L2CAP signaling channel. |
| Group frame (G-frame) | G-frame is a PDU exclusively used on Connectionless L2CAP channels in the Basic L2CAP mode. It contains a complete SDU as its payload, encapsulated by a specific header. |
| 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.<br>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 accommodate the L2CAP PDU transport to HCI data packet or baseband packet sizes.<br>Fragmentation of PDUs may be applied in all L2CAP modes.<br>Note: in version 1.1, Fragmentation and Recombination was referred to as “Segmentation and Reassembly“. |
| Recombination | The reverse procedure corresponding to fragmentation, resulting in an L2CAP PDU re-established from fragments. In the receive path, full or partial recombination operations may occur in the Controller and/or the Host, and the location of recombination does not necessarily correspond to where fragmentations occurs on the transmit side. |
| 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. |

*Table 1.1: Terminology*



LifeScan, Inc. Ex. 1019
Page 1101 of 1420
IPR2025-00103

| 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.<br>Note: in the absence of segmentation, or in the Basic L2CAP Mode, the Maximum Transmission Unit is the equivalent to the Maximum PDU payload Size and shall be made equal in the configuration parameters. |
| Signaling MTU ($MTU_{sig}$) | The maximum size of command information that the L2CAP layer entity is capable of accepting. The $MTU_{sig}$, refers to the signaling 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 1 transmission permitted).<br>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. |

*Table 1.1: Terminology*



LifeScan, Inc. Ex. 1019
Page 1102 of 1420
IPR2025-00103

# 2 GENERAL OPERATION

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

## 2.1 CHANNEL IDENTIFIERS

A channel identifier (CID) is the local name representing a logical channel endpoint on the device. The null identifier (0x0000) is an illegal identifier and shall never be used as a destination endpoint. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that two simultaneously active L2CAP channels shall not share the same CID. Table 2.1 on page 27 summarizes the definition and partitioning of the CID name space.

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

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

*Table 2.1: CID name space*

## 2.2 OPERATION BETWEEN DEVICES

Figure 2.1 on page 28 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID identifies each endpoint of the channel. The connectionless channels restrict data flow to a single direction. These channels are used to support a channel 'group' where the CID on the source represents one or more remote devices. There are also a number of CIDs reserved for special purposes. The signaling channel is one example of a reserved channel. This channel is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of connection oriented and connectionless channels. Support for a signaling channel within an L2CAP entity is mandatory.



LifeScan, Inc. Ex. 1019
Page 1103 of 1420
IPR2025-00103

Note: it is assumed that an L2CAP signaling channel is available immediately when an ACL logical transport is established between two devices, and L2CAP traffic is enabled on the L2CAP logical link.
Another CID is reserved for all incoming connectionless data traffic. In the example below, a CID is used to represent a group consisting of device #3 and #4. Traffic sent from this channel ID is directed to the remote channel reserved for connectionless data traffic.

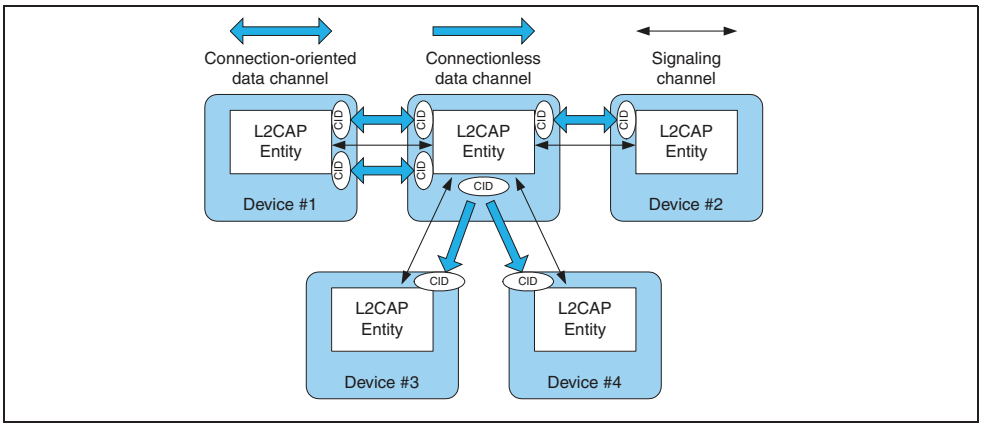



*Figure 2.1: Channels between devices*

Table 2.2 on page 28 describes the various channels and their source and destination identifiers. A CID is allocated to identify the local endpoint and shall be in the range 0x0040 to 0xFFFF. Section 6 on page 65 describes the state machine associated with each connection-oriented channel. Section 3.1 on page 31 describes the packet format associated with bi-directional channels and Section 3.2 on page 32 describes the packet format associated with uni-directional channels.

| Channel Type | Local CID (sending) | Remote CID (receiving) |
|---|---|---|
| Connection-oriented | Dynamically allocated | Dynamically allocated |
| Connectionless data | Dynamically allocated | 0x0002 (fixed) |
| Signaling | 0x0001 (fixed) | 0x0001 (fixed) |

*Table 2.2: Types of Channel Identifiers*



LifeScan, Inc. Ex. 1019
Page 1104 of 1420
IPR2025-00103

## 2.3 OPERATION BETWEEN LAYERS

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

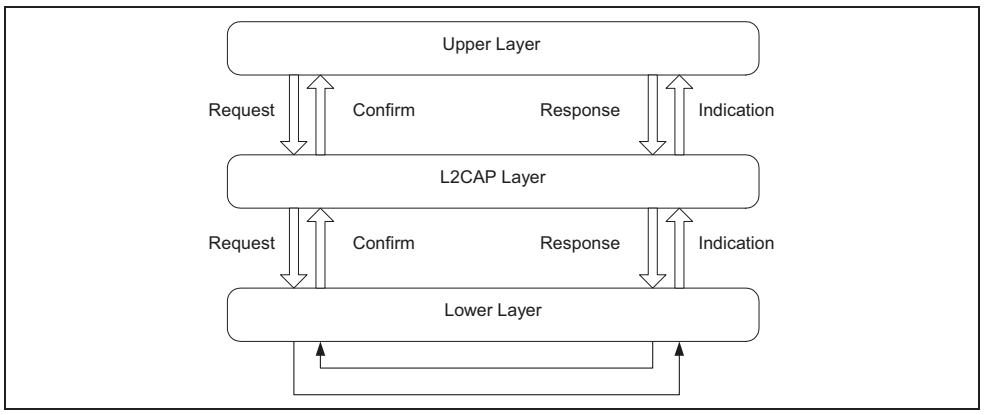



*Figure 2.2: L2CAP transaction model.*

## 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:

- Basic L2CAP Mode (equivalent to L2CAP specification in Bluetooth v1.1) [1]
- 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 sequence numbers in the PDUs are used to control buffering, and a TxWindow size is used to limit the required buffer space and/or provide a method for flow control. In addition to the window size, the Token Bucket size parameter of the flow specification can be used to

1. Specification of the Bluetooth System v1.1 (Feb 22nd 2001): volume 1, part D.



LifeScan, Inc. Ex. 1019
Page 1105 of 1420
IPR2025-00103

dimension the buffers; in particular on channels that do not use flow and error control.

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

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



LifeScan, Inc. Ex. 1019
Page 1106 of 1420
IPR2025-00103

# 3 DATA PACKET FORMAT

L2CAP is packet-based but follows a communication model based on *channels*. A channel represents a data flow between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless.
All packet fields shall use Little Endian byte order.

## 3.1 CONNECTION-ORIENTED CHANNEL IN BASIC L2CAP MODE

Figure 3.1 on page 31 illustrates the format of the L2CAP PDU within a connection-oriented channel. In basic L2CAP mode, the L2CAP PDU on a connection-oriented channel is also referred to as a "B-frame".

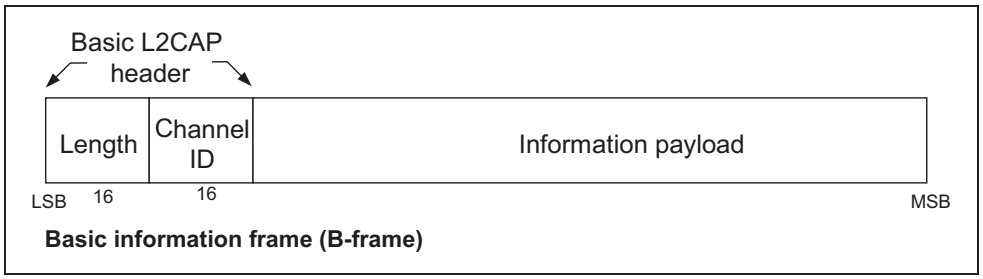



*Figure 3.1: PDU format in Basic L2CAP mode on connection-oriented channels (field sizes in bits)*

The fields shown are:

- *Length: 2 octets (16 bits)*

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

- *Channel ID: 2 octets*

  The channel ID (CID) identifies the destination channel endpoint of the packet.

- *Information payload: 0 to 65535 octets*

  This contains the payload received from the upper layer protocol (outgoing packet), or delivered to the upper layer protocol (incoming packet). The MTU is determined during channel configuration (see Section 5.1 on page 55). The minimum supported MTU for the signaling PDUs ($MTU_{sig}$) is 48 octets (see Section 4 on page 39).



LifeScan, Inc. Ex. 1019
Page 1107 of 1420
IPR2025-00103

## 3.2 CONNECTIONLESS DATA CHANNEL IN BASIC L2CAP MODE

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

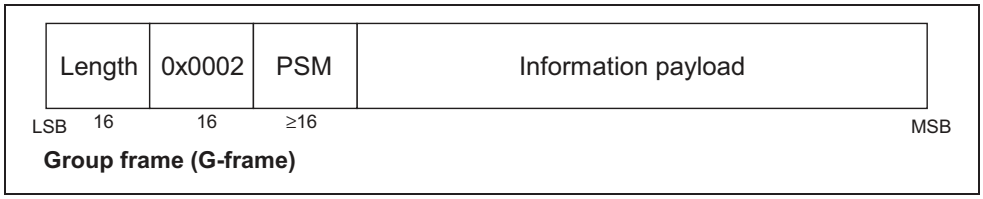



*Figure 3.2: L2CAP PDU format in Basic L2CAP mode on Connectionless channel*

The fields shown are:

- *Length: 2 octets*

  Length indicates the size of information payload plus the PSM field in octets.

- *Channel ID: 2 octets*

  Channel ID (0x0002) reserved for connectionless traffic.

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

  For information on the PSM field see Section 4.2 on page 42.

- *Information payload: 0 to 65533 octets*

  The payload information to be distributed to all members of the piconet. Implementations shall support a connectionless MTU ($MTU_{cnl}$) of 48 octets on connectionless channels. Devices may also explicitly change to a larger or smaller connectionless MTU ($MTU_{cnl}$).

  Note: the maximum size of the Information payload field decreases accordingly if the PSM field is extended beyond the two octet minimum.



LifeScan, Inc. Ex. 1019
Page 1108 of 1420
IPR2025-00103

## 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.

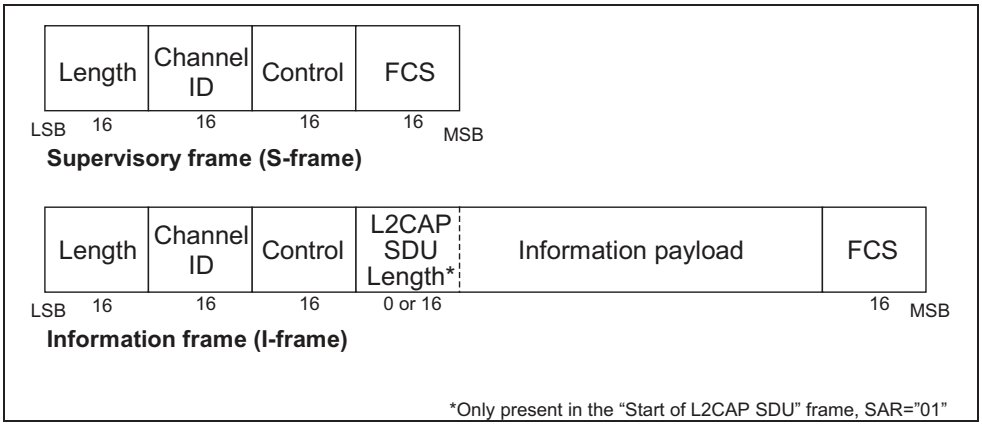



*Figure 3.3: L2CAP PDU formats in Flow Control and Retransmission Modes*

### 3.3.1 L2CAP header fields

- *Length: 2 octets*

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

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

  If the L2CAP SDU length field is present then the maximum number of Information octets in one I-frame is 65529 octets. If the L2CAP SDU length field is not present then the maximum number of Information octets in one I-frame is 65531 octets.

- *Channel ID: 2 octets*

  This field contains the Channel Identification (CID).



LifeScan, Inc. Ex. 1019
Page 1109 of 1420
IPR2025-00103

### 3.3.2 Control field (2 octets)

The Control field identifies the type of frame. The control field will contain sequence numbers where applicable. Its coding is shown in Table 3.1 on page 34. There are two different frame types, Information frame types and Supervisory frame types. Information and Supervisory frames types are distinguished by the rightmost bit in the Control field, as shown in Table 3.1 on page 34.

- *Information frame format (I-frame)*

  The I-frames are used to transfer information between L2CAP entities. Each I-frame has a TxSeq(Send sequence number), ReqSeq(Receive sequence number) which may or may not acknowledge additional I-frames received by the data link layer entity, and a retransmission bit (R bit) that affects whether I-frames are retransmitted.

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

- *Supervisory frame format (S-frame)*

  S-frames are used to acknowledge I-frames and request retransmission of I-frames. Each S-frame has an ReqSeq sequence number which may acknowledge additional I-frames received by the data link layer entity, and a retransmission bit (R bit) that affects whether I-frames are retransmitted.

  Defined types of S-frames are RR (Receiver Ready) and REJ (Reject).

| Frame type | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| I | SAR | | ReqSeq | | | | | | R | TxSeq | | | | | | 0 |
| S | X | X | ReqSeq | | | | | | R | X | X | X | S | | 0 | 1 |

X denotes reserved bits. Shall be coded 0.

*Table 3.1: Control Field formats*

- *Send Sequence Number - TxSeq (6 bits)*

  The send sequence number is used to number each I-frame, to enable sequencing and retransmission.

- *Receive Sequence Number - ReqSeq (6 bits)*

  The receive sequence number is used by the receiver side to acknowledge I-frames, and in the REJ frame to request the retransmission of an I-frame with a specific send sequence number.



LifeScan, Inc. Ex. 1019
Page 1110 of 1420
IPR2025-00103

- *Retransmission Disable Bit - R (1 bit)*

  The Retransmission Disable bit is used to implement Flow Control. The receiver sets the bit when its internal receive buffer is full, this happens when one or more I-frames have been received but the SDU reassembly function has not yet pulled all the frames received. When the sender receives a frame with the Retransmission Disable bit set it shall disable the RetransmissionTimer, this causes the sender to stop retransmitting I-frames.

  R=0: Normal operation. Sender uses the RetransmissionTimer to control retransmission of I-frames. Sender does not use the MonitorTimer.

  R=1: Receiver side requests sender to postpone retransmission of I-frames. Sender monitors signaling with the MonitorTimer. Sender does not use the RetransmissionTimer.

  The functions of ReqSeq and R are independent.

- *Segmentation and Reassembly - SAR (2 bits)*

  The SAR bits define whether an L2CAP SDU is segmented. For segmented SDUs, the SAR bits also define which part of an SDU is in this I-frame, thus allowing one L2CAP SDU to span several I-frames.

  An I-frame with SAR="Start of L2CAP SDU" also contains a length indicator, specifying the number of information octets in the total L2CAP SDU. The encoding of the SAR bits is shown in Table 3.2.

| | |
|---|---|
| 00 | Unsegmented L2CAP SDU |
| 01 | Start of L2CAP SDU |
| 10 | End of L2CAP SDU |
| 11 | Continuation of L2CAP SDU |

*Table 3.2: SAR control element format.*

- *Supervisory function - S (2 bits)*

  The S-bits mark the type of S-frame. There are two types defined: RR (Receiver Ready) and REJ (Reject). The encoding is shown in Table 3.3.

| | |
|---|---|
| 00 | RR - Receiver Ready |
| 01 | REJ - Reject |
| 10 | Reserved |
| 11 | Reserved |

*Table 3.3: S control element format: type of S-frame.*



LifeScan, Inc. Ex. 1019
Page 1111 of 1420
IPR2025-00103

### 3.3.3 L2CAP SDU length field (2 octets)

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

### 3.3.4 Information payload field (0 to 65531 octets)

The information payload field consists of an integral number of octets. The maximum number of octets in this field is the same as the negotiated value of the MPS configuration parameter. The maximum number of octets in this field is also limited by the range of the basic L2CAP header length field. For I-frames without an SDU length field, this limits the maximum number of octets in the field to 65531. For I-frames with an SDU length field, SAR=01, this limits maximum number of octets in the field to 65529. Thus, even if an MPS of 65531 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 65529.



LifeScan, Inc. Ex. 1019
Page 1112 of 1420
IPR2025-00103

### 3.3.5 Frame check sequence (2 octets)

The Frame Check Sequence (FCS) is 2 octets. The FCS is constructed using the generator polynomial g(D) = $D^{16} + D^{15} + D^2 + 1$ (see Figure 3.4). The 16 bit LFSR is initially loaded with the value 0x0000, as depicted in Figure 3.5. The switch S is set in position 1 while data is shifted in, LSB first for each octet. After the last bit has entered the LFSR, the switch is set in position 2, and the register contents are transmitted from right to left (i.e. starting with position 15, then position 14, etc.). The FCS covers the Basic L2CAP header, Control, L2CAP-SDU length and Information payload fields, if present, as shown in Figure 3.3 on page 33.

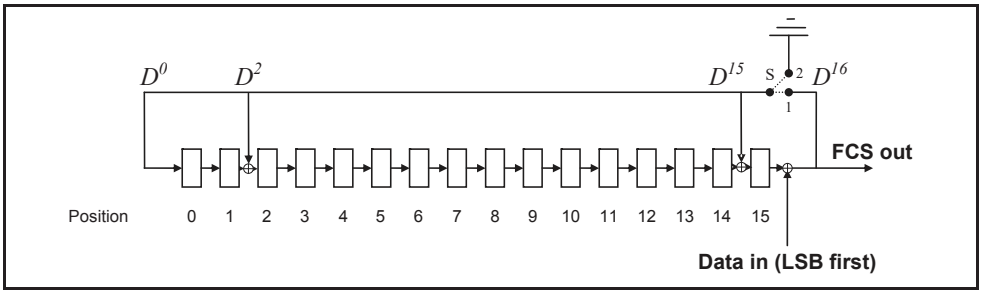



*Figure 3.4: The LFSR circuit generating the FCS.*

| 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 3.5: Initial state of the FCS generating circuit.*

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

1. ***I Frame***
   Length = 14
   Control = 0x0002 (SAR=0, ReqSeq=0, R=0, TxSeq=1)
   Information Payload = 00 01 02 03 04 05 06 07 08 09 (10 octets, hexadecimal notation)
   ==> FCS = 0x6138
   ==> Data to Send = 0E 00 40 00 02 00 00 01 02 03 04 05 06 07 08 09 38 61 (hexadecimal notation)
2. ***RR Frame***
   Length = 4
   Control = 0x0101 (ReqSeq=1, R=0, S=0)
   ==> FCS = 0x14D4
   ==> Data to Send = 04 00 40 00 01 01 D4 14 (hexadecimal notation)



LifeScan, Inc. Ex. 1019
Page 1113 of 1420
IPR2025-00103

### 3.3.6 Invalid frame detection

A received PDU shall be regarded as invalid, if one of the following conditions occurs:

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

These error conditions may be used for error reporting.



LifeScan, Inc. Ex. 1019
Page 1114 of 1420
IPR2025-00103

# 4 SIGNALING PACKET FORMATS

This section describes the signaling commands passed between two L2CAP entities on peer devices. All signaling commands are sent to the signaling channel with CID 0x0001. This signaling channel is available as soon as an ACL logical transport is set up and L2CAP traffic is enabled on the L2CAP logical link. Figure 4.1 on page 39 illustrates the general format of L2CAP PDUs containing signaling commands (C-frames). Multiple commands may be sent in a single C-frame. Commands take the form of Requests and Responses. All L2CAP implementations shall support the reception of C-frames with a payload length that does not exceed the signaling MTU. The minimum supported payload length for the C-frame ($MTU_{sig}$) 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 containing the supported $MTU_{sig}$. Implementations must be able to handle the reception of multiple commands in an L2CAP packet.

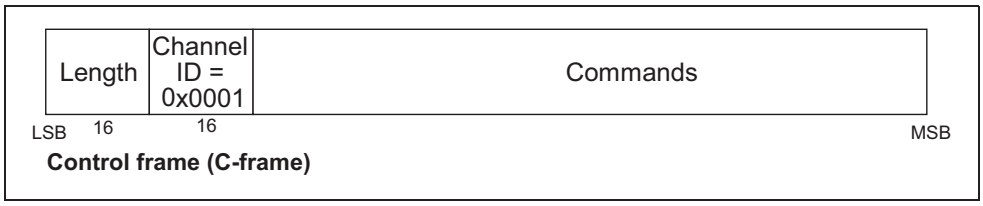



*Figure 4.1: L2CAP PDU format on the signaling channel*

Figure 4.2 displays the general format of all signaling commands.

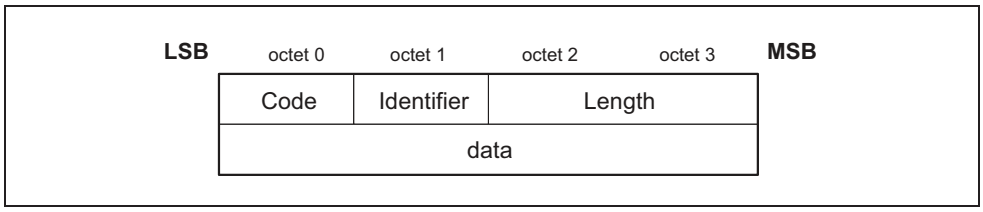



*Figure 4.2: Command format*



LifeScan, Inc. Ex. 1019
Page 1115 of 1420
IPR2025-00103

The fields shown are:

- *Code (1 octet)*

  The Code field is one octet long and identifies the type of command. When a packet is received with an unknown Code field, a Command Reject packet (defined in Section 4.1 on page 41) is sent in response.

  Table 4.1 on page 40 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

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

*Table 4.1: Signaling Command Codes*

- *Identifier (1 octet)*

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

  RTX and ERTX timers are used to determine when the remote end point is not responding to signaling requests. On the expiration of a RTX or ERTX timer, the same identifier shall be used if a duplicate Request is re-sent as stated in Section 6.2 on page 73.

  A device receiving a duplicate request should reply with a duplicate response. A command response with an invalid identifier is silently discarded. Signaling identifier 0x00 is an illegal identifier and shall never be used in any command.

- *Length (2 octets)*

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

- *Data (0 or more octets)*

  The Data field is variable in length. The Code field determines the format of the Data field. The length field determines the length of the data field.

## 4.1 COMMAND REJECT (CODE 0x01)

A Command Reject packet shall be sent in response to a command packet with an unknown command code or when sending the corresponding response is inappropriate. Figure 4.3 on page 41 displays the format of the packet. The identifier shall match the identifier of the command packet being rejected. Implementations shall always send these packets in response to unidentified signaling packets. Command Reject packets should not be sent in response to an identified Response packet.

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

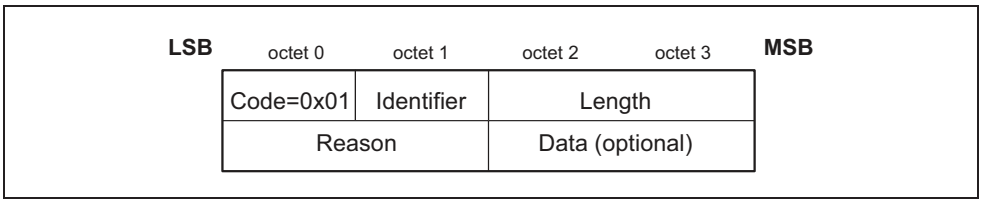



*Figure 4.3: Command Reject packet*

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

- *Reason (2 octets)*

  The Reason field describes why the Request packet was rejected, and is set to one of the Reason codes in Table 4.2.

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

*Table 4.2: Reason Code Descriptions*



LifeScan, Inc. Ex. 1019
Page 1117 of 1420
IPR2025-00103

- *Data (0 or more octets)*

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

  If a command refers to an invalid channel then the Reason code 0x0002 will be returned. Typically a channel is invalid because it does not exist. The data field shall be 4 octets containing the local (first) and remote (second) channel endpoints (relative to the sender of the Command Reject) of the disputed channel. The remote endpoint is the source CID from the rejected command. The local endpoint is the destination CID from the rejected command. If the rejected command contains only one of the channel endpoints, the other one shall be replaced by the null CID 0x0000.

| Reason value | Data Length | Data value |
|---|---|---|
| 0x0000 | 0 octets | N/A |
| 0x0001 | 2 octets | Actual $MTU_{sig}$ |
| 0x0002 | 4 octets | Requested CID |

*Table 4.3: Reason Data values*

## 4.2 CONNECTION REQUEST (CODE 0x02)

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

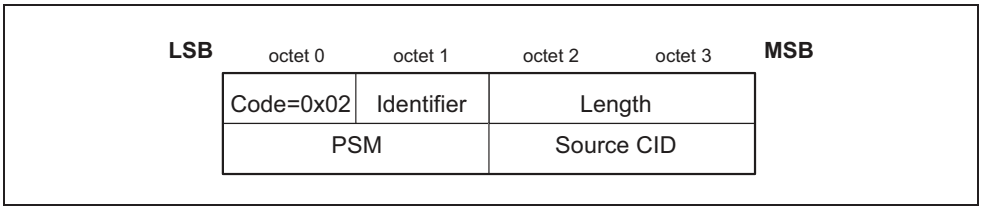



*Figure 4.4: Connection Request Packet*



LifeScan, Inc. Ex. 1019
Page 1118 of 1420
IPR2025-00103

The data fields are:

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

  The PSM field is at least two octets in length. The structure of the PSM field is based on the ISO 3309 extension mechanism for address fields. All PSM values shall be ODD, that is, the least significant bit of the least significant octet must be ’1’. Also, all PSM values shall have the least significant bit of the most significant octet equal to ’0’. This allows the PSM field to be extended beyond 16 bits. PSM values are separated into two ranges. Values in the first range are assigned by the Bluetooth SIG and indicate protocols. The second range of values are dynamically allocated and used in conjunction with the Service Discovery Protocol (SDP). The dynamically assigned values may be used to support multiple implementations of a particular protocol.

  PSM values are defined in the Assigned Numbers document.

- *Source CID - SCID (2 octets)*

  The source CID is two octets in length and represents a channel endpoint on the device sending the request. Once the channel has been configured, data packets flowing to the sender of the request shall be sent to this CID. Thus, the Source CID represents the channel endpoint on the device sending the request and receiving the response.



LifeScan, Inc. Ex. 1019
Page 1119 of 1420
IPR2025-00103

## 4.3 CONNECTION RESPONSE (CODE 0x03)

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

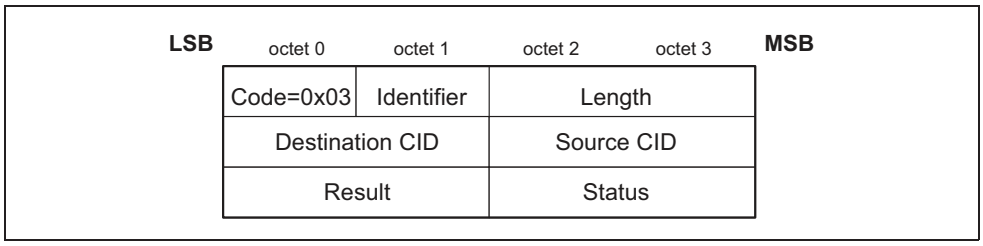



*Figure 4.5: Connection Response Packet*

The data fields are:

- *Destination Channel Identifier - DCID (2 octets)*

  This field contains the channel endpoint on the device sending this Response packet. Thus, the Destination CID represents the channel endpoint on the device receiving the request and sending the response.

- *Source Channel Identifier - SCID (2 octets)*

  This field contains the channel endpoint on the device receiving this Response packet. This is copied from the SCID field of the connection request packet.

- *Result (2 octets)*

  The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed or is pending. A logical channel is established on the receipt of a successful result. Table 4.4 on page 44 defines values for this field. The DCID and SCID fields shall be ignored when the result field indicates the connection was refused.

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

*Table 4.4: Result values*



LifeScan, Inc. Ex. 1019
Page 1120 of 1420
IPR2025-00103

- *Status (2 octets)*

  Only defined for Result = Pending. Indicates the status of the connection. The status is set to one of the values shown in Table 4.5 on page 45.

| Value | Description |
|---|---|
| 0x0000 | No further information available |
| 0x0001 | Authentication pending |
| 0x0002 | Authorization pending |
| Other | Reserved |

*Table 4.5: Status values*

## 4.4 CONFIGURATION REQUEST (CODE 0x04)

Configuration Request packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to re-negotiate this contract whenever appropriate. During a re-negotiation session, all data traffic on the channel should be suspended pending the outcome of the negotiation. Each configuration parameter in a Configuration Request shall be related exclusively to either the outgoing or the incoming data traffic but not both of them. In Section 5 on page 55, the various configuration parameters and their relation to the outgoing or incoming data traffic are shown. If an L2CAP entity receives a Configuration Request while it is waiting for a response it shall not block sending the Configuration Response, otherwise the configuration process may deadlock.

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

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

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

See Section 7.1 on page 77 for details of the configuration procedure.



LifeScan, Inc. Ex. 1019
Page 1121 of 1420
IPR2025-00103

Figure 4.6 defines the format of the Configuration Request packet.

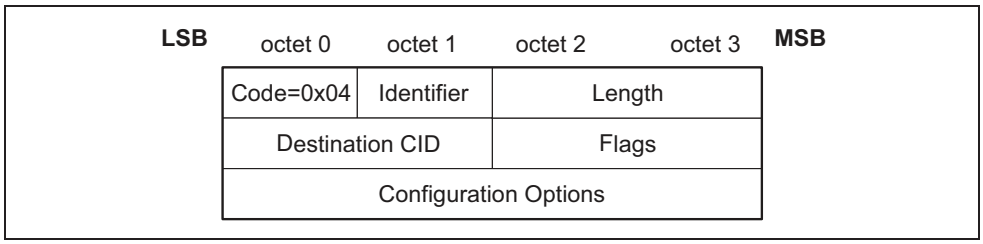



*Figure 4.6: Configuration Request Packet*

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field contains the channel endpoint on the device receiving this Request packet.

- *Flags (2 octets)*

  Figure 4.7 shows the two-octet Flags field. Note the most significant bit is shown on the left.

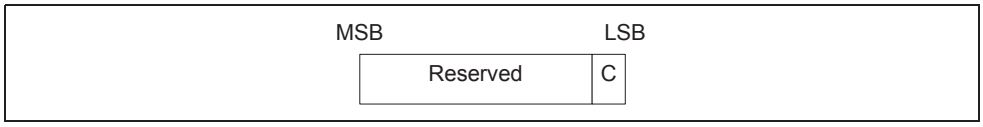



*Figure 4.7: Configuration Request Flags field format*

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

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

When used in the Configuration Request, the continuation flag indicates the responder should expect to receive multiple request packets. The responder shall reply to each Configuration Request packet. The responder may reply to each Configuration Request with a Configuration Response containing the same option(s) present in the Request (except for those error conditions more appropriate for a Command Reject), or the responder may reply with a "Success" Configuration Response packet containing no options, delaying those options until the full Request has been received. The Configuration Request packet with the continuation flag cleared shall be treated as the Configuration Request event in the channel state machine.

When used in the Configuration Response, the continuation flag shall be set to one if the flag is set to one in the Request. If the continuation flag is set to one in the Response when the matching Request has the flag set to zero, it indicates the responder has additional options to send to the requestor. In this situation, the requestor shall send null-option Configuration Requests (with continuation flag set to zero) to the responder until the responder replies with a Configuration Response where the continuation flag is set to zero. The Configuration Response packet with the continuation flag set to zero shall be treated as the Configuration Response event in the channel state machine.

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

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

- *Configuration Options*

  A list of the parameters and their values to be negotiated shall be provided in the Configuration Options field. These are defined in Section 5 on page 55. A Configuration Request may contain no options (referred to as an empty or null configuration request) and can be used to request a response. For an empty configuration request the length field is set to 0x0004.



LifeScan, Inc. Ex. 1019
Page 1123 of 1420
IPR2025-00103

## 4.5 CONFIGURATION RESPONSE (CODE 0X05)

Configuration Response packets shall be sent in reply to Configuration Request packets except when the error condition is covered by a Command Reject response. Each configuration parameter value (if any is present) in a Configuration Response reflects an 'adjustment' to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding Configuration Request. For example, if a configuration request relates to traffic flowing from device A to device B, the sender of the configuration response may adjust this value for the same traffic flowing from device A to device B, but the response can not adjust the value in the reverse direction.

The options sent in the Response depend on the value in the Result field. Figure 4.8 on page 48 defines the format of the Configuration Response packet. See also Section 7.1 on page 77 for details of the configuration process.

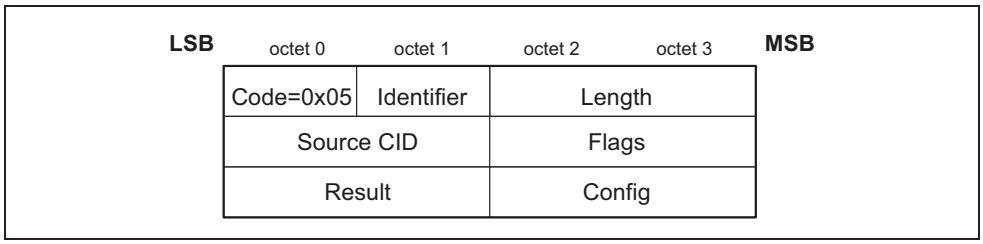



*Figure 4.8: Configuration Response Packet*

The data fields are:

- *Source CID - SCID (2 octets)*

  This field contains the channel endpoint on the device receiving this Response packet. The device receiving the Response shall check that the Identifier field matches the same field in the corresponding configuration request command and the SCID matches its local CID paired with the original DCID.

- *Flags (2 octets)*

  Figure 4.9 displays the two-octet Flags field. Note the most significant bit is shown on the left.

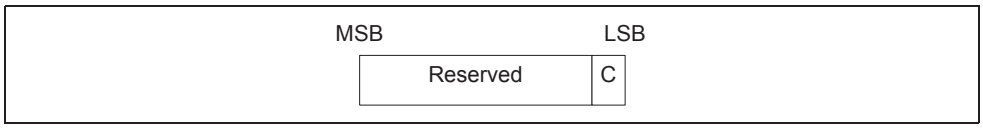



*Figure 4.9: Configuration Response Flags field format*

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 or not the Request was acceptable. See Table 4.6 on page 49 for possible result codes.

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

*Table 4.6: Configuration Response Result codes*

- *Configuration Options*

  This field contains the list of parameters being configured. These are defined in Section 5 on page 55. On a successful result, these parameters contain the return values for any wild card parameter values (see Section 5.3 on page 58) contained in the request.

  On an unacceptable parameters failure (Result=0x0001) the rejected parameters shall be sent in the response with the values that would have been accepted if sent in the original request. Any missing configuration parameters are assumed to have their most recently accepted values and they too shall be included in the Configuration Response if they need to be changed.

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

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

  On an unknown option failure (Result=0x0003), the option types not understood by the recipient of the Request shall be included in the Response unless they are hints. Hints are those options in the Request that are skipped if not understood (see Section 5 on page 55). Hints shall not be included in the Response and shall not be the sole cause for rejecting the Request.

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

## 4.6 DISCONNECTION REQUEST (CODE 0x06)

Terminating an L2CAP channel requires that a disconnection request be sent and acknowledged by a disconnection response. Figure 4.10 on page 50 shows a disconnection request. The receiver shall ensure that both source and destination CIDs match before initiating a channel disconnection.

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

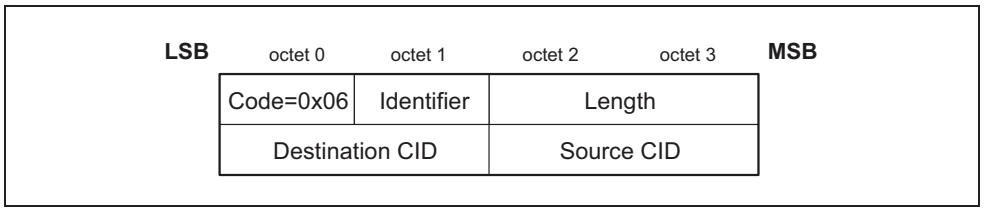



*Figure 4.10: Disconnection Request Packet*

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field specifies the endpoint of the channel to be disconnected on the device receiving this request.

- *Source CID - SCID (2 octets)*

  This field specifies the endpoint of the channel to be disconnected on the device sending this request.

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



LifeScan, Inc. Ex. 1019
Page 1126 of 1420
IPR2025-00103

## 4.7 DISCONNECTION RESPONSE (CODE 0x07)

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

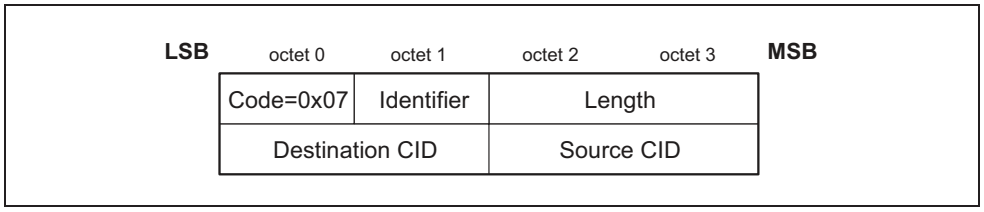



*Figure 4.11: Disconnection Response Packet*

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field identifies the channel endpoint on the device sending the response.

- *Source CID - SCID (2 octets)*

  This field identifies the channel endpoint on the device receiving the response.

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

## 4.8 ECHO REQUEST (CODE 0x08)

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

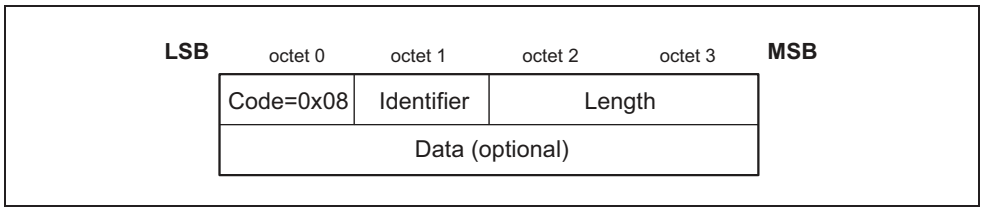



*Figure 4.12: Echo Request Packet*



LifeScan, Inc. Ex. 1019
Page 1127 of 1420
IPR2025-00103

## 4.9 ECHO RESPONSE (CODE 0x09)

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

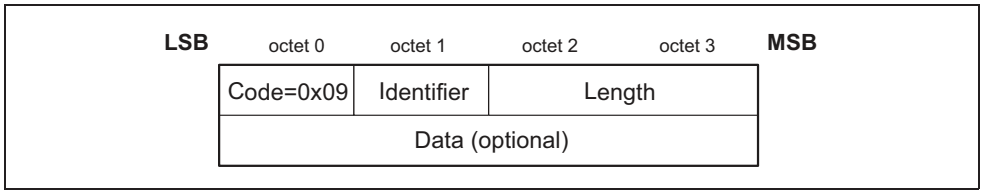



*Figure 4.13: Echo Response Packet*

## 4.10 INFORMATION REQUEST (CODE 0X0A)

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

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

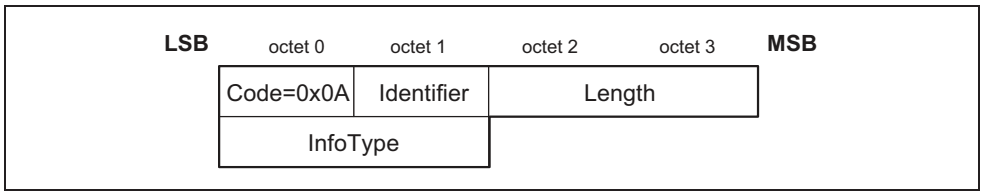



*Figure 4.14: Information Request Packet*

The data fields are:

- *InfoType (2 octets)*

  The InfoType defines the type of implementation specific information being requested. See Section 4.11 on page 53 for details on the type of information requested.

| Value | Description |
|---|---|
| 0x0001 | Connectionless MTU |
| 0x0002 | Extended features supported |
| Other | Reserved |

*Table 4.7: InfoType definitions*



LifeScan, Inc. Ex. 1019
Page 1128 of 1420
IPR2025-00103

## 4.11 INFORMATION RESPONSE (CODE 0X0B)

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

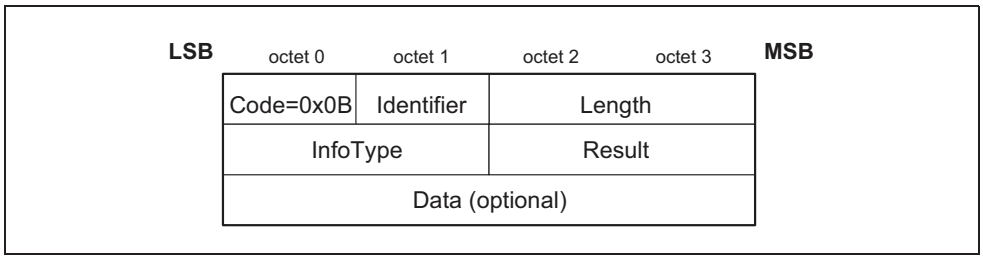



*Figure 4.15: Information Response Packet*

The data fields are:

- *InfoType (2 octets)*

  The InfoType defines the type of implementation specific information that was requested. This value shall be copied from the InfoType field in the Information Request.

- *Result (2 octets)*

  The Result contains information about the success of the request. If result is "Success", the data field contains the information as specified in Table 4.9 on page 54. If result is "Not supported", no data shall be returned.

| Value | Description |
|---|---|
| 0x0000 | Success |
| 0x0001 | Not supported |
| Other | Reserved |

*Table 4.8: Information Response Result values*

- *Data (0 or more octets)*

  The contents of the Data field depends on the InfoType.
  For InfoType = 0x0001 the data field contains the remote entity's 2-octet acceptable connectionless MTU. The default value is defined in Section 3.2 on page 32.

  For InfoType = 0x0002, the data field contains the 4 octet L2CAP extended feature mask. The feature mask refers to the extended features that the L2CAP entity sending the Information Response supports. The feature bits contained in the L2CAP feature mask are specified in Section 4.12 on page 54.

Note: L2CAP entities of versions prior to version 1.2, receiving an Information Request with InfoType = 0x0002 for an L2CAP feature discovery, return an Information Response with result code "Not supported". L2CAP entities at version 1.2 or later that have an all zero extended features mask may return an Information Response with result code "Not supported".

| InfoType | Data | Data Length (octets) |
|---|---|---|
| 0x0001 | Connectionless MTU | 2 |
| 0x0002 | Extended feature mask | 4 |

*Table 4.9: Information Response Data fields*

## 4.12 EXTENDED FEATURE MASK

The features are represented as a bit mask in the Information Response data field (see Section 4.11 on page 53). For each feature a single bit is specified which shall be set to 1 if the feature is supported and set to 0 otherwise. All unknown, reserved, or unassigned feature bits shall be set to 0.

The feature mask shown in Table 4.10 on page 54 consists of 4 octets (numbered octet 0 ... 3), with bit numbers 0 ... 7 each. Within the Information Response packet data field, bit 0 of octet 0 is aligned leftmost, bit 7 of octet 3 is aligned rightmost.

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

| No. | Supported feature | Octet | Bit |
|---|---|---|---|
| 0 | Flow control mode | 0 | 0 |
| 1 | Retransmission mode | 0 | 1 |
| 2 | Bi-directional QoS[1] | 0 | 2 |
| 31 | Reserved for feature mask extension | 3 | 7 |

*Table 4.10: Extended feature mask.*

1. Peer side supports upper layer control of the Link Manager's Bi-directional QoS, see Section 5.3 on page 58 for more details.



LifeScan, Inc. Ex. 1019
Page 1130 of 1420
IPR2025-00103

# 5 CONFIGURATION PARAMETER OPTIONS

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

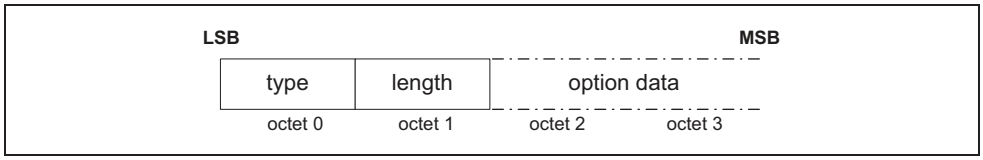



*Figure 5.1: Configuration option format*

The configuration option fields are:

- *Type (1 octet)*

  The option type field defines the parameters being configured. The most significant bit of the type determines the action taken if the option is not recognized.

  0 - option must be recognized; if the option is not recognized then refuse the configuration request

  1 - option is a hint; if the option is not recognized then skip the option and continue processing

- *Length (1 octet)*

  The length field defines the number of octets in the option data. Thus an option type without option data has a length of 0.

- *Option data*

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

## 5.1 MAXIMUM TRANSMISSION UNIT (MTU)

This option specifies the maximum SDU size the sender of this option is capable of accepting for a channel. The type is 0x01, and the payload length is 2 octets, carrying the two-octet MTU size value as the only information element (see Figure 5.2 on page 56). Unlike the B-Frame length field, the I-Frame length field may be greater then the configured MTU because it includes the octet lengths of the Control, L2CAP SDU Length (when present), and frame check sequence fields as well as the Information octets.

MTU is not a negotiated value, it is an informational parameter that each device can specify independently. It indicates to the remote device that the local device can receive, in this channel, an MTU larger than the minimum required. All L2CAP implementations shall support a minimum MTU of 48 octets, however some protocols and profiles explicitly require support for a



LifeScan, Inc. Ex. 1019
Page 1131 of 1420
IPR2025-00103

larger MTU. The minimum MTU for a channel is the larger of the L2CAP minimum 48 octet MTU and any MTU explicitly required by the protocols and profiles using that channel. (Note: the MTU is only affected by the profile directly using the channel. For example, if a service discovery transaction is initiated by a non service discovery profile, that profile does not affect the MTU of the L2CAP channel used for service discovery).

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

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

The signaling described in Section 4.5 on page 48 may be used to reject an MTU smaller than the minimum MTU for a channel. The "failure-unacceptable parameters" result sent to reject the MTU shall include the proposed value of MTU that the remote device intends to transmit. It is implementation specific whether the local device continues the configuration process or disconnects the channel.

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

The MTU to be used on this channel for the traffic flowing in the opposite direction will be established when the remote device sends its own Configuration Request as explained in Section 4.4 on page 45.

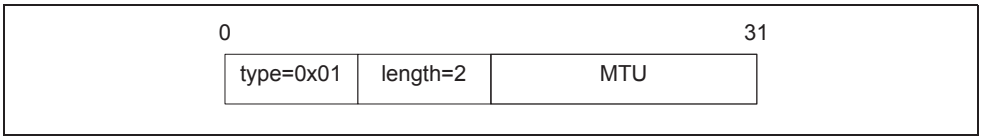



*Figure 5.2: MTU Option Format*



LifeScan, Inc. Ex. 1019
Page 1132 of 1420
IPR2025-00103

The option data field is:

- Maximum Transmission Unit - MTU (2 octets)

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

## 5.2 FLUSH TIMEOUT OPTION

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

If the remote device returns a negative response to this option and the local device cannot honor the proposed value, then it shall either continue the configuration process by sending a new request with the original value, or disconnect the channel. The flush timeout applies to all channels on the same ACL logical transport but may be overridden on a packet by packet basis by marking individual L2CAP packets as non-automatically-flushable via the Packet_Boundary_Flag in the HCI ACL Data Packet (see section 1.1 on page 19).

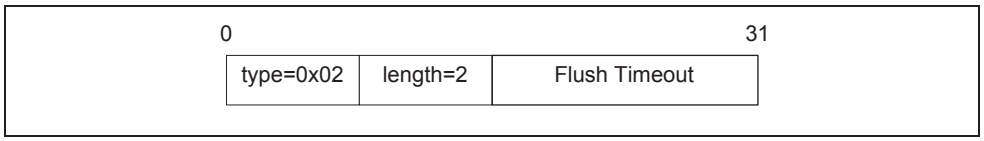



*Figure 5.3: Flush Timeout option format.*

The option data field is:

- *Flush Timeout*

  This value is the Flush Timeout in milliseconds. This is an asymmetric value and the sender of the Request shall specify its flush timeout value if it differs from the default value of 0xFFFF.

  Possible values are:

  0x0001 - no retransmissions at the baseband level should be performed since the minimum polling interval is 1.25 ms.

  0x0002 to 0xFFFE - Flush Timeout used by the baseband.

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



LifeScan, Inc. Ex. 1019
Page 1133 of 1420
IPR2025-00103

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

## 5.3 QUALITY OF SERVICE (QOS) OPTION

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

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

In a configuration request, this option describes the outgoing traffic flow from the device sending the request. In a positive Configuration Response, this option describes the incoming traffic flow agreement to the device sending the response. In a negative Configuration Response, this option describes the preferred incoming traffic flow to the device sending the response.

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

The remote device's Configuration Response contains information that depends on the value of the result field (see Section 4.5 on page 48). If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is "Failure – unacceptable parameters", the response shall include a list of outgoing flow specification parameters and parameter values that would make a new Connection Request from the local device acceptable by the remote device. Both explicitly referenced in a Configuration Request or implied configuration parameters can be included in a Configuration Response. Recall that any missing configuration parameters from a Configuration Request are assumed to have their most recently accepted values.

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

1. Internet Engineering Task Force, "A Proposed Flow Specification", RFC 1363, September 1992.



LifeScan, Inc. Ex. 1019
Page 1134 of 1420
IPR2025-00103

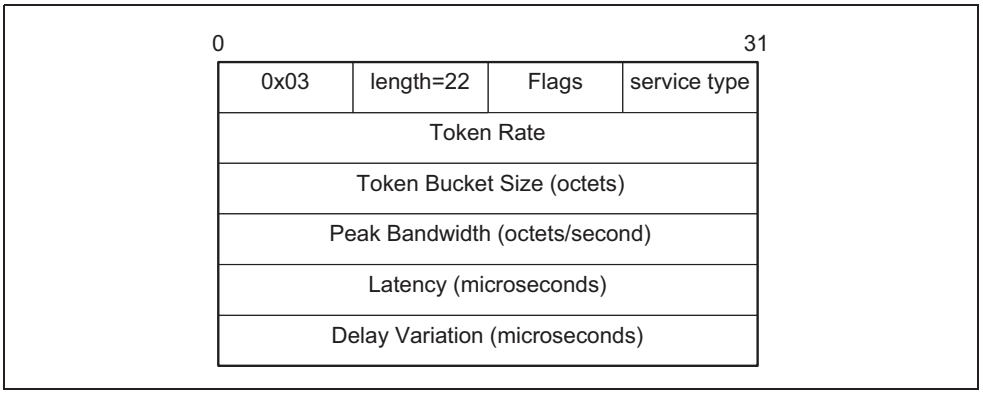



*Figure 5.4: Quality of Service (QoS) option format containing Flow Specification.*

The option data fields are:

- *Flags (1 octet)*

  Reserved for future use and shall be set to 0 and ignored by the receiver.

- *Service Type (1 octet)*

  This field indicates the level of service required. Table 5.1 on page 59 defines the different services available. The default value is ‘Best effort’.

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

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

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

*Table 5.1: Service type definitions*

- *Token Rate (4 octets)*

  The value of this field represents the average data rate with which the application transmits data. The application may send data at this rate continuously. On a short time scale the application may send data in excess of the average data rate, dependent on the specified Token Bucket Size and Peak Bandwidth (see below). The Token Bucket Size and Peak Bandwidth allow the application to transmit data in a 'bursty' fashion.

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

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

The Token Rate is specified in octets per second. The value 0x00000000 indicates no token rate is specified. This is the default value and means “do not care”. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token rate available. The meaning of this value depends on the service type. For best effort, the value is a hint that the application wants as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

- *Token Bucket Size (4 octets)*

  The Token Bucket Size specifies a limit on the 'burstiness' with which the application may transmit data. The application may offer a burst of data equal to the Token Bucket Size instantaneously, limited by the Peak Bandwidth (see below). The Token Bucket Size is specified in octets.

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

  The value of 0x00000000 means that no token bucket is needed; this is the default value. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token bucket available. The meaning of this value depends on the service type. For best effort, the value indicates the application wants a bucket as big as possible. For Guaranteed service the value represents the maximum L2CAP SDU size.

  The Token Bucket Size is a property of the traffic carried over the L2CAP channel. The Maximum Transmission Unit (MTU) is a property of an L2CAP implementation. For the Guaranteed service the Token Bucket Size shall be smaller or equal to the MTU.



LifeScan, Inc. Ex. 1019
Page 1136 of 1420
IPR2025-00103

- *Peak Bandwidth (4 octets)*

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

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

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

- *Access Latency (4 octets)*

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

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

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

- *Delay Variation (4 octets)*

  The value of this field is the difference, in microseconds, between the maximum and minimum possible delay of an L2CAP SDU between two L2CAP peers. The Delay Variation is a purely informational parameter. The value 0xFFFFFFFF means “do not care” and is the default value.

## 5.4 RETRANSMISSION AND FLOW CONTROL OPTION

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

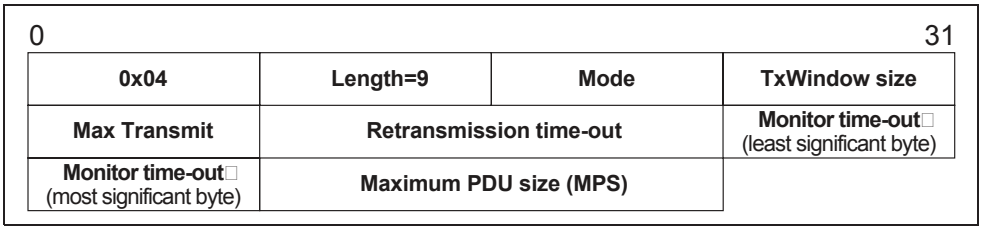



*Figure 5.5: Retransmission and Flow Control option format.*

The option data fields are:

- *Mode (1 octet)*

  The field contain the requested mode of the link. Possible values are shown in Table 5.2 on page 62.

| Value | Description |
|---|---|
| 0x00 | Basic L2CAP mode |
| 0x01 | Retransmission Mode |
| 0x02 | Flow control mode |
| Other values | Reserved for future use |

*Table 5.2: Mode definitions.*

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

  Retransmission mode should be enabled if a reliable channel has been requested, or if the L2CAP Flush Time-Out 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.



LifeScan, Inc. Ex. 1019
Page 1138 of 1420
IPR2025-00103

- *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).

  MaxTransmit 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. MaxTransmit shall not be set to zero.

- *Retransmission time-out (2 octets)*

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

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

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

- *Monitor time-out (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 MonitorTimer, see below).

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

- *Maximum PDU payload Size - MPS (2 octets)*

  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.

The settings are configured separately for the two directions of an L2CAP connection. For example, an L2CAP connection can be configured as Flow Control



LifeScan, Inc. Ex. 1019
Page 1139 of 1420
IPR2025-00103

mode in one direction and Retransmission mode in the other direction. If Basic L2CAP mode is configured in one direction and Retransmission mode or Flow control mode is configured in the other direction on the same L2CAP channel then the channel shall not be used.
Note: this asymmetric configuration only occurs during configuration.



LifeScan, Inc. Ex. 1019
Page 1140 of 1420
IPR2025-00103

# 6 STATE MACHINE

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

## 6.1 GENERAL RULES FOR THE STATE MACHINE:

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

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

- CLOSED – channel not connected.
- WAIT_CONNECT – a connection request has been received, but only a connection response with indication “pending” can be sent.
- WAIT_CONNECT_RSP – a connection request has been sent, pending a positive connect response.
- CONFIG – the different options are being negotiated for both sides; this state comprises a number of substates, see Section 6.1.3 on page 67
- OPEN – user data transfer state.
- WAIT_DISCONNECT – a disconnect request has been sent, pending a disconnect response.

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

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

- *OpenChannel_Req* – a local L2CAP entity is requested to set up a new connection-oriented channel.
- *OpenChannel_Rsp* – a local L2CAP entity is requested to finally accept or refuse a pending connection request.
- *ConfigureChannel_Req* – a local L2CAP entity is requested to initiate an outgoing configuration request.
- *CloseChannel_Req* – a local L2CAP entity is requested to close a channel.
- *SendData_Req* – a local L2CAP entity is requested to transmit an SDU.
- *ReconfigureChannel_Req* – a local L2CAP entity is requested to reconfigure the parameters of a connection-oriented channel.



LifeScan, Inc. Ex. 1019
Page 1141 of 1420
IPR2025-00103

There is a single state machine for each L2CAP connection-oriented channel that is active. A state machine is created for each new L2CAP_ConnectReq received. The state machine always starts in the CLOSED state.

To simplify the state event tables, the RTX and ERTX timers, as well as the handling of request retransmissions are described in Section 6.2 on page 73 and not included in the state tables.

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

The following states and transitions are illustrated in Figure 6.1 on page 75.

### 6.1.1 CLOSED state

| 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 |

*Table 6.1: CLOSED state event table.*

**Notes:**

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



LifeScan, Inc. Ex. 1019
Page 1142 of 1420
IPR2025-00103

### 6.1.2 WAIT_CONNECT_RSP state

| 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 |
| L2CAP_ConfigRsp | - | Ignore | WAIT_CONNECT_RSP |
| L2CAP_DisconnectRsp | - | Ignore | WAIT_CONNECT_RSP |
| L2CAP_Data | - | Ignore | WAIT_CONNECT_RSP |

*Table 6.2: WAIT_CONNECT_RSP state event table.*

**Notes:**

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

### 6.1.3 WAIT_CONNECT state

| 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 |

*Table 6.3: WAIT_CONNECT state event table.*

**Notes:**

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



LifeScan, Inc. Ex. 1019
Page 1143 of 1420
IPR2025-00103

### 6.1.4 CONFIG state

As it is also described in Section 7.1 on page 77, both L2CAP entities initiate a configuration request during the configuration process. This means that each device adopts an initiator role for the outgoing configuration request, and an acceptor role for the incoming configuration request. Configurations in both directions may occur sequentially, but can also occur in parallel.

The following substates are distinguished within the CONFIG state:

- WAIT_CONFIG – a device has sent or received a connection response, but has neither initiated a configuration request yet, nor received a configuration request with acceptable parameters.
- WAIT_SEND_CONFIG – for the initiator path, a configuration request has not yet been initiated, while for the response path, a request with acceptable options has been received.
- WAIT_CONFIG_REQ_RSP – for the initiator path, a request has been sent but a positive response has not yet been received, and for the acceptor path, a request with acceptable options has not yet been received.
- WAIT_CONFIG_RSP – the acceptor path is complete after having responded to acceptable options, but for the initiator path, a positive response on the recent request has not yet been received.
- WAIT_CONFIG_REQ – the initiator path is complete after having received a positive response, but for the acceptor path, a request with acceptable options has not yet been received.

According to Section 6.1.1 on page 66 and Section 6.1.2 on page 67, the CONFIG state is entered via WAIT_CONFIG substate from either the CLOSED state, the WAIT_CONNECT state, or the WAIT_CONNECT_RSP state. The CONFIG state is left for the OPEN state if both the initiator and acceptor paths complete successfully.

For better overview, separate tables are given: Table 6.4 shows the success transitions; therein, transitions on one of the minimum paths (no previous non-success transitions) are shaded. Table 6.5 on page 69 shows the non-success transitions within the configuration process, and Table 6.6 on page 70 shows further transition cause by events not belonging to the configuration process itself. The following configuration states and transitions are illustrated in Figure 6.2 on page 76.

| 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 |

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

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| WAIT_CONFIG_REQ_RSP | L2CAP_ConfigReq | Options acceptable | Send L2CAP_Config Rsp (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_Config Rsp (success) | OPEN |
| WAIT_SEND_CONFIG | *ConfigureChannel_Req* | - | Send L2CAP_Config Req | WAIT_CONFIG_RSP |
| WAIT_CONFIG_RSP | L2CAP_ConfigRsp | Remote side accepts options | - | OPEN |

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

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| WAIT_CONFIG | L2CAP_ConfigReq | Options not acceptable | Send L2CAP_Config Rsp (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_Config Rsp (fail) | WAIT_CONFIG_REQ_RSP |
| WAIT_CONFIG_REQ_RSP | L2CAP_ConfigRsp | Remote side rejects options | Send L2CAP_Config Req (new options) | WAIT_CONFIG_REQ_RSP |
| WAIT_CONFIG_REQ | L2CAP_ConfigReq | Options not acceptable | Send L2CAP_Config Rsp (fail) | WAIT_CONFIG_REQ |
| WAIT_CONFIG_REQ | L2CAP_ConfigRsp | - | Ignore | WAIT_CONFIG_REQ |
| WAIT_CONFIG_RSP | L2CAP_ConfigRsp | Remote side rejects options | Send L2CAP_Config Req (new options) | WAIT_CONFIG_RSP |

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



LifeScan, Inc. Ex. 1019
Page 1145 of 1420
IPR2025-00103

| Previous state | Event | Condition | Action | Next State |
|---|---|---|---|---|
| CONFIG (any substate) | *CloseChannel_Req* | Any internal reason to stop | Send L2CAP_ Disconnect Req | WAIT_ DISCONNECT |
| CONFIG (any substate) | L2CAP_Disconnect Req | - | Send L2CAP_ Disconnect Rsp | CLOSED |
| CONFIG (any substate) | L2CAP_Disconnect Rsp | - | Ignore | CONFIG (remain in sub-state) |
| CONFIG (any substate) | L2CAP_Data | - | Process the PDU | CONFIG (remain in sub-state) |

*Table 6.6: CONFIG state/substates event table: events not related to configuration process.*

**Notes:**

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



LifeScan, Inc. Ex. 1019
Page 1146 of 1420
IPR2025-00103

### 6.1.5 OPEN state

| Event | Condition | Action | Next State |
|---|---|---|---|
| *SendData_req* | - | Send L2CAP_Data packet according to configured mode | OPEN |
| *ReconfigureChannel_req* | - | Complete outgoing SDU<br>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<br>Send L2CAP_ConfigRsp (ok) | CONFIG (substate WAIT_CONFIG_REQ) |
| L2CAP_ConfigReq | Incoming config. options not acceptable | Complete outgoing SDU<br>Send L2CAP_ConfigRsp (fail) | OPEN |
| L2CAP_DisconnectReq | - | Send L2CAP_DisconnectRsp | CLOSED |
| L2CAP_DisconnectRsp | - | Ignore | OPEN |
| L2CAP_Data | - | Process the PDU | OPEN |

*Table 6.7: OPEN state event table.*

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

### 6.1.6 WAIT_DISCONNECT state

| Event | Condition | Action | Next State |
|---|---|---|---|
| L2CAP_ConnectRsp | - | Ignore | WAIT_DISCONNECT |
| L2CAP_ConfigReq | - | Send L2CAP_CommandReject with reason Invalid CID | WAIT_DISCONNECT |
| L2CAP_ConfigRsp | - | Ignore | WAIT_DISCONNECT |
| L2CAP_DisconnectReq | - | Send L2CAP_DisconnectRsp | CLOSED |
| L2CAP_DisconnectRsp | - | - | CLOSED |

*Table 6.8: WAIT_DISCONNECT state event table.*

| Event | Condition | Action | Next State |
|---|---|---|---|
| L2CAP_Data | - | Ignore | WAIT_DISCONNECT |

*Table 6.8: WAIT_DISCONNECT state event table.*



LifeScan, Inc. Ex. 1019
Page 1148 of 1420
IPR2025-00103

## 6.2 TIMERS EVENTS

### 6.2.1 RTX

The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signaling requests. This timer is started when a signaling request (see Section 7 on page 77) is sent to the remote device. This timer is disabled when the response is received. If the initial timer expires, a duplicate Request message may be sent or the channel identified in the request may be disconnected. If a duplicate Request message is sent, the RTX timeout value shall be reset to a new value at least double the previous value. When retransmitting the Request message, the context of the same state shall be assumed as with the original transmission. If a Request message is received that is identified as a duplicate (retransmission), it shall be processed in the context of the same state which applied when the original Request message was received.

Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before terminating the channel identified by the Requests. The one exception is the signaling CID that should never be terminated. The decision should be based on the flush timeout of the signaling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level.

For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq and enter WAIT_DISCONNECT state. Channels can be transitioned directly to the CLOSED state.

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



LifeScan, Inc. Ex. 1019
Page 1149 of 1420
IPR2025-00103

### 6.2.2 ERTX

The Extended Response Timeout eXpired (ERTX) timer is used in place of the RTX timer when it is suspected the remote endpoint is performing additional processing of a request signal. This timer is started when the remote endpoint responds that a request is pending, e.g., when an L2CAP_ConnectRsp event with a "connect pending" result (0x0001) is received. This timer is disabled when the formal response is received or the physical link is lost. If the initial timer expires, a duplicate Request may be sent or the channel may be disconnected.

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

The value of this timer is implementation-dependent but the minimum initial value is 60 seconds and the maximum initial value is 300 seconds. Similar to RTX, there MUST be at least one ERTX timer for each outstanding request that received a Pending response. There should be at most one (RTX or ERTX) associated with each outstanding request. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 300 seconds. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq and enter WAIT_DISCONNECT state. Channels should be transitioned directly to the CLOSED state.



LifeScan, Inc. Ex. 1019
Page 1150 of 1420
IPR2025-00103

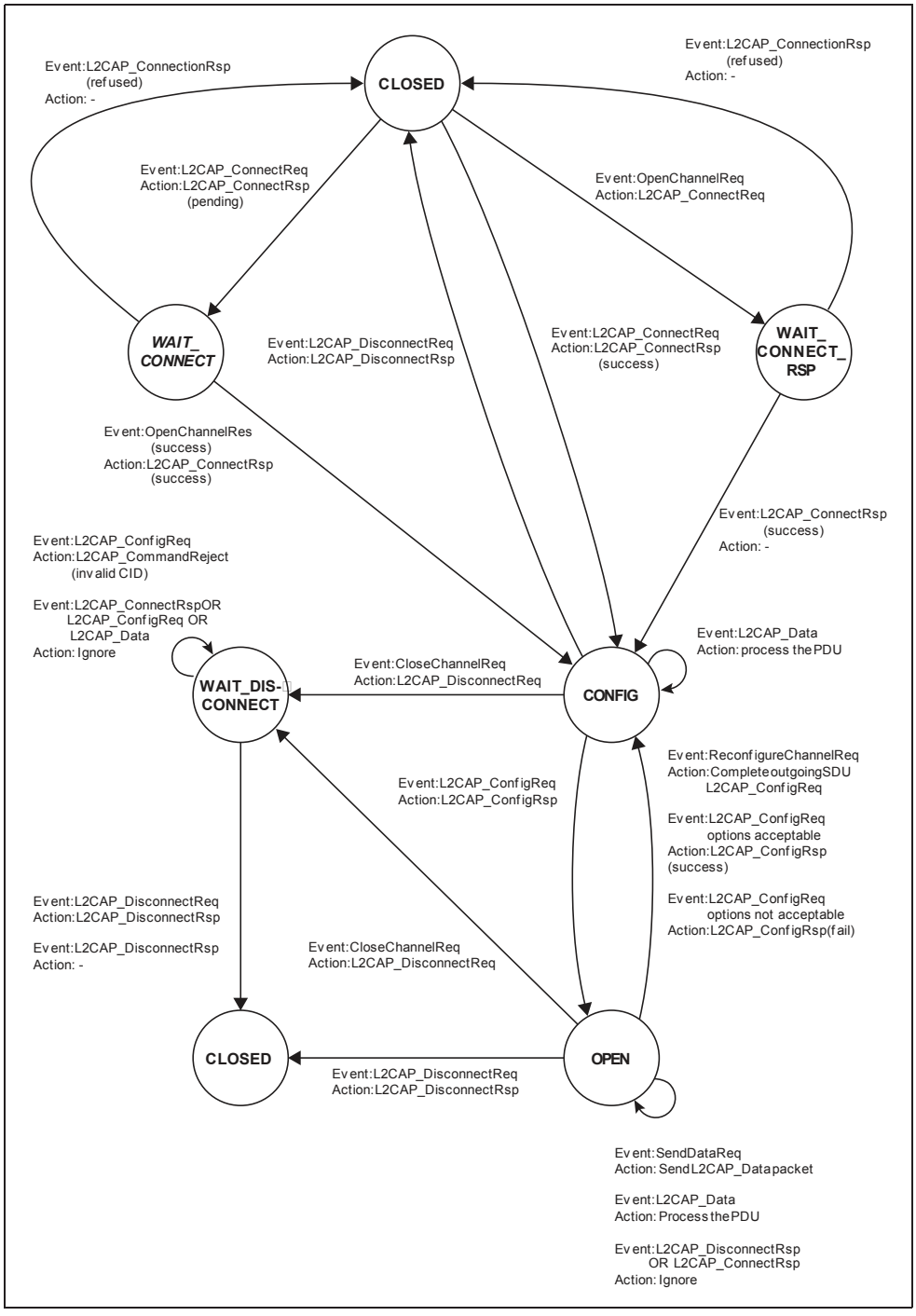



*Figure 6.1: States and transitions.*



LifeScan, Inc. Ex. 1019
Page 1151 of 1420
IPR2025-00103

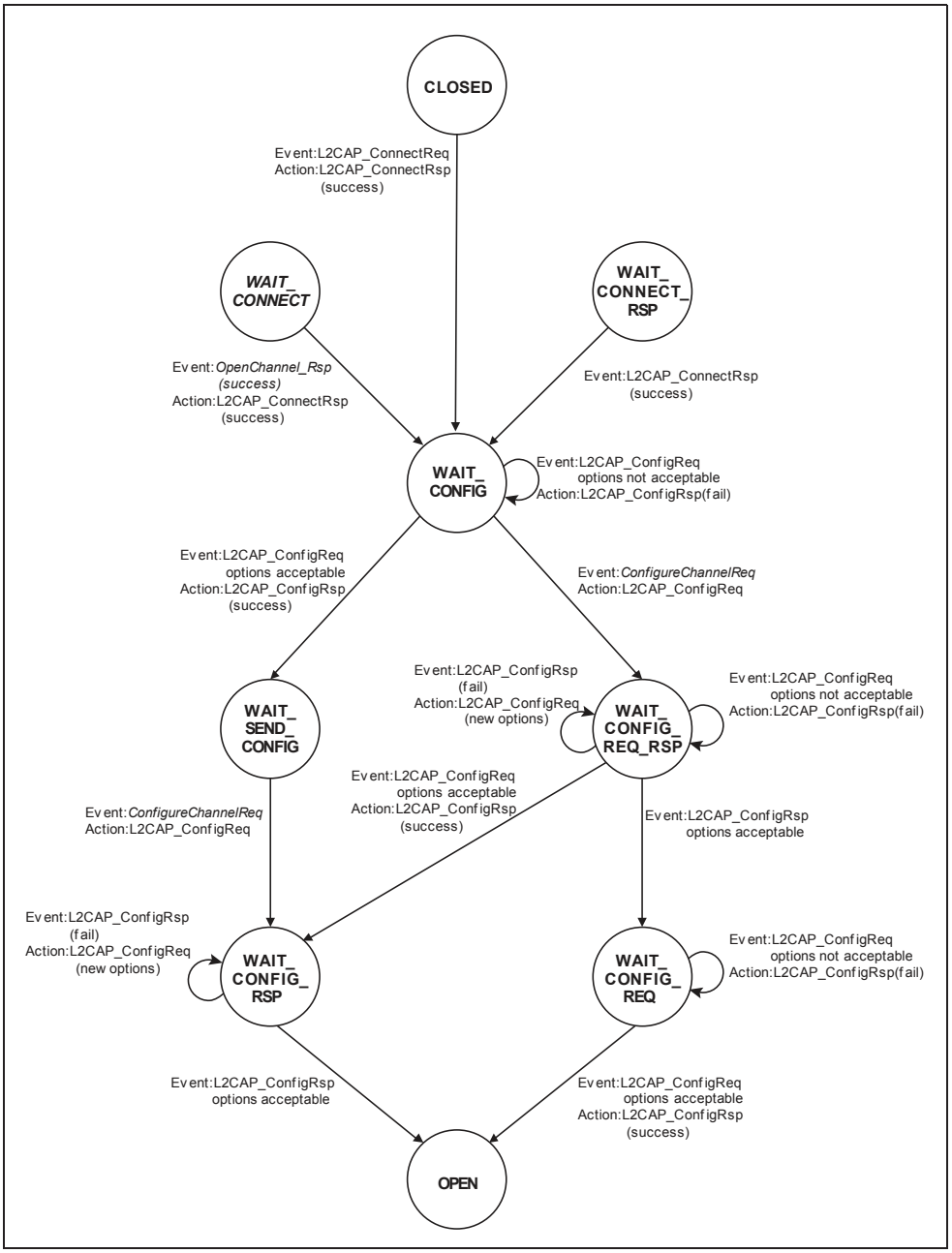



*Figure 6.2: Configuration states and transitions.*



LifeScan, Inc. Ex. 1019
Page 1152 of 1420
IPR2025-00103

# 7 GENERAL PROCEDURES

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

Procedures for the flow control and retransmission modes are described in Section 8 on page 85.

## 7.1 CONFIGURATION PROCESS

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

1. Informing the remote side of the non-default parameters that the local side will accept using a Configuration Request
2. Remote side responds, agreeing or disagreeing with these values, including the default ones, using a Configuration Response.
3. The local and remote devices repeat steps (1) and (2) until agreement on all parameters is reached.

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

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



LifeScan, Inc. Ex. 1019
Page 1153 of 1420
IPR2025-00103

### 7.1.1 Request path

The Request Path can configure the following:

- requester's incoming MTU.
- requester's outgoing flush timeout.
- requester's outgoing QoS.
- requester's incoming flow and error control information.

Table 7.1 on page 78 defines the configuration options that may be placed in a Configuration Request.

| Parameter | Description |
|---|---|
| MTU | Incoming MTU information |
| FlushTO | Outgoing flush timeout |
| QoS | Outgoing QoS information |
| RFCMode | Incoming Retransmission and Flow Control Mode |

*Table 7.1: Parameters allowed in Request*

The state machine for the configuration process is described in Section 6 on page 65.

### 7.1.2 Response path

The Response Path can configure the following:

- responder's outgoing MTU, that is the remote side's incoming MTU.
- remote side's flush timeout.
- responder's incoming QoS Flow Specification (remote side's outgoing QoS Flow Specification).
- responder's Outgoing Flow and Error Control information

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

| Parameter | Description |
|---|---|
| MTU | Outgoing MTU information |
| FlushTO | Incoming flush timeout |
| QoS | Incoming QoS information |
| RFCMode | Outgoing Retransmission and Flow Control Mode |

*Table 7.2: Parameters allowed in Response*

## 7.2 FRAGMENTATION AND RECOMBINATION

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

### 7.2.1 Fragmentation of L2CAP PDUs

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

The two LLID bits defined in the first octet of baseband payload (also called the frame header) are used to signal the start and continuation of L2CAP PDUs. LLID shall be ’10’ for the first segment in an L2CAP PDU and ’01’ for a continuation segment. An illustration of fragmentation is shown in Figure 7.1 on page 79. An example of how fragmentation might be used in a device with HCI is shown in Figure 7.2 on page 80.

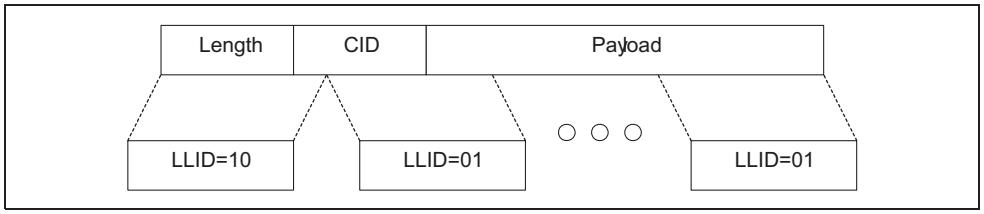



*Figure 7.1: L2CAP fragmentation.*

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 baseband packets. Thus, both L2CAP and the link controller use the same mechanism to control the size of fragments.



LifeScan, Inc. Ex. 1019
Page 1155 of 1420
IPR2025-00103

### 7.2.2 Recombination of L2CAP PDUs

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

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

An L2CAP implementation shall use the length field in the header of L2CAP PDUs, see Section 3 on page 31, as a consistency check and shall discard any L2CAP PDUs that fail to match the length field. If channel reliability is not needed, packets with invalid lengths may be silently discarded. For reliable channels, an L2CAP implementation shall indicate to the upper layer that the channel has become unreliable. Reliable channels are defined by having an infinite flush timeout value as specified in Section 5.2 on page 57. For higher data integrity L2CAP should be operated in the Retransmission Mode.

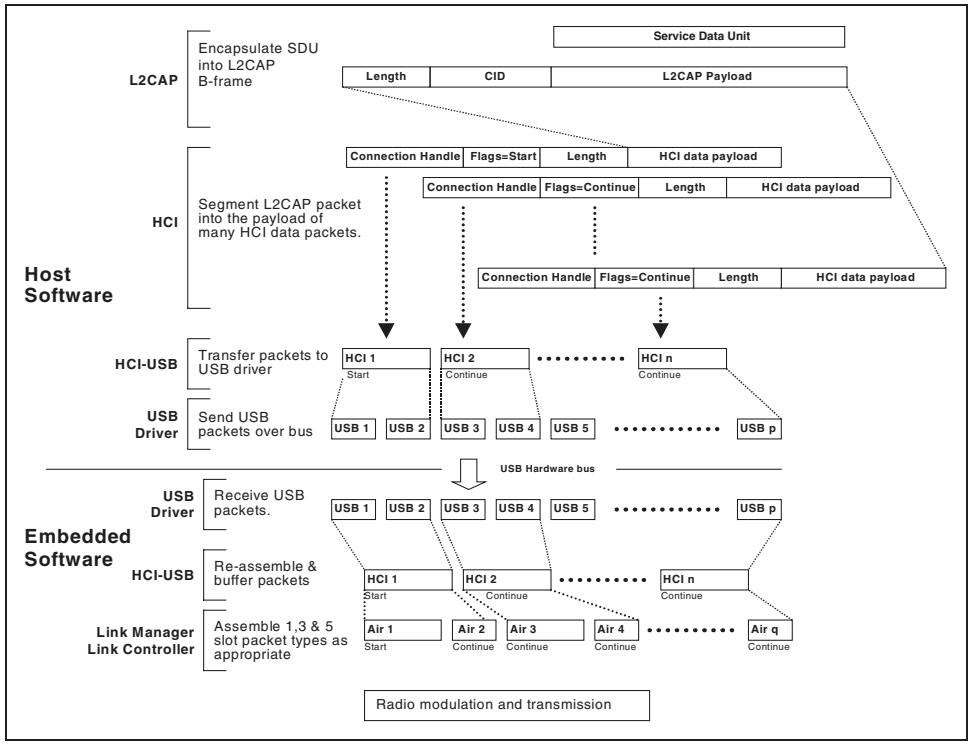



*Figure 7.2: Example of fragmentation processes in a device with HCI.*



LifeScan, Inc. Ex. 1019
Page 1156 of 1420
IPR2025-00103

## 7.3 ENCAPSULATION OF SDUs

All SDUs are encapsulated into one or more L2CAP PDUs.

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

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

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

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

### 7.3.1 Segmentation of L2CAP SDUs

In Flow Control or Retransmission modes, incoming SDUs may be broken down into segments, which shall then be individually encapsulated with L2CAP protocol elements (header and checksum fields) to form I-frames. I-frames are subject to flow control and may be subject to retransmission procedures. The header carries a 2 bit SAR field that is used to identify whether the I-frame is a 'start', 'end' or 'continuation' packet or whether it carries a complete, unsegmented SDU. Figure 7.3 on page 81 illustrates segmentation and fragmentation.

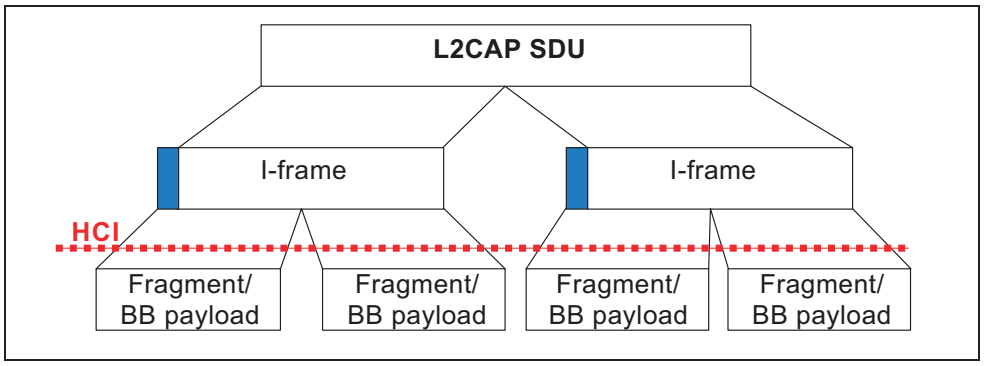



*Figure 7.3: Segmentation and fragmentation of an SDU.*



LifeScan, Inc. Ex. 1019
Page 1157 of 1420
IPR2025-00103

### 7.3.2 Reassembly of L2CAP SDUs

The receiving side uses the SAR field of incoming 'I-frames' for the reassembly process. The L2CAP SDU length field, present in the "start of SDU" I-frame, is an extra integrity check, and together with the sequence numbers may be used to indicate lost L2CAP SDUs to the application. Figure 7.3 on page 81 illustrates segmentation and fragmentation.

### 7.3.3 Segmentation and fragmentation

Figure 7.4 on page 82 illustrates the use of segmentation and fragmentation operations to transmit a single SDU. Note that while SDUs and L2CAP PDUs are transported in peer-to-peer fashion, the fragment size used by the Fragmentation and Recombination routines is implementation specific and may not be the same in the sender and the receiver. The over-the-air sequence of baseband packets as created by the sender is common to both devices.

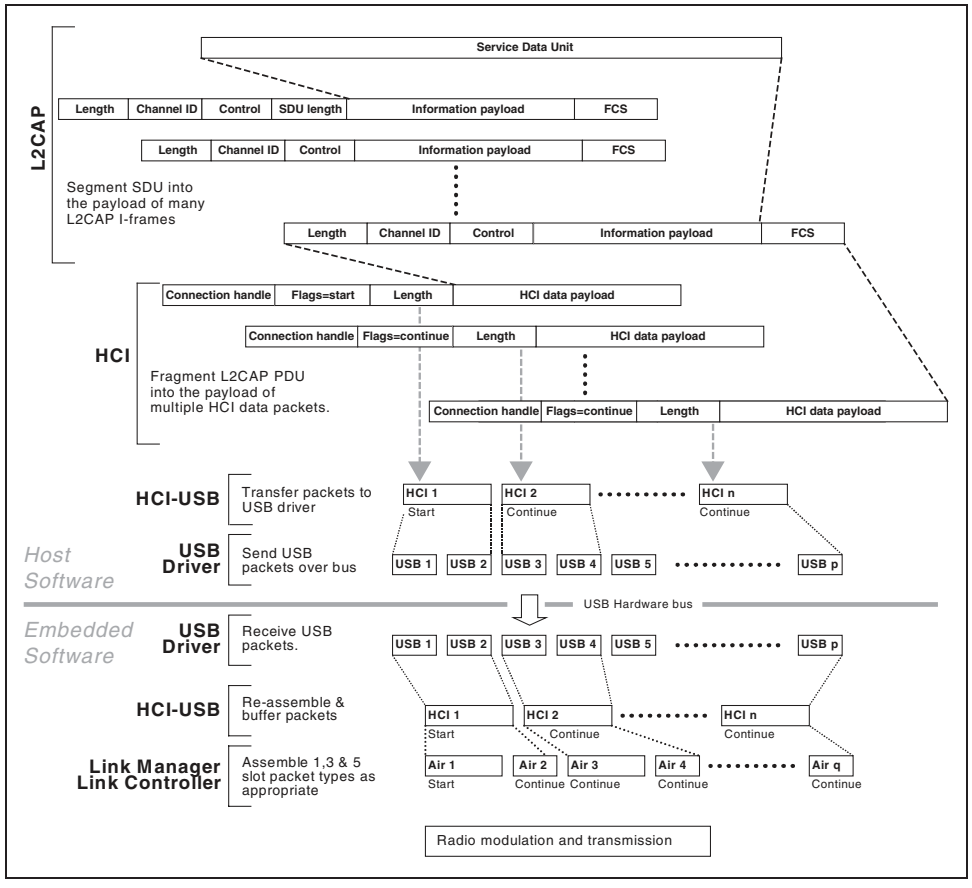



*Figure 7.4: Example of segmentation and fragment processes in a device with HCI*[1]

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



LifeScan, Inc. Ex. 1019
Page 1158 of 1420
IPR2025-00103

## 7.4 DELIVERY OF ERRONEOUS L2CAP SDUS

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

## 7.5 OPERATION WITH FLUSHING

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

When there is more than one L2CAP channel mapped to the same ACL logical transport, the automatic flush time-out does not discriminate between L2CAP channels. The exception is packets marked as non-automatically-flushable via the Packet_Boundary_Flag in the HCI ACL Data Packet (see section 1.1 on page 19). The automatic flush time-out flushes a specific automatically-flushable L2CAP PDU. The HCI Flush command flushes all outstanding L2CAP PDUs for the ACL logical transport including L2CAP PDUs marked as non-automatically-flushable. Therefore, care has to be taken when using the Automatic Flush Time-out and the HCI Flush command. The HCI Enhanced Flush command should be used instead:

1. For any connection to be reliable at the L2CAP level, it shall either mark all the packets associated with the reliable connection as non-automatically-flushable (if it is mapped to an ACL logical transport with a finite automatic flush time-out) or use L2CAP retransmission mode. In retransmission mode, loss of flushed L2CAP PDUs on the channel is detected by the L2CAP ARQ mechanism and they are retransmitted. Another approach is shown in 4 below. Note that L2CAP retransmission mode might be used for other purposes such as the need for a residual error rate that is much smaller than the baseband can deliver. In this situation both L2CAP retransmission mode and the Non-Flushable Packet Boundary Flag feature can be used at the same time.
2. There is only one automatic flush time-out setting per ACL logical transport. Therefore, all time bounded L2CAP channels on an ACL logical transport with an automatic flush time-out setting should configure the same flush time-out value at the L2CAP level.
3. If Automatic Flush Time-out is used, then it should be taken into account that it only flushes one L2CAP PDU. If one PDU has timed out and needs flushing, then other automatically-flushable packets on the same logical transport are also likely to need flushing. Therefore, flushing can be handled by the



LifeScan, Inc. Ex. 1019
Page 1159 of 1420
IPR2025-00103

HCI Enhanced Flush command so that all outstanding automatically-flushable L2CAP PDUs are flushed.

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

## 7.6 CONNECTIONLESS DATA CHANNEL

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

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

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

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

Connectionless data channels shall not be used with Retransmission Mode or Flow Control Mode.



LifeScan, Inc. Ex. 1019
Page 1160 of 1420
IPR2025-00103

# 8 PROCEDURES FOR FLOW CONTROL AND RETRANSMISSION

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

## 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 for the "Extended features supported" information type (0x0002). If the information retrieval is not successful or the "Extended features mask" bit is not set for the wanted mode, the mode shall not be suggested in a configuration request.

## 8.2 FUNCTION OF PDU TYPES FOR FLOW CONTROL AND RETRANSMISSION

Two frame formats are defined for flow control and retransmission modes (see Section 3.3 on page 33). The I-frame is used to transport user information instead of the B-frame. The S-frame is used for signaling.

### 8.2.1 Information frame (I-frame)

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

### 8.2.2 Supervisory Frame (S-frame)

The S-frame is used to control the transmission of I-frames. The S-frame has two formats: Receiver Ready (RR) and Reject (REJ).

#### *8.2.2.1 Receiver Ready (RR)*

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

1. Acknowledge I-frames numbered up to and including ReqSeq - 1.
2. Enable or disable retransmission of I-frames by updating the receiver with the current status of the Retransmission Disable Bit.

The RR frame has no information field.



LifeScan, Inc. Ex. 1019
Page 1161 of 1420
IPR2025-00103

#### 8.2.2.2 Reject (REJ)

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

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



LifeScan, Inc. Ex. 1019
Page 1162 of 1420
IPR2025-00103

## 8.3 VARIABLES AND SEQUENCE NUMBERS

The sending peer uses the following variables and Sequence numbers:

- TxSeq – the send Sequence number used to sequentially number each new I-frame transmitted.
- NextTxSeq – the Sequence number to be used in the next new I-frame transmitted.
- ExpectedAckSeq – the Sequence number of the next I-frame expected to be acknowledged by the receiving peer.

The receiving peer uses the following variables and Sequence numbers:

- ReqSeq – The Sequence number sent in an acknowledgement frame to request transmission of I-frame with TxSeq = ReqSeq and acknowledge receipt of I-frames up to and including (ReqSeq-1)
- ExpectedTxSeq – the value of TxSeq expected in the next I-frame.
- BufferSeq – When segmented I-frames are buffered this is used to delay acknowledgement of received I-frame so that new I-frame transmissions do not cause buffer overflow.

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

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

### 8.3.1 Sending peer

#### *8.3.1.1 Send sequence number TxSeq*

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

#### *8.3.1.2 Send state variable NextTXSeq*

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

The value of NextTxSeq shall be incremented by 1 after each in-sequence I-frame transmission, and shall not exceed ExpectedAckSeq by more than the maximum number of outstanding I-frames (TxWindow). The value of TxWindow shall be in the range 1 to 32.

#### *8.3.1.3 Acknowledge state variable ExpectedAckSeq*

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. An acknowledge state variable ExpectedAck-Seq shall be maintained for each remote endpoint. ExpectedAckSeq is the sequence number of the next in-sequence I-frame that the remote receiving peer is expected to acknowledge. (ExpectedAckSeq – 1 equals the TxSeq of the last acknowledged I-frame). When the link is created ExpectedAckSeq shall be initialized to 0.
Note that if the next acknowledgement acknowledges a single I-frame then it's 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-ExpectedAck-Seq) mod 64. Furthermore, from the description of NextTXSeq, it can be seen that (NextTXSeq-ExpectedAckSeq) mod 64 ≤ TxWindow.

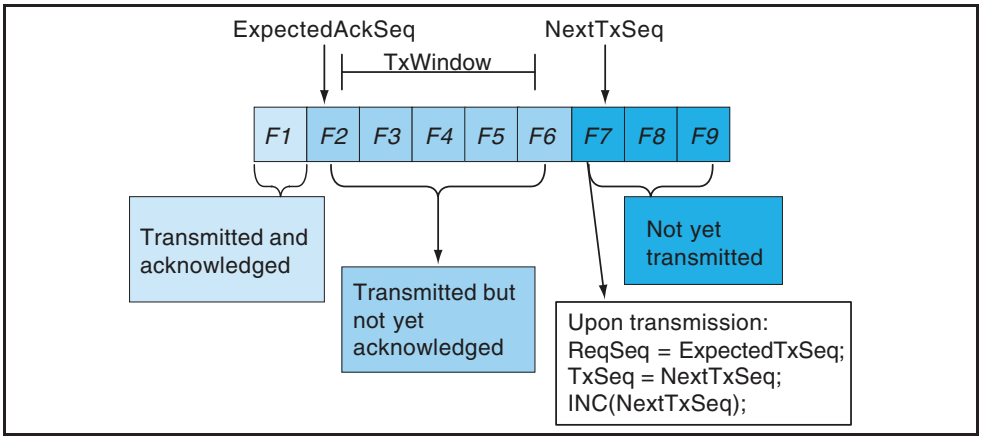



*Figure 8.1: Example of the transmitter side*

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

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



LifeScan, Inc. Ex. 1019
Page 1164 of 1420
IPR2025-00103

### 8.3.2 Receiving peer

#### *8.3.2.1 Receive sequence number ReqSeq*

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

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

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

#### *8.3.2.2 Receive state variable, ExpectedTxSeq*

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

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

#### *8.3.2.3 Buffer state variable BufferSeq*

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

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

I-Frames shall have sequence numbers in the range ExpectedTxSeq $\leq$ TxSeq $<$ (BufferSeq + TxWindow).



LifeScan, Inc. Ex. 1019
Page 1165 of 1420
IPR2025-00103

On receipt of an I-frame with TxSeq equal to ExpectedTxSeq, ExpectedTxSeq shall be incremented by 1 regardless of how many I-frames with TxSeq greater than ExpectedTxSeq were previously received.

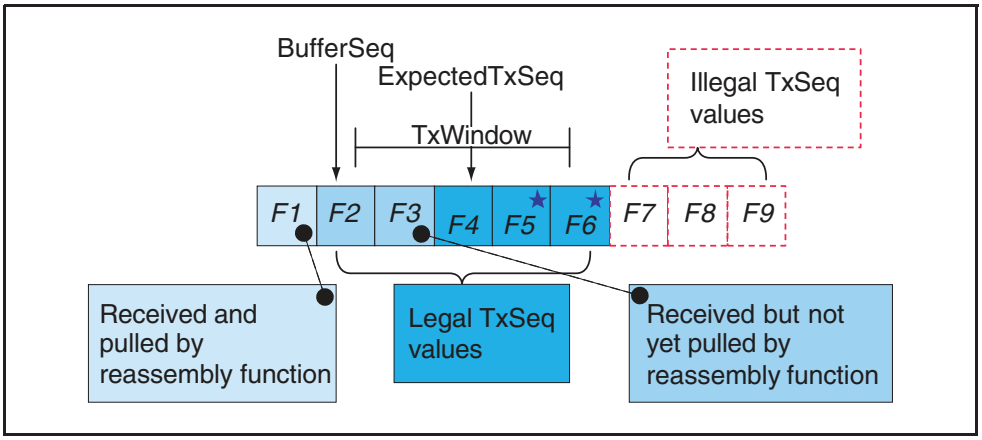



*Figure 8.2: Example of the receiver side*

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

In Figure 8.2 on page 90 there are several I-frames in a buffer awaiting the SDU reassembly function to pull them and the TxWindow is full. The receiver would usually disable retransmission by setting the Retransmission Disable Bit to 1 and send an RR back to the sending side. This tells the transmitting peer that there is no point in performing retransmissions. Both sides will send S-frames to make sure the peer entity knows the current value of the Retransmission Disable Bit.



LifeScan, Inc. Ex. 1019
Page 1166 of 1420
IPR2025-00103

## 8.4 RETRANSMISSION MODE

### 8.4.1 Transmitting frames

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

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

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

#### *8.4.1.1 Last received R was set to zero*

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

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

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

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

d) If no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted, and the RetransmissionTimer is running, then the retransmission timer shall be stopped and the monitor timer shall be started.



LifeScan, Inc. Ex. 1019
Page 1167 of 1420
IPR2025-00103

The table below summarizes actions when the RetransmissionTimer is in use and R=0.

| Unacknowledged I-frames sent = Retransmission Timer is running | Retrans-mission Timer has elapsed | New I-frames are waiting | Transmit Action | Timer Action |
|---|---|---|---|---|
| True | True | True or False | Retransmit un-acknowledged I-frame | Restart Retransmission Timer |
| True | False | True | Transmit new I-frame | No timer action |
| True | False | False | No transmit action | No Timer action |
| False | False | True | Transmit new I-frame | Restart Retransmis-sion Timer |
| False | False | False | No Transmit action | If MonitorTimer is not running then restart MonitorTimer |

*Table 8.1: Summary of actions when the RetransmissionTimer is in use and R=0.*

If the RetransmissionTimer is not in use, no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted

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

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

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

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



LifeScan, Inc. Ex. 1019
Page 1168 of 1420
IPR2025-00103

#### *8.4.1.2 Last received R was set to one*

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

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

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

### 8.4.2 Receiving I-frames

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

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

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

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

### 8.4.3 I-frames pulled by the SDU reassembly function

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

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

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

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

### 8.4.4 Sending and receiving acknowledgements

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

#### *8.4.4.1 Sending acknowledgements*

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

#### *8.4.4.2 Receiving acknowledgements*

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

The following rules shall be applied:

1. If the RetransmissionDisableBit changed value from 0 to 1 (stop retransmissions) then the receiving entity shall
    a) If the RetransmissionTimer is running then stop it and start the MonitorTimer.
    b) Store the state of the RetransmissionDisableBit received.
2. If the RetransmissionDisableBit changed value from 1 to 0 (start retransmissions) then the receiving entity shall
    a) Store the state of the RetransmissionDisableBit received.
    b) If there are any I-frames that have been sent but not acknowledged, then stop the MonitorTimer and start the RetransmissionTimer.
    c) Any buffered I-frames shall be transmitted according to Section 8.4.1 on page 91.
3. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame, and the RetransmissionDisableBit equals 1 (retransmissions stopped), then the receiving entity shall
    a) Follow the rules in Section 8.4.1 on page 91.
4. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame and the RetransmissionDisableBit equals 0 (retransmissions started) then the receiving entity shall
    a) If the RetransmissionTimer is running, then stop it.
    b) If any unacknowledged I-frames have been sent then the RetransmissionTimer shall be restarted.



LifeScan, Inc. Ex. 1019
Page 1170 of 1420
IPR2025-00103

c) Follow the rules in Section 8.4.1 on page 91.

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

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

### 8.4.5 Receiving REJ frames

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

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

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

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

### 8.4.6 Waiting acknowledgements

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

1. If the TransmitCounter is less than MaxTransmit then:
    a) Increment the TransmitCounter
    b) Retransmit the last unacknowledged I-frame, according to Section 8.4.1 on page 91.
2. If the TransmitCounter is equal to MaxTransmit this channel to the peer entity shall be assumed lost. The channel shall move to the CLOSED state and appropriate action shall be taken to report this to the upper layers.

### 8.4.7 Exception conditions

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

#### *8.4.7.1 TxSeq Sequence error*

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

The TxSeq sequence error may be due to three different causes:

- *Duplicated I-frame*

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

- *Out-of-sequence I-frame*

  The out-of-sequence I-frame is identified by a TxSeq within the legal range. The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4 on page 94.

  A REJ exception is triggered, and an REJ frame with ReqSeq equal to ExpectedTxSeq shall be sent to initiate recovery. The received I-frame shall be discarded.

- *Invalid TxSeq*

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

#### *8.4.7.2 ReqSeq Sequence error*

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

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

#### *8.4.7.3 Timer recovery error*

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

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

#### *8.4.7.4 Invalid frame*

Any frame received which is invalid (as defined in Section 3.3.6 on page 38) shall be discarded, and no action shall be taken as a result of that frame.

## 8.5 FLOW CONTROL MODE

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

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

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

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

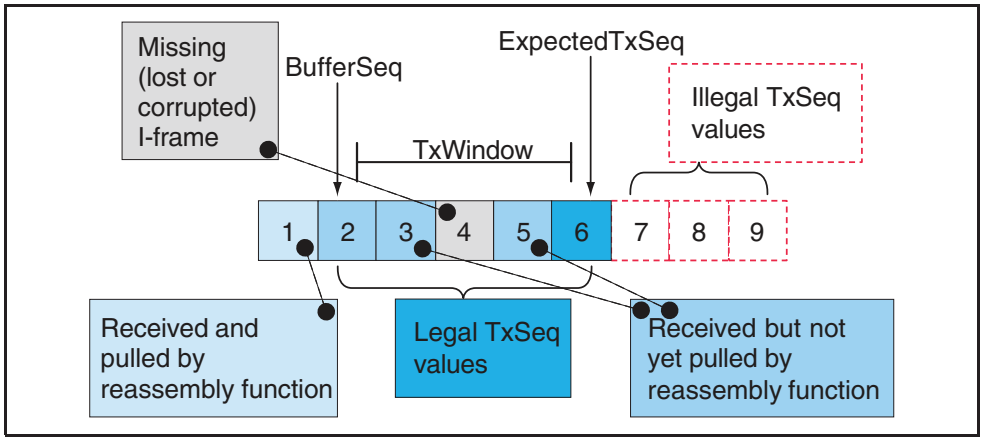



*Figure 8.3: Overview of the receiver side when operating in flow control mode*

### 8.5.1 Transmitting I-frames

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

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

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

The control field parameter ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTXSeq and NextTXSeq shall be incremented by one.



LifeScan, Inc. Ex. 1019
Page 1173 of 1420
IPR2025-00103

### 8.5.2 Receiving I-frames

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

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

### 8.5.3 I-frames pulled by the SDU reassembly function

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

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

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

### 8.5.4 Sending and receiving acknowledgements

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

#### *8.5.4.1 Sending acknowledgements*

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

#### *8.5.4.2 Receiving acknowledgements*

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

1. If any outstanding I-frames were acknowledged then
    a) Stop the RetransmissionTimer
    b) If there are still unacknowledged I-frames then restart the RetransmissionTimer, otherwise start the MonitorTimer.
    c) Transmit any I-frames awaiting transmission according to Section 8.5.1 on page 97.

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

### 8.5.5 Waiting acknowledgements

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

1. If I-frames are waiting to be sent
    a) The RetransmissionTimer is restarted
    b) I-frames awaiting transmission are transmitted according to Section 8.5.1 on page 97.
2. If there are no I-frames waiting to be sent
    a) If there are still unacknowledged I-frames the RetransmissionTimer is restarted, otherwise the MonitorTimer is started.

### 8.5.6 Exception conditions

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

#### *8.5.6.1 TxSeq Sequence error*

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

The TxSeq sequence error may be due to three different causes:

- *Duplicated I-frame*

  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to ExpectedTxSeq – 1. The ReqSeq shall be processed according to Section 8.5.4 on page 98. The Information field shall be discarded since it has already been received.

- *Out-of-sequence I-frame*

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

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

- *Invalid TxSeq*

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

#### 8.5.6.2 ReqSeq Sequence error

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

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

#### 8.5.6.3 Timer recovery error

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

#### 8.5.6.4 Invalid frame

Any frame received that is invalid (as defined in Section 3.3.6 on page 38) 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.

# 9 LIST OF FIGURES

Figure 1.1: L2CAP within protocol layers ....21
Figure 1.2: L2CAP architectural blocks ....22
Figure 2.1: Channels between devices ....30
Figure 2.2: L2CAP transaction model. ....31
Figure 3.1: PDU format in Basic L2CAP mode on connection-oriented channels (field sizes in bits) ....33
Figure 3.2: L2CAP PDU format in Basic L2CAP mode on Connectionless channel ....34
Figure 3.3: L2CAP PDU formats in Flow Control and Retransmission Modes ....35
Figure 3.4: The LFSR circuit generating the FCS. ....39
Figure 3.5: Initial state of the FCS generating circuit. ....39
Figure 4.1: L2CAP PDU format on the signaling channel ....41
Figure 4.2: Command format ....41
Figure 4.3: Command Reject packet ....43
Figure 4.4: Connection Request Packet ....44
Figure 4.5: Connection Response Packet ....46
Figure 4.6: Configuration Request Packet ....48
Figure 4.7: Configuration Request Flags field format ....48
Figure 4.8: Configuration Response Packet ....50
Figure 4.9: Configuration Response Flags field format ....50
Figure 4.10: Disconnection Request Packet ....52
Figure 4.11: Disconnection Response Packet ....53
Figure 4.12: Echo Request Packet ....53
Figure 4.13: Echo Response Packet ....54
Figure 4.14: Information Request Packet ....54
Figure 4.15: Information Response Packet ....55
Figure 5.1: Configuration option format ....57
Figure 5.2: MTU Option Format ....58
Figure 5.3: Flush Timeout option format. ....59
Figure 5.4: Quality of Service (QoS) option format containing Flow Specification. ....61
Figure 5.5: Retransmission and Flow Control option format. ....64
Figure 6.1: States and transitions. ....77
Figure 6.2: Configuration states and transitions. ....78
Figure 7.1: L2CAP fragmentation. ....81
Figure 7.2: Example of fragmentation processes in a device with HCI. ....82
Figure 7.3: Segmentation and fragmentation of an SDU. ....83
Figure 7.4: Example of segmentation and fragment processes in a device with HCI ....84
Figure 8.1: Example of the transmitter side ....90



LifeScan, Inc. Ex. 1019
Page 1177 of 1420
IPR2025-00103

Figure 8.2: Example of the receiver side ..... 92
Figure 8.3: Overview of the receiver side when operating in flow control mode ..... 99



LifeScan, Inc. Ex. 1019
Page 1178 of 1420
IPR2025-00103

# 10 LIST OF TABLES

Table 1.1: Terminology....26
Table 2.1: CID name space....29
Table 2.2: Types of Channel Identifiers....30
Table 3.1: Control Field formats....36
Table 3.2: SAR control element format....37
Table 3.3: S control element format: type of S-frame....37
Table 4.1: Signaling Command Codes....42
Table 4.2: Reason Code Descriptions....43
Table 4.3: Reason Data values....44
Table 4.4: Result values....46
Table 4.5: Status values....47
Table 4.6: Configuration Response Result codes....51
Table 4.7: InfoType definitions....54
Table 4.8: Information Response Result values....55
Table 4.9: Information Response Data fields....56
Table 4.10: Extended feature mask....56
Table 5.1: Service type definitions....61
Table 5.2: Mode definitions....64
Table 6.1: CLOSED state event table....68
Table 6.2: WAIT_CONNECT_RSP state event table....69
Table 6.3: WAIT_CONNECT state event table....69
Table 6.4: CONFIG state/substates event table: success transitions within configuration process.70
Table 6.5: CONFIG state/substates event table: non-success transitions within configuration process.71
Table 6.6: CONFIG state/substates event table: events not related to configuration process.72
Table 6.7: OPEN state event table....73
Table 6.8: WAIT_DISCONNECT state event table....73
Table 7.1: Parameters allowed in Request....80
Table 7.2: Parameters allowed in Response....80
Table 8.1: Summary of actions when the RetransmissionTimer is in use and R=0.94



LifeScan, Inc. Ex. 1019
Page 1179 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1180 of 1420
IPR2025-00103

# APPENDIX A: CONFIGURATION MSCs

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

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

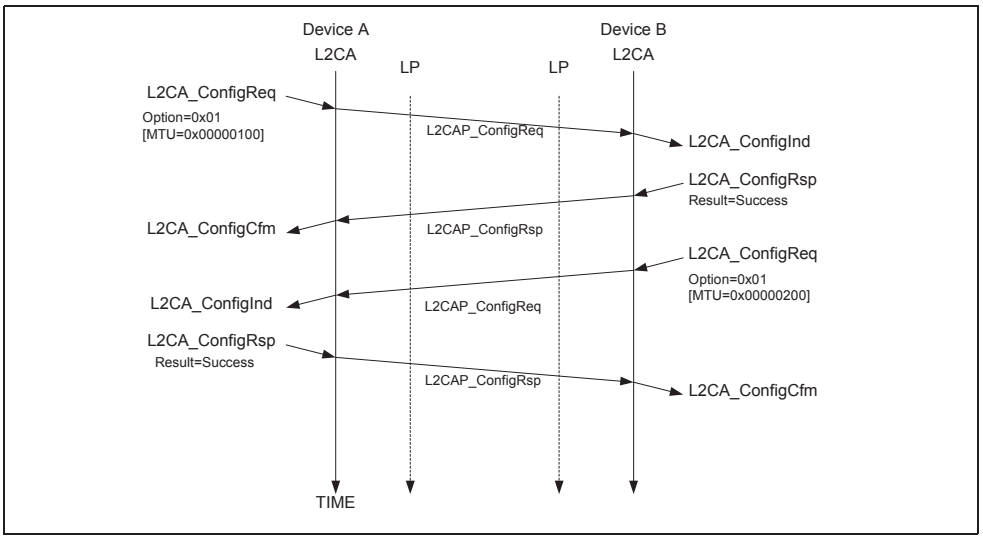



*Figure I: Basic MTU exchange*

Figure II on page 106 illustrates how two devices interoperate even though one device supports more options than the other does. Device A is an upgraded version. It uses a hypothetically defined option type 0x20 for link-level security. Device B rejects the command using the Configuration Response packet with result 'unknown parameter' informing Device A that option 0x20 is not understood. Device A then resends the request omitting option 0x20. Device B notices that it does not need to such a large MTU and accepts the request but includes in the response the MTU option informing Device A that Device B will not send an L2CAP packet with a payload larger than 0x80 octets over this channel. On receipt of the response, Device A could reduce the buffer allocated to hold incoming traffic.



LifeScan, Inc. Ex. 1019
Page 1181 of 1420
IPR2025-00103

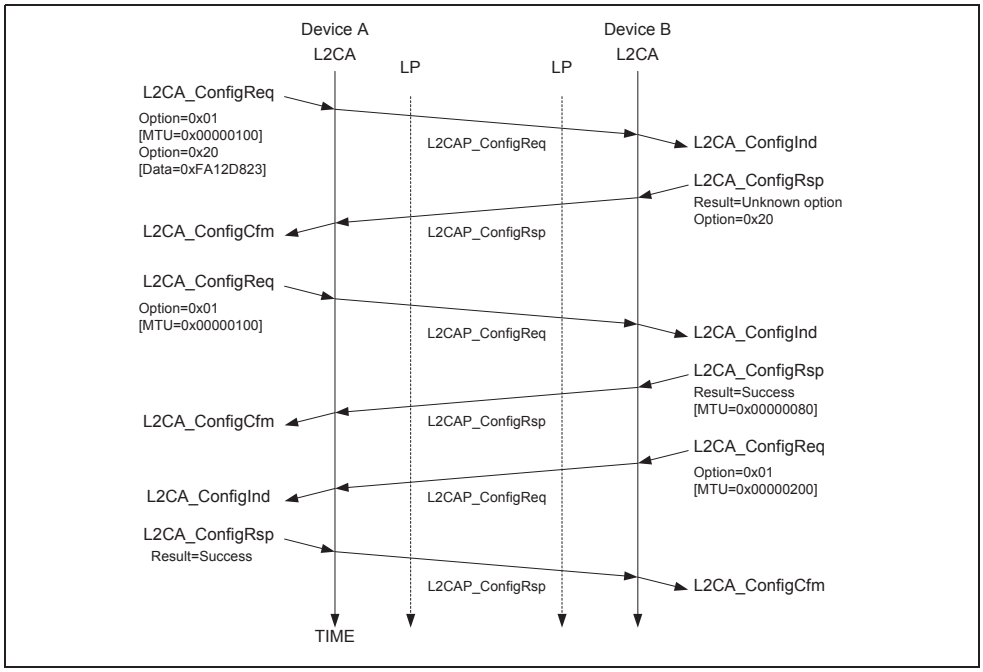



*Figure II: Dealing with Unknown Options*

Figure III on page 107 illustrates an unsuccessful configuration request. There are two problems described by this example. The first problem is that the configuration request is placed in an L2CAP packet that cannot be accepted by the remote device, due to its size. The remote device informs the sender of this problem using the Command Reject message. Device A then resends the configuration options using two smaller L2CAP_ConfigReq messages.

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



LifeScan, Inc. Ex. 1019
Page 1182 of 1420
IPR2025-00103

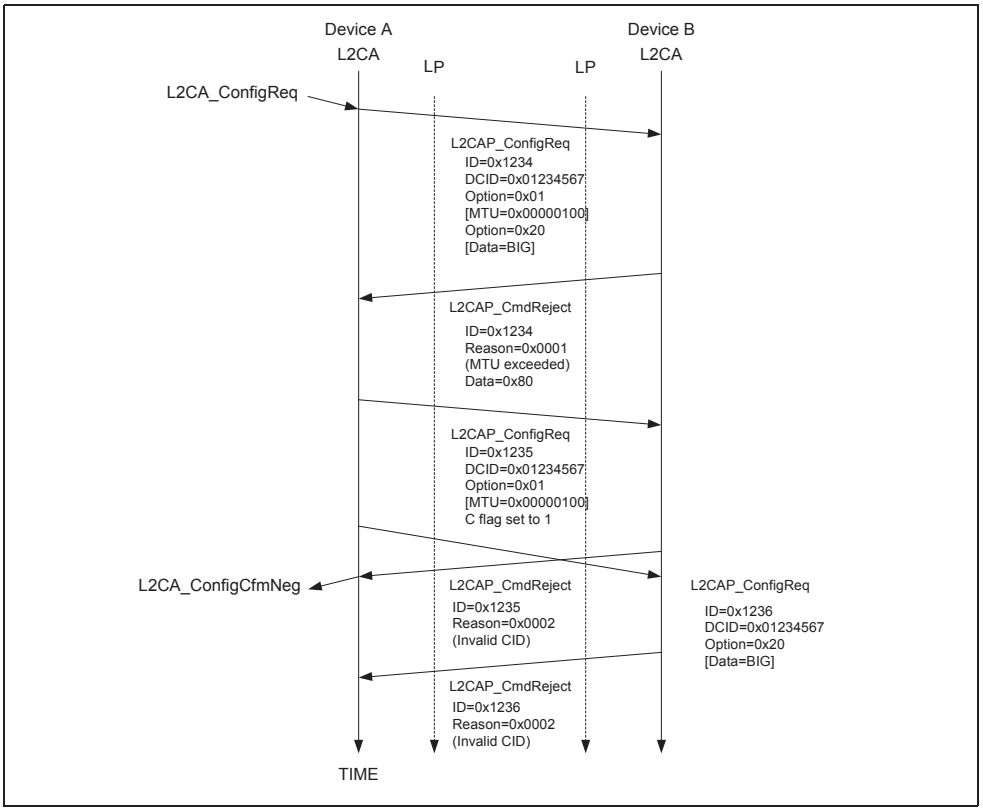



*Figure III: Unsuccessful Configuration Request*



LifeScan, Inc. Ex. 1019
Page 1183 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1184 of 1420
IPR2025-00103

# SERVICE DISCOVERY PROTOCOL (SDP)

***This specification defines a protocol for locating services provided by or available through a Bluetooth device.***



LifeScan, Inc. Ex. 1019
Page 1185 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1186 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 113**
1.1 General Description ..... 113
1.2 Motivation ..... 113
1.3 Requirements ..... 113
1.4 Non-requirements and Deferred Requirements ..... 114
1.5 Conventions ..... 114
1.5.1 Bit And Byte Ordering Conventions ..... 114

**2 Overview ..... 115**
2.1 SDP Client-Server Interaction ..... 115
2.2 Service Record ..... 116
2.3 Service Attribute ..... 117
2.3.1 Attribute ID ..... 118
2.3.2 Attribute Value ..... 119
2.4 Service Class ..... 119
2.4.1 A Printer Service Class Example ..... 119
2.5 Searching for Services ..... 121
2.5.1 UUID ..... 121
2.5.2 Service Search Patterns ..... 122
2.6 Browsing for Services ..... 122
2.6.1 Example Service Browsing Hierarchy ..... 123

**3 Data Representation ..... 125**
3.1 Data Element ..... 125
3.2 Data Element Type Descriptor ..... 125
3.3 Data Element Size Descriptor ..... 126
3.4 Data Element Examples ..... 127

**4 Protocol Description ..... 128**
4.1 Transfer Byte Order ..... 128
4.2 Protocol Data Unit Format ..... 128
4.3 Partial Responses and Continuation State ..... 130
4.4 Error Handling ..... 130
4.4.1 SDP_ErrorResponse PDU ..... 131
4.5 ServiceSearch Transaction ..... 131
4.5.1 SDP_ServiceSearchRequest PDU ..... 131
4.5.2 SDP_ServiceSearchResponse PDU ..... 133
4.6 ServiceAttribute Transaction ..... 135
4.6.1 SDP_ServiceAttributeRequest PDU ..... 135
4.6.2 SDP_ServiceAttributeResponse PDU ..... 137
4.7 ServiceSearchAttribute Transaction ..... 138



LifeScan, Inc. Ex. 1019
Page 1187 of 1420
IPR2025-00103

4.7.1 SDP_ServiceSearchAttributeRequest PDU ..... 138
4.7.2 SDP_ServiceSearchAttributeResponse PDU ..... 140

**5 Service Attribute Definitions ..... 142**
5.1 Universal Attribute Definitions ..... 142
5.1.1 ServiceRecordHandle Attribute ..... 142
5.1.2 ServiceClassIDList Attribute ..... 143
5.1.3 ServiceRecordState Attribute ..... 143
5.1.4 ServiceID Attribute ..... 143
5.1.5 ProtocolDescriptorList Attribute ..... 144
5.1.6 AdditionalProtocolDescriptorList Attribute ..... 145
5.1.7 BrowseGroupList Attribute ..... 146
5.1.8 LanguageBaseAttributeIDList Attribute ..... 146
5.1.9 ServiceInfoTimeToLive Attribute ..... 147
5.1.10 ServiceAvailability Attribute ..... 148
5.1.11 BluetoothProfileDescriptorList Attribute ..... 148
5.1.12 DocumentationURL Attribute ..... 149
5.1.13 ClientExecutableURL Attribute ..... 149
5.1.14 IconURL Attribute ..... 150
5.1.15 ServiceName Attribute ..... 150
5.1.16 ServiceDescription Attribute ..... 151
5.1.17 ProviderName Attribute ..... 151
5.1.18 Reserved Universal Attribute IDs ..... 151
5.2 ServiceDiscoveryServer Service Class Attribute Definitions ... 152
5.2.1 ServiceRecordHandle Attribute ..... 152
5.2.2 ServiceClassIDList Attribute ..... 152
5.2.3 VersionNumberList Attribute ..... 152
5.2.4 ServiceDatabaseState Attribute ..... 153
5.2.5 Reserved Attribute IDs ..... 153
5.3 BrowseGroupDescriptor Service Class Attribute Definitions ... 153
5.3.1 ServiceClassIDList Attribute ..... 153
5.3.2 GroupID Attribute ..... 154
5.3.3 Reserved Attribute IDs ..... 154

**6 Security ..... 155**

**7 Appendix ..... 156**



LifeScan, Inc. Ex. 1019
Page 1188 of 1420
IPR2025-00103

# 1 INTRODUCTION

## 1.1 GENERAL DESCRIPTION

The service discovery protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services.

## 1.2 MOTIVATION

Service Discovery in the Bluetooth environment, where the set of services that are available changes dynamically based on the RF proximity of devices in motion, is qualitatively different from service discovery in traditional network-based environments. The service discovery protocol defined in this specification is intended to address the unique characteristics of the Bluetooth environment. See section , “Appendix A – Background Information,” on page 157, for further information on this topic.

## 1.3 REQUIREMENTS

The following capabilities have been identified as requirements for version 1.0 of the Service Discovery Protocol.

1. SDP shall provide the ability for clients to search for needed services based on specific attributes of those services.
2. SDP shall permit services to be discovered based on the class of service.
3. SDP shall enable browsing of services without a priori knowledge of the specific characteristics of those services.
4. SDP shall provide the means for the discovery of new services that become available when devices enter RF proximity with a client device as well as when a new service is made available on a device that is in RF proximity with the client device.
5. SDP shall provide a mechanism for determining when a service becomes unavailable when devices leave RF proximity with a client device as well as when a service is made unavailable on a device that is in RF proximity with the client device.
6. SDP shall provide for services, classes of services, and attributes of services to be uniquely identified.
7. SDP shall allow a client on one device to discover a service on another device without consulting a third device.
8. SDP should be suitable for use on devices of limited complexity.
9. SDP shall provide a mechanism to incrementally discover information about the services provided by a device. This is intended to minimize the quantity



LifeScan, Inc. Ex. 1019
Page 1189 of 1420
IPR2025-00103

of data that must be exchanged in order to determine that a particular service is not needed by a client.

10. SDP should support the caching of service discovery information by intermediary agents to improve the speed or efficiency of the discovery process.
11. SDP should be transport independent.
12. SDP shall function while using L2CAP as its transport protocol.
13. SDP shall permit the discovery and use of services that provide access to other service discovery protocols.
14. SDP shall support the creation and definition of new services without requiring registration with a central authority.

## 1.4 NON-REQUIREMENTS AND DEFERRED REQUIREMENTS

The Bluetooth SIG recognizes that the following capabilities are related to service discovery. These items are not addressed in SDP version 1.0. However, some may be addressed in future revisions of the specification.

1. SDP 1.0 does not provide access to services. It only provides access to information about services.
2. SDP 1.0 does not provide brokering of services.
3. SDP 1.0 does not provide for negotiation of service parameters.
4. SDP 1.0 does not provide for billing of service use.
5. SDP 1.0 does not provide the means for a client to control or change the operation of a service.
6. SDP 1.0 does not provide an event notification when services, or information about services, become unavailable.
7. SDP 1.0 does not provide an event notification when attributes of services are modified.
8. This specification does not define an application programming interface for SDP.
9. SDP 1.0 does not provide support for service agent functions such as service aggregation or service registration.

## 1.5 CONVENTIONS

### 1.5.1 Bit And Byte Ordering Conventions

When multiple bit fields are contained in a single byte and represented in a drawing in this specification, the more significant (high-order) bits are shown toward the left and less significant (low-order) bits toward the right.

Multiple-byte fields are drawn with the more significant bytes toward the left and the less significant bytes toward the right. Multiple-byte fields are transferred in network byte order. See Section 4.1 on page 128.



LifeScan, Inc. Ex. 1019
Page 1190 of 1420
IPR2025-00103

# 2 OVERVIEW

## 2.1 SDP CLIENT-SERVER INTERACTION

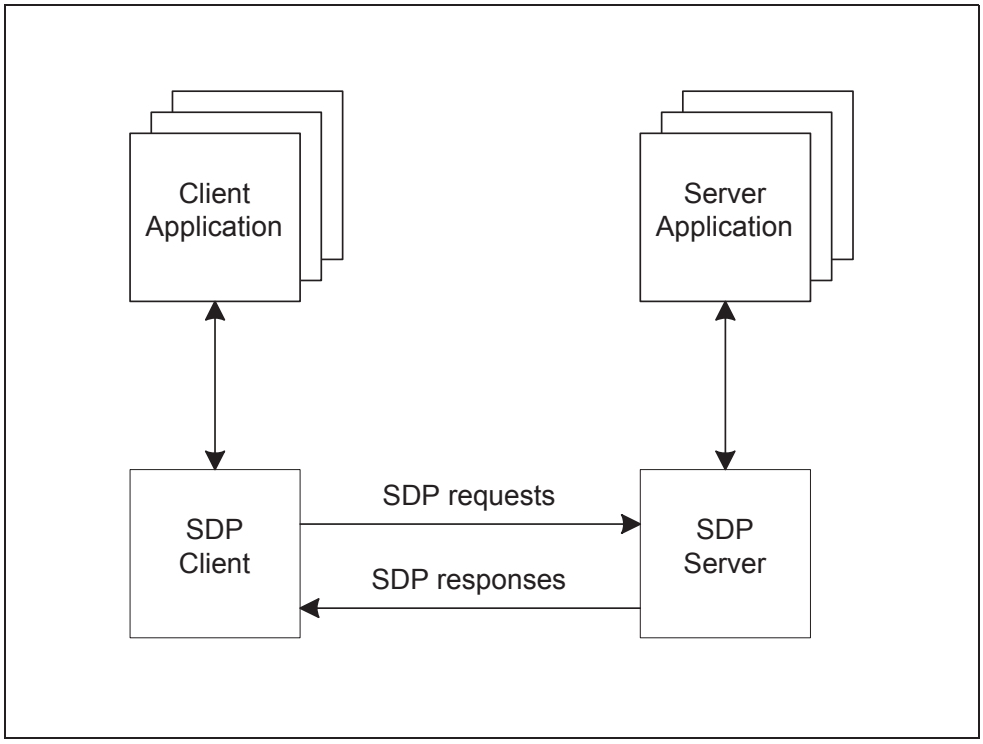



*Figure 2.1:*

The service discovery mechanism provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service.

As far as the Service Discovery Protocol (SDP) is concerned, the configuration shown in Figure 2.1 can be simplified to that shown in Figure 2.2.

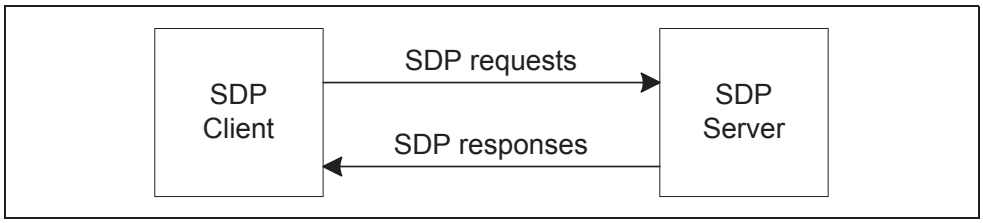



*Figure 2.2:*



LifeScan, Inc. Ex. 1019
Page 1191 of 1420
IPR2025-00103

SDP involves communication between an SDP server and an SDP client. The server maintains a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client can retrieve information from a service record maintained by the SDP server by issuing an SDP request.

If the client, or an application associated with the client, decides to use a service, it opens a separate connection to the service provider in order to utilize the service. SDP provides a mechanism for discovering services and their attributes (including associated service access protocols), but it does not provide a mechanism for utilizing those services (such as delivering the service access protocols).

There shall be a maximum of one active SDP server per Bluetooth device. If a Bluetooth device acts only as a client, it needs no SDP server. A single Bluetooth device can function both as an SDP server and as an SDP client. If multiple applications on a device provide services, an SDP server can act on behalf of those service providers to handle requests for information about the services that they provide.

Similarly, multiple client applications can utilize an SDP client to query servers on behalf of the client applications.

The set of SDP servers that are available to an SDP client will change dynamically based on the RF proximity of the servers to the client. When a server becomes available, a potential client must be notified by a means other than SDP so that the client can use SDP to query the server about its services. Similarly, when a server leaves proximity or becomes unavailable for any reason, there is no explicit notification via the service discovery protocol. However, the client can use SDP to poll the server and may infer that the server is not available if it no longer responds to requests.

Additional information regarding application interaction with SDP shall be contained in the Bluetooth Service Discovery Profile document.

## 2.2 SERVICE RECORD

A service is any entity that can provide information, perform an action, or control a resource on behalf of another entity. A service may be implemented as software, hardware, or a combination of hardware and software.

All of the information about a service that is maintained by an SDP server is contained within a single service record. The service record shall only be a list of service attributes.



LifeScan, Inc. Ex. 1019
Page 1192 of 1420
IPR2025-00103

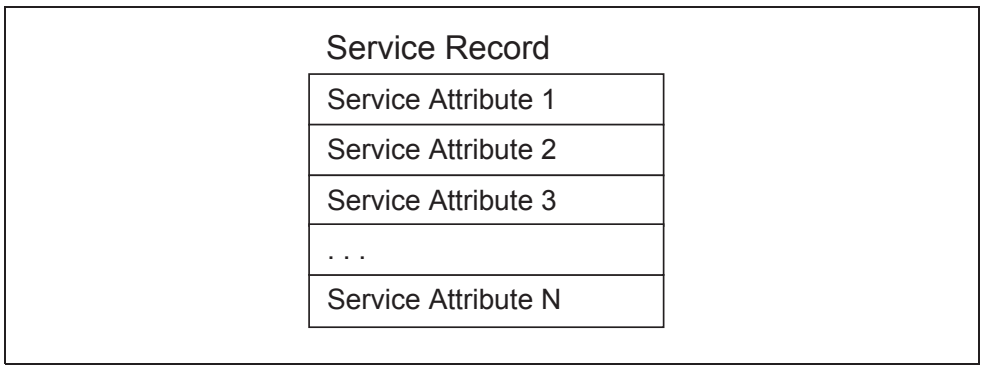



*Figure 2.3: Service Record*

A service record handle is a 32-bit number that shall uniquely identiy each service record within an SDP server. It is important to note that, in general, each handle is unique only within each SDP server. If SDP server S1 and SDP server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on S1 will be meaningless if presented to S2.

The service discovery protocol does not provide a mechanism for notifying clients when service records are added to or removed from an SDP server. While an L2CAP (Logical Link Control and Adaptation Protocol) connection is established to a server, a service record handle acquired from the server shall remain valid unless the service record it represents is removed. If a service is removed from the server, further requests to the server (during the L2CAP connection in which the service record handle was acquired) using the service's (now stale) record handle shall result in an error response indicating an invalid service record handle. An SDP server shall ensure that no service record handle values are re-used while an L2CAP connection remains established. Service record handles shall remain valid across successive L2CAP connections while the ServiceDatabaseState attribute value remains unchanged. Further, service record handles should remain valid until such time that the corresponding service is permanently removed or changes in an incompatible way. See the ServiceRecordState and ServiceDatabaseState attributes in Section 5 on page 142.

A device may have a service record with a service record handle of 0x00000000 representing the SDP server itself. This service record contains attributes for the SDP server and the protocol it supports. For example, one of its attributes is the list of SDP protocol versions supported by the server.

## 2.3 SERVICE ATTRIBUTE

Each service attribute describes a single characteristic of a service. Some examples of service attributes are:



LifeScan, Inc. Ex. 1019
Page 1193 of 1420
IPR2025-00103

| | |
|---|---|
| ServiceClassIDList | Identifies the type of service represented by a service record. In other words, the list of classes of which the service is an instance |
| ServiceID | Uniquely identifies a specific instance of a service |
| ProtocolDescriptorList | Specifies the protocol stack(s) that may be used to utilize a service |
| ProviderName | The textual name of the individual or organization that provides a service |
| IconURL | Specifies a URL that refers to an icon image that may be used to represent a service |
| ServiceName | A text string containing a human-readable name for the service |
| ServiceDescription | A text string describing the service |

See Section 5.1 on page 142, for attribute definitions that are common to all service records. Service providers can also define their own service attributes.

A service attribute consists of two components: an attribute ID and an attribute value.

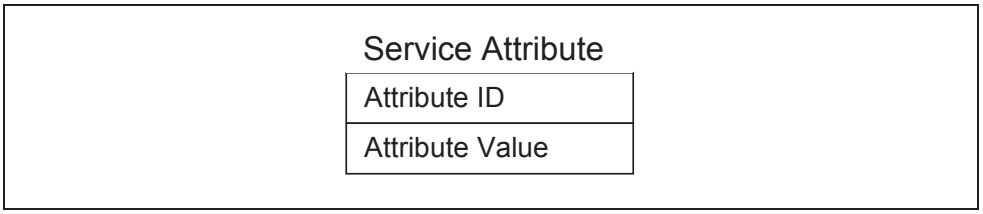



*Figure 2.4: Service Attribute*

### 2.3.1 Attribute ID

An attribute ID is a 16-bit unsigned integer that distinguishes each service attribute from other service attributes within a service record. The attribute ID also identifies the semantics of the associated attribute value.

A service class definition specifies each of the attribute IDs for a service class and assigns a meaning to the attribute value associated with each attribute ID. Each attribute ID is defined to be unique only within each service class.

All services belonging to a given service class assign the same meaning to each particular attribute ID. See Section 2.4 on page 119.

In the Service Discovery Protocol, an attribute ID is represented as a data element. See Section 3 on page 125.



LifeScan, Inc. Ex. 1019
Page 1194 of 1420
IPR2025-00103

### 2.3.2 Attribute Value

The attribute value is a variable length field whose meaning is determined by the attribute ID associated with it and by the service class of the service record in which the attribute is contained. In the Service Discovery Protocol, an attribute value is represented as a data element. (See Section 3 on page 125.) Generally, any type of data element is permitted as an attribute value, subject to the constraints specified in the service class definition that assigns an attribute ID to the attribute and assigns a meaning to the attribute value. See Section 5 on page 142, for attribute value examples.

## 2.4 SERVICE CLASS

Each service is an instance of a service class. The service class definition provides the definitions of all attributes contained in service records that represent instances of that class. Each attribute definition specifies the numeric value of the attribute ID, the intended use of the attribute value, and the format of the attribute value. A service record contains attributes that are specific to a service class as well as universal attributes that are common to all services.

Each service class is also assigned a unique identifier. This service class identifier is contained in the attribute value for the ServiceClassIDList attribute, and is represented as a UUID (see Section 2.5.1 on page 121). Since the format and meanings of many attributes in a service record are dependent on the service class of the service record, the ServiceClassIDList attribute is very important. Its value shall be examined or verified before any class-specific attributes are used. Since all of the attributes in a service record must conform to all of the service's classes, the service class identifiers contained in the ServiceClassIDList attribute are related. Typically, each service class is a subclass of another class whose identifier is contained in the list. A service subclass definition differs from its superclass in that the subclass contains additional attribute definitions that are specific to the subclass. The service class identifiers in the ServiceClassIDList attribute should be listed in order from the most specific class to the most general class.

When a new service class is defined that is a subclass of an existing service class, the new service class retains all of the attributes defined in its superclass. Additional attributes may be defined that are specific to the new service class. In other words, the mechanism for adding new attributes to some of the instances of an existing service class is to create a new service class that is a subclass of the existing service class.

### 2.4.1 A Printer Service Class Example

A color postscript printer with duplex capability might conform to 4 ServiceClass definitions and have a ServiceClassIDList with UUIDs (See Section 2.5.1 on page 121.) representing the following ServiceClasses:



LifeScan, Inc. Ex. 1019
Page 1195 of 1420
IPR2025-00103

DuplexColorPostscriptPrinterServiceClassID,
ColorPostscriptPrinterServiceClassID,
PostscriptPrinterServiceClassID,
PrinterServiceClassID

*Note that this example is only illustrative. This may not be a practical printer class hierarchy.*



LifeScan, Inc. Ex. 1019
Page 1196 of 1420
IPR2025-00103

## 2.5 SEARCHING FOR SERVICES

The Service Search transaction allows a client to retrieve the service record handles for particular service records based on the values of attributes contained within those service records. Once an SDP client has a service record handle, it can request the values of specific attributes.

The capability search for service records based on the values of arbitrary attributes is not provided. Rather, the capability is provided to search only for attributes whose values are Universally Unique Identifiers[1] (UUIDs). Important attributes of services that can be used to search for a service are represented as UUIDs.

### 2.5.1 UUID

A UUID is a universally unique identifier that is guaranteed to be unique across all space and all time. UUIDs can be independently created in a distributed fashion. No central registry of assigned UUIDs is required. A UUID is a 128-bit value.

To reduce the burden of storing and transferring 128-bit UUID values, a range of UUID values has been pre-allocated for assignment to often-used, registered purposes. The first UUID in this pre-allocated range is known as the Bluetooth Base UUID and has the value 00000000-0000-1000-8000-00805F9B34FB, from the Bluetooth Assigned Numbers document. UUID values in the pre-allocated range have aliases that are represented as 16-bit or 32-bit values. These aliases are often called 16-bit and 32-bit UUIDs, but it is important to note that each actually represents a 128-bit UUID value.

The full 128-bit value of a 16-bit or 32-bit UUID may be computed by a simple arithmetic operation.

128_bit_value = 16_bit_value * $2^{96}$ + Bluetooth_Base_UUID

128_bit_value = 32_bit_value * $2^{96}$ + Bluetooth_Base_UUID

A 16-bit UUID may be converted to 32-bit UUID format by zero-extending the 16-bit value to 32-bits. An equivalent method is to add the 16-bit UUID value to a zero-valued 32-bit UUID.

Note that two 16-bit UUIDs may be compared directly, as may two 32-bit UUIDs or two 128-bit UUIDs. If two UUIDs of differing sizes are to be compared, the shorter UUID must be converted to the longer UUID format before comparison.

1. The format of UUIDs is defined by the International Organization for Standardization in ISO/IEC 11578:1996. “Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)”



LifeScan, Inc. Ex. 1019
Page 1197 of 1420
IPR2025-00103

### 2.5.2 Service Search Patterns

A service search pattern is a list of UUIDs used to locate matching service records. A service search pattern matches a service record if each and every UUID in the service search pattern is contained within any of the service record's attribute values. The UUIDs need not be contained within any specific attributes or in any particular order within the service record. The service search pattern matches if the UUIDs it contains constitute a subset of the UUIDs in the service record's attribute values. The only time a service search pattern does not match a service record is if the service search pattern contains at least one UUID that is not contained within the service record's attribute values. Note also that a valid service search pattern must contain at least one UUID.

## 2.6 BROWSING FOR SERVICES

Normally, a client searches for services based on some desired characteristic(s) (represented by a UUID) of the services. However, there are times when it is desirable to discover which types of services are described by an SDP server's service records without any a priori information about the services. This process of looking for any offered services is termed browsing. In SDP, the mechanism for browsing for services is based on an attribute shared by all service classes. This attribute is called the BrowseGroupList attribute. The value of this attribute contains a list of UUIDs. Each UUID represents a browse group with which a service may be associated for the purpose of browsing.

When a client desires to browse an SDP server's services, it creates a service search pattern containing the UUID that represents the root browse group. All services that may be browsed at the top level are made members of the root browse group by having the root browse group's UUID as a value within the BrowseGroupList attribute.

Normally, if an SDP server has relatively few services, all of its services will be placed in the root browse group. However, the services offered by an SDP server may be organized in a browse group hierarchy, by defining additional browse groups below the root browse group. Each of these additional browse groups is described by a service record with a service class of BrowseGroupDescriptor.

A browse group descriptor service record defines a new browse group by means of its Group ID attribute. In order for a service contained in one of these newly defined browse groups to be browseable, the browse group descriptor service record that defines the new browse group must in turn be browseable. The hierarchy of browseable services that is provided by the use of browse group descriptor service records allows the services contained in an SDP server to be incrementally browsed and is particularly useful when the SDP server contains many service records.



LifeScan, Inc. Ex. 1019
Page 1198 of 1420
IPR2025-00103

### 2.6.1 Example Service Browsing Hierarchy

Here is a fictitious service browsing hierarchy that illuminates the manner in which browse group descriptors are used. Browse group descriptor service records are identified with (G); other service records with (S).

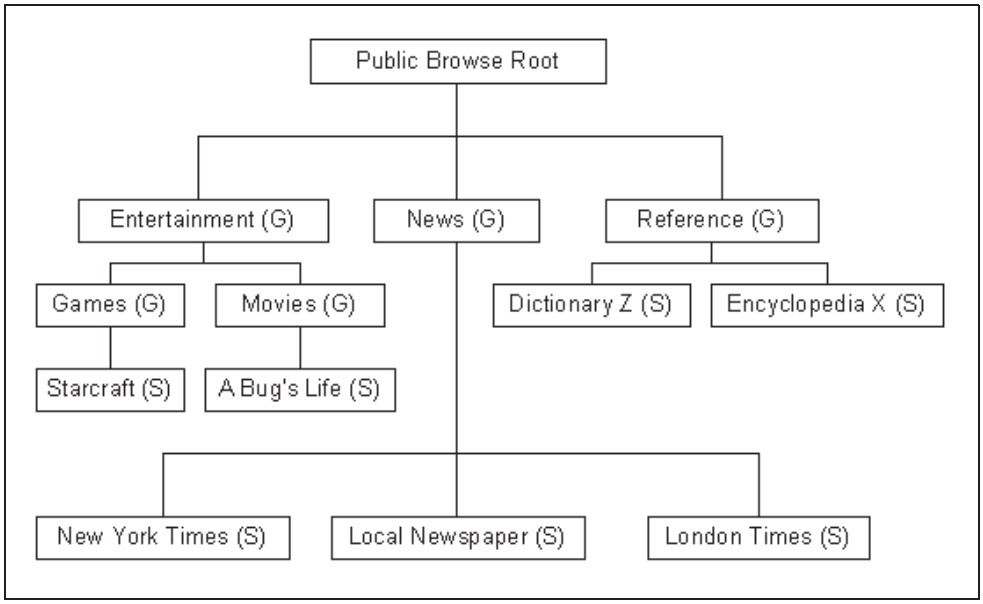



*Figure 2.5:*



LifeScan, Inc. Ex. 1019
Page 1199 of 1420
IPR2025-00103

This table shows the services records and service attributes necessary to implement the browse hierarchy.

| Service Name | Service Class | Attribute Name | Attribute Value |
|---|---|---|---|
| Entertainment | BrowseGroupDescriptor | BrowseGroupList | PublicBrowseRoot |
| | | GroupID | EntertainmentID |
| News | BrowsegroupDescriptor | BrowseGroupList | PublicBrowseRoot |
| | | GroupID | NewsID |
| Reference | BrowseGroupDescriptor | BrowseGroupList | PublicBrowseRoot |
| | | GroupID | ReferenceID |
| Games | BrowseGroupDescriptor | BrowseGroupList | EntertainmentID |
| | | GroupID | GamesID |
| Movies | BrowseGroupDescriptor | BrowseGroupList | EntertainmentID |
| | | GroupID | MoviesID |
| Starcraft | Video Game Class ID | BrowseGroupList | GamesID |
| A Bug's Life | Movie Class ID | BrowseGroupList | MovieID |
| Dictionary Z | Dictionary Class ID | BrowseGroupList | ReferenceID |
| Encyclopedia X | Encyclopedia Class ID | BrowseGroupList | ReferenceID |
| New York Times | Newspaper ID | BrowseGroupList | NewspaperID |
| London Times | Newspaper ID | BrowseGroupList | NewspaperID |
| Local Newspaper | Newspaper ID | BrowseGroupList | NewspaperID |

*Table 2.1:*



LifeScan, Inc. Ex. 1019
Page 1200 of 1420
IPR2025-00103

# 3 DATA REPRESENTATION

Attribute values can contain information of various types with arbitrary complexity; thus enabling an attribute list to be generally useful across a wide variety of service classes and environments.

SDP defines a simple mechanism to describe the data contained within an attribute ID, attribute ID range, and attribute value. The primitive construct used is the data element.

## 3.1 DATA ELEMENT

A data element is a typed data representation. It consists of two fields: a header field and a data field. The header field, in turn, is composed of two parts: a type descriptor and a size descriptor. The data is a sequence of bytes whose length is specified in the size descriptor (described in Section 3.3 on page 126) and whose meaning is (partially) specified by the type descriptor.

## 3.2 DATA ELEMENT TYPE DESCRIPTOR

A data element type is represented as a 5-bit type descriptor. The type descriptor is contained in the most significant (high-order) 5 bits of the first byte of the data element header. The following types have been defined.

| Type Descriptor Value | Valid Size Descriptor Values | Type Description |
|---|---|---|
| 0 | 0 | Nil, the null type |
| 1 | 0, 1, 2, 3, 4 | Unsigned Integer |
| 2 | 0, 1, 2, 3, 4 | Signed twos-complement integer |
| 3 | 1, 2, 4 | UUID, a universally unique identifier |
| 4 | 5, 6, 7 | Text string |
| 5 | 0 | Boolean[1] |
| 6 | 5, 6, 7 | Data element sequence, a data element whose data field is a sequence of data elements |
| 7 | 5, 6, 7 | Data element alternative, data element whose data field is a sequence of data elements from which one data element is to be selected. |
| 8 | 5, 6, 7 | URL, a uniform resource locator |
| 9-31 | | Reserved |

*Table 3.1: Data Element Type.*



LifeScan, Inc. Ex. 1019
Page 1201 of 1420
IPR2025-00103

1. False is represented by the value 0, and true is represented by the value 1. However, to maximize interoperability, any non-zero value received must be accepted as representing true.

## 3.3 DATA ELEMENT SIZE DESCRIPTOR

The data element size descriptor is represented as a 3-bit size index followed by 0, 8, 16, or 32 bits. The size index is contained in the least significant (low-order) 3 bits of the first byte of the data element header. The size index is encoded as follows.

| Size Index | Additional bits | Data Size |
|---|---|---|
| 0 | 0 | 1 byte. Exception: if the data element type is nil, the data size is 0 bytes. |
| 1 | 0 | 2 bytes |
| 2 | 0 | 4 bytes |
| 3 | 0 | 8 bytes |
| 4 | 0 | 16 bytes |
| 5 | 8 | The data size is contained in the additional 8 bits, which are interpreted as an unsigned integer. |
| 6 | 16 | The data size is contained in the additional 16 bits, which are interpreted as an unsigned integer. |
| 7 | 32 | The data size is contained in the additional 32 bits, which are interpreted as an unsigned integer. |

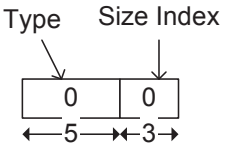
*Table 3.2:*



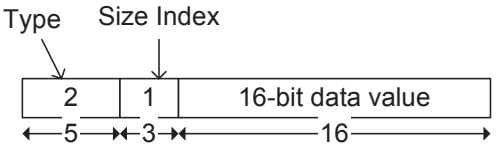
LifeScan, Inc. Ex. 1019
Page 1202 of 1420
IPR2025-00103

## 3.4 DATA ELEMENT EXAMPLES

Nil is represented as:




A 16-bit signed integer is represented as:




The 3 character ASCII string "Hat" is represented as:

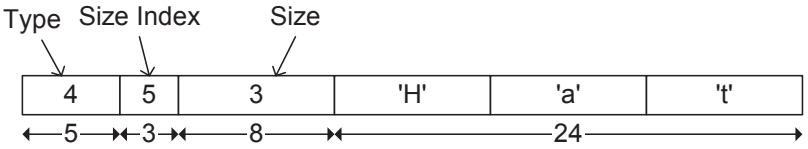



*Figure 3.1:*



LifeScan, Inc. Ex. 1019
Page 1203 of 1420
IPR2025-00103

# 4 PROTOCOL DESCRIPTION

SDP is a simple protocol with minimal requirements on the underlying transport. It can function over a reliable packet transport (or even unreliable, if the client implements timeouts and repeats requests as necessary).

SDP uses a request/response model where each transaction consists of one request protocol data unit (PDU) and one response PDU. In the case where SDP is used with the Bluetooth L2CAP transport protocol, no more than one SDP request PDU per connection to a given SDP server may be outstanding at a given instant. In other words, a client does not issue a further request to a server prior to receiving a response to the current request before issuing another request on the same L2CAP connection. Limiting SDP to sending one unacknowledged request PDU provides a simple form of flow control.

The protocol examples found in Appendix B – Example SDP Transactions, could be helpful in understanding the protocol transactions.

## 4.1 TRANSFER BYTE ORDER

The service discovery protocol shall transfer multiple-byte fields in standard network byte order (Big Endian), with more significant (high-order) bytes being transferred before less-significant (low-order) bytes.

## 4.2 PROTOCOL DATA UNIT FORMAT

Every SDP PDU consists of a PDU header followed by PDU-specific parameters. The header contains three fields: a PDU ID, a Transaction ID, and a ParameterLength. Each of these header fields is described here. Parameters may include a continuation state parameter, described below; PDU-specific parameters for each PDU type are described later in separate PDU descriptions.

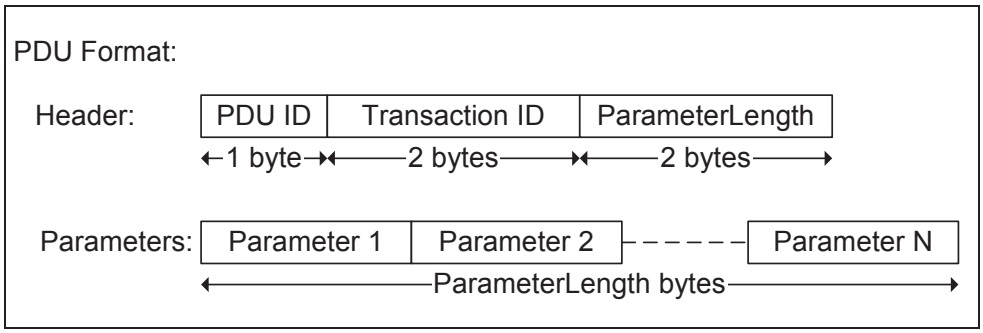



*Figure 4.1:*



LifeScan, Inc. Ex. 1019
Page 1204 of 1420
IPR2025-00103

*PDU ID:* *Size: 1 Byte*

| Value | Parameter Description |
|---|---|
| N | The PDU ID field identifies the type of PDU. I.e. its meaning and the specific parameters. |
| 0x00 | Reserved |
| 0x01 | SDP_ErrorResponse |
| 0x02 | SDP_ServiceSearchRequest |
| 0x03 | SDP_ServiceSearchResponse |
| 0x04 | SDP_ServiceAttributeRequest |
| 0x05 | SDP_ServiceAttributeResponse |
| 0x06 | SDP_ServiceSearchAttributeRequest |
| 0x07 | SDP_ServiceSearchAttributeResponse |
| 0x07-0xFF | Reserved |

*TransactionID:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The TransactionID field uniquely identifies request PDUs and is used to match response PDUs to request PDUs. The SDP client can choose any value for a request's TransactionID provided that it is different from all outstanding requests. The TransactionID value in response PDUs is required to be the same as the request that is being responded to.<br>Range: 0x0000 – 0xFFFF |

*ParameterLength:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The ParameterLength field specifies the length (in bytes) of all parameters contained in the PDU.<br>Range: 0x0000 – 0xFFFF |



LifeScan, Inc. Ex. 1019
Page 1205 of 1420
IPR2025-00103

## 4.3 PARTIAL RESPONSES AND CONTINUATION STATE

Some SDP requests may require responses that are larger than can fit in a single response PDU. In this case, the SDP server shall generate a partial response along with a continuation state parameter. The continuation state parameter can be supplied by the client in a subsequent request to retrieve the next portion of the complete response. The continuation state parameter is a variable length field whose first byte contains the number of additional bytes of continuation information in the field. The format of the continuation information is not standardized among SDP servers. Each continuation state parameter is meaningful only to the SDP server that generated it.

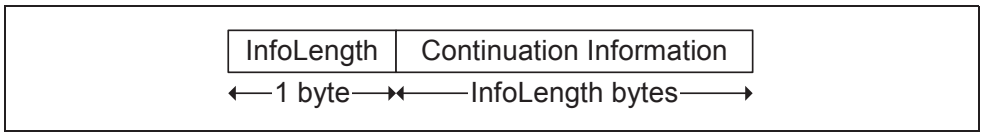



*Figure 4.2: Continuation State Format*

After a client receives a partial response and the accompanying continuation state parameter, it can re-issue the original request (with a new transaction ID) and include the continuation state in the new request indicating to the server that the remainder of the original response is desired. The maximum allowable value of the InfoLength field is 16 (0x10).

Note that an SDP server can split a response at any arbitrary boundary when it generates a partial response. The SDP server can select the boundary based on the contents of the reply, but is not required to do so.

## 4.4 ERROR HANDLING

Each transaction consists of a request and a response PDU. Generally, each type of request PDU has a corresponding type of response PDU. However, if the server determines that a request is improperly formatted or for any reason the server cannot respond with the appropriate PDU type, it shall respond with an SDP_ErrorResponse PDU.

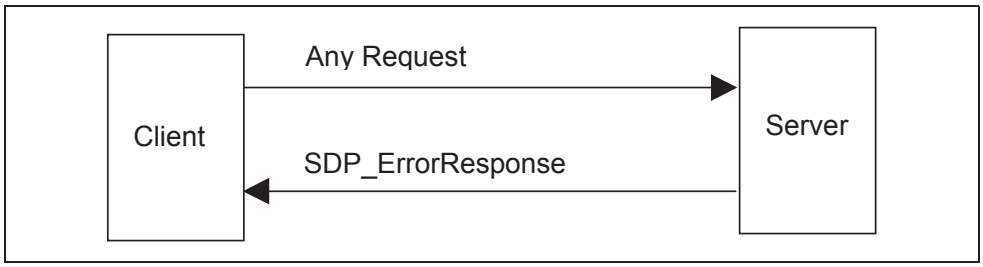



*Figure 4.3:*



LifeScan, Inc. Ex. 1019
Page 1206 of 1420
IPR2025-00103

### 4.4.1 SDP_ErrorResponse PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ErrorResponse | 0x01 | ErrorCode |

**Description:**

The SDP server generates this PDU type in response to an improperly formatted request PDU or when the SDP server, for whatever reason, cannot generate an appropriate response PDU.

**PDU Parameters:**

*ErrorCode:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The ErrorCode identifies the reason that an SDP_ErrorResponse PDU was generated. |
| 0x0000 | Reserved |
| 0x0001 | Invalid/unsupported SDP version |
| 0x0002 | Invalid Service Record Handle |
| 0x0003 | Invalid request syntax |
| 0x0004 | Invalid PDU Size |
| 0x0005 | Invalid Continuation State |
| 0x0006 | Insufficient Resources to satisfy Request |
| 0x0007-0xFFFF | Reserved |

## 4.5 SERVICESEARCH TRANSACTION

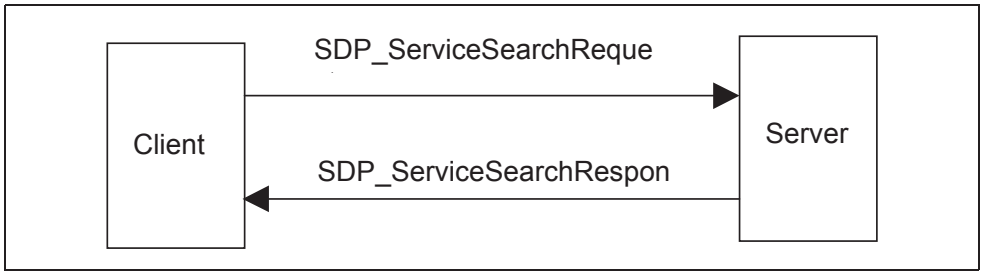



*Figure 4.4:*

### 4.5.1 SDP_ServiceSearchRequest PDU



LifeScan, Inc. Ex. 1019
Page 1207 of 1420
IPR2025-00103

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ServiceSearchRequest | 0x02 | ServiceSearchPattern, MaximumServiceRecordCount, ContinuationState |

**Description:**

The SDP client generates an SDP_ServiceSearchRequest to locate service records that match the service search pattern given as the first parameter of the PDU. Upon receipt of this request, the SDP server shall examine its service record data base and return an SDP_ServiceSearchResponse containing the service record handles of service records within its service record database that match the given service search pattern, or an appropriate error response.

Note that no mechanism is provided to request information for all service records. However, see Section 2.6 on page 122 for a description of a mechanism that permits browsing for non-specific services without a priori knowledge of the services.

**PDU Parameters:**

*ServiceSearchPattern:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The ServiceSearchPattern is a data element sequence where each element in the sequence is a UUID. The sequence shall contain at least one UUID. The maximum number of UUIDs in the sequence is 12[1]. The list of UUIDs constitutes a service search pattern. |

1. The value of 12 has been selected as a compromise between the scope of a service search and the size of a search request PDU. It is not expected that more than 12 UUIDs will be useful in a service search pattern.

*MaximumServiceRecordCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | MaximumServiceRecordCount is a 16-bit count specifying the maximum number of service record handles to be returned in the response(s) to this request. The SDP server shall not return more handles than this value specifies. If more than N service records match the request, the SDP server determines which matching service record handles to return in the response(s).<br>Range: 0x0001-0xFFFF |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|



LifeScan, Inc. Ex. 1019
Page 1208 of 1420
IPR2025-00103

| | |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N is required to be less than or equal to 16. If no continuation state is to be provided in the request, N is set to 0. |

### 4.5.2 SDP_ServiceSearchResponse PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ServiceSearchResponse | 0x03 | TotalServiceRecordCount, CurrentServiceRecordCount, ServiceRecordHandleList, ContinuationState |

**Description:**

The SDP server generates an SDP_ServiceSearchResponse upon receipt of a valid SDP_ServiceSearchRequest. The response contains a list of service record handles for service records that match the service search pattern given in the request. If a partial response is generated, it shall contain an integral number of complete service record handles; a service record handle value shall not be split across multiple PDUs.



LifeScan, Inc. Ex. 1019
Page 1209 of 1420
IPR2025-00103

### PDU Parameters:

*TotalServiceRecordCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The TotalServiceRecordCount is an integer containing the number of service records that match the requested service search pattern. If no service records match the requested service search pattern, this parameter is set to 0. N should never be larger than the MaximumServiceRecordCount value specified in the SDP_ServiceSearchRequest. When multiple partial responses are used, each partial response contains the same value for TotalServiceRecordCount.<br>Range: 0x0000-0xFFFF |

*CurrentServiceRecordCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The CurrentServiceRecordCount is an integer indicating the number of service record handles that are contained in the next parameter. If no service records match the requested service search pattern, this parameter is set to 0. N should never be larger than the TotalServiceRecordCount value specified in the current response.<br>Range: 0x0000-0xFFFF |

*ServiceRecordHandleList:* *Size: (CurrentServiceRecordCount*4) Bytes*

| Value | Parameter Description |
|---|---|
| List of 32-bit handles | The ServiceRecordHandleList contains a list of service record handles. The number of handles in the list is given in the CurrentServiceRecordCount parameter. Each of the handles in the list refers to a service record that matches the requested service search pattern. Note that this list of service record handles does not have the format of a data element. It contains no header fields, only the 32-bit service record handles. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is contained in the PDU, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response. |



LifeScan, Inc. Ex. 1019
Page 1210 of 1420
IPR2025-00103

## 4.6 SERVICEATTRIBUTE TRANSACTION

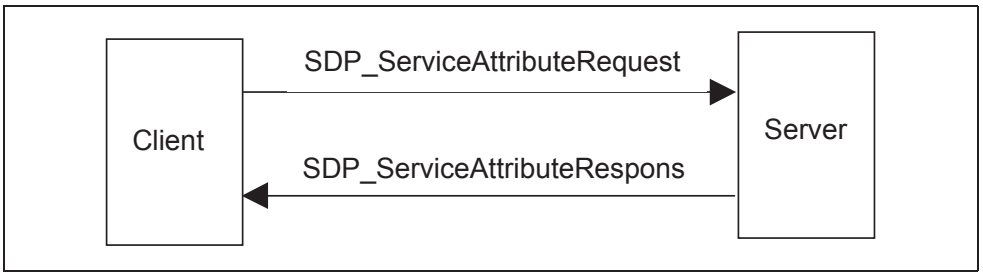



*Figure 4.5:*

### 4.6.1 SDP_ServiceAttributeRequest PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ServiceAttributeRequest | 0x04 | ServiceRecordHandle, MaximumAttributeByteCount, AttributeIDList, ContinuationState |

**Description:**

The SDP client generates an SDP_ServiceAttributeRequest to retrieve specified attribute values from a specific service record. The service record handle of the desired service record and a list of desired attribute IDs to be retrieved from that service record are supplied as parameters.

**Command Parameters:**

*ServiceRecordHandle:* *Size: 4 Bytes*

| Value | Parameter Description |
|---|---|
| 32-bit handle | The ServiceRecordHandle parameter specifies the service record from which attribute values are to be retrieved. The handle is obtained via a previous SDP_ServiceSearch transaction. |



LifeScan, Inc. Ex. 1019
Page 1211 of 1420
IPR2025-00103

*MaximumAttributeByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | MaximumAttributeByteCount specifies the maximum number of bytes of attribute data to be returned in the response to this request. The SDP server shall not return more than N bytes of attribute data in the response PDU. If the requested attributes require more than N bytes, the SDP server determines how to segment the list. In this case the client may request each successive segment by issuing a request containing the continuation state that was returned in the previous response PDU. Note that in the case where multiple response PDUs are needed to return the attribute data, MaximumAttributeByteCount specifies the maximum size of the portion of the attribute data contained in each response PDU.<br>Range: 0x0007-0xFFFF |

*AttributeIDList:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeIDList is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. Each attribute ID is encoded as a 16-bit unsigned integer data element. Each attribute ID range is encoded as a 32-bit unsigned integer data element, where the high order 16 bits are interpreted as the beginning attribute ID of the range and the low order 16 bits are interpreted as the ending attribute ID of the range. The attribute IDs contained in the AttributeIDList shall be listed in ascending order without duplication of any attribute ID values. Note that all attributes can be requested by specifying a range of 0x0000-0xFFFF. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N shall be less than or equal to 16. If no continuation state is to be provided in the request, N shall be set to 0. |



LifeScan, Inc. Ex. 1019
Page 1212 of 1420
IPR2025-00103

### 4.6.2 SDP_ServiceAttributeResponse PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ServiceAttributeResponse | 0x05 | AttributeListByteCount, AttributeList, ContinuationState |

**Description:**

The SDP server generates an SDP_ServiceAttributeResponse upon receipt of a valid SDP_ServiceAttributeRequest. The response contains a list of attributes (both attribute ID and attribute value) from the requested service record.

**PDU Parameters:**

*AttributeListByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The AttributeListByteCount contains a count of the number of bytes in the AttributeList parameter. N shall never be larger than the MaximumAttributeByteCount value specified in the SDP_ServiceAttributeRequest.<br>Range: 0x0002-0xFFFF |

*AttributeList:* *Size: AttributeListByteCount*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeList is a data element sequence containing attribute IDs and attribute values. The first element in the sequence contains the attribute ID of the first attribute to be returned. The second element in the sequence contains the corresponding attribute value. Successive pairs of elements in the list contain additional attribute ID and value pairs. Only attributes that have non-null values within the service record and whose attribute IDs were specified in the SDP_ServiceAttributeRequest are contained in the AttributeList. Neither an attribute ID nor an attribute value is placed in the AttributeList for attributes in the service record that have no value. The attributes are listed in ascending order of attribute ID value. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is given, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response. |



LifeScan, Inc. Ex. 1019
Page 1213 of 1420
IPR2025-00103

## 4.7 SERVICESEARCHATTRIBUTE TRANSACTION

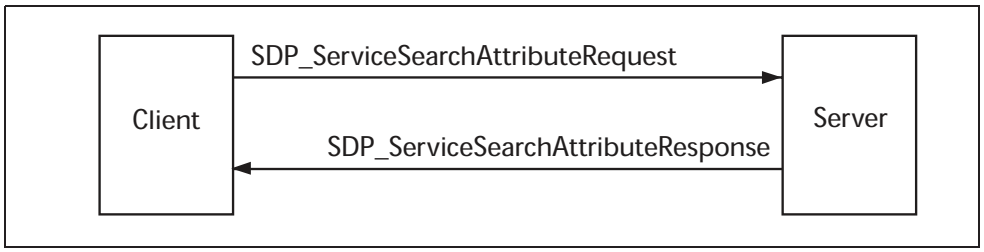



*Figure 4.6:*

### 4.7.1 SDP_ServiceSearchAttributeRequest PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ServiceSearchAttributeRequest | 0x06 | ServiceSearchPattern, MaximumAttributeByteCount, AttributeIDList, ContinuationState |

**Description:**

The SDP_ServiceSearchAttributeRequest transaction combines the capabilities of the SDP_ServiceSearchRequest and the SDP_ServiceAttributeRequest into a single request. As parameters, it contains both a service search pattern and a list of attributes to be retrieved from service records that match the service search pattern. The SDP_ServiceSearchAttributeRequest and its response are more complex and can require more bytes than separate SDP_ServiceSearch and SDP_ServiceAttribute transactions. However, using SDP_ServiceSearchAttributeRequest can reduce the total number of SDP transactions, particularly when retrieving multiple service records.

Note that the service record handle for each service record is contained in the ServiceRecordHandle attribute of that service and may be requested along with other attributes.

**PDU Parameters:**

*ServiceSearchPattern:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The ServiceSearchPattern is a data element sequence where each element in the sequence is a UUID. The sequence must contain at least one UUID. The maximum number of UUIDs in the sequence is 12[1]. The list of UUIDs constitutes a service search pattern. |

1. The value of 12 has been selected as a compromise between the scope of a service search and the size of a search request PDU. It is not expected that more than 12 UUIDs will be useful in a service search pattern.



LifeScan, Inc. Ex. 1019
Page 1214 of 1420
IPR2025-00103

*MaximumAttributeByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | MaximumAttributeByteCount specifies the maximum number of bytes of attribute data to be returned in the response to this request. The SDP server shall not return more than N bytes of attribute data in the response PDU. If the requested attributes require more than N bytes, the SDP server determines how to segment the list. In this case the client may request each successive segment by issuing a request containing the continuation state that was returned in the previous response PDU. Note that in the case where multiple response PDUs are needed to return the attribute data, MaximumAttributeByteCount specifies the maximum size of the portion of the attribute data contained in each response PDU.<br>Range: 0x0007-0xFFFF |

*AttributeIDList:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeIDList is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. Each attribute ID is encoded as a 16-bit unsigned integer data element. Each attribute ID range is encoded as a 32-bit unsigned integer data element, where the high order 16 bits are interpreted as the beginning attribute ID of the range and the low order 16 bits are interpreted as the ending attribute ID of the range. The attribute IDs contained in the AttributeIDList shall be listed in ascending order without duplication of any attribute ID values. Note that all attributes may be requested by specifying a range of 0x0000-0xFFFF. |

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N shall be less than or equal to 16. If no continuation state is to be provided in the request, N shall set to 0. |

### 4.7.2 SDP_ServiceSearchAttributeResponse PDU

| PDU Type | PDU ID | Parameters |
|---|---|---|
| SDP_ServiceSearchAttributeResponse | 0x07 | AttributeListsByteCount, AttributeLists, ContinuationState |

**Description:**

The SDP server generates an SDP_ServiceSearchAttributeResponse upon receipt of a valid SDP_ServiceSearchAttributeRequest. The response contains a list of attributes (both attribute ID and attribute value) from the service records that match the requested service search pattern.

**PDU Parameters:**

*AttributeListsByteCount:* *Size: 2 Bytes*

| Value | Parameter Description |
|---|---|
| N | The AttributeListsByteCount contains a count of the number of bytes in the AttributeLists parameter. N shall never be larger than the MaximumAttributeByteCount value specified in the SDP_ServiceSearchAttributeRequest.<br>Range: 0x0002-0xFFFF |

*AttributeLists:* *Size: Varies*

| Value | Parameter Description |
|---|---|
| Data Element Sequence | The AttributeLists is a data element sequence where each element in turn is a data element sequence representing an attribute list. Each attribute list contains attribute IDs and attribute values from one service record. The first element in each attribute list contains the attribute ID of the first attribute to be returned for that service record. The second element in each attribute list contains the corresponding attribute value. Successive pairs of elements in each attribute list contain additional attribute ID and value pairs. Only attributes that have non-null values within the service record and whose attribute IDs were specified in the SDP_ServiceSearchAttributeRequest are contained in the AttributeLists. Neither an attribute ID nor attribute value is placed in AttributeLists for attributes in the service record that have no value. Within each attribute list, the attributes are listed in ascending order of attribute ID value. |



LifeScan, Inc. Ex. 1019
Page 1216 of 1420
IPR2025-00103

*ContinuationState:* *Size: 1 to 17 Bytes*

| Value | Parameter Description |
|---|---|
| Continuation State | ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is given, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response. |



LifeScan, Inc. Ex. 1019
Page 1217 of 1420
IPR2025-00103

# 5 SERVICE ATTRIBUTE DEFINITIONS

The service classes and attributes contained in this document are necessarily a partial list of the service classes and attributes supported by SDP. Only service classes that directly support the SDP server are included in this document. Additional service classes will be defined in other documents and possibly in future revisions of this document. Also, it is expected that additional attributes will be discovered that are applicable to a broad set of services; these may be added to the list of Universal attributes in future revisions of this document.

## 5.1 UNIVERSAL ATTRIBUTE DEFINITIONS

Universal attributes are those service attributes whose definitions are common to all service records. Note that this does not mean that every service record contains values for all of these service attributes. However, if a service record has a service attribute with an attribute ID allocated to a universal attribute, the attribute value shall conform to the universal attribute's definition.

Only two attributes are required to exist in every service record instance. They are the ServiceRecordHandle (attribute ID 0x0000) and the ServiceClassIDList (attribute ID 0x0001). All other service attributes are optional within a service record.

### 5.1.1 ServiceRecordHandle Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceRecordHandle | 0x0000 | 32-bit unsigned integer |

**Description:**

A service record handle is a 32-bit number that uniquely identifies each service record within an SDP server. It is important to note that, in general, each handle is unique only within each SDP server. If SDP server S1 and SDP server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on S1 will, in general, be meaningless if presented to S2. Service record handle values 0x00000001-0x0000FFFF are reserved.



LifeScan, Inc. Ex. 1019
Page 1218 of 1420
IPR2025-00103

### 5.1.2 ServiceClassIDList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceClassIDList | 0x0001 | Data Element Sequence |

**Description:**

The ServiceClassIDList attribute consists of a data element sequence in which each data element is a UUID representing the service classes that a given service record conforms to. The UUIDs are listed in order from the most specific class to the most general class. The ServiceClassIDList shall contain at least one service class UUID.

### 5.1.3 ServiceRecordState Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceRecordState | 0x0002 | 32-bit unsigned integer |

**Description:**

The ServiceRecordState is a 32-bit integer that is used to facilitate caching of ServiceAttributes. If this attribute is contained in a service record, its value shall bechanged when any other attribute value is added to, deleted from or changed within the service record. This permits a client to check the value of this single attribute. If its value has not changed since it was last checked, the client knows that no other attribute values within the service record have changed.

### 5.1.4 ServiceID Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceID | 0x0003 | UUID |

**Description:**

The ServiceID is a UUID that universally and uniquely identifies the service instance described by the service record. This service attribute is particularly useful if the same service is described by service records in more than one SDP server.



LifeScan, Inc. Ex. 1019
Page 1219 of 1420
IPR2025-00103

### 5.1.5 ProtocolDescriptorList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ProtocolDescriptorList | 0x0004 | Data Element Sequence or Data Element Alternative |

**Description:**

The ProtocolDescriptorList attribute describes one or more protocol stacks that can be used to gain access to the service described by the service record.

If the ProtocolDescriptorList describes a single stack, it takes the form of a data element sequence in which each element of the sequence is a protocol descriptor. Each protocol descriptor is, in turn, a data element sequence whose first element is a UUID identifying the protocol and whose successive elements are protocol-specific parameters. Potential protocol-specific parameters are a protocol version number and a connection-port number. The protocol descriptors are listed in order from the lowest layer protocol to the highest layer protocol used to gain access to the service.

If it is possible for more than one kind of protocol stack to be used to gain access to the service, the ProtocolDescriptorList takes the form of a data element alternative where each member is a data element sequence as described in the previous paragraph.

**Protocol Descriptors**

A protocol descriptor identifies a communications protocol and provides protocol-specific parameters. A protocol descriptor is represented as a data element sequence. The first data element in the sequence shall be the UUID that identifies the protocol. Additional data elements optionally provide protocol-specific information, such as the L2CAP protocol/service multiplexer (PSM) and the RFCOMM server channel number (CN) shown below.

**ProtocolDescriptorList Examples**

These examples are intended to be illustrative. The parameter formats for each protocol are not defined within this specification.

In the first two examples, it is assumed that a single RFCOMM instance exists on top of the L2CAP layer. In this case, the L2CAP protocol specific information (PSM) points to the single instance of RFCOMM. In the last example, two different and independent RFCOMM instances are available on top of the L2CAP layer. In this case, the L2CAP protocol specific information (PSM) points to a distinct identifier that distinguishes each of the RFCOMM instances. According to the L2CAP specification, this identifier takes values in the range 0x1000-0xFFFF.



LifeScan, Inc. Ex. 1019
Page 1220 of 1420
IPR2025-00103

*IrDA-like printer*

( ( L2CAP, PSM=RFCOMM ), ( RFCOMM, CN=1 ), ( PostscriptStream ) )

*IP Network Printing*

( ( L2CAP, PSM=RFCOMM ), ( RFCOMM, CN=2 ), ( PPP ), ( IP ), ( TCP ), ( IPP ) )

Synchronization Protocol Descriptor Example

( ( L2CAP, PSM=0x1001 ), ( RFCOMM, CN=1 ), ( Obex ), ( vCal ) )

( ( L2CAP, PSM=0x1002 ), ( RFCOMM, CN=1 ), ( Obex ), ( otherSynchronisationApplication ) )

### 5.1.6 AdditionalProtocolDescriptorList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| AdditionalProtocolDescriptorList | 0x000D | Data Element Sequence |

**Description:**

The AdditionalProtocolDescriptorLists attribute contains a sequence of ProtocolDescriptorList-elements. Each element having the same format as the ProtocolDescriptorList described in section 5.1.5. The ordering of the elements is significant and should be specified and fixed in Profiles that make use of this attribute.

The AdditionalProtocolDescriptorLists attribute supports services that require more channels in addition to the service described in Section 5.1.5 . If the AdditionalProtocolDescriptorLists attribute is included in a service record, the ProtocolDescriptorList attribute must be included.

**AdditionalProtocolDescriptorList Examples**

The following is just an illustrative example and is not meant to refer a real specified service or protocols.

```
Attribute    Attribute Value type            Attribute Value

ProtocolDescriptorList

   ProtocolDescriptor #0DataElementSequence

         ProtocolID    UUID       L2CAP

         Param: PSM    PSM        FooDataProtocol

   ProtocolDescriptor #1DataElementSequence
```



LifeScan, Inc. Ex. 1019
Page 1221 of 1420
IPR2025-00103

```
         ProtocolID          UUID       FooDataProtocol
AdditionalProtocolDescriptorLists
   ProtocolDescriptorList #0  DataElementSequence
   ProtocolDescriptor #0      DataElementSequence
              ProtocolID      UUID       L2CAP
              Param: PSM      PSM        FooControlProtocol
         ProtocolDescriptor #1DataElementSequence
              ProtocolID      UUID       FooControlProtocol
```

### 5.1.7 BrowseGroupList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| BrowseGroupList | 0x0005 | Data Element Sequence |

**Description:**

The BrowseGroupList attribute consists of a data element sequence in which each element is a UUID that represents a browse group to which the service record belongs. The top-level browse group ID, called PublicBrowseRoot and representing the root of the browsing hierarchy, has the value 00001002-0000-1000-8000-00805F9B34FB (UUID16: 0x1002) from the Bluetooth Assigned Numbers document. If one or more human readable attributes are contained in a service record, the LanguageBaseAttributeIDList attribute should be included in that service record.

### 5.1.8 LanguageBaseAttributeIDList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| LanguageBaseAttributeIDList | 0x0006 | Data Element Sequence |

**Description:**

In order to support human-readable attributes for multiple natural languages in a single service record, a base attribute ID is assigned for each of the natural languages used in a service record. The human-readable universal attributes are then defined with an attribute ID offset from each of these base values, rather than with an absolute attribute ID.

The LanguageBaseAttributeIDList attribute is a list in which each member contains a language identifier, a character encoding identifier, and a base attribute ID for each of the natural languages used in the service record. The Language-

BaseAttributeIDList attribute consists of a data element sequence in which each element is a 16-bit unsigned integer. The elements are grouped as triplets (threes).

The first element of each triplet contains an identifier representing the natural language. The language is encoded according to ISO 639:1988 (E/F): “Code for the representation of names of languages”.

The second element of each triplet contains an identifier that specifies a character encoding used for the language. Values for character encoding can be found in IANA's database[1], and have the values that are referred to as MIBEnum values. The recommended character encoding is UTF-8.

The third element of each triplet contains an attribute ID that serves as the base attribute ID for the natural language in the service record. Different service records within a server may use different base attribute ID values for the same language.

To facilitate the retrieval of human-readable universal attributes in a principal language, the base attribute ID value for the primary language supported by a service record shall be 0x0100. Also, if a LanguageBaseAttributeIDList attribute is contained in a service record, the base attribute ID value contained in its first element shall be 0x0100.

### 5.1.9 ServiceInfoTimeToLive Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceInfoTimeToLive | 0x0007 | 32-bit unsigned integer |

**Description:**

The ServiceTimeToLive attribute is a 32-bit integer that contains the number of seconds for which the information in a service record is expected to remain valid and unchanged. This time interval is measured from the time that the attribute value is retrieved from the SDP server. This value does not imply a guarantee that the service record will remain available or unchanged. It is simply a hint that a client can use to determine a suitable polling interval to re-validate the service record contents.

1. See http://www.isi.edu/in-notes/iana/assignments/character-sets



LifeScan, Inc. Ex. 1019
Page 1223 of 1420
IPR2025-00103

### 5.1.10 ServiceAvailability Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceAvailability | 0x0008 | 8-bit unsigned integer |

**Description:**

The ServiceAvailability attribute is an 8-bit unsigned integer that represents the relative ability of the service to accept additional clients. A value of 0xFF indicates that the service is not currently in use and is thus fully available, while a value of 0x00 means that the service is not accepting new clients. For services that support multiple simultaneous clients, intermediate values indicate the relative availability of the service on a linear scale.

For example, a service that can accept up to 3 clients should provide ServiceAvailability values of 0xFF, 0xAA, 0x55, and 0x00 when 0, 1, 2, and 3 clients, respectively, are utilizing the service. The value 0xAA is approximately (2/3) * 0xFF and represents 2/3 availability, while the value 0x55 is approximately (1/3)*0xFF and represents 1/3 availability. Note that the availability value is be approximated as

( 1 - ( current_number_of_clients / maximum_number_of_clients ) ) * 0xFF

When the maximum number of clients is large, this formula must be modified to ensure that ServiceAvailability values of 0x00 and 0xFF are reserved for their defined meanings of unavailability and full availability, respectively.

Note that the maximum number of clients a service can support may vary according to the resources utilized by the service's current clients.

A non-zero value for ServiceAvailability does not guarantee that the service will be available for use. It should be treated as a hint or an approximation of availability status.

### 5.1.11 BluetoothProfileDescriptorList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| BluetoothProfileDescriptorList | 0x0009 | Data Element Sequence |

**Description:**

The BluetoothProfileDescriptorList attribute consists of a data element sequence in which each element is a profile descriptor that contains information about a Bluetooth profile to which the service represented by this service record conforms. Each profile descriptor is a data element sequence whose first element is the UUID assigned to the profile and whose second element is a 16-bit profile version number.



LifeScan, Inc. Ex. 1019
Page 1224 of 1420
IPR2025-00103

Each version of a profile is assigned a 16-bit unsigned integer profile version number, which consists of two 8-bit fields. The higher-order 8 bits contain the major version number field and the lower-order 8 bits contain the minor version number field. The initial version of each profile has a major version of 1 and a minor version of 0. When upward compatible changes are made to the profile, the minor version number will be incremented. If incompatible changes are made to the profile, the major version number will be incremented.

### 5.1.12 DocumentationURL Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| DocumentationURL | 0x000A | URL |

**Description:**

This attribute is a URL which points to documentation on the service described by a service record.

### 5.1.13 ClientExecutableURL Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ClientExecutableURL | 0x000B | URL |

**Description:**

This attribute contains a URL that refers to the location of an application that can be used to utilize the service described by the service record. Since different operating environments require different executable formats, a mechanism has been defined to allow this single attribute to be used to locate an executable that is appropriate for the client device's operating environment. In the attribute value URL, the first byte with the value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired operating environment before the URL is to be used.

The list of standardized strings representing operating environments is contained in the Bluetooth Assigned Numbers document.

For example, assume that the value of the ClientExecutableURL attribute is http://my.fake/public/*/client.exe. On a device capable of executing SH3 WindowsCE files, this URL would be changed to http://my.fake/public/sh3-microsoft-wince/client.exe. On a device capable of executing Windows 98 binaries, this URL would be changed to http://my.fake/public/i86-microsoft-win98/client.exe.



LifeScan, Inc. Ex. 1019
Page 1225 of 1420
IPR2025-00103

### 5.1.14 IconURL Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| IconURL | 0x000C | URL |

**Description:**

This attribute contains a URL that refers to the location of an icon that can be used to represent the service described by the service record. Since different hardware devices require different icon formats, a mechanism has been defined to allow this single attribute to be used to locate an icon that is appropriate for the client device. In the attribute value URL, the first byte with the value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired icon format before the URL is to be used.

The list of standardized strings representing icon formats is contained in the Bluetooth Assigned Numbers document.

For example, assume that the value of the IconURL attribute is http://my.fake/public/icons/*. On a device that prefers 24 x 24 icons with 256 colors, this URL would be changed to http://my.fake/public/icons/24x24x8.png. On a device that prefers 10 x 10 monochrome icons, this URL would be changed to http://my.fake/public/icons/10x10x1.png.

### 5.1.15 ServiceName Attribute

| Attribute Name | Attribute ID Offset | Attribute Value Type |
|---|---|---|
| ServiceName | 0x0000 | String |

**Description:**

The ServiceName attribute is a string containing the name of the service represented by a service record. It should be brief and suitable for display with an Icon representing the service. The offset 0x0000 is added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.



LifeScan, Inc. Ex. 1019
Page 1226 of 1420
IPR2025-00103

### 5.1.16 ServiceDescription Attribute

| Attribute Name | Attribute ID Offset | Attribute Value Type |
|---|---|---|
| ServiceDescription | 0x0001 | String |

**Description:**

This attribute is a string containing a brief description of the service. It should be less than 200 characters in length. The offset 0x0001 is added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

### 5.1.17 ProviderName Attribute

| Attribute Name | Attribute ID Offset | Attribute Value Type |
|---|---|---|
| ProviderName | 0x0002 | String |

**Description:**

This attribute is a string containing the name of the person or organization providing the service. The offset 0x0002 is added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

### 5.1.18 Reserved Universal Attribute IDs

Attribute IDs in the range of 0x000D-0x01FF are reserved.



LifeScan, Inc. Ex. 1019
Page 1227 of 1420
IPR2025-00103

## 5.2 SERVICEDISCOVERYSERVER SERVICE CLASS ATTRIBUTE DEFINITIONS

This service class describes service record that contains attributes of service discovery server itself. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the ServiceDiscoveryServerService-ClassID. All of the universal attributes may be included in service records of the ServiceDiscoveryServer class.

### 5.2.1 ServiceRecordHandle Attribute

Described in the universal attribute definition for ServiceRecordHandle.

**Value**

A 32-bit integer with the value 0x000000000.

### 5.2.2 ServiceClassIDList Attribute

Described in the universal attribute definition for ServiceClassIDList.

**Value**

A UUID representing the ServiceDiscoveryServerServiceClassID.

### 5.2.3 VersionNumberList Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| VersionNumberList | 0x0200 | Data Element Sequence |

**Description:**

The VersionNumberList is a data element sequence in which each element of the sequence is a version number supported by the SDP server.

A version number is a 16-bit unsigned integer consisting of two fields. The higher-order 8 bits contain the major version number field and the low-order 8 bits contain the minor version number field. The initial version of SDP has a major version of 1 and a minor version of 0. When upward compatible changes are made to the protocol, the minor version number will be incremented. If incompatible changes are made to SDP, the major version number will be incremented. This guarantees that if a client and a server support a common major version number, they can communicate if each uses only features of the specification with a minor version number that is supported by both client and server.



LifeScan, Inc. Ex. 1019
Page 1228 of 1420
IPR2025-00103

### 5.2.4 ServiceDatabaseState Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| ServiceDatabaseState | 0x0201 | 32-bit unsigned integer |

**Description:**

The ServiceDatabaseState is a 32-bit integer that is used to facilitate caching of service records. If this attribute exists, its value shall be changed when any of the other service records are added to or deleted from the server's database. If this value has not changed since the last time a client queried its value, the client knows that a) none of the other service records maintained by the SDP server have been added or deleted; and b) any service record handles acquired from the server are still valid. A client shall query this attribute's value when a connection to the server is established, prior to using any service record handles acquired during a previous connection.

Note that the ServiceDatabaseState attribute does not change when existing service records are modified, including the addition, removal, or modification of service attributes. A service record's ServiceRecordState attribute indicates when that service record is modified.

### 5.2.5 Reserved Attribute IDs

Attribute IDs in the range of 0x0202-0x02FF are reserved.

## 5.3 BROWSEGROUPDESCRIPTOR SERVICE CLASS ATTRIBUTE DEFINITIONS

This service class describes the ServiceRecord provided for each BrowseGroupDescriptor service offered on a Bluetooth device. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the BrowseGroupDescriptorServiceClassID. Note that all of the universal attributes may be included in service records of the BrowseGroupDescriptor class.

### 5.3.1 ServiceClassIDList Attribute

Described in the universal attribute definition for ServiceClassIDList.

**Value**

A UUID representing the BrowseGroupDescriptorServiceClassID.



LifeScan, Inc. Ex. 1019
Page 1229 of 1420
IPR2025-00103

### 5.3.2 GroupID Attribute

| Attribute Name | Attribute ID | Attribute Value Type |
|---|---|---|
| GroupID | 0x0200 | UUID |

**Description:**

This attribute contains a UUID that can be used to locate services that are members of the browse group that this service record describes.

### 5.3.3 Reserved Attribute IDs

Attribute IDs in the range of 0x0201-0x02FF are reserved.



LifeScan, Inc. Ex. 1019
Page 1230 of 1420
IPR2025-00103

# 6 SECURITY

In Security Mode 4, SDP should use no security but may use security (an authenticated or unauthenticated link key with encryption). See [Part C] Section 5.2.2 on page 197).



LifeScan, Inc. Ex. 1019
Page 1231 of 1420
IPR2025-00103

# 7 APPENDIX



LifeScan, Inc. Ex. 1019
Page 1232 of 1420
IPR2025-00103

# APPENDIX A – BACKGROUND INFORMATION

## A.1. Service Discovery

As computing continues to move to a network-centric model, finding and making use of services that may be available in the network becomes increasingly important. Services can include common ones such as printing, paging, FAXing, and so on, as well as various kinds of information access such as teleconferencing, network bridges and access points, eCommerce facilities, and so on — most any kind of service that a server or service provider might offer. In addition to the need for a standard way of discovering available services, there are other considerations: getting access to the services (finding and obtaining the protocols, access methods, “drivers” and other code necessary to utilize the service), controlling access to the services, advertising the services, choosing among competing services, billing for services, and so on. This problem is widely recognized; many companies, standards bodies and consortia are addressing it at various levels in various ways. Service Location Protocol (SLP), Jini™, and Salutation™, to name just a few, all address some aspect of service discovery.

## A.2. Bluetooth Service Discovery

Bluetooth Service Discovery Protocol (SDP) addresses service discovery specifically for the Bluetooth environment. It is optimized for the highly dynamic nature of Bluetooth communications. SDP focuses primarily on discovering services available from or through Bluetooth devices. SDP does not define methods for accessing services; once services are discovered with SDP, they can be accessed in various ways, depending upon the service. This might include the use of other service discovery and access mechanisms such as those mentioned above; SDP provides a means for other protocols to be used along with SDP in those environments where this can be beneficial. While SDP can coexist with other service discovery protocols, it does not require them. In Bluetooth environments, services can be discovered using SDP and can be accessed using other protocols defined by Bluetooth.



LifeScan, Inc. Ex. 1019
Page 1233 of 1420
IPR2025-00103

# APPENDIX B – EXAMPLE SDP TRANSACTIONS

The following are simple examples of typical SDP transactions. These are meant to be illustrative of SDP flows. The examples do not consider:

- Caching (in a caching system, the SDP client would make use of the ServiceRecordState and ServiceDatabaseState attributes);
- Service availability (if this is of interest, the SDP client should use the ServiceAvailability and/or ServiceTimeToLive attributes);
- SDP versions (the VersionNumberList attribute could be used to determine compatible SDP versions);
- SDP Error Responses (an SDP error response is possible for any SDP request that is in error); and
- Communication connection (the examples assume that an L2CAP connection is established).

The examples are meant to be illustrative of the protocol. The format used is `ObjectName[ObjectSizeInBytes] {SubObjectDefinitions}`, but this is not meant to illustrate an interface. The `ObjectSizeInBytes` is the size of the object in decimal. The `SubObjectDefinitions` (inside of `{}` characters) are components of the immediately enclosing object. Hexadecimal values shown as lower-case letters, such as for transaction IDs and service handles, are variables (the particular value is not important for the illustration, but each such symbol always represents the same value). Comments are included in this manner: `/* comment text */`.

Numeric values preceded by “0x” are hexadecimal, while those preceded by “0b” are binary. All other numeric values are decimal.

## B.1. SDP Example 1 – ServiceSearchRequest

The first example is that of an SDP client searching for a generic printing service. The client does not specify a particular type of printing service. In the example, the SDP server has two available printing services. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceSearchRequest, specifying the PrinterServiceClassID (represented as a DataElement with a 32-bit UUID value of `ppp...ppp`) as the only element of the ServiceSearchPattern. The PrinterServiceClassID is assumed to be a 32-bit UUID and the data element type for it is illustrated. The TransactionID is illustrated as `tttt`.
2. SDP server to SDP client: SDP_ServiceSearchResponse, returning handles to two printing services, represented as `qqqqqqqq` for the first printing service and `rrrrrrrr` for the second printing service. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (`tttt`).



LifeScan, Inc. Ex. 1019
Page 1234 of 1420
IPR2025-00103

```
/* Sent from SDP Client to SDP server */
SDP_ServiceSearchRequest[15] {
  PDUID[1] {
    0x02
  }
  TransactionID[2] {
    0xtttt
  }
  ParameterLength[2] {
    0x000A
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* PrinterServiceClassID */
        0b00011 0b010 0xpppppppp
      }
    }
  }
  MaximumServiceRecordCount[2] {
    0x0003
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}


/* Sent from SDP server to SDP client */
SDP_ServiceSearchResponse[18] {
  PDUID[1] {
    0x03
  }
  TransactionID[2] {
    0xtttt
  }
  ParameterLength[2] {
    0x000D
  }
  TotalServiceRecordCount[2] {
    0x0002
  }
  CurrentServiceRecordCount[2] {
    0x0002
  }
  ServiceRecordHandleList[8] {
    /* print service 1 handle */
    0xqqqqqqqq
    /* print service 2 handle */
    0xrrrrrrrr
  }
  ContinuationState[1] {
    /* no continuation state */
```

```
    0x00
  }
}
```

## B.2. SDP Example 2 – ServiceAttributeTransaction

The second example continues the first example. In Example 1, the SDP client obtained handles to two printing services. In Example 2, the client uses one of those service handles to obtain the ProtocolDescriptorList attribute for that printing service. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceAttributeRequest, presenting the previously obtained service handle (the one denoted as `qqqqqqqq`) and specifying the ProtocolDescriptorList attribute ID (AttributeID 0x0004) as the only attribute requested (other attributes could be retrieved in the same transaction if desired). The TransactionID is illustrated as `uuuu` to distinguish it from the TransactionID of Example 1.
2. SDP server to SDP client: SDP_ServiceAttributeResponse, returning the ProtocolDescriptorList for the specified printing service. This protocol stack is assumed to be ( (L2CAP), (RFCOMM, 2), (PostscriptStream) ). The ProtocolDescriptorList is a data element sequence in which each element is, in turn, a data element sequence whose first element is a UUID representing the protocol, and whose subsequent elements are protocol-specific parameters. In this example, one such parameter is included for the RFCOMM protocol, an 8-bit value indicating RFCOMM server channel 2. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (`uuuu`). The Attributes returned are illustrated as a data element sequence where the protocol descriptors are 32-bit UUIDs and the RFCOMM server channel is a data element with an 8-bit value of 2.

```
 /* Sent from SDP Client to SDP server */
SDP_ServiceAttributeRequest[17] {
  PDUID[1] {
    0x04
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x000C
  }
  ServiceRecordHandle[4] {
    0xqqqqqqqq
  }
  MaximumAttributeByteCount[2] {
    0x0080
  }
  AttributeIDList[5] {
    DataElementSequence[5] {
```



LifeScan, Inc. Ex. 1019
Page 1236 of 1420
IPR2025-00103

```
      0b00110 0b101 0x03
      AttributeID[3] {
        0b00001 0b001 0x0004
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP server to SDP client */
SDP_ServiceAttributeResponse[38] {
  PDUID[1] {
    0x05
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x0021
  }
  AttributeListByteCount[2] {
    0x001E
  }
  AttributeList[30] {
    DataElementSequence[30] {
      0b00110 0b101 0x1C
      Attribute[28] {
        AttributeID[3] {
          0b00001 0b001 0x0004
        }
        AttributeValue[25] {
          /* ProtocolDescriptorList */
          DataElementSequence[25] {
            0b00110 0b101 0x17
            /* L2CAP protocol descriptor */
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* L2CAP Protocol UUID */
                0b00011 0b010 <32-bit L2CAP UUID>
              }
            }
            /* RFCOMM protocol descriptor */
            DataElementSequence[9] {
              0b00110 0b101 0x07
              UUID[5] {
                /* RFCOMM Protocol UUID */
                0b00011 0b010 <32-bit RFCOMM UUID>
              }
              /* parameter for server 2 */
              Uint8[2] {
                0b00001 0b000 0x02
              }
            }
```

```
            /* PostscriptStream protocol descriptor */
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* PostscriptStream Protocol UUID */
                0b00011 0b010 <32-bit PostscriptStream UUID>
              }
            }
          }
        }
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
```

## B.3. SDP Example 3 – ServiceSearchAttributeTransaction

The third example is a form of service browsing, although it is not generic browsing in that it does not make use of SDP browse groups. Instead, an SDP client is searching for available Synchronization services that can be presented to the user for selection. The SDP client does not specify a particular type of synchronization service. In the example, the SDP server has three available synchronization services: an address book synchronization service and a calendar synchronization service (both from the same provider), and a second calendar synchronization service from a different provider. The SDP client is retrieving the same attributes for each of these services; namely, the data formats supported for the synchronization service (vCard, vCal, ICal, etc.) and those attributes that are relevant for presenting information to the user about the services. Also assume that the maximum size of a response is 400 bytes. Since the result is larger than this, the SDP client will repeat the request supplying a continuation state parameter to retrieve the remainder of the response. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceSearchAttributeRequest, specifying the generic SynchronisationServiceClassID (represented as a data element whose 32-bit UUID value is `sss...sss`) as the only element of the ServiceSearchPattern. The SynchronisationServiceClassID is assumed to be a 32-bit UUID. The requested attributes are the ServiceRecordHandle (attribute ID 0x0000), ServiceClassIDList (attribute ID 0x0001), IconURL (attribute ID 0x000C), ServiceName (attribute ID 0x0100), ServiceDescription (attribute ID 0x0101), and ProviderName (attributeID 0x0102) attributes; as well as the service-specific SupportedDataStores (AttributeID 0x0301). Since the first two attribute IDs (0x0000 and 0x0001) and three other attribute IDs(0x0100, 0x0101, and 0x0102 are consecutive, they are specified as attribute ranges. The TransactionID is illustrated as `vvvv` to distinguish it from the TransactionIDs of the other Examples.



LifeScan, Inc. Ex. 1019
Page 1238 of 1420
IPR2025-00103

Note that values in the service record's primary language are requested for the text attributes (ServiceName, ServiceDescription and ProviderName) so that absolute attribute IDs may be used, rather than adding offsets to a base obtained from the LanguageBaseAttributeIDList attribute.

2. SDP server to SDP client: SDP_ServiceSearchAttributeResponse, returning the specified attributes for each of the three synchronization services. In the example, each ServiceClassIDList is assumed to contain a single element, the generic SynchronisationServiceClassID (a 32-bit UUID represented as sss...sss). Each of the other attributes contain illustrative data in the example (the strings have illustrative text; the icon URLs are illustrative, for each of the respective three synchronization services; and the SupportedDataStore attribute is represented as an unsigned 8-bit integer where 0x01 = vCard2.1, 0x02 = vCard3.0, 0x03 = vCal1.0 and 0x04 = iCal). Note that one of the service records (the third for which data is returned) has no ServiceDescription attribute. The attributes are returned as a data element sequence, where each element is in turn a data element sequence representing a list of attributes. Within each attribute list, the ServiceClassIDList is a data element sequence while the remaining attributes are single data elements. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (0xvvvv). Since the entire result cannot be returned in a single response, a non-null continuation state is returned in this first response.

   Note that the total length of the initial data element sequence (487 in the example) is indicated in the first response, even though only a portion of this data element sequence (368 bytes in the example, as indicated in the AttributeLists byte count) is returned in the first response. The remainder of this data element sequence is returned in the second response (without an additional data element header).

3. SDP client to SDP server: SDP_ServiceSearchAttributeRequest, with the same parameters as in step 1, except that the continuation state received from the server in step 2 is included as a request parameter. The TransactionID is changed to 0xwww to distinguish it from previous request.
4. SDP server to SDP client: SDP_ServiceSearchAttributeResponse, with the remainder of the result computed in step 2 above. Since all of the remaining result fits in this second response, a null continuation state is included.

```
/* Part 1 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[33] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x001B
  }
  ServiceSearchPattern[7] {
```



LifeScan, Inc. Ex. 1019
Page 1239 of 1420
IPR2025-00103

```
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
      }
    }
  }
  MaximumAttributeByteCount[2] {
    0x0190
  }
AttributeIDList[18] {
    DataElementSequence[18] {
      0b00110 0b101 0x10
      AttributeIDRange[5] {
        0b00001 0b010 0x00000001
      }
      AttributeID[3] {
        0b00001 0b001 0x000C
      }
      AttributeIDRange[5] {
        0b00001 0b010 0x01000102
      }
      AttributeID[3] {
        0b00001 0b001 0x0301
      }    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Part 2 -- Sent from SDP server to SDP client */
SdpSDP_ServiceSearchAttributeResponse[384] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x017B
  }
  AttributeListByteCount[2] {
    0x0170
  }
  AttributeLists[368] {
    DataElementSequence[487] {
      0b00110 0b110 0x01E4
      DataElementSequence[178] {
        0b00110 0b101 0xB0
        Attribute[8] {
          AttributeID[3] {
            0b00001 0b001 0x0000
          }
          AttributeValue[5] {
```

```
            /* service record handle */
            0b00001 0b010 0xhhhhhhhh
          }
        }
        Attribute[10] {
          AttributeID[3] {
            0b00001 0b001 0x0001
          }
AttributeValue[7] {
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* SynchronisationServiceClassID */
                0b00011 0b010 0xssssssss
              }
            }
          }
        }
        Attribute[35] {
          AttributeID[3] {
            0b00001 0b001 0x000C
          }
          AttributeValue[32] {
            /* IconURL; '*' replaced by client application */
            0b01000 0b101 0x1E
            "http://Synchronisation/icons/*"
          }
        }
        Attribute[22] {
          AttributeID[3] {
            0b00001 0b001 0x0100
          }
          AttributeValue[19] {
            /* service name */
            0b00100 0b101 0x11
            "Address Book Sync"
          }
        }
        Attribute[59] {
          AttributeID[3] {
            0b00001 0b001 0x0101
          }
          AttributeValue[56] {
            /* service description */
            0b00100 0b101 0x36
            "Synchronisation Service for"
            " vCard Address Book Entries"
          }
        }
        Attribute[37] {
          AttributeID[3] {
            0b00001 0b001 0x0102
          }
          AttributeValue[34] {
            /* service provider */
            0b00100 0b101 0x20
            "Synchronisation Specialists Inc."
```



LifeScan, Inc. Ex. 1019
Page 1241 of 1420
IPR2025-00103

```
        }
      }
      Attribute[5] {
        AttributeID[3] {
          0b00001 0b001 0x0301
        }
        AttributeValue[2] {
          /* Supported Data Store 'phonebook' */
          0b00001 0b000 0x01
        }
      }
    }
    DataElementSequence[175] {
      0b00110 0b101 0xAD
      Attribute[8] {
        AttributeID[3] {
          0b00001 0b001 0x0000
        }
        AttributeValue[5] {
          /* service record handle */
          0b00001 0b010 0xmmmmmmmm
        }
      }
      Attribute[10] {
        AttributeID[3] {
          0b00001 0b001 0x0001
        }
                  AttributeValue[7] {
          DataElementSequence[7] {
            0b00110 0b101 0x05
            UUID[5] {
              /* SynchronisationServiceClassID */
              0b00011 0b010 0xssssssss
            }
          }
        }
      }
      Attribute[35] {
        AttributeID[3] {
          0b00001 0b001 0x000C
        }
        AttributeValue[32] {
          /* IconURL; '*' replaced by client application */
          0b01000 0b101 0x1E
          "http://Synchronisation/icons/*"
        }
      }
      Attribute[21] {
        AttributeID[3] {
          0b00001 0b001 0x0100
        }
        AttributeValue[18] {
          /* service name */
          0b00100 0b101 0x10
          "Appointment Sync"
        }
      }
```



LifeScan, Inc. Ex. 1019
Page 1242 of 1420
IPR2025-00103

```
        Attribute[57] {
          AttributeID[3] {
            0b00001 0b001 0x0101
          }
          AttributeValue[54] {
            /* service description */
            0b00100 0b101 0x34
            "Synchronisation Service for"
            " vCal Appointment Entries"
          }
        }
        Attribute[37] {
          AttributeID[3] {
            0b00001 0b001 0x0102
          }
          AttributeValue[34] {
            /* service provider */
            0b00100 0b101 0x20
            "Synchronisation Specialists Inc."
          }
        }
        Attribute[5] {
          AttributeID[3] {
            0b00001 0b001 0x0301
          }
          AttributeValue[2] {
            /* Supported Data Store 'calendar' */
            0b00001 0b000 0x03
          }
        }
      }
    /* } Data element sequence of attribute lists */
    /* is not completed in this PDU. */
  }
  ContinuationState[9] {
    /* 8 bytes of continuation state */
    0x08 0xzzzzzzzzzzzzzzzz
  }
}

/* Part 3 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[41] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xwwww
  }
  ParameterLength[2] {
    0x0024
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
```



LifeScan, Inc. Ex. 1019
Page 1243 of 1420
IPR2025-00103

```
      }
    }
  }
  MaximumAttributeByteCount[2] {
    0x0180
  }
    AttributeIDList[18] {
    DataElementSequence[18] {
      0b00110 0b101 0x10
      AttributeIDRange[5] {
        0b00001 0b010 0x00000001
      }
      AttributeID[3] {
        0b00001 0b001 0x000C
      }
      AttributeIDRange[5] {
        0b00001 0b010 0x01000102
      }
      AttributeID[3] {
        0b00001 0b001 0x0301
      }
    }
  }
  ContinuationState[9] {
    /* same 8 bytes of continuation state */
    /* received in part 2 */
    0x08 0xzzzzzzzzzzzzzzzz
  }
}

Part 4 -- Sent from SDP server to SDP client

SdpSDP_ServiceSearchAttributeResponse[115] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xwwww
  }
  ParameterLength[2] {
    0x006E
  }
  AttributeListByteCount[2] {
    0x006B
  }
  AttributeLists[107] {
    /* Continuing the data element sequence of */
    /* attribute lists begun in Part 2. */
      DataElementSequence[107] {
        0b00110 0b101 0x69
        Attribute[8] {
          AttributeID[3] {
            0b00001 0b001 0x0000
          }
          AttributeValue[5] {
            /* service record handle */
            0b00001 0b010 0xffffffff
```

# GENERIC ACCESS PROFILE

***This profile defines the generic procedures related to discovery of Bluetooth devices (idle mode procedures) and link management aspects of connecting to Bluetooth 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.***



LifeScan, Inc. Ex. 1019
Page 1245 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1246 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 176**
1.1 Scope ..... 176
1.2 Symbols and conventions ..... 176
1.2.1 Requirement status symbols ..... 176
1.2.2 Signaling diagram conventions ..... 177
1.2.3 Notation for timers and counters ..... 177
**2 Profile overview ..... 178**
2.1 Profile stack ..... 178
2.2 Configurations and roles ..... 178
2.3 User requirements and scenarios ..... 179
2.4 Profile fundamentals ..... 180
2.5 Conformance ..... 180
**3 User interface aspects ..... 181**
3.1 The user interface level ..... 181
3.2 Representation of Bluetooth parameters ..... 181
3.2.1 Bluetooth device address (BD_ADDR) ..... 181
3.2.2 Bluetooth device name (the user-friendly name) ..... 181
3.2.3 Bluetooth passkey (Bluetooth PIN) ..... 182
3.2.4 Class of Device ..... 183
3.3 Pairing ..... 184
**4 Modes ..... 185**
4.1 Discoverability modes ..... 185
4.1.1 Non-discoverable mode ..... 186
4.1.2 Limited discoverable mode ..... 186
4.1.3 General discoverable mode ..... 187
4.2 Connectability modes ..... 189
4.2.1 Non-connectable mode ..... 189
4.2.2 Connectable mode ..... 189
4.3 Bondable modes ..... 191
4.3.1 Non-bondable mode ..... 191
4.3.2 Bondable mode ..... 191
**5 Security Aspects ..... 193**
5.1 Authentication ..... 193
5.1.1 Purpose ..... 193
5.1.2 Term on UI level ..... 193
5.1.3 Procedure ..... 194



LifeScan, Inc. Ex. 1019
Page 1247 of 1420
IPR2025-00103

5.1.4 Conditions ..... 194
5.2 Security modes ..... 194
5.2.1 Legacy Security Modes ..... 195
5.2.2 Security mode 4 (service level enforced security) ..... 197
**6 Idle Mode Procedures ..... 212**
6.1 General inquiry ..... 212
6.1.1 Purpose ..... 212
6.1.2 Term on UI level ..... 212
6.1.3 Description ..... 213
6.1.4 Conditions ..... 213
6.2 Limited inquiry ..... 213
6.2.1 Purpose ..... 213
6.2.2 Term on UI level ..... 214
6.2.3 Description ..... 214
6.2.4 Conditions ..... 214
6.3 Name discovery ..... 215
6.3.1 Purpose ..... 215
6.3.2 Term on UI level ..... 215
6.3.3 Description ..... 216
6.3.4 Conditions ..... 217
6.4 Device discovery ..... 217
6.4.1 Purpose ..... 217
6.4.2 Term on UI level ..... 217
6.4.3 Description ..... 218
6.4.4 Conditions ..... 218
6.5 Bonding ..... 218
6.5.1 Purpose ..... 218
6.5.2 Term on UI level ..... 218
6.5.3 Description ..... 219
6.5.4 Conditions ..... 220
**7 Establishment procedures ..... 222**
7.1 Link establishment ..... 222
7.1.1 Purpose ..... 222
7.1.2 Term on UI level ..... 222
7.1.3 Description ..... 223
7.1.4 Conditions ..... 224
7.2 Channel establishment ..... 225
7.2.1 Purpose ..... 225



LifeScan, Inc. Ex. 1019
Page 1248 of 1420
IPR2025-00103

7.2.2 Term on UI level ....225
7.2.3 Description ....225
7.2.4 Conditions ....226
7.3 Connection establishment ....227
7.3.1 Purpose ....227
7.3.2 Term on UI level ....227
7.3.3 Description ....227
7.3.4 Conditions ....228
7.4 Establishment of additional connection ....228

**8 Extended Inquiry Response Data Format ....229**
8.1 EIR Data Type Definitions ....230
8.1.1 Service Class UUIDs ....230
8.1.2 Local Name ....231
8.1.3 Flags ....231
8.1.4 Manufacturer Specific Data ....232
8.1.5 TX Power Level ....232
8.1.6 Secure Simple Pairing Out of Band (OOB) ....232
8.2 Example Extended Inquiry Response ....233

**9 Definitions ....235**
9.1 General definitions ....235
9.2 Connection-related definitions ....235
9.3 Device-related definitions ....236
9.4 Procedure-related definitions ....237
9.5 Security-related definitions ....237

**10 Appendix A (Normative): Timers and constants ....239**

**11 Appendix B (Informative): Information Flows of Related Procedures ....240**
11.1 LMP-authentication ....240
11.2 LMP-pairing ....241
11.3 Service discovery ....242

**12 References ....243**



LifeScan, Inc. Ex. 1019
Page 1249 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1250 of 1420
IPR2025-00103

# FOREWORD

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 Bluetooth air interface.



LifeScan, Inc. Ex. 1019
Page 1251 of 1420
IPR2025-00103

# 1 INTRODUCTION

## 1.1 SCOPE

The purpose of the Generic Access Profile is:

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 Bluetooth devices, and that multi-profile 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.

## 1.2 SYMBOLS AND CONVENTIONS

### 1.2.1 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.

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.



LifeScan, Inc. Ex. 1019
Page 1252 of 1420
IPR2025-00103

### 1.2.2 Signaling diagram conventions

The following arrows are used in diagrams describing procedures:

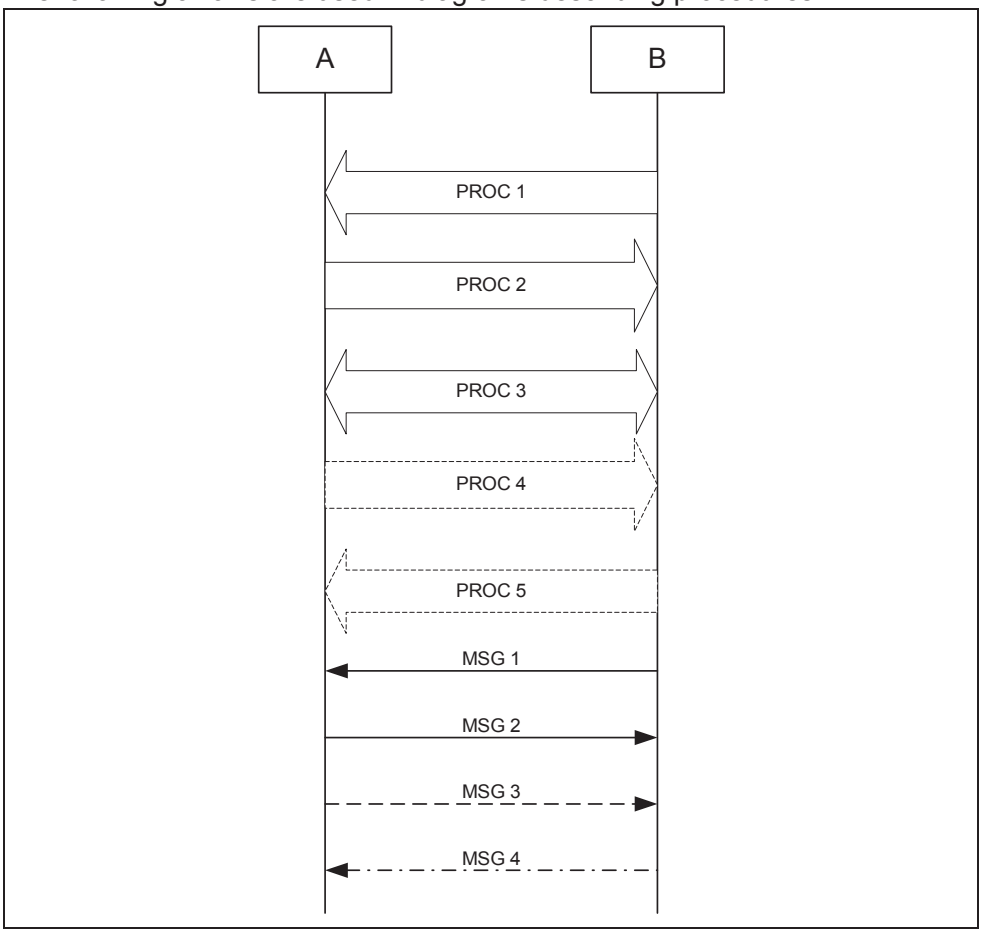



*Figure 1.1: Arrows used in signaling diagramsF*

In the figure above, the following cases are shown: PROC1 is a sub-procedure initiated by B. PROC2 is a sub-procedure initiated by A. PROC3 is a sub-procedure where the initiating side is undefined (may be both A or B). Dashed arrows denote optional steps. PROC4 indicates an optional sub-procedure initiated by A, and PROC5 indicates an optional sub-procedure 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.

### 1.2.3 Notation for timers and counters

Timers are introduced specific to this profile. To distinguish them from timers used in the Bluetooth protocol specifications and other profiles, these timers are named in the following format: '$T_{GAP}$*(nnn)*'.



LifeScan, Inc. Ex. 1019
Page 1253 of 1420
IPR2025-00103

# 2 PROFILE OVERVIEW

## 2.1 PROFILE STACK

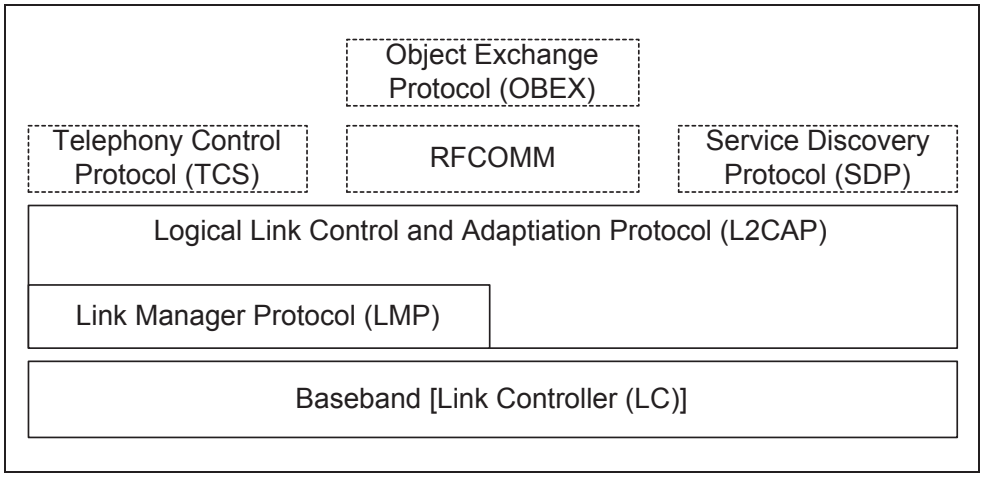



*Figure 2.1: Profile stack covered by this profile.*

The main purpose of this profile is to describe the use of the lower layers of the Bluetooth protocol stack (LC and LMP). To describe security related alternatives, also higher layers (L2CAP, RFCOMM and OBEX) are included.

## 2.2 CONFIGURATIONS AND ROLES

For the descriptions in this profile of the roles that the two devices involved in a Bluetooth communication can take, 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. 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.



LifeScan, Inc. Ex. 1019
Page 1254 of 1420
IPR2025-00103

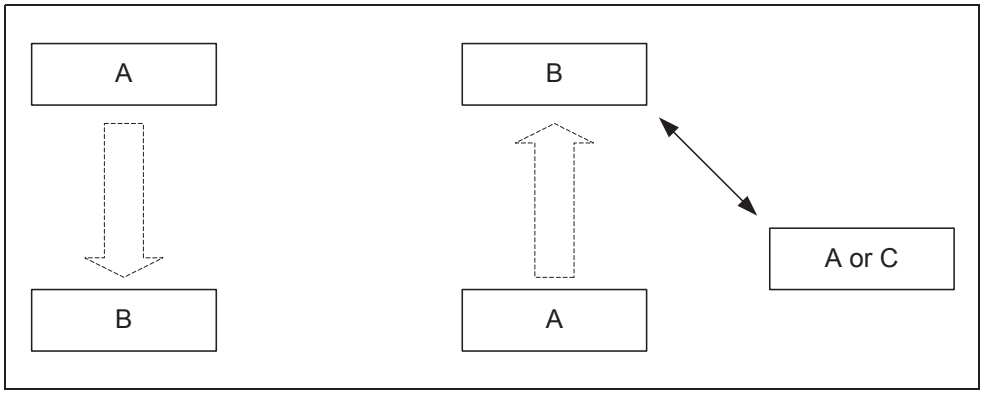



*Figure 2.2: This profile covers procedures initiated by one device (A) towards another device (B) that may or may not have an existing Bluetooth link active.*

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.

## 2.3 USER REQUIREMENTS AND SCENARIOS

The Bluetooth user should in principle be able to connect a Bluetooth device to any other Bluetooth device. Even if the two connected devices don't share any common application, it should be possible for the user to find this out using basic Bluetooth 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 Bluetooth capabilities by different names on the user interface level or implement basic procedures to be executed in different orders.



LifeScan, Inc. Ex. 1019
Page 1255 of 1420
IPR2025-00103

## 2.4 PROFILE FUNDAMENTALS

This profile states the requirements on names, values and coding schemes used for names of parameters and procedures experienced on the user interface level.

This profile 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.

This profile defines the general procedures that can be used for discovering identities, names and basic capabilities of other Bluetooth devices that are in a mode where they can be discoverable. Only procedures where no channel or connection establishment is used are specified.

This profile defines the general procedure for how to create bonds (i.e. dedicated exchange of link keys) between Bluetooth devices.

This profile describes the general procedures that can be used for establishing connections to other Bluetooth devices that are in mode that allows them to accept connections and service requests.

## 2.5 CONFORMANCE

Bluetooth devices that do not conform to any other Bluetooth profile shall conform to this profile to ensure basic interoperability.

Bluetooth devices that conform to another Bluetooth profile may use adaptations of the generic procedures as specified by that other profile. They shall, however, be compatible with devices compliant to this profile at least on the level of the supported generic procedures.

If conformance to this profile is claimed, all capabilities indicated mandatory for this profile shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the Bluetooth certification program.



LifeScan, Inc. Ex. 1019
Page 1256 of 1420
IPR2025-00103

# 3 USER INTERFACE ASPECTS

## 3.1 THE USER INTERFACE LEVEL

In the context of this specification, the user interface level refers to places (such as displays, dialog boxes, manuals, packaging, advertising, etc.) where users of Bluetooth devices encounters names, values and numerical representation of Bluetooth terminology and parameters.

This profile specifies the generic terms that should be used on the user interface level.

## 3.2 REPRESENTATION OF BLUETOOTH PARAMETERS

### 3.2.1 Bluetooth device address (BD_ADDR)

#### *3.2.1.1 Definition*

BD_ADDR is the address used by a Bluetooth device as defined in [2]. It is received from a remote device during the device discovery procedure.

#### *3.2.1.2 Term on user interface level*

When the Bluetooth address is referred to on UI level, the term ’Bluetooth Device Address’ should be used.

#### *3.2.1.3 Representation*

On BB level the BD_ADDR is represented as 48 bits [2].

On the UI level the Bluetooth address shall be represented as 12 hexadecimal characters, possibly divided into sub-parts separated by’:’.
(E.g., ’000C3E3A4B69’ or ’00:0C:3E:3A:4B:69’.) At UI level, any number shall have the MSB -> LSB (from left to right) ’natural’ ordering (e.g., the number ’16’ shall be shown as ’0x10’).

### 3.2.2 Bluetooth device name (the user-friendly name)

#### *3.2.2.1 Definition*

The Bluetooth device name is the user-friendly name that a Bluetooth device presents itself with. It is a character string returned in LMP_name_res as response to a LMP_name_req.



LifeScan, Inc. Ex. 1019
Page 1257 of 1420
IPR2025-00103

#### *3.2.2.2 Term on user interface level*

When the Bluetooth device name is referred to on UI level, the term 'Bluetooth Device Name' should be used.

#### *3.2.2.3 Representation*

The Bluetooth device name can be up to 248 bytes maximum according to [3]. It shall be encoded according to UTF-8 (i.e. name entered on UI level may be down to 82 characters outside the Unicode range 0x00-0x7F are used).

A device can not expect that a general remote device is able to handle more than the first 40 characters of the Bluetooth device name. If a remote device has limited display capabilities, it may use only the first 20 characters.

### 3.2.3 Bluetooth passkey (Bluetooth PIN)

#### *3.2.3.1 Definition*

The Bluetooth PIN is used to authenticate two Bluetooth devices (that have not previously exchanged link keys) to each other and create a trusted relationship between them. The PIN is used in the pairing procedure (see Section 11.2 on page 241) to generate the initial link key that is used for further authentication.

The PIN may be entered on UI level but may also be stored in the device; e.g. in the case of a device without sufficient MMI for entering and displaying digits.

#### *3.2.3.2 Terms at user interface level*

When the Bluetooth PIN is referred to on UI level, the term 'Bluetooth Passkey' should be used.

#### *3.2.3.3 Representation*

The Bluetooth PIN has different representations on different levels. PINBB is used on baseband level, and PINUI is used on user interface level. PINBB is the PIN used by [1] for calculating the initialization key during the pairing procedure. PINUI is the character representation of the PIN that is entered on UI level. The transformation from PINUI to PINBB shall be according to UTF-8. According to [1], PINBB can be 128 bits (16 bytes).

PIN codes may be up to 16 characters. In order to take advantage of the full level of security all PINs should be 16 characters long. Variable PINs should be composed of alphanumeric characters chosen from within the Unicode range 0x00-0x7F. If the PIN contains any decimal digits these shall be encoded using the Unicode Basic Latin characters (e.g. code points 0x30 to 0x39) (Note 1).



LifeScan, Inc. Ex. 1019
Page 1258 of 1420
IPR2025-00103

For compatibility with devices with numeric keypads fixed PINs shall be composed of only decimal digits, and variable PINS may be composed of only decimal digits.

If a device supports entry of characters outside the Unicode range 0x00-0x7F other Unicode code points may be used (Note 2), except the Halfwidth and Fullwidth Forms from within the Unicode range FF00 - FFEF shall not be used (Note 3).

**Examples:**

| **User-entered code** | **Corresponding $PIN_{BB}$[0..length-1] (value as a sequence of octets in hexadecimal notation)** |
|---|---|
| '0196554200906493' | length = 16, value = 0x30 0x31 0x39 0x36 0x35 0x35 0x34 0x32 0x30 0x30 0x39 0x30 0x36 0x34 0x39 0x33 |
| 'Børnelitteratur' | length = 16, value = 0x42 0xC3 0xB8 0x72 0x6e 0x65 0x6c 0x69 0x74 0x74 0x65 0x72 0x61 0x74 0x75 0x72 |

Note 1: This is to prevent interoperability problems since there are decimal digits at other code points (e.g. the Fullwidth digits at code points 0xff10 to 0xff19).

Note 2: Unicode characters outside the Basic Latin range (0x00 - 0x7F) encode to multiple bytes, therefore when characters outside the Basic Latin range are used the maximum number of characters in the PINUI will be less than 16. The second example illustrates a case where a 16 character string encodes to 15 bytes because the character ø is outside the Basic Latin range and encodes to two bytes (0xC3 0xB8).

Note 3: This is to prevent interoperability problems since the Halfwidth and Fullwidth forms contain alternative variants of ASCII, Katakana, Hangul, punctuation and symbols. All of the characters in the Halfwidth and Fullwidth forms have other more commonly used Unicode code points.

### 3.2.4 Class of Device

#### *3.2.4.1 Definition*

Class of device is a parameter received during the device discovery procedure, indicating the type of device and which types of service that are supported.



LifeScan, Inc. Ex. 1019
Page 1259 of 1420
IPR2025-00103

#### *3.2.4.2 Term on user interface level*

The information within the Class of Device parameter should be referred to as 'Bluetooth Device Class' (i.e. the major and minor device class fields) and 'Bluetooth Service Type' (i.e. the service class field). The terms for the defined Bluetooth Device Types and Bluetooth Service Types are defined in [12].

When using a mix of information found in the Bluetooth Device Class and the Bluetooth Service Type, the term 'Bluetooth Device Type' should be used.

#### *3.2.4.3 Representation*

The Class of device is a bit field and is defined in [12]. The UI-level representation of the information in the Class of device is implementation specific.

## 3.3 PAIRING

Two procedures are defined that make use of the pairing procedure defined on LMP level (LMP-pairing, see Section 11.2 on page 241). Either the user initiates the bonding procedure and enters the passkey with the explicit purpose of creating a bond (and maybe also a secure relationship) between two Bluetooth devices, or the user is requested to enter the passkey during the establishment procedure since the devices did not share a common link key beforehand. In the first case, the user is said to perform "bonding (with entering of passkey)" and in the second case the user is said to "authenticate using the passkey."



LifeScan, Inc. Ex. 1019
Page 1260 of 1420
IPR2025-00103

# 4 MODES

| Procedure | Ref. | Support |
|---|---|---|
| Discoverability modes: | 4.1 | |
| Non-discoverable mode | | C1 |
| Discoverable mode | | O |
| Limited discoverable mode | | C3 |
| General discoverable mode | | C4 |
| Connectability modes: | 4.2 | |
| Non-connectable mode | | O |
| Connectable mode | | M |
| Bondable modes: | 4.3 | |
| Non-bondable mode | | O |
| Bondable mode | | C2 |
| C1: If limited discoverable mode is supported, non-discoverable mode is mandatory, otherwise optional. | | |
| C2: If the bonding procedure is supported, support for bondable mode is mandatory, otherwise optional. | | |
| C3: If Discoverable mode is supported, support for Limited Discoverable mode is mandatory, otherwise optional. | | |
| C4: If Discoverable mode is supported, support for General Discoverable mode is mandatory, otherwise optional | | |

*Table 4.1: Conformance requirements related to modes defined in this section*

## 4.1 DISCOVERABILITY MODES

With respect to inquiry, a Bluetooth device shall be either in non-discoverable 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 [2].

When a Bluetooth device is in non-discoverable mode it does not respond to inquiry.

A Bluetooth 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 Bluetooth device is made discoverable, it may be unable to



LifeScan, Inc. Ex. 1019
Page 1261 of 1420
IPR2025-00103

respond to inquiry due to other baseband activity [2]. A Bluetooth device that does not respond to inquiry is called a silent device.

After being made discoverable, the Bluetooth 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 Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of discovery.

### 4.1.1 Non-discoverable mode

#### *4.1.1.1 Definition*

When a Bluetooth device is in non-discoverable mode, it shall never enter the INQUIRY_RESPONSE state.

#### *4.1.1.2 Term on UI-level*

Bluetooth device is ’non-discoverable’ or in ’non-discoverable mode’.

### 4.1.2 Limited discoverable mode

#### *4.1.2.1 Definition*

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 LIAC).

A Bluetooth device should not be in limited discoverable mode for more than $T_{GAP}(104)$. The scanning for the limited inquiry access code can be done either in parallel or in sequence with the scanning of the general inquiry access code. When in limited discoverable mode, one of the following options shall be used.

- *Parallel scanning*

  When a Bluetooth device is in limited discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

  If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.



LifeScan, Inc. Ex. 1019
Page 1262 of 1420
IPR2025-00103

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and non-Interlaced Inquiry scan is used.

In all cases the Bluetooth 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 Bluetooth 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 $T_{GAP}(101)$ and 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.

#### *4.1.2.2 Conditions*

When a device is in limited discoverable mode it shall set bit no 13 in the Major Service Class part of the Class of Device/Service field [12].

#### *4.1.2.3 Term on UI-level*

Bluetooth device is ’discoverable’ or in ’discoverable mode’.

### 4.1.3 General discoverable mode

#### *4.1.3.1 Definition*

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).



LifeScan, Inc. Ex. 1019
Page 1263 of 1420
IPR2025-00103

#### *4.1.3.2 Conditions*

When a Bluetooth device is in general discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Non-interlaced Inquiry scan is used.

In all cases the Bluetooth 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 a LIAC inquiry.

#### *4.1.3.3 Term on UI-level*

Bluetooth device is ’discoverable’ or in ’discoverable mode’.



LifeScan, Inc. Ex. 1019
Page 1264 of 1420
IPR2025-00103

## 4.2 CONNECTABILITY MODES

With respect to paging, a Bluetooth device shall be either in non-connectable mode or connectable mode. Paging is defined in [2].

When a Bluetooth device is in non-connectable mode it does not respond to paging. When a Bluetooth 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 Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of connection.

### 4.2.1 Non-connectable mode

#### *4.2.1.1 Definition*

When a Bluetooth device is in non-connectable mode it shall never enter the PAGE_SCAN state.

#### *4.2.1.2 Term on UI-level*

Bluetooth device is ’non-connectable’ or in ’non-connectable mode’.

### 4.2.2 Connectable mode

#### *4.2.2.1 Definition*

When a Bluetooth device is in connectable mode it shall periodically enter the PAGE_SCAN state. The device makes page scan using the Bluetooth device address, BD_ADDR. Connection speed is a trade-off between power consumption / 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 Bluetooth 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 the other device either does not have an estimate of the Bluetooth 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,



LifeScan, Inc. Ex. 1019
Page 1265 of 1420
IPR2025-00103

$T_{GAP}(106)$, 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 Bluetooth 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 Bluetooth device shall page scan for at least $T_{GAP}(101)$.

In all cases the Bluetooth 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 the following table:

| Scenario | Page Scan Interval | Page Scan Window | Scan Type |
|---|---|---|---|
| R0 (1.28s) | $T_{GAP}(107)$ | $T_{GAP}(107)$ | Normal scan |
| Fast R1 (100ms) | $T_{GAP}(106)$ | $T_{GAP}(101)$ | Interlaced scan |
| Medium R1 (1.28s) | $T_{GAP}(107)$ | $T_{GAP}(101)$ | Interlaced scan |
| Slow R1 (1.28s) | $T_{GAP}(107)$ | $T_{GAP}(101)$ | Normal scan |
| Fast R2 (2.56s) | $T_{GAP}(108)$ | $T_{GAP}(101)$ | Interlaced scan |
| Slow R2 (2.56s) | $T_{GAP}(108)$ | $T_{GAP}(101)$ | Normal scan |

*Table 4.2: Page scan parameters for connection speed scenarios*

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the connection time.

#### *4.2.2.2 Term on UI-level*

Bluetooth device is ’connectable’ or in ’connectable mode’.



LifeScan, Inc. Ex. 1019
Page 1266 of 1420
IPR2025-00103

## 4.3 BONDABLE MODES

With respect to bonding, a Bluetooth device shall be either in non-bondable mode or in bondable mode. In bondable mode the Bluetooth device accepts bonding initiated by the remote device, and in non-bondable mode it does not.

### 4.3.1 Non-bondable mode

#### *4.3.1.1 Definition*

When a Bluetooth device is in non-bondable mode it shall not accept a pairing request that results in bonding. Devices in non-bondable mode may accept connections that do not request or require bonding.

A device in non-bondable mode shall respond to a received LMP_in_rand with LMP_not_accepted with the reason *pairing not allowed*. An HCI-compliant host stack shall respond to an HCI_PIN_Code_Request event with the HCI_PIN_Code_Request_Negative_Reply command.

When both devices support Secure Simple Pairing and are in non-bondable mode, the local host shall respond to an IO capability request with the Authentication_Requirements parameter requesting dedicated bonding or general bonding with a negative response. An HCI-compliant host stack shall respond to an HCI_IO_Capabilities_Request event with an HCI_IO_Capabilities_ Request_Negative_Reply command.

#### *4.3.1.2 Term on UI-level*

Bluetooth device is ’non-bondable’ or in ’non-bondable mode’ or “does not accept bonding”.

### 4.3.2 Bondable mode

#### *4.3.2.1 Definition*

When a Bluetooth device is in bondable mode, and Secure Simple Pairing is not supported by either the local or remote device, the local device shall respond to a received LMP_in_rand with LMP_accepted (or with LMP_in_rand if it has a fixed PIN). An HCI-compliant host stack shall respond to an HCI_PIN_Code_Request event with the HCI_PIN_Code_Request_Reply command.

When both devices support Secure Simple Pairing, the local host shall respond to a user confirmation request with a positive response. An HCI-compliant host stack shall respond to an HCI_User_Confirmation_Request event with an HCI_User_Confirmation_Request_Reply command or an HCI_User_Passkey_Request event with an HCI_User_Passkey_Request_Reply command.



LifeScan, Inc. Ex. 1019
Page 1267 of 1420
IPR2025-00103

#### *4.3.2.2 Term on UI-level*

Bluetooth device is ’bondable’ or in ’bondable mode’ or “accepts bonding”.



LifeScan, Inc. Ex. 1019
Page 1268 of 1420
IPR2025-00103

# 5 SECURITY ASPECTS

| | Procedure | Ref. | Support |
|---|---|---|---|
| 1 | Authentication | 5.1 | M |
| **2** | **Security modes** | **5.2** | |
| | Security mode 1 | | Excluded |
| | Security mode 2 | | O.1 |
| | Security mode 3 | | Excluded |
| | Security mode 4 | | M |
| O.1: Security Mode 2 may only be used for backwards compatibility when the remote device does not support Secure Simple Pairing. | | | |

*Table 5.1: Conformance requirements related to the generic authentication procedure and the security modes defined in this section*

## 5.1 AUTHENTICATION

### 5.1.1 Purpose

The generic authentication procedure describes how the LMP-authentication and LMP-pairing procedures are used when authentication is initiated by one Bluetooth device towards another, depending on if a link key exists or not and if pairing is allowed or not.

### 5.1.2 Term on UI level

'Bluetooth authentication'.



LifeScan, Inc. Ex. 1019
Page 1269 of 1420
IPR2025-00103

### 5.1.3 Procedure

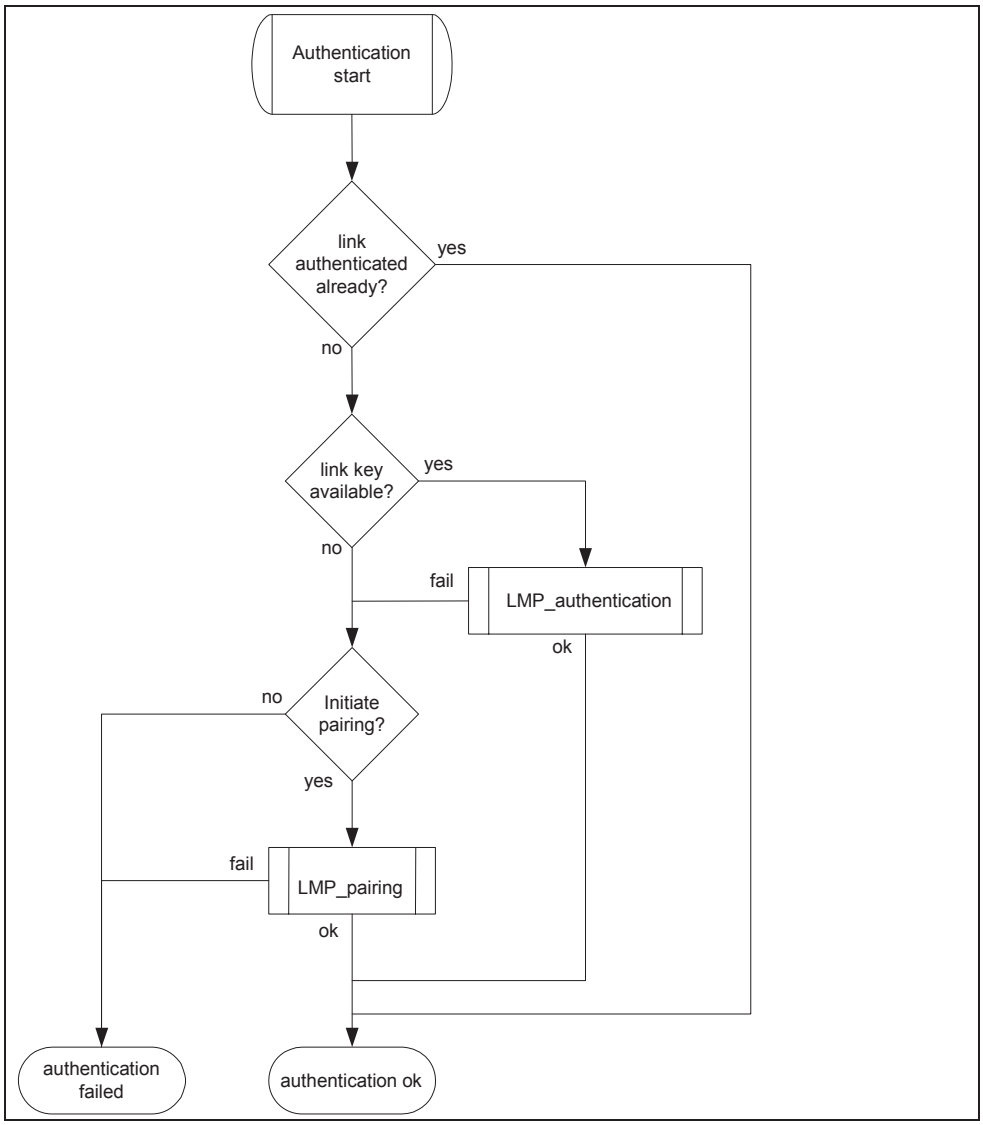



*Figure 5.1: Definition of the generic authentication procedure.*

### 5.1.4 Conditions

The local device shall initiate authentication after link establishment. The remote device may initiate security during or after link establishment.

## 5.2 SECURITY MODES

The following flow chart provides an overview of the channel establishment procedures.



LifeScan, Inc. Ex. 1019
Page 1270 of 1420
IPR2025-00103

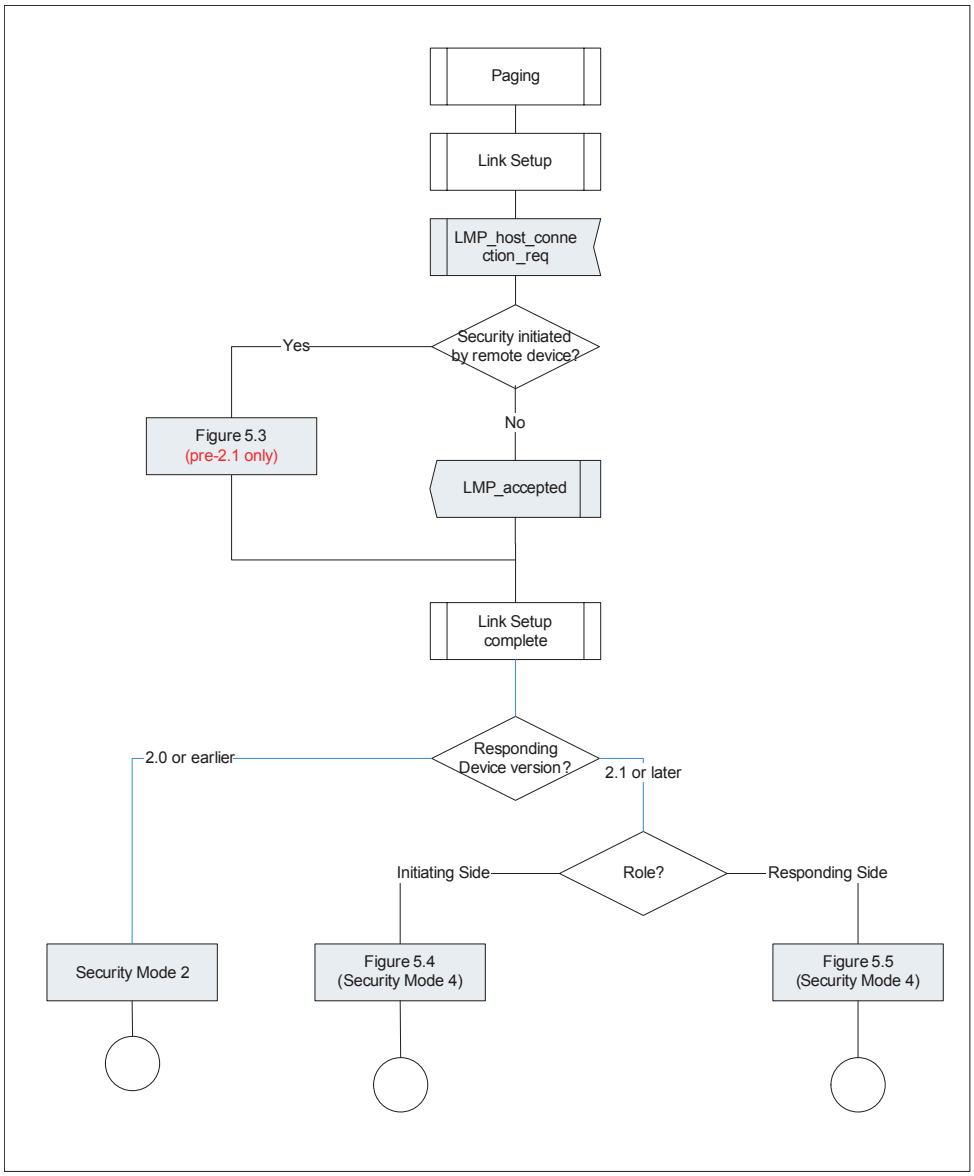



*Figure 5.2: Channel establishment with security.*

A device may support two security modes simultaneously: security mode 2 for backwards compatibility with remote devices that do not support Secure Simple Pairing and security mode 4 for devices that support Secure Simple Pairing

### 5.2.1 Legacy Security Modes

The following security modes apply only to devices compliant to Bluetooth Core Specifications v2.0 + EDR and earlier. This information is included only for reference.



LifeScan, Inc. Ex. 1019
Page 1271 of 1420
IPR2025-00103

#### 5.2.1.1 Security mode 1 (non-secure)

When a remote Bluetooth device is in security mode 1 it will never initiate any security procedure (i.e., it will never send LMP_au_rand, LMP_in_rand or LMP_encryption_mode_req).

#### 5.2.1.2 Security mode 2 (service level enforced security)

When a remote Bluetooth device is in security mode 2 it will 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. (The behavior of a device in security mode 2 is further described in [11].) Whether a security procedure is initiated or not depends on the security requirements of the requested channel or service.

A Bluetooth 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 point of view) as a special case of security mode 2 where no service has registered any security requirements.*

#### 5.2.1.3 Security modes 3 (link level enforced security)

When a remote Bluetooth device is in security mode 3 it will initiate security procedures before it sends LMP_setup_complete. (The behavior of a device in security mode 3 is as described in [3].)

A Bluetooth device in security mode 3 may reject the host connection request (respond with LMP_not_accepted to the LMP_host_connection_req) based on settings in the host (e.g. only communication with pre-paired devices allowed).



LifeScan, Inc. Ex. 1019
Page 1272 of 1420
IPR2025-00103

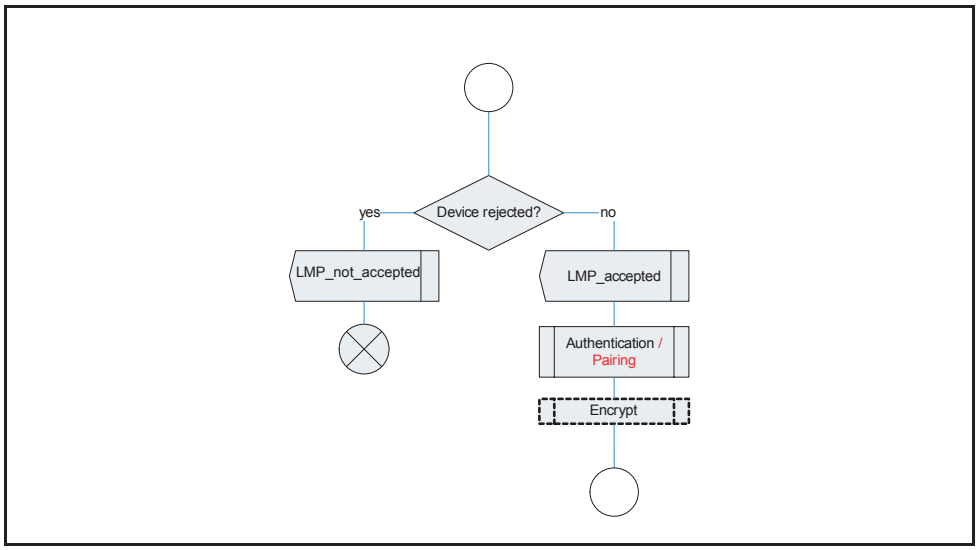



*Figure 5.3: Security mode 3 with a legacy remote device*

### 5.2.2 Security mode 4 (service level enforced security)

A Bluetooth device in security mode 4 shall classify the security requirements of its services using at least the following attributes (in order of decreasing security):

- Authenticated link key required
- Unauthenticated link key required
- No security required

An authenticated link key is a link key where either the numeric comparison, out-of-band or passkey entry simple pairing association models were used. An authenticated link key has protection against man-in-the-middle (MITM) attacks. To ensure that an authenticated link key is created during the Simple Pairing procedure, the Authentication_Requirements parameter should be set to one of the MITM Protection Required options. An *unauthenticated link key* is a link key where the just works Secure Simple Pairing association model was used. An unauthenticated link key does not have protection against MITM attacks.

When both devices support Secure Simple Pairing, GAP shall default to requiring an unauthenticated link key and enabling encryption. A profile or protocol may define services that require more security (e.g. an authenticated link key). or no security. To allow an unauthenticated link key to be created during the Simple Pairing procedure, the Authentication_Requirements parameter may be set to one of the MITM Protection Not Required options.

When the device is in Bondable Mode, it shall enable Secure Simple Pairing mode prior to entering Connectable Mode or establishing a link.



LifeScan, Inc. Ex. 1019
Page 1273 of 1420
IPR2025-00103

A Bluetooth device in security mode 4 shall respond to authentication requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons.

A Bluetooth device in security mode 4 enforces its security requirements before it attempts to access services offered by a remote device and before it grants access to services it offers to remote devices. Service access may occur via L2CAP channels or via channels established by protocols above L2CAP such as RFCOMM.

#### *5.2.2.1 Security for Access to Remote Service (Initiating Side)*

When the responding device does not support Secure Simple Pairing, it may disconnect the link while the initiating device is requesting the PIN to be entered by the user. This may occur due to the lack of an L2CAP channel being present for longer than an implementation-specific amount of time (e.g. a few seconds). When this occurs, the initiating device shall allow the user to complete entering the PIN and shall then re-page the remote device and restart the pairing procedure (see[Vol. 2, Part C] Section 4.2.2 on page 242) using the PIN entered.

##### *5.2.2.1.1 Pairing Required for Access to Remote Service*

When a Bluetooth device in security mode 4 initiates access to a remote service and a sufficient link-key is not available, the local device shall perform pairing procedures and enable encryption before sending a channel establishment request (L2CAP_ConnectReq or higher layer channel establishment request such as that of RFCOMM).

See section 5.2.2.8 on page 208 for details on determining whether or not a link key is sufficient.

If pairing does not succeed, the local device shall not send a channel establishment request. The local device may re-try pairing up to three (3) times. If pairing fails three consecutive times, the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

If the link key generated is not at least as good as the expected or required type, the local device shall fail the establishment and may disconnect the ACL link with error code 0x05 - Authentication Failure.

##### *5.2.2.1.2 Authentication Required for Access to Remote Service*

When a Bluetooth device in security mode 4 initiates access to a remote service and a sufficient link key is available for the remote device, it shall authenticate the remote device and enable encryption before sending a channel



LifeScan, Inc. Ex. 1019
Page 1274 of 1420
IPR2025-00103

establishment request (L2CAP_ConnectReq or higher layer channel establishment request such as that of RFCOMM).

See section 5.2.2.8 on page 208 for details on determining whether or not a link key is sufficient.

If authentication does not succeed, the local device shall not enable encryption and shall not send a channel establishment request. The host may then notify the user and offer to perform pairing.



LifeScan, Inc. Ex. 1019
Page 1275 of 1420
IPR2025-00103

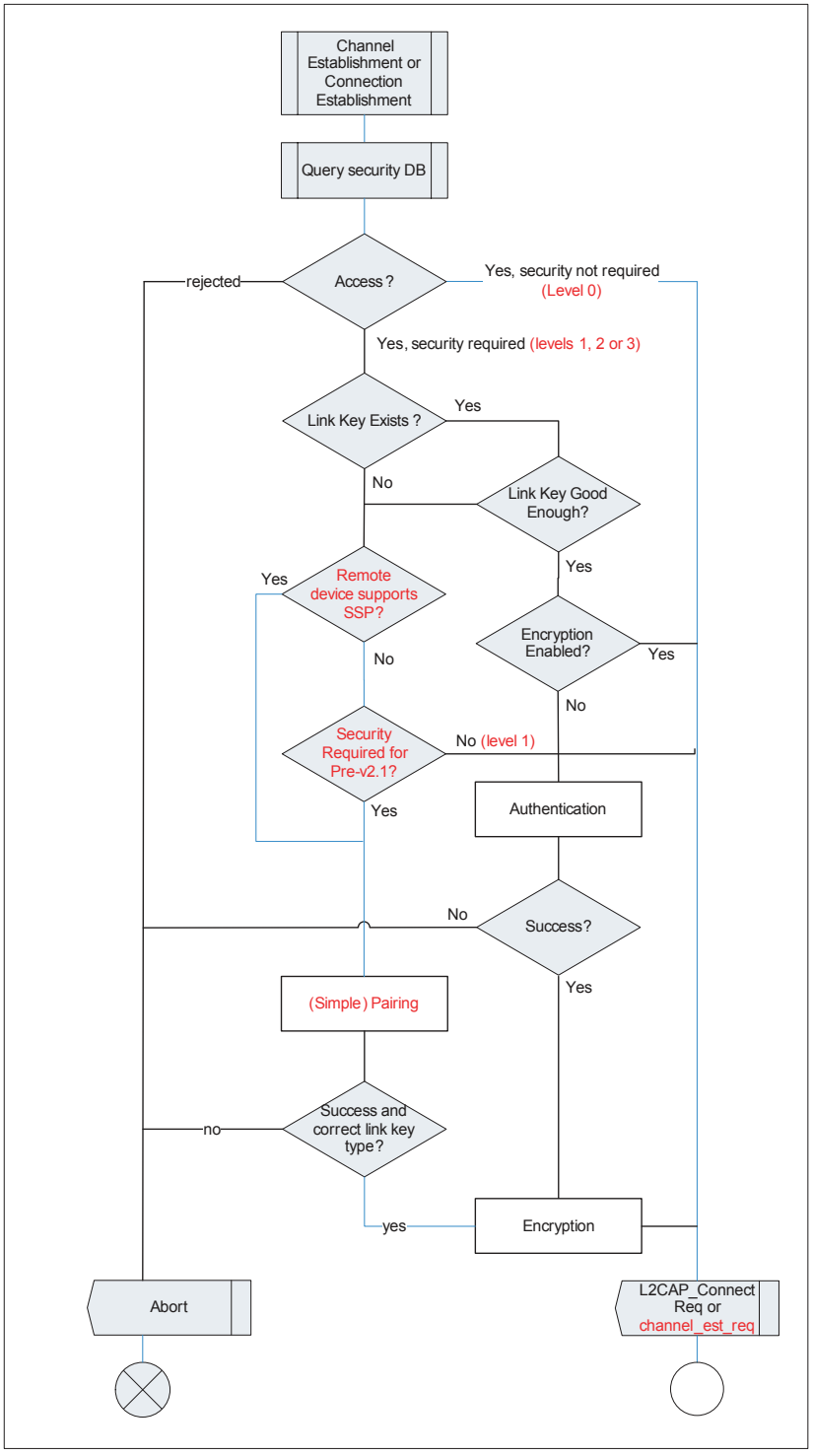



*Figure 5.4: Channel establishment using security mode 4 for initiating side*



LifeScan, Inc. Ex. 1019
Page 1276 of 1420
IPR2025-00103

#### *5.2.2.2 Security for Access to Local Service by Remote Device (Responding Side)*

When L2CAP is the channel establishment protocol being used for the requested service, an L2CAP_ConnectRsp signalling packet shall be sent by the responding device containing the result code 0x0001 (connection pending) following receipt of an L2CAP_ConnectReq and prior to initiating security procedures which can result in prompting the local user for input, e.g. pairing using a PIN or Secure Simple Pairing using either the Passkey entry or Numerical Comparison association models. This will stop the L2CAP RTX timer on the remote device (which may be as short as 1 second) and will invoke the ERTX timer on the remote device, which is a minimum duration of 60 seconds.

See [Vol. 3, Part A] Section 6.2 on page 73, for additional information on L2CAP RTX and ERTX timers. See also [Vol. 3, Part A] Section 4.3 on page 44, for additional information on the L2CAP_ConnectRsp signalling packet, and the defined result codes.

Higher layer channel establishment protocols should be designed to restrict timeouts to be 30 seconds or longer to allow for user input, or provide mechanisms to dynamically extend timeouts when user input may be required.

If authentication or pairing fails when a remote device is attempting to access a local service, the local device shall send a negative response to the channel establishment request (L2CAP_ConnectReq or channel_est_req) indicating a security issue if possible. If the channel establishment protocol is L2CAP, then the result code 0x0003 (connection refused - security block) shall be sent in the L2CAP_ConnectRsp signal.

If the remote device has indicated support for Secure Simple Pairing, a channel establishment request is received for a service other than SDP, and encryption has not yet been enabled, then the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

##### *5.2.2.2.1 Pairing Required for Access to Local Service by Remote Device*

When a remote device attempts to access a service offered by a Bluetooth device that isin security mode 4 and a sufficient link key exists, the local device shall authenticate the remote device after the channel establishment request is received but before a channel establishment confirmation (L2CAP_ConnectRsp with result code of 0x0000 or a higher level channel establishment confirmation such as that of RFCOMM) is sent.

See Section 5.2.2.6 for details on determining whether or not a link key is sufficient.

If pairing does not succeed, then the local device shall not send a channel establishment confirmation. The local device may retry pairing up to three (3)



LifeScan, Inc. Ex. 1019
Page 1277 of 1420
IPR2025-00103

times. If pairing fails three consecutive times, then the local device shall disconnect the ACL link with error code 0x05 - Authentication Failure.

If the link-key generated is not at least as good as the expected or required type, then the local device shall fail the channel establishment and may disconnect the ACL link with error code 0x05 - Authentication Failure.

#### *5.2.2.2.2 Authentication Required for Access to Local Service by Remote Device*

When a remote device attempts to access a service offered by a Bluetooth device, which is in security mode 4 and pairing is required due to the link key being absent or insufficient, the local device shall perform pairing procedures and enable encryption after the channel establishment request is received and before a channel establishment confirmation (L2CAP_ConnectRsp with result code of 0x0000 or a higher level channel establishment response such as that of RFCOMM) is sent.

See Section 5.2.2.6 for details on determining whether or not a link key is sufficient.

If authentication does not succeed, then the local device shall not send a channel establishment confirmation. The host may at this point notify the user and offer to perform pairing.

A Bluetooth device in security mode 4 shall respond to authentication and pairing requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons. However, authentication of the remote device shall be performed after the receipt of the channel establishment request is received, and before the channel establishment response is sent.



LifeScan, Inc. Ex. 1019
Page 1278 of 1420
IPR2025-00103

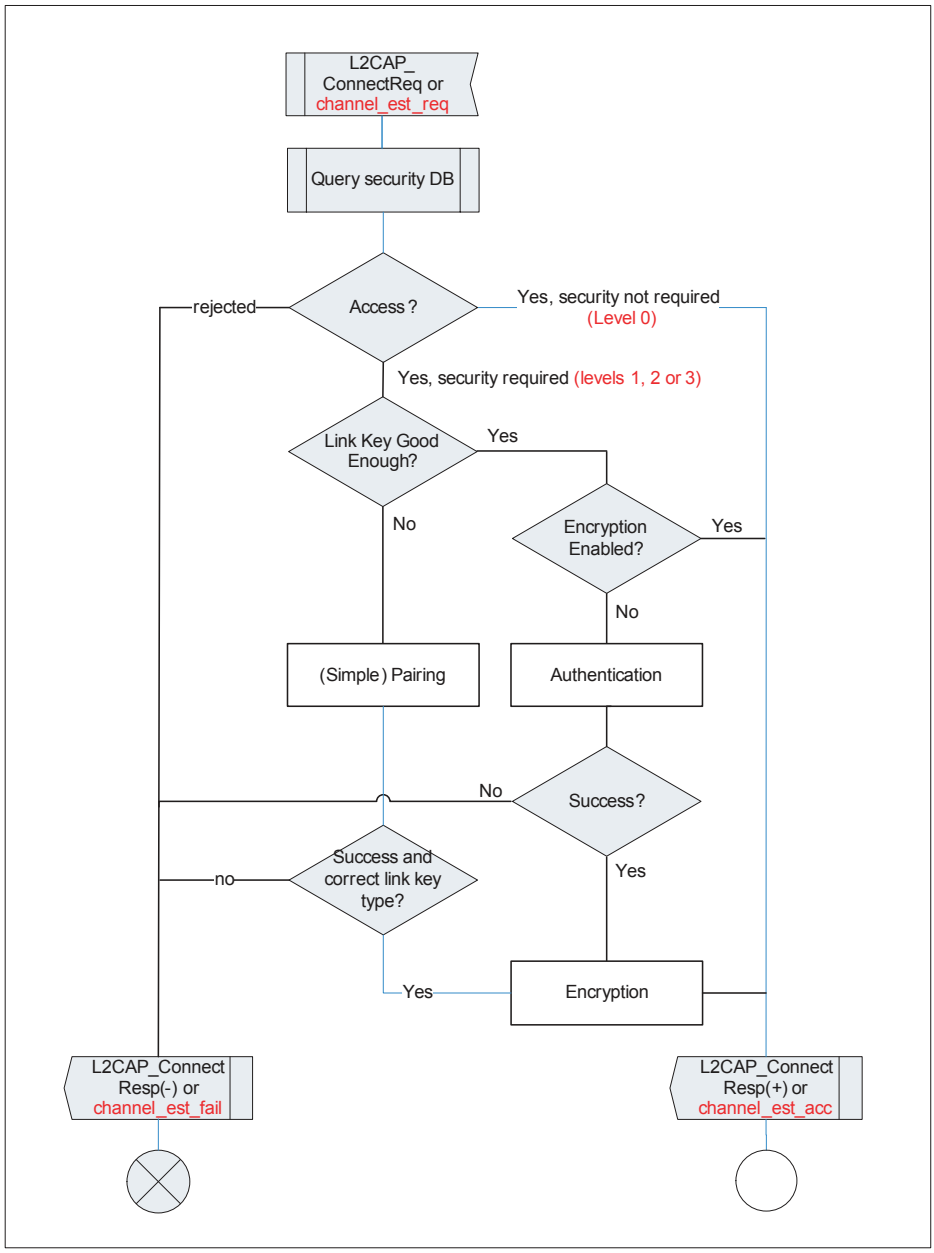



*Figure 5.5: Channel establishment using security mode 4 for the responding side*

### *5.2.2.3 Simple Pairing after Authentication Failure*

When both devices support Secure Simple Pairing all non-SDP connections are encrypted regardless of whether security was required or whether the devices are bonded or not. The initial connection between the two devices will result in a link key through Secure Simple Pairing. Depending on whether or not bonding was performed and the security policy of the initiating device, the

link key may or may not be stored. When the link key is stored, subsequent connections to the same device will use authentication but this may fail if the remote device has deleted the link key. The table below defines what shall be done depending on the type of the link key and whether bonding was performed or not.

| Link Key Type | Devices Bonded? | Action to take when Authentication Fails |
|---|---|---|
| Combination | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate pairing<br>• Option 2: Notify user and ask if pairing is ok<br>Option 2 is recommended. |
| Combination | Yes | Notify user of security failure |
| Unauthenticated | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate secure simple pairing<br>• Option 2: Notify user and ask if secure simple pairing is ok.<br>Option 1 is recommended. |
| Unauthenticated | Yes | Notify user of security failure |
| Authenticated | No | Depends on security policy of the device:<br>• Option 1: Automatically initiate secure simple pairing<br>• Option 2: Notify user and ask if secure simple pairing is ok<br>Option 2 is recommended. |
| Authenticated | Yes | Notify user of security failure |

*Table 5.1: Simple Pairing after Authentication Failure*

Note that non-bonded authenticated or unauthenticated link keys may be considered disposable by either device and may be deleted at any time.

#### *5.2.2.4 IO Capabilities*

Once a connection is established, if the Host determines that security is necessary and both devices support Secure Simple Pairing, the devices perform an IO capability exchange. The purpose of the IO capability exchange is to determine the authentication algorithm used in the Authentication Stage 1 phase of Simple Pairing.

The input and output capabilities are described in Table 5.2:

| Capability | Description |
|---|---|
| No input | Device does not have the ability to indicate 'yes' or 'no' |

*Table 5.2: User Input Capabilities*



LifeScan, Inc. Ex. 1019
Page 1280 of 1420
IPR2025-00103

| Capability | Description |
|---|---|
| Yes / No | Device has at least two buttons that are mapped easily to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see note 1 below). |
| Keyboard | Device has a numeric keyboard that can input the numbers '0' through '9' and a confirmation. Device also has two buttons that can be easily mapped to 'yes' and 'no' or the device has a mechanism whereby the user can indicate either 'yes' or 'no' (see note 1 below). |

*Table 5.2: User Input Capabilities*

Note: 'yes' could be indicated by pressing a button within a certain time limit otherwise 'no' would be assumed.

| Capability | Description |
|---|---|
| No output | Device does not have the ability to display or communicate a 6 digit decimal number |
| Numeric output | Device has the ability to display or communicate a 6 digit decimal number |

*Table 5.3: User Output Capabilities*

### *5.2.2.5 Mapping of Input / Output Capabilities to IO Capability*

The individual input and output capabilities are mapped to a single IO capability which is later used to determine which authentication algorithm will be used.

| Local input capacity / Local output capacity | **No Output** | **Numeric Output** |
|---|---|---|
| **No input** | NoInputNoOutput | DisplayOnly |
| **Yes / No** | NoInputNoOutput | DisplayYesNo |
| **Keyboard** | KeyboardOnly | DisplayYesNo |

*Table 5.4: IO Capability mapping*

When a device has OOB authentication information from the remote device, it will indicate it in the LMP_io_capability_res PDU. When either device has OOB information, the OOB authentication method will be used.

The Host may allow the Link Manager to ignore the IO capabilities and use the Numeric Comparison protocol with automatic accept by setting the Authentication_Requirements parameter to one of the MITM Protection *Not Required* options.



LifeScan, Inc. Ex. 1019
Page 1281 of 1420
IPR2025-00103

### 5.2.2.6 IO and OOB Capability Mapping to Authentication Stage 1 Method

Determining which association model to use in Authentication Stage 1 is performed in three steps. First, the devices look at the OOB Authentication Data Present parameter received in the remote IO capabilities. If either device has received OOB authentication data then the OOB association model is used. The event of receiving the OOB information is indicated by a device to its peer in the IO Capability Exchange step of simpleL pairing.

| Device A / Device B | **Has not received remote OOB authentication data** | **Has received remote OOB authentication table** |
|---|---|---|
| **Has not received remote OOB authentication data** | Use the IO capability mapping table | Use OOB association with<br>ra = 0<br>rb from OOB |
| **Has received remote OOB authentication data** | Use OOB association with<br>ra from OOB<br>rb = 0 | Use OOB association with<br>ra from OOB<br>rb from OOB |

*Table 5.5: IO and OOB capability mapping*

Second, if neither device has received OOB authentication data and if both devices have set the Authentication_Requirements parameter to one of the MITM Protection Not Required options, authentication stage 1 shall function as if both devices set their IO capabilities to DisplayOnly (e.g. Numeric comparison with automatic confirmation on both devices).

Finally, if neither device has received OOB authentication data and if one or both devices have set the Authentication_Requirements parameter to one of the *MITM Protection Required* options, the IO and OOB capabilities are mapped to the authentication stage 1 method as defined in the following table. A Host that has set the Authentication_Requirements parameter to one of the *MITM Protection Required* options shall verify that the resulting Link Key is an Authenticated Combination Key (see Section 7.7.24 in Host Controller Interface Functional Specification). The table also lists whether the combination key results in an authenticated or unauthenticated link key.



LifeScan, Inc. Ex. 1019
Page 1282 of 1420
IPR2025-00103

| Initiator A / B Responder | Display Only | DisplayYesNo | KeyboardOnly | NoInputNoOutput |
|---|---|---|---|---|
| **DisplayOnly** | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated | Numeric Comparison with automatic confirmation on device B only.<br><br>Unauthenticated | Passkey Entry: Responder Display, Initiator Input.<br><br>Authenticated | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated |
| **DisplayYesNo** | Numeric Comparison with automatic confirmation on device A only.<br><br>Unauthenticated | Numeric Comparison: Both Display, Both Confirm.<br><br>Authenticated | Passkey Entry: Responder Display, Initiator Input.<br><br>Authenticated | Numeric Comparison with automatic confirmation on device A only.<br><br>Unauthenticated |
| **Keyboard Only** | Passkey Entry: Initiator Display, Responder Input.<br><br>Authenticated | Passkey Entry: Initiator Display, Responder Input.<br><br>Authenticated | Passkey Entry: Initiator and Responder Input.<br><br>Authenticated | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated |
| **NoInputNoOutput** | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated | Numeric Comparison with automatic confirmation on device B only.<br><br>Unauthenticated | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated | Numeric Comparison with automatic confirmation on both devices.<br><br>Unauthenticated |

*Table 5.6: IO Capability Mapping to Authentication Stage 1*

Note: The "DisplayOnly" IO capability only provides unidirectional authentication.

### *5.2.2.7 Out of Band (OOB)*

An out of band mechanism may also be used to communicate discovery information as well as other information related to the pairing process.

The contents of the OOB data block are:

**Mandatory contents:**



LifeScan, Inc. Ex. 1019
Page 1283 of 1420
IPR2025-00103

- Length (2 bytes)
- BD_ADDR (6 bytes)

**Optional contents:**

- Class of Device (3 bytes)
- Simple Pairing Hash C (16 bytes)
- Simple Pairing Randomizer R (16 bytes)
- Local name (variable length)
- Other information

The length field includes all bytes in the OOB data block including the length field itself. The BD_ADDR will be a fixed field in the beginning of the OOB data block. Following the BD_ADDR will be zero or more EIR tag fields containing optional contents. The EIR tag format will be used for the optional contents.

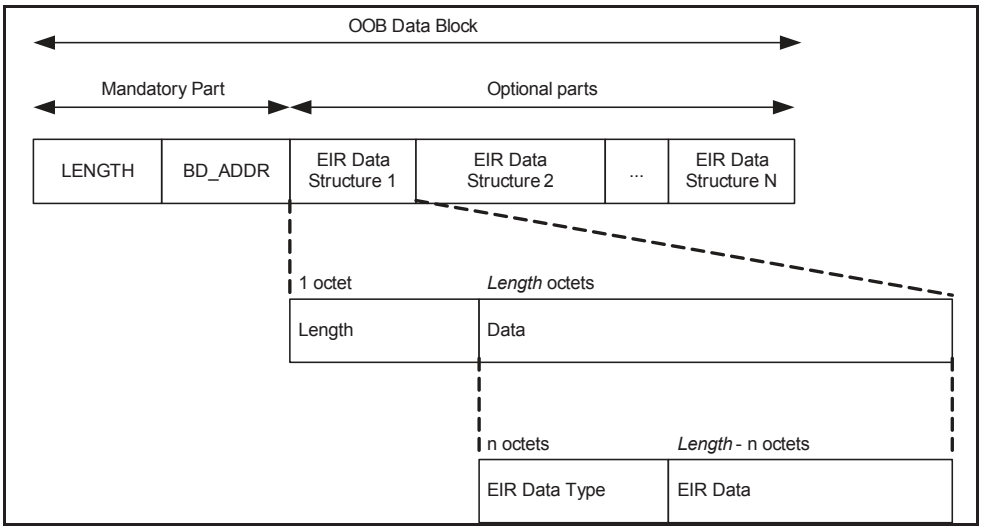



*Figure 5.6: OOB Data Block Format*

If Simple Pairing fails when one or both devices have OOB Authentication Data present, both devices shall discard the OOB Authentication Data and the device that originally initiated authentication shall re-initiate authentication. Note that although the user may get involved in authentication as defined by the IO capabilities of the two devices, falling back to the in-band authentication method will prevent deadlock conditions when one or both devices has stale OOB Authentication Data.

### *5.2.2.8 Security Database*

A Bluetooth device in security mode 4 shall classify the security requirements of its services using at least the following levels attributes (in order of decreas-



LifeScan, Inc. Ex. 1019
Page 1284 of 1420
IPR2025-00103

ing security) for use when pairing with remote devices supporting Secure Simple Pairing:

- Level 3, for services with the following attributes:
  MITM protection required
  Encryption required
  User interaction acceptable
- Level 2, for services with the following attributes:
  MITM protection not required
  Encryption required
- Level 1, for services with the following attributes:
  MITM protection not required
  Minimal user interaction desired

One additional level is permitted for SDP, and shall not be used for any other service:

- Level 0: Service requires the following:
  MITM protection not required
  No encryption required
  No user interaction required

The security level required for each service offered should be stored in a security database that is accessed to determine the type of link key that is required for to allow access to the respective service. Table 5.7 on page 209 shows the type of link key required for each security level for both remote devices that support Secure Simple Pairing (v2.1 + EDR remote devices) and for those that do not (pre-v2.1 + EDR remote devices).

| Security Level Required for Service | Link Key type required for remote devices | Link Key type required for pre-v2.1 remote devices | Comments |
|---|---|---|---|
| Level 3<br>• Strong MITM protection desired<br>• Encryption desired<br>• User interaction acceptable. | Authenticated | Combination (16 digit PIN recommended) | High Security |
| Level 2<br>• MITM protection not necessary<br>• Encryption desired | Unauthenticated | Combination | Medium Security |

*Table 5.7: Security Level mapping to link key requirements*



LifeScan, Inc. Ex. 1019
Page 1285 of 1420
IPR2025-00103

| Security Level Required for Service | Link Key type required for remote devices | Link Key type required for pre-v2.1 remote devices | Comments |
|---|---|---|---|
| Level 1<br>• MITM protection not necessary<br>• Encryption not necessary[1]<br>• Minimal user interaction desired. | Unauthenticated | None | Low Security |
| Level 0<br>• MITM protection not necessary<br>• Encryption not necessary<br>• No user interaction desired. | None | None | Permitted only for SDP |

*Table 5.7: Security Level mapping to link key requirements*

1. Though encryption is not necessary for the service for Level 1, this specification mandates the use of encryption when the remote device is v2.1+EDR for all services other than SDP.

An *authenticated* link key is a link key where either the numeric comparison, out-of-band or passkey entry simple pairing association models were used. An authenticated link key has protection against MITM attacks. To ensure that an authenticated link key is created during the Simple Pairing procedure, the Authentication_Requirements parameter should be set to one of the *MITM Protection Required* options.

An *unauthenticated* link key is a link key where the "Just Works" just works simple pairing association model was used (see [Vol. 1, Part A] Section 5.4 on page 59). An unauthenticated link key does not have protection against MITM attacks. To allow an unauthenticated link key to be created during the Simple Pairing procedure, the Authentication_Requirements parameter may be set to one of the *MITM Protection Not Required* options.

A combination link key is a link key where the v2.0 pairing mechanism was used to generate the link-key (see [Vol. 2, Part C] Section 4.2.2.4 on page 244).

When both devices support Secure Simple Pairing, GAP shall default to requiring an unauthenticated link key and enabling encryption. A profile or protocol may define services that require more security (for example, an authenticated link key) or no security in the case of SDP.

When the device is in Bondable Mode, it shall enable Secure Simple Pairing mode prior to entering Connectable Mode or establishing a link.



LifeScan, Inc. Ex. 1019
Page 1286 of 1420
IPR2025-00103

A Bluetooth device in security mode 4 shall respond to authentication and pairing requests during link establishment when the remote device is in security mode 3 for backwards compatibility reasons. See Section 5.2.1.3 for more information.

The remote Controller's and remote Host's support for Secure Simple Pairing shall be determined by the Link Manager Secure Simple Pairing (Host Support) feature bit.

A previously generated link key is considered "sufficient" if the link key type is of the type required for the service, or of a higher strength. Authenticated link keys are considered higher strength than Unauthenticated or Combination keys. Unauthenticated link keys are considered higher strength than Combination keys.



LifeScan, Inc. Ex. 1019
Page 1287 of 1420
IPR2025-00103

# 6 IDLE MODE PROCEDURES

The inquiry and discovery procedures described here are applicable only to the device that initiates them (A). The requirements on the behavior of B is according to the modes specified in Section 4 on page 185 and to [3].

| | Procedure | Ref. | Support |
|---|---|---|---|
| 1 | General inquiry | 6.1 | C1 |
| 2 | Limited inquiry | 6.2 | C1 |
| 3 | Name Discovery | 6.3 | O |
| 4 | Device discovery | 6.4 | O |
| 5 | Bonding | 6.5 | O |

C1: If initiation of bonding is supported, support for simple inquiry procedure is mandatory, otherwise optional.

(Note: support for LMP-pairing is mandatory [3].)

## 6.1 GENERAL INQUIRY

### 6.1.1 Purpose

The purpose of the general inquiry procedure is to provide the initiator with the Bluetooth device address, clock, Class of Device, page scan mode, and extended inquiry response information 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 General Inquiry Access Code). 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.

### 6.1.2 Term on UI level

'Bluetooth Device Inquiry.'



LifeScan, Inc. Ex. 1019
Page 1288 of 1420
IPR2025-00103

### 6.1.3 Description

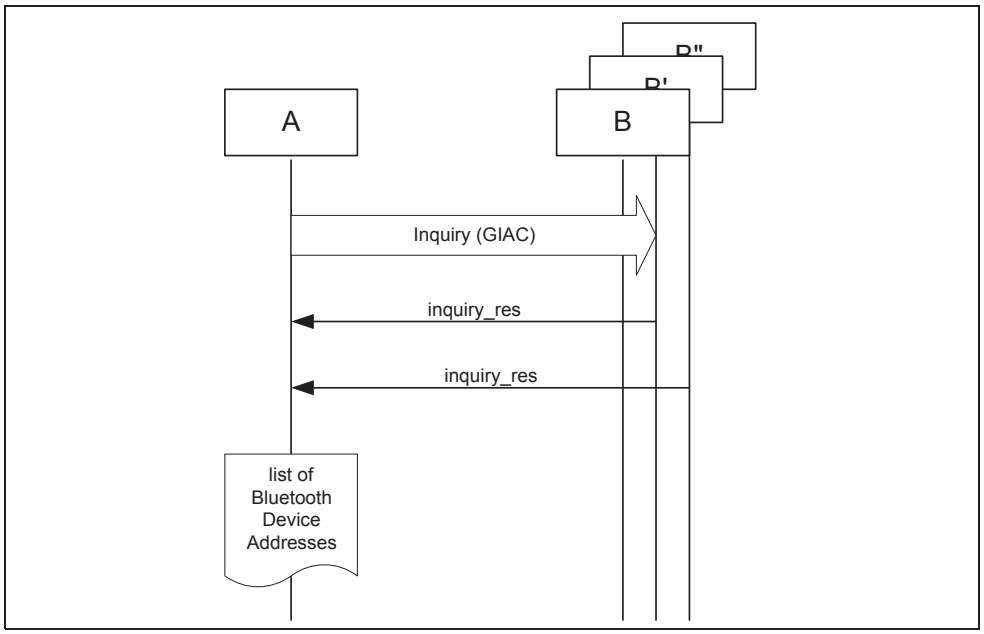



*Figure 6.1: General inquiry ,where B is a device in non-discoverable mode, B´ is a device in limited discoverable mode and B" is a device in general discoverable mode. (Note that all discoverable devices are discovered using general inquiry, independent of which discoverable mode they are in.)*

### 6.1.4 Conditions

[ID236]When general inquiry is initiated by a Bluetooth 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 Bluetooth device shall perform inquiry using the GIAC.

In order for Device A to receive inquiry responses, the remote devices in range have to be made discoverable (limited or general).

## 6.2 LIMITED INQUIRY

### 6.2.1 Purpose

The purpose of the limited inquiry procedure is to provide the initiator with the Bluetooth device address, clock, Class of Device, page scan mode, and extended inquiry response information 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 Inquiry Access Code, in addition to scanning for inquiry messages with the General Inquiry Access Code.



LifeScan, Inc. Ex. 1019
Page 1289 of 1420
IPR2025-00103

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 $T_{GAP}(103)$.

### 6.2.2 Term on UI level

'Bluetooth Device Inquiry'.

### 6.2.3 Description

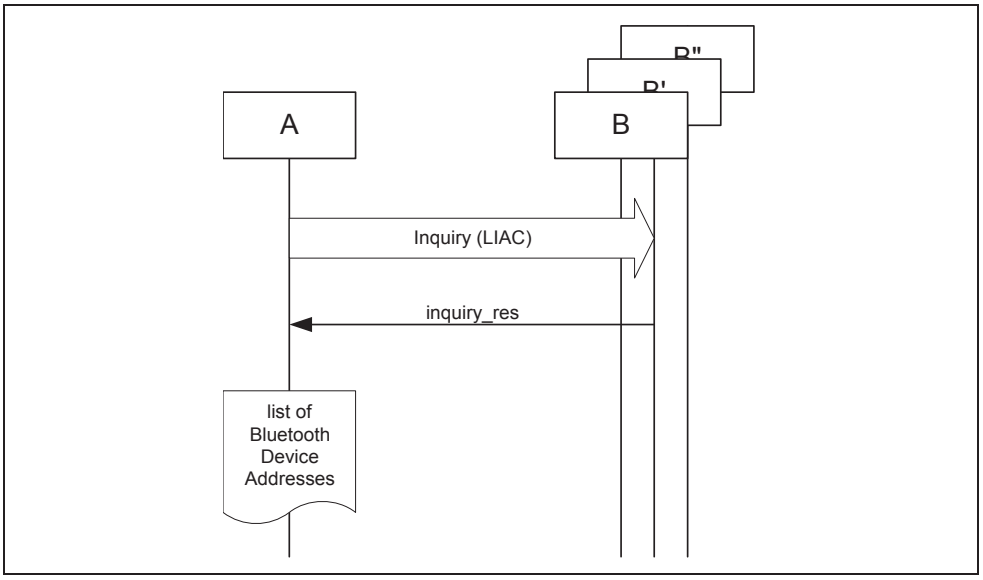



*Figure 6.2: Limited inquiry where B is a device in non-discoverable mode, B' is a device in limited discoverable mode and B" is a device in general discoverable mode. (Note that only limited discoverable devices can be discovered using limited inquiry.)*

### 6.2.4 Conditions

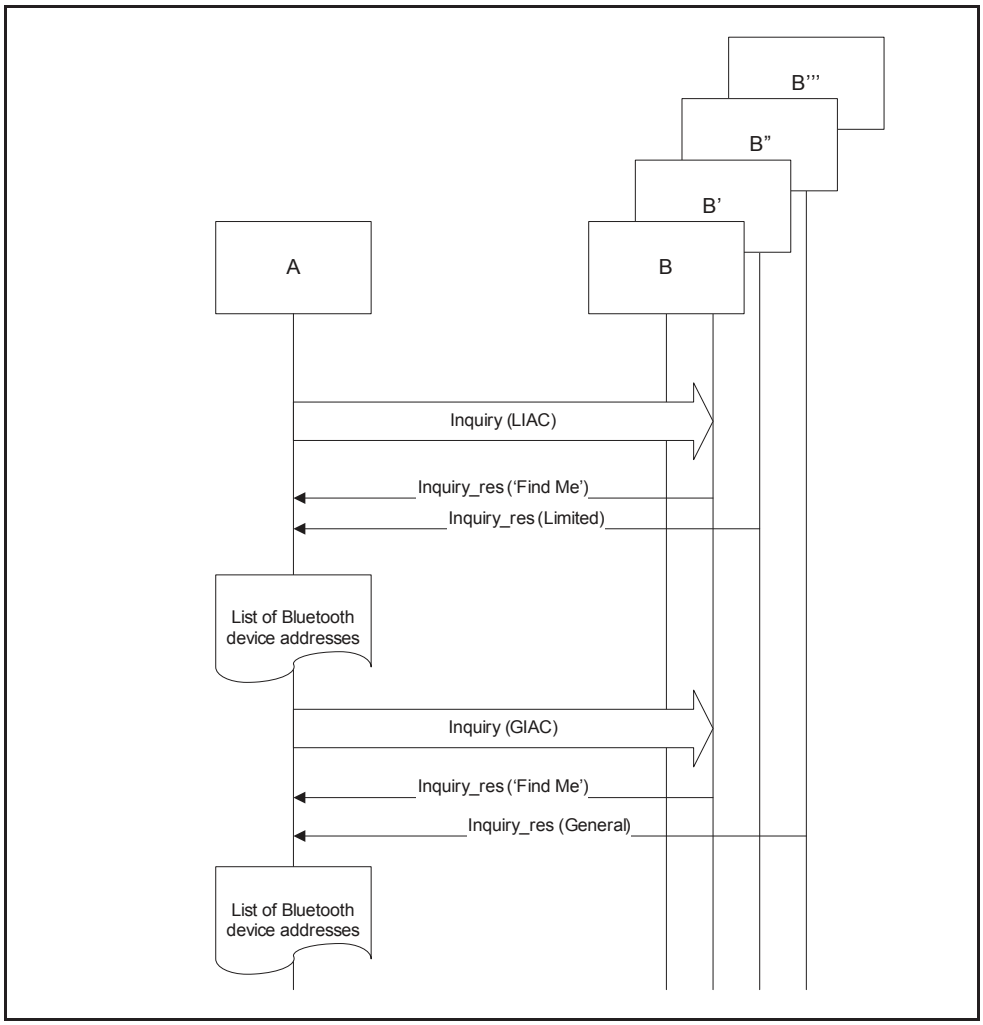
[ID237]When limited inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the LIAC.

In order for Device A to receive inquiry responses, the remote devices in range has to be made limited discoverable.



LifeScan, Inc. Ex. 1019
Page 1290 of 1420
IPR2025-00103

## 6.3 NAME DISCOVERY




### 6.3.1 Purpose

The purpose of name discovery is to provide the initiator with the Bluetooth Device Name of connectable devices (i.e. devices in range that will respond to paging).

### 6.3.2 Term on UI level

'Bluetooth Device Name Discovery'.



LifeScan, Inc. Ex. 1019
Page 1291 of 1420
IPR2025-00103

### 6.3.3 Description

#### *6.3.3.1 Name request*

Name request is the procedure for retrieving the Bluetooth Device Name from a connectable Bluetooth device. It is not necessary to perform the full link establishment procedure (see Section 7.1 on page 222) in order to just to get the name of another device.

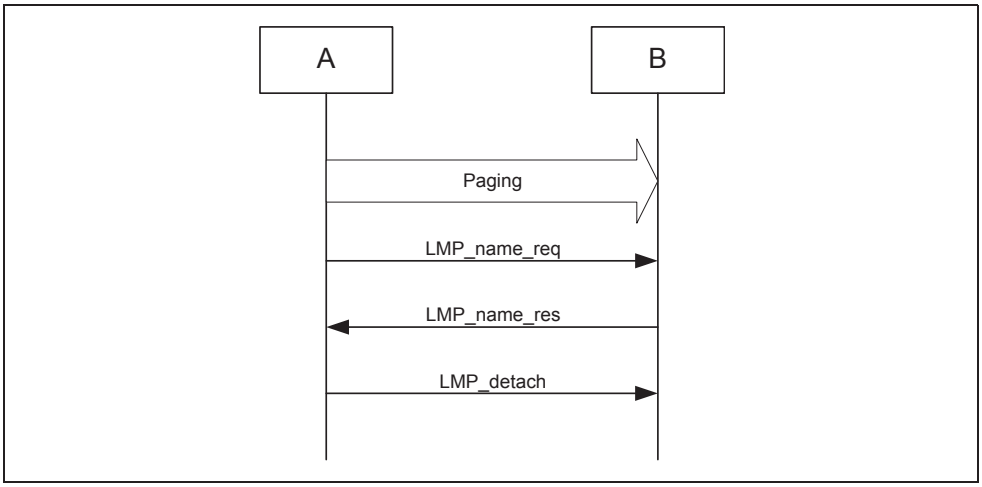



*Figure 6.3: Name request procedure.*

#### *6.3.3.2 Name discovery*

Name discovery is the procedure for retrieving the Bluetooth Device Name from connectable Bluetooth devices by performing name request towards known devices (i.e. Bluetooth devices for which the Bluetooth Device Addresses are available).

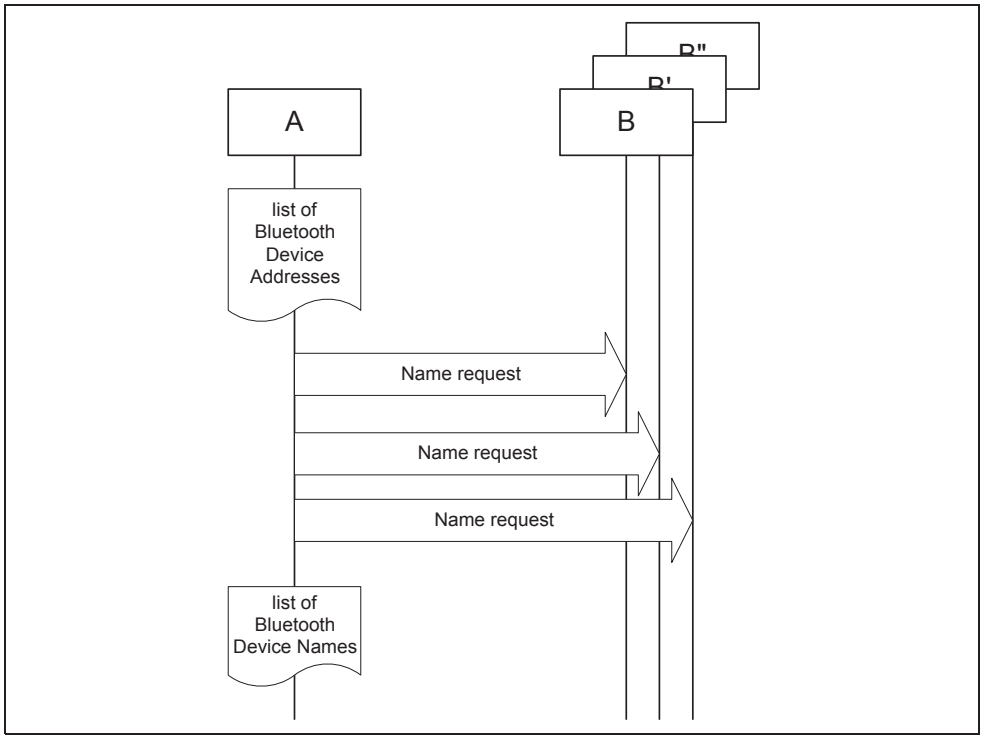



*Figure 6.4: Name discovery procedure.*

### 6.3.4 Conditions

In the name request procedure, the initiator will use the Device Access Code of the remote device as retrieved immediately beforehand – normally through an inquiry procedure.

## 6.4 DEVICE DISCOVERY

### 6.4.1 Purpose

The purpose of device discovery is to provide the initiator with the Bluetooth Device Address, clock, Class of Device, used page scan mode Bluetooth Device Name, and extended inquiry response information of discoverable devices.

### 6.4.2 Term on UI level

'Bluetooth Device Discovery'.



LifeScan, Inc. Ex. 1019
Page 1293 of 1420
IPR2025-00103

### 6.4.3 Description

During the device discovery procedure, first an inquiry (either general or limited) is performed, and then name discovery is done towards some or all of the devices that responded to the inquiry. If the initiator of the device discovery receives a complete local name or a shortened local name that is considered long enough, via an extended inquiry response from a remote device, the initiator should not do a separate name discovery for that device.

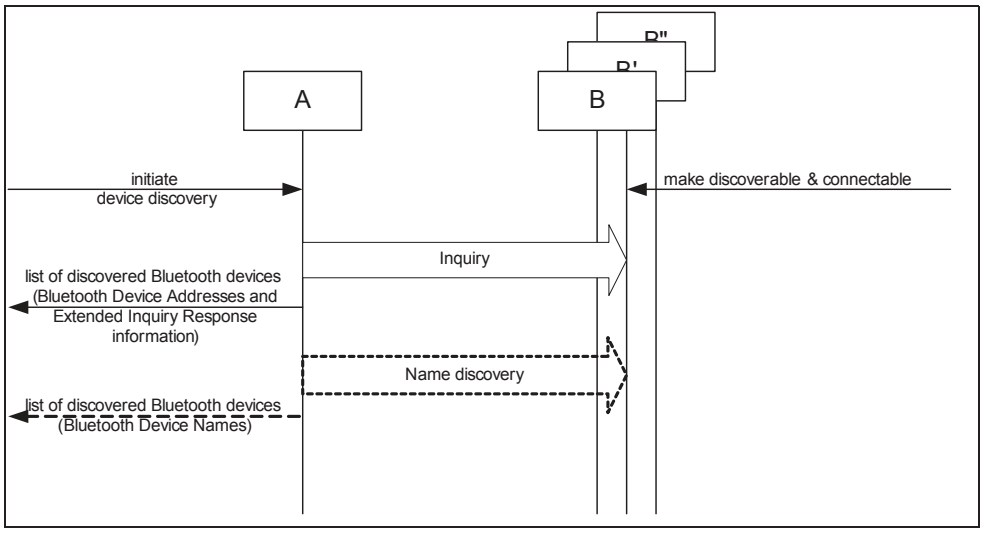



*Figure 6.5: Device discovery procedure.*

### 6.4.4 Conditions

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).

## 6.5 BONDING

### 6.5.1 Purpose

The purpose of bonding is to create a relation between two Bluetooth 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 Bluetooth devices, to be used for future authentication. In addition to pairing, the bonding procedure can involve higher layer initialization procedures.

### 6.5.2 Term on UI level

'Bluetooth Bonding



LifeScan, Inc. Ex. 1019
Page 1294 of 1420
IPR2025-00103

### 6.5.3 Description

Two forms of bonding are described in the following sections: General Bonding and Dedicated Bonding.

#### *6.5.3.1 General Bonding*

General Bonding refers to the process of performing bonding during connection setup or channel establishment procedures as a precursor to accessing a service.

When the devices that are performing General Bonding both support Secure Simple Pairing, the Authentication_Requirements parameter should be set to MITM Protection Not Required – General Bonding unless the security policy of an available local service requires MITM Protection in which case the Authentication_Requirements parameter shall be set to MITM Protection Required – General Bonding.

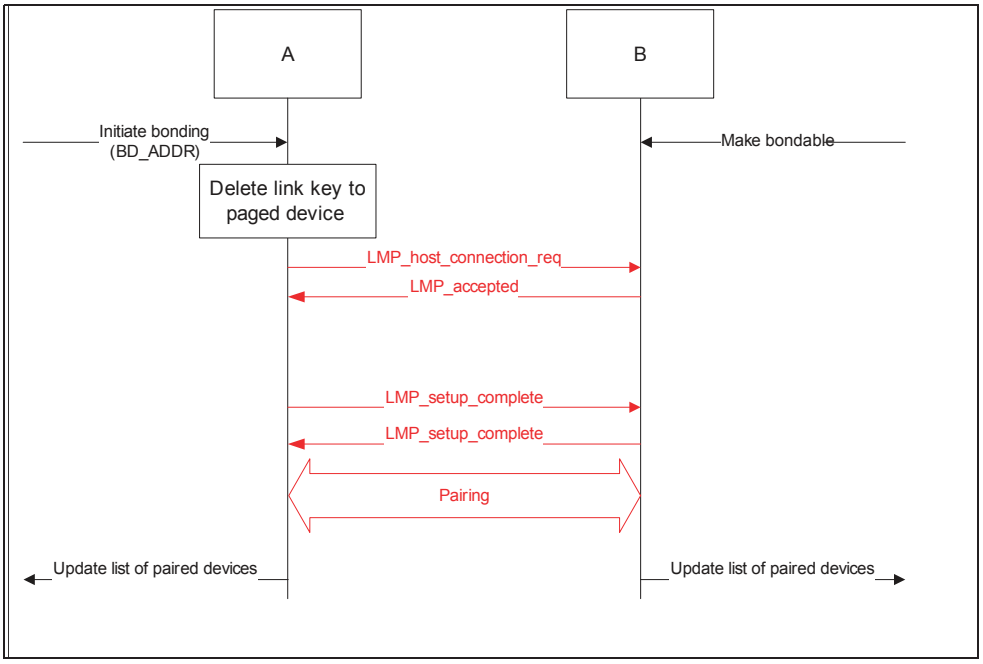



*Figure 6.6: General description of general bonding as being the link establishment procedure executed under specific conditions on both devices, followed by authentication and an optional higher layer initalization process.*

#### *6.5.3.2 Dedicated Bonding*

Dedicated Bonding refers to a procedure wherein one device connects to another only for the purpose of pairing without accessing a particular service. The main difference with dedicated bonding, as compared to a pairing done



LifeScan, Inc. Ex. 1019
Page 1295 of 1420
IPR2025-00103

during link or channel establishment, is that for bonding it is the paging device (A) that must initiate the authentication.

When the devices that are performing Dedicated Bonding both support Secure Simple Pairing, the Authentication_Requirements parameter should be set to *MITM Protection Not Required – Dedicated Bonding* unless the security policy of an available local service requires MITM Protection in which case the Authentication Required parameter shall be set to *MITM Protection Required – Dedicated Bonding*.

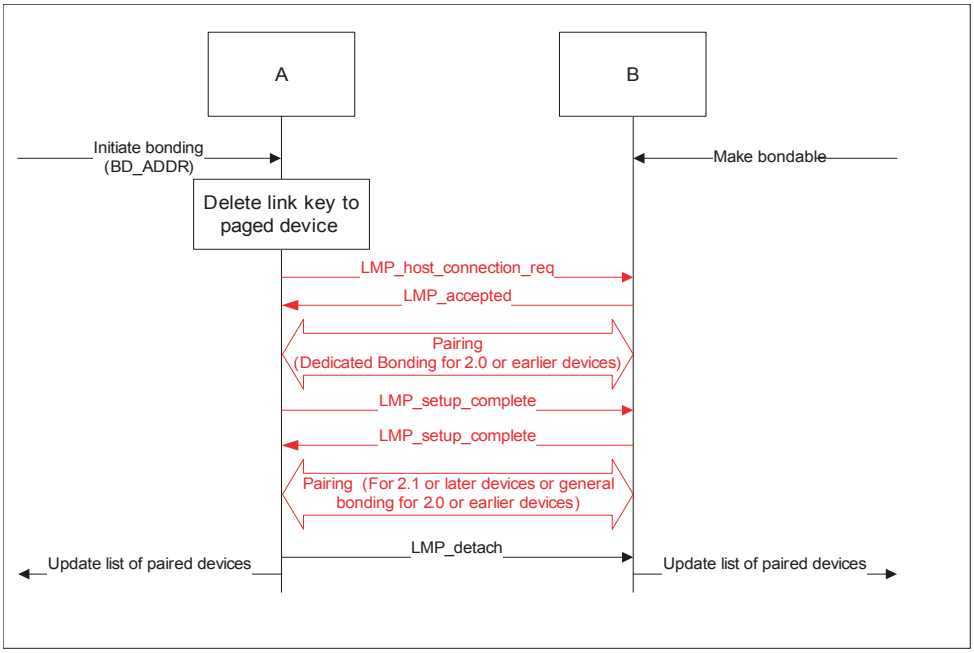



*Figure 6.7: Dedicated Bonding as performed when the purpose of the procedure is only to create and exchange a link key between two Bluetooth devices.*

### 6.5.4 Conditions

Before bonding can be initiated, the initiating device (A) must know the Device Access Code of the device to pair with. This is normally done by first performing device discovery. A Bluetooth Device that can initiate bonding (A) should use limited inquiry, and a Bluetooth Device that accepts bonding (B) should support the limited discoverable mode.

Bonding is in principle the same as link establishment with the conditions:

- The paged device (B) shall be set into bondable 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.



LifeScan, Inc. Ex. 1019
Page 1296 of 1420
IPR2025-00103

- 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.



LifeScan, Inc. Ex. 1019
Page 1297 of 1420
IPR2025-00103

# 7 ESTABLISHMENT PROCEDURES

| | Procedure | Ref. | Support in A | Support in B |
|---|---|---|---|---|
| 1 | Link establishment | 7.1 | M | M |
| 2 | Channel establishment | 7.2 | O | M |
| 3 | Connection establishment | 7.3 | O | O |
| | | | | |
| | | | | |

*Table 7.1: Establishment procedures*

The establishment procedures defined here do not include any discovery part. Before establishment procedures are initiated, the information provided during device discovery (in the FHS packet or the extended inquiry response packet of the inquiry response or in the response to a name request) has to be available in the initiating device. This information is:

- The Bluetooth Device Address (BD_ADDR) from which the Device Access Code 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 Class of device;
- The Device name;
- The supported Service Classes.

## 7.1 LINK ESTABLISHMENT

### 7.1.1 Purpose

The purpose of the link establishment procedure is to establish a physical link (of ACL type) between two Bluetooth devices using procedures from [2] and [3].

### 7.1.2 Term on UI level

'Bluetooth link establishment'



LifeScan, Inc. Ex. 1019
Page 1298 of 1420
IPR2025-00103

### 7.1.3 Description

In this sub-section, the paging device (A) is in security mode 3.During link establishment, the paging device cannot during link establishment distinguish if the paged device (B) is in security mode 1, 2 or 4.

#### *7.1.3.1 B in security mode 1, 2, or 4*

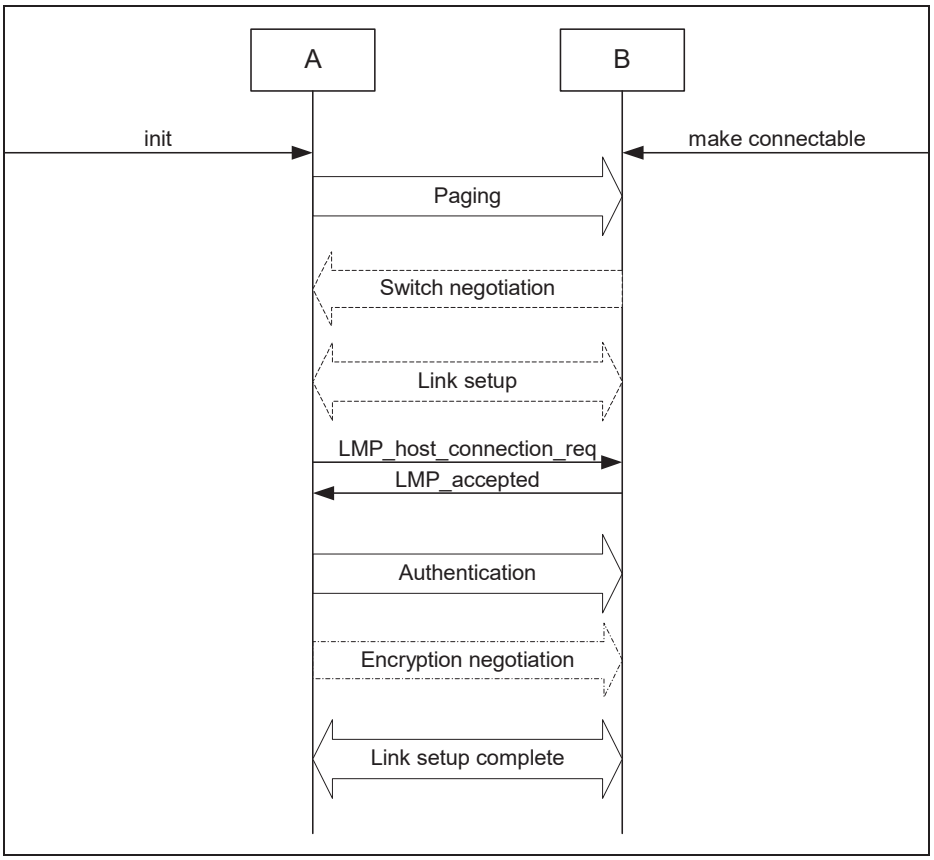



*Figure 7.1: Link establishment procedure when the paging device (A) is in security mode 3 and the paged device (B) is in security mode 1, 2, or 4.*



LifeScan, Inc. Ex. 1019
Page 1299 of 1420
IPR2025-00103

### *7.1.3.2 B in security mode 3*

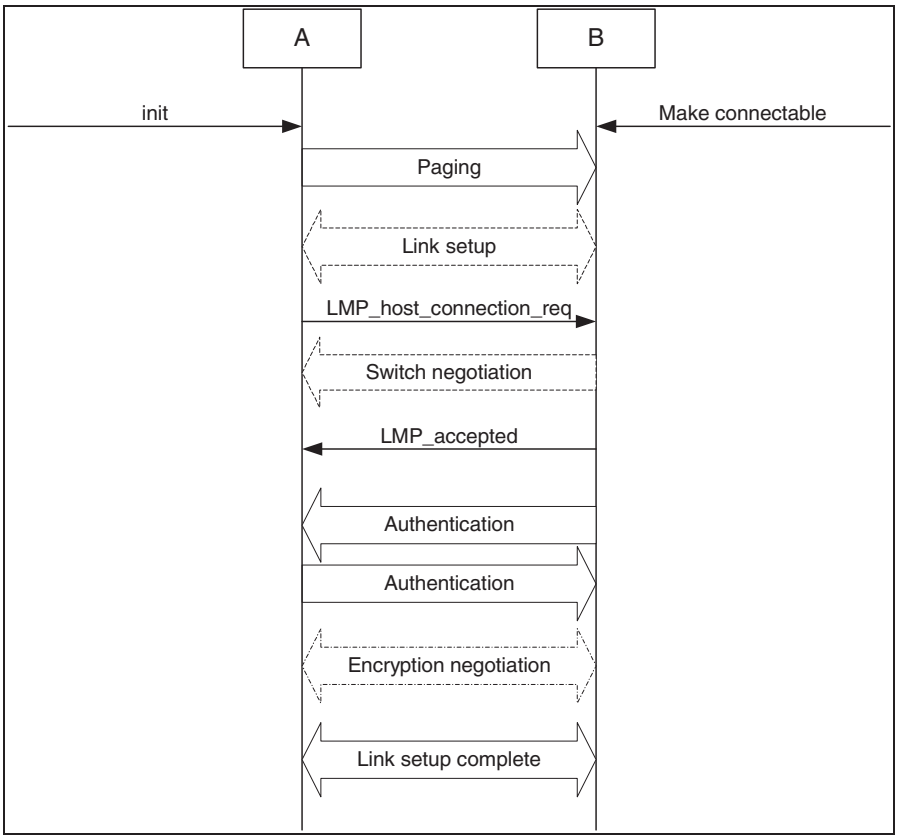



*Figure 7.2: Link establishment procedure when both the paging device (A) and the paged device (B) are in security mode 3.*

## 7.1.4 Conditions

The paging procedure shall be according to [2] and the paging device should use the Device access code and page mode received through a previous inquiry. When paging is completed, a physical link between the two Bluetooth 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 or not.

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.



LifeScan, Inc. Ex. 1019
Page 1300 of 1420
IPR2025-00103

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.

The paging device shall send LMP_host_connection_req during link establishment (i.e. before channel establishment) and may initiate authentication only after having sent LMP_host_connection_request.

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. When both devices are satisfied, they send LMP_setup_complete.

Link establishment is completed when both devices have sent LMP_setup_complete.

## 7.2 CHANNEL ESTABLISHMENT

### 7.2.1 Purpose

The purpose of the channel establishment procedure is to establish a Bluetooth channel (L2CAP channel) between two Bluetooth devices using [2].

### 7.2.2 Term on UI level

'Bluetooth channel establishment'.

### 7.2.3 Description

In this sub-section, the initiator (A) is in security mode 3. During channel establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.



LifeScan, Inc. Ex. 1019
Page 1301 of 1420
IPR2025-00103

#### *7.2.3.1 B in security mode 2 or 4*

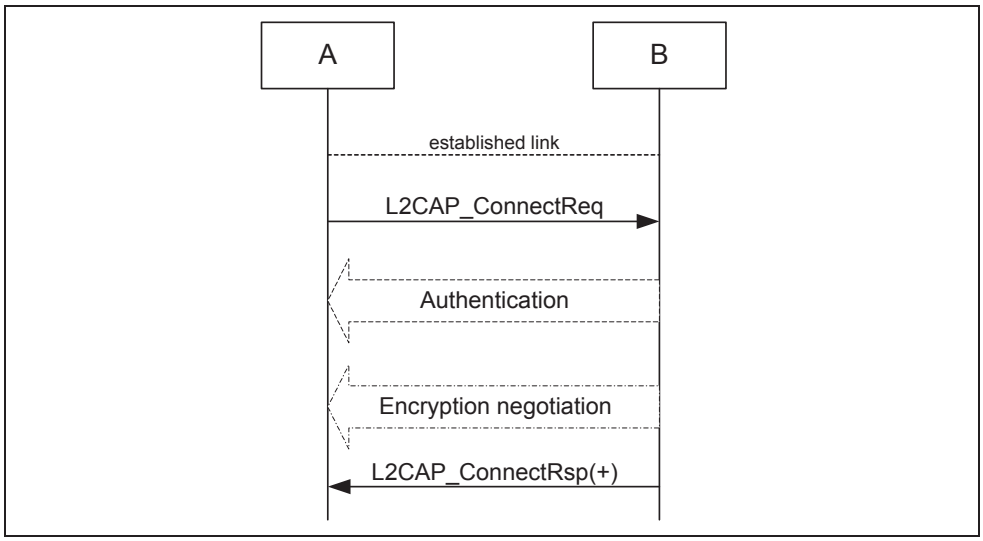



*Figure 7.3: Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2 or 4.*

#### *7.2.3.2 B in security mode 1 or 3*

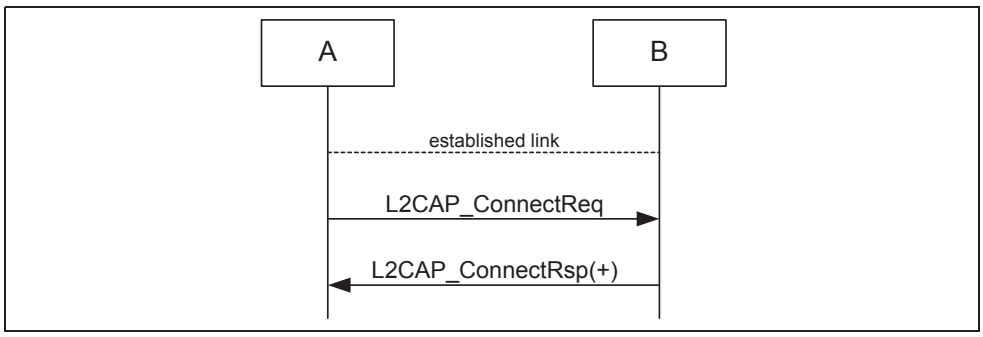



*Figure 7.4: Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3.*

### 7.2.4 Conditions

Channel establishment starts after link establishment is completed when the initiator sends a channel establishment request (L2CAP_ConnectReq).

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).



LifeScan, Inc. Ex. 1019
Page 1302 of 1420
IPR2025-00103

## 7.3 CONNECTION ESTABLISHMENT

### 7.3.1 Purpose

The purpose of the connection establishment procedure is to establish a connection between applications on two Bluetooth devices.

### 7.3.2 Term on UI level

'Bluetooth connection establishment'

### 7.3.3 Description

In this sub-section, the initiator (A) is in security mode 3. During connection establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.

#### *7.3.3.1 B in security mode 2 or 4*

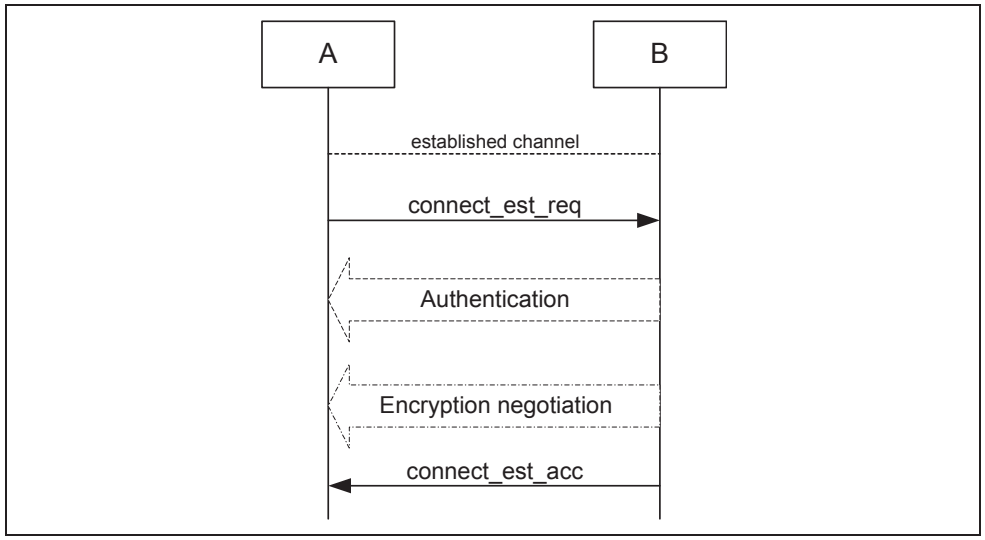



*Figure 7.5: Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2 or 4*

#### *7.3.3.2 B in security mode 1 or 3*

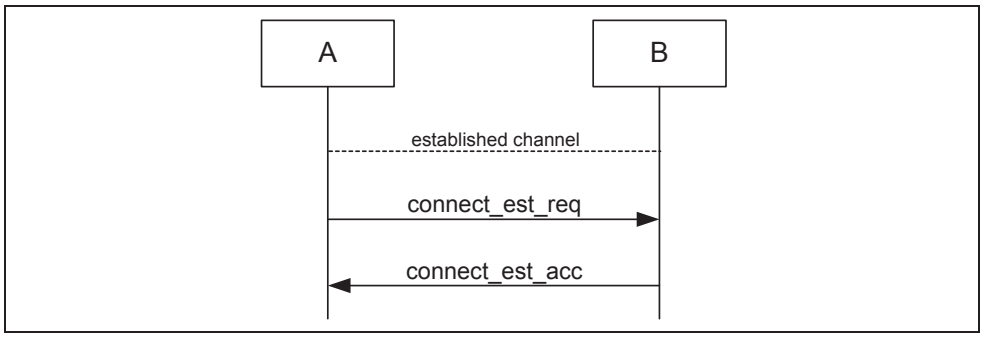



*Figure 7.6: Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3.*

### 7.3.4 Conditions

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 [6] in the case of a Bluetooth telephony application Cordless Telephony Profile, or initialization of RFCOMM and establishment of DLC [5] in the case of a serial port-based application Serial Port Profile (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).

## 7.4 ESTABLISHMENT OF ADDITIONAL CONNECTION

When a Bluetooth device has established one connection with another Bluetooth device, it may be available for establishment of:

- A second connection on the same channel, and/or
- A second channel on the same logical link, and/or
- A second physical link.

If the new establishment procedure is to be towards the same device, the security part of the establishment depends on the security modes used. If the new establishment procedure is to be towards 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 co-incident radio and baseband events have to be handled).



LifeScan, Inc. Ex. 1019
Page 1304 of 1420
IPR2025-00103

# 8 EXTENDED INQUIRY RESPONSE DATA FORMAT

The extended inquiry response data format is shown in Figure 8.1. The data is 240 octets and consists of a significant part and a non-significant part. The significant part contains a sequence of data structures. Each data structure shall have a length field of one octet, which contains the Length value, and a data field of Length octets. The first n octets of the data field contain the extended inquiry response (EIR) data type. The content of the remaining Length - n octets in the data field depends on the value of the EIR data type and is called the EIR data. The non-significant part extends the extended inquiry response to 240 octets and shall contain all-zero octets.

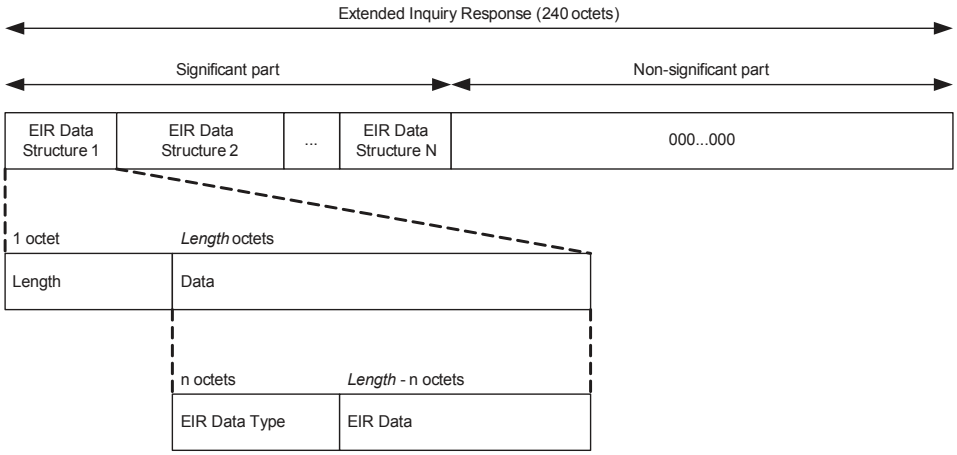



*Figure 8.1: Extended Inquiry Response data format*

[1] The extended inquiry response data types are specified in the Assigned Numbers document.

If the length field is set to zero, then the data field has zero octets. This shall only occur to allow an early termination of the tagged data.

To reduce interference, the host should try to minimize the amount of EIR data such that the baseband can use a 1-slot or 3-slot EIR packet. This is advantageous because it reduces interference and maximizes the probability that the EIR packet will be received. If applications on a host provide more than 240 bytes of extended inquiry response data, it is up to the host to limit it to 240 octets. .

The EIR data shall always be sent during inquiry response state. EIR data can contain device name, Tx power level, service class UUIDs, as well as manufacturers data, as defined in Section 8.1. In selecting the packet type to be used, FEC (DM1 or DM3) should be considered to maximize the range.



LifeScan, Inc. Ex. 1019
Page 1305 of 1420
IPR2025-00103

The Host shall include the device name in the EIR data according to the following rules:

1. If the device does not have a device name (i.e. 0 octet) and
    a) There is no other data to be sent in the EIR then host shall send a name tag with zero length and the type field set to indicate that this is the complete name (i.e. total of 2 octets with length = 1)
    b) There is other important data to be sent in the EIR packet and a zero octet name tag will not fit, then host may avoid sending the name tag.
2. If the device has a device name (greater than 0 octet) and
    a) It is too long be included in the EIR packet (given the choice of packet type and any other data that is being sent) then host may send a shorted version of the name (even 0 octet) and shall mark the name as 'shortened' to inform the receiver that a remote name request is required obtain the full name if the name is needed
    b) There are no other data to be sent in the EIR packet (given the choice of packet type selected) then host shall maximise the length of the device name to be sent, this may be complete or shortened name (e.g. if DM1 packet is chosen and device name characters equates to greater than 15 octets, then host sends first few characters that equates to 15 octets or less with shortened flag)

Note: It is not necessary to understand each and every EIR data type. If a host does not understand a given EIR data type value it should just skip over Length octets and look for the next EIR data structure.

## 8.1 EIR DATA TYPE DEFINITIONS

This section defines the basic EIR data types. Additional EIR data types may be defined in profile specifications.

All EIR data type values are listed in the Bluetooth Assigned Numbers document.

All numerical multi-byte entities and values associated with the following data types shall use little-endian byte order.

### 8.1.1 Service Class UUIDs

A list of service class UUIDs (see Section 2.4, “Service Class,” on page 119) may be included in the extended inquiry response data packet. There are three types of UUIDs that may be returned: 16-bit Bluetooth UUIDs, 32-bit Bluetooth UUIDs, and global 128-bit UUIDs. Each size is assigned two data type values. In addition to the size, the data type values also indicate if the list of UUIDs is complete or there are more UUIDs available via SDP. An extended inquiry



LifeScan, Inc. Ex. 1019
Page 1306 of 1420
IPR2025-00103

response shall not contain more than one list for each size of UUIDs. If a device has no service class UUIDs of a certain size the corresponding field in the extended inquiry response may be omitted.

| Value | Description | |
|---|---|---|
| 0x02/0x03 | 16-bit Service Class UUIDs | 0x02 = More 16-bit UUIDs available |
| | | 0x03 = Complete list of 16-bit UUIDs |
| 0x04/0x05 | 32-bit Service Class UUIDs | 0x04 = More 32-bit UUIDs available |
| | | 0x05 = Complete list of 32-bit UUIDs |
| 0x06/0x07 | 128-bit Service Class UUIDs | 0x06 = More 128-bit UUIDs available |
| | | 0x07 = Complete list of 128-bit UUIDs |

### 8.1.2 Local Name

When included in the extended inquiry response, this shall be the same as, or a shortened version of, the local name assigned to the device. The data type value indicates if the name is complete or shortened. If the name is shortened, the complete name can be read using the remote name request procedure. An extended inquiry response shall not contain more than one instance of this data type.

| Value | Description | |
|---|---|---|
| **0x08/0x09** | **Local Name** | **0x08 = Shortened local name** |
| | | **0x09 = Complete local name** |

### 8.1.3 Flags

This data type is allocated for flag bits. These can for example be used for boolean information that would otherwise require a whole byte (not counting the Length and Data Type fields). The flag bits shall be included in the extended inquiry response if any of the bits are non-zero. If all flag bits are zero they may be omitted. An extended inquiry response shall not contain more than one instance of this data type.

Note: The Flags field can be zero or more octets long. This allows the Flags field to be extended while using the minimum number of octets within the data packet. Whole octets that are 0x00 are not transmitted as long as all other octets after that octet are also 0x00.

No flag bits are currently defined.

| Value | Description |
|---|---|
| 0x01 | Flags |

### 8.1.4 Manufacturer Specific Data

This data type is used for manufacturer specific data. The first two data octets shall contain a company identifier code from the Assigned Numbers - Company Identifiers document. The interpretation of any other octets within the data shall be defined by the manufacturer specified by the company identifier.

| Value | Description |
|---|---|
| 0xFF | Manufacturer Specific Data |

### 8.1.5 TX Power Level

The TX Power Level tag indicates the transmitted power level of the EIR packet.

| Value | Description |
|---|---|
| 0x0A | TX Power Level (1 byte)<br>0xXX:-127 to +127dBm<br>Note: when the TX Power Level tag is not present, the TX power level of the EIR packet is unknown. |

The TX Power Level tag may be used to calculate path loss on a received EIR packet using the following equation:

pathloss = Tx Power Level – EIR_RSSI

For example, if Tx Power Level = +4 (dBm) and the RSSI on the EIR packet is -60 (dBm) then the total pathloss is +4 – (-60) = +64 dB. If a second EIR packet were received at -40dBm with a Tx Power Level tag = +15dBm the resulting pathloss would be +55dB. An application could use these pathloss values to choose which device it thinks might be closer (the one with the lower pathloss value).

Unfortunately, due to fading and varying antenna, circuit, and chip characteristics, these resulting pathloss values will have uncertainty. Some of the uncertainty (for example, due to fading) may be able to be removed if multiple EIR packets are received from the same device.

### 8.1.6 Secure Simple Pairing Out of Band (OOB)

An out of band mechanism may be used in Secure Simple Pairing to communicate discovery information as well as other information related to the pairing

process. These tags use the standard EIR tag format. They shall only be used over an out-of-band mechanism.

| Value | Descrption |
|---|---|
| NA | OOB Optional Data Length (2 bytes)<br>0xXXXX: 0 to 65535 bytes<br>This field contains the length of the optional part |
| NA | BD_ADDR (6 bytes)<br>Format defined in [Vol. 2, Part B] Section 1.2 on page 62 |

*Table 8.1: Mandatory OOB Tags*

| Value | Desciption |
|---|---|
| 0x0D | Class of Device (3 bytes)<br>Format defined in Assigned Numbers |
| 0x0E | Simple Pairing Hash C (16 bytes)<br>Format defined in [Vol. 2, Part H] Section 7.2.2 on page 892 |
| 0x0F | Simple Pairing Randomizer R (16 bytes)<br>Format defined in [Vol. 2, Part H] Section 7.2.2 on page 892 |

*Table 8.2: Optional OOB Tags*

## 8.2 EXAMPLE EXTENDED INQUIRY RESPONSE

This is an example extended inquiry response for a phone with PANU, Hands-free Audio Gateway, and Intercom Service:

| Value | Notes |
|---|---|
| 0x06 | Length of this Data |
| 0x09 | Complete Local Name |
| 0x50 | 'P' |
| 0x68 | 'h' |
| 0x6F | 'o' |
| 0x6E | 'n' |
| 0x65 | 'e' |
| 0x07 | Length of this Data |
| 0x03 | Complete list of 16-bit Service Class UUIDs |
| 0x15 | PANU service class UUID |
| 0x11 | |
| 0x1F | Handsfree Audio Gateway service class UUID |



LifeScan, Inc. Ex. 1019
Page 1309 of 1420
IPR2025-00103

| | |
|---|---|
| 0x11 | |
| 0x09 | Cordless Telephony service class UUID |
| 0x11 | |
| 0x00 | End of Data (Not transmitted over the air) |



LifeScan, Inc. Ex. 1019
Page 1310 of 1420
IPR2025-00103

# 9 DEFINITIONS

In the following, terms written with capital letters refer to states.

## 9.1 GENERAL DEFINITIONS

**Mode .** A set of directives that defines how a device will respond to certain events.

**Idle .** As seen from a remote device, a Bluetooth device is idle, or is in idle mode, when there is no link established between them.

**Bond .** A relation between two Bluetooth devices defined by creating, exchanging and storing a common link key. The bond is created through the bonding or LMP-pairing procedures.

## 9.2 CONNECTION-RELATED DEFINITIONS

**Physical channel .** A synchronized Bluetooth baseband-compliant RF hopping sequence.

**Piconet .** A set of Bluetooth devices sharing the same physical channel defined by the master parameters (clock and BD_ADDR).

**Physical link .** A Baseband-level connection[1] between two devices established using paging. A physical link comprises a sequence of transmission slots on a physical channel alternating between master and slave transmission slots.

**ACL link .** An asynchronous (packet-switched) connection[1] between two devices created on LMP level. Traffic on an ACL link uses ACL packets to be transmitted.

**SCO link .** A synchronous (circuit-switched) connection[1] for reserved bandwidth communications; e.g. voice between two devices, created on the LMP level by reserving slots periodically on a physical channel. Traffic on an SCO link uses SCO packets to be transmitted. SCO links can be established only after an ACL link has first been established.

**Link .** Shorthand for an ACL link.

**PAGE .** A baseband state where a device transmits page trains, and processes any eventual responses to the page trains.

**PAGE_SCAN .** A baseband state where a device listens for page trains.

1. The term 'connection' used here is not identical to the definition below. It is used in the absence of a more concise term.



LifeScan, Inc. Ex. 1019
Page 1311 of 1420
IPR2025-00103

**Page .** The transmission by a device of page trains containing the Device Access Code of the device to which the physical link is requested.

**Page scan .** The listening by a device for page trains containing its own Device Access Code.

**Channel .** A logical connection on L2CAP level between two devices serving a single application or higher layer protocol.

**Connection .** A connection between two peer applications or higher layer protocols mapped onto a channel.

**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.)

**Connect (to service) .** The establishment of a connection to a service. If not already done, this includes establishment of a physical link, link and channel as well.

## 9.3 DEVICE-RELATED DEFINITIONS

**Discoverable device .** A Bluetooth device in range that will respond to an inquiry (normally in addition to responding to page).

**Silent device .** A Bluetooth device appears as silent to a remote device if it does not respond to inquiries made by the remote device. A device may be silent due to being non-discoverable or due to baseband congestion while being discoverable.

**Connectable device .** A Bluetooth device in range that will respond to a page.

**Trusted device .** A paired device that is explicitly marked as trusted.

**Paired device .** A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).

**Pre-paired device .** A Bluetooth device with which a link key was exchanged, and the link key is stored, before link establishment.

**Un-paired device .** A Bluetooth device for which there was no exchanged link key available before connection establishment was request.

**Known device .** A Bluetooth device for which at least the BD_ADDR is stored.

**Un-known device .** A Bluetooth device for which no information (BD_ADDR, link key or other) is stored.

**Authenticated device .** A Bluetooth device whose identity has been verified during the lifetime of the current link, based on the authentication procedure.



LifeScan, Inc. Ex. 1019
Page 1312 of 1420
IPR2025-00103

## 9.4 PROCEDURE-RELATED DEFINITIONS

**Paging .** A procedure for establishing a physical link of ACL type on baseband level, consisting of a page action of the initiator and a page scan action of the responding device.

**Link establishment .** A procedure for establishing a link on LMP level. A link is established when both devices have agreed that LMP setup is completed.

**Channel establishment .** A procedure for establishing a channel on L2CAP level.

**Connection establishment .** A procedure for creating a connection mapped onto a channel.

**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).

**Creation of a secure connection.** A procedure of establishing a connection, including authentication and encryption.

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

**Name discovery .** A procedure for retrieving the user-friendly name (the Bluetooth device name) of a connectable device.

**Service discovery .** Procedures for querying and browsing for services offered by or through another Bluetooth device.

## 9.5 SECURITY-RELATED DEFINITIONS

**Authentication .** A generic procedure based on LMP-authentication if a link key exists or on LMP-pairing if no link key exists.

**LMP-authentication .** An LMP level procedure 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 BD_ADDR of the non-initiating device. The secret key used can be a previously exchanged link key.

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

**Authorize .** The act of granting a specific Bluetooth device access to a specific service. It may be based upon user confirmation, or given the existence of a trusted relationship.



LifeScan, Inc. Ex. 1019
Page 1313 of 1420
IPR2025-00103

**LMP-pairing .** A procedure that authenticates two devices, based on a 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 steps: creation of an initialization key (based on a random number and a PIN), creation and exchange of a common link key and LMP-authentication based on the common link key.

**Bonding .** A dedicated procedure for performing the first authentication, where a common link key is created and stored for future use.

**Trusting .** The marking of a paired device as trusted. Trust marking can be done by the user, or automatically by the device (e.g. when in bondable mode) after a successful pairing.



LifeScan, Inc. Ex. 1019
Page 1314 of 1420
IPR2025-00103

# 10 APPENDIX A (NORMATIVE): TIMERS AND CONSTANTS

The following timers are required by this profile.

| Timer name | Recommended value | Description | Comment |
|---|---|---|---|
| $T_{GAP}(100)$ | 10.24 s | Normal time span that a Bluetooth device performs inquiry. | Used during inquiry and device discovery. |
| $T_{GAP}(101)$ | 10.625 ms | Minimum time in INQUIRY_SCAN. | A discoverable Bluetooth 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 Bluetooth device shall not be in a discoverable mode less than $T_{GAP}(103)$. | Minimum time to be discoverable. |
| $T_{GAP}(104)$ | 1 min. | A Bluetooth device should not be in limited discoverable mode more than $T_{GAP}(104)$. | Recommended upper limit. |
| $T_{GAP}(105)$ | 100ms | Maximum time between INQUIRY_SCAN enterings | Recommended value |
| $T_{GAP}(106)$ | 100ms | Maximum time between PAGE_SCAN enterings | Recommended value |
| $T_{GAP}(107)$ | 1.28s | Maximum time between PAGE_SCAN enterings (R1 page scan) | Recommended value |
| $T_{GAP}(108)$ | 2.56s | Maximum time between PAGE_SCAN enterings (R2 page scan) | Recommended value |

*Table 10.1: Defined GAP timers*



LifeScan, Inc. Ex. 1019
Page 1315 of 1420
IPR2025-00103

# 11 APPENDIX B (INFORMATIVE): INFORMATION FLOWS OF RELATED PROCEDURES

## 11.1 LMP-AUTHENTICATION

The specification of authentication on link level is found in [3].

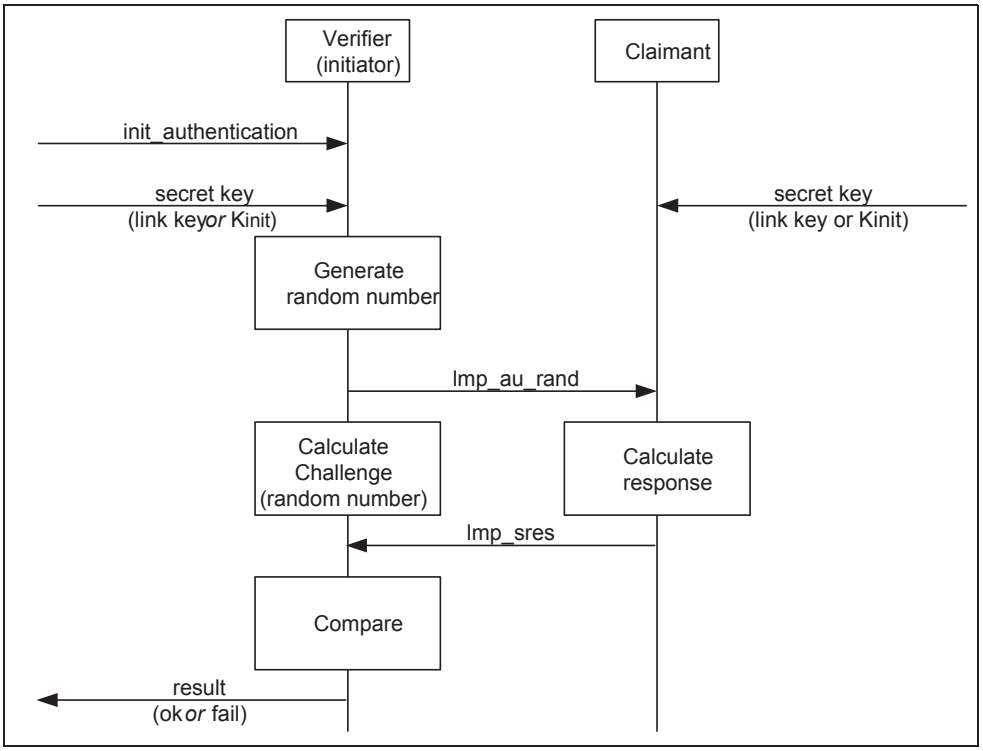



*Figure 11.1: LMP-authentication as defined by [3].*

The secret key used here is an already exchanged link key.



LifeScan, Inc. Ex. 1019
Page 1316 of 1420
IPR2025-00103

## 11.2 LMP-PAIRING

The specification of pairing on link level is found in [3].

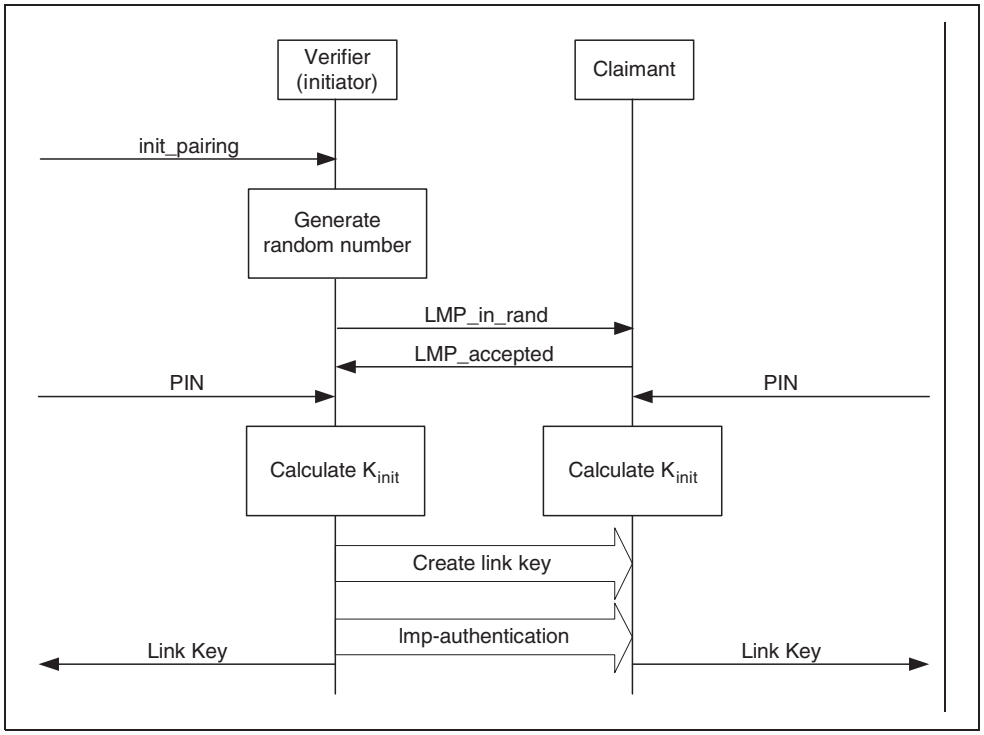



*Figure 11.2: LMP-pairing as defined in [3].*

The PIN used here is $PN_{BB}$.

The create link key procedure is described in [Vol. 2, Part C] Section 4.2.2.4 on page 244 and [Vol. 2, Part H] Section 3.2 on page 861. 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.



LifeScan, Inc. Ex. 1019
Page 1317 of 1420
IPR2025-00103

## 11.3 SERVICE DISCOVERY

The Service Discovery Protocol [7] 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 Service Discovery Protocol are described in the Service Discovery Application Profile. This is just an example.

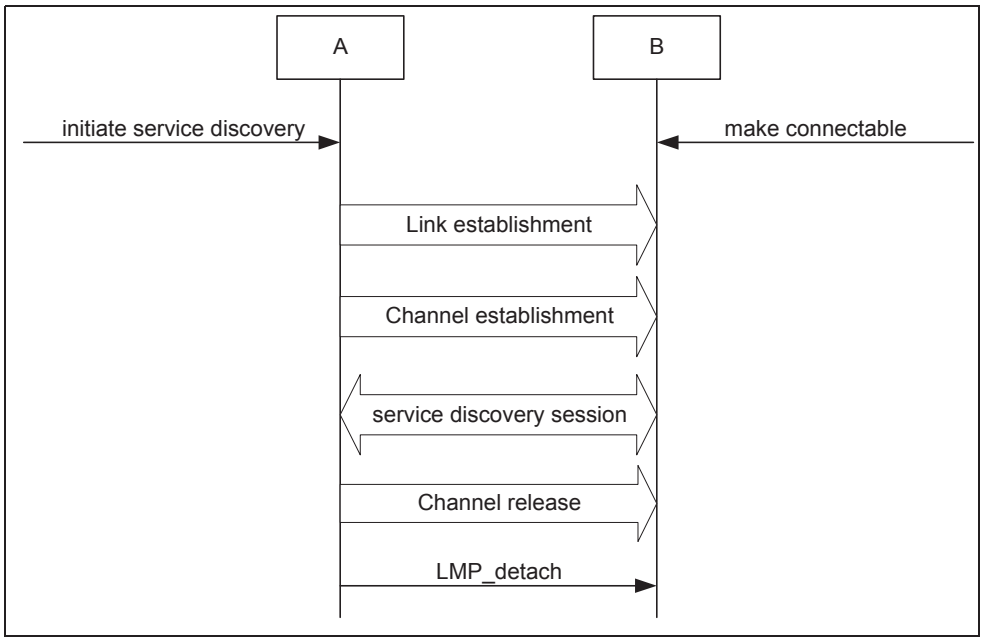



*Figure 11.3: Service discovery procedure.*



LifeScan, Inc. Ex. 1019
Page 1318 of 1420
IPR2025-00103

# 12 REFERENCES

[2] Baseband Specification

[3] Link Manager Protocol

[4] Logical Link Control and Adaptation Protocol

[5] RFCOMM

[6] Telephony Control Specification

[7] Service Discovery Protocol

[8] Service Discovery Application Profile

[9] Cordless Telephony Profile

[10] Serial Port Profile

[11] Security Architecture (white paper)

[12] Bluetooth Assigned Numbers



LifeScan, Inc. Ex. 1019
Page 1319 of 1420
IPR2025-00103





LifeScan, Inc. Ex. 1019
Page 1320 of 1420
IPR2025-00103

# TEST SUPPORT



LifeScan, Inc. Ex. 1019
Page 1321 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1322 of 1420
IPR2025-00103

# CONTENTS

**1 Test Methodology ..... 249**
1.1 Test Scenarios ..... 249
1.1.1 Test setup ..... 249
1.1.2 Transmitter Test ..... 250
1.1.2.1 Packet Format ..... 251
1.1.2.2 Pseudorandom Sequence ..... 252
1.1.2.3 Control of Transmit Parameters ..... 253
1.1.2.4 Power Control ..... 254
1.1.2.5 Switch Between Different Frequency Settings ..... 253
1.1.2.6 Adaptive Frequency Hopping ..... 253
1.1.3 LoopBack test ..... 255
1.1.4 Pause test ..... 258
1.2 References ..... 258
**2 Test Control Interface (TCI) ..... 259**
2.1 Introduction ..... 259
2.1.1 Terms used ..... 259
2.1.2 Usage of the interface ..... 259
2.2 TCI Configurations ..... 260
2.2.1 Bluetooth RF requirements ..... 260
2.2.1.1 Required interfaces ..... 260
2.2.2 Bluetooth protocol requirements ..... 260
2.2.2.1 Required interfaces ..... 261
2.2.3 Bluetooth profile requirements ..... 262
2.2.3.1 Required interfaces ..... 262
2.3 TCI Configuration and Usage ..... 263
2.3.1 Transport layers ..... 263
2.3.1.1 Physical bearer ..... 263
2.3.1.2 Software bearer ..... 263
2.3.2 Baseband and link manager qualification ..... 264
2.3.3 HCI qualification ..... 266



LifeScan, Inc. Ex. 1019
Page 1323 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1324 of 1420
IPR2025-00103

# 1 TEST METHODOLOGY

This section describes the test mode for hardware and low-level functionality tests of Bluetooth devices. The test mode includes transmitter tests (packets with constant bit patterns) and loop back tests.

The test mode supports testing of the Bluetooth transmitter and receiver. It is intended mainly for certification/compliance testing of the radio and baseband layer, and may also be used for regulatory approval or in-production and after-sales testing.

## 1.1 TEST SCENARIOS

A device in test mode shall not support normal operation. For security reasons the test mode is designed such that it offers no benefit to the user. Therefore, no data output or acceptance on a HW or SW interface shall be allowed.

### 1.1.1 Test setup

The setup consists of a device under test (DUT) and a tester. Optionally, additional measurement equipment may be used.

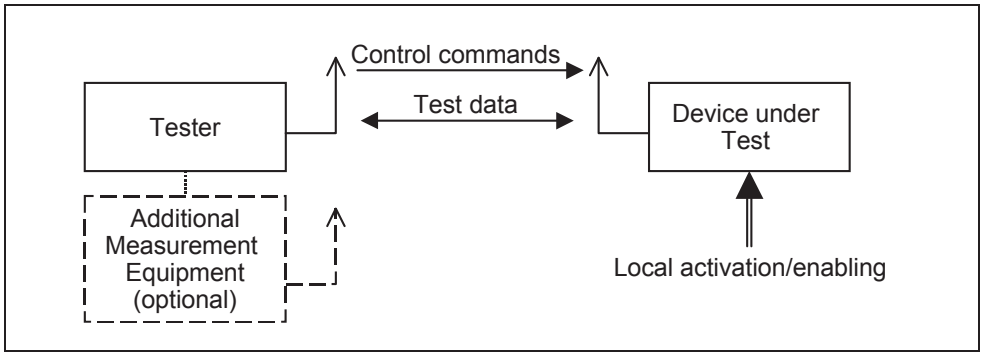



*Figure 1.1: Setup for Test Mode*

Tester and DUT form a piconet where the tester acts as master and has full control over the test procedure. The DUT acts as slave.

The control is done via the air interface using LMP commands (see [Vol. 2, Part C] Section 4.7.3 on page 299). Hardware interfaces to the DUT may exist, but are not subject to standardization.

The test mode is a special state of the Bluetooth model. For security and type approval reasons, a Bluetooth device in test mode shall not support normal operation. When the DUT leaves the test mode it enters the standby state. After power-off the Bluetooth device shall return to standby state.



LifeScan, Inc. Ex. 1019
Page 1325 of 1420
IPR2025-00103

### 1.1.2 Transmitter Test

The Bluetooth device transmits a constant bit pattern. This pattern is transmitted periodically with packets aligned to the slave TX timing of the piconet formed by tester and DUT. The same test packet is repeated for each transmission.

The transmitter test is started when the master sends the first POLL packet. In non-hopping mode agreed frequency is used for this POLL packet.

The tester (master) transmits control or POLL packets in the master-to-slave transmission slots. The DUT (slave) shall transmit packets in the following slave-to-master transmission slot. The tester's polling interval is fixed and defined by the LMP_test_control PDU. The device under test may transmit its burst according to the normal timing even if no packet from the tester was received. In this case, the ARQN bit is shall be set to NAK.

The burst length may exceed the length of a one slot packet. In this case the tester may take the next free master TX slot for polling. The timing is illustrated in Figure 1.2.

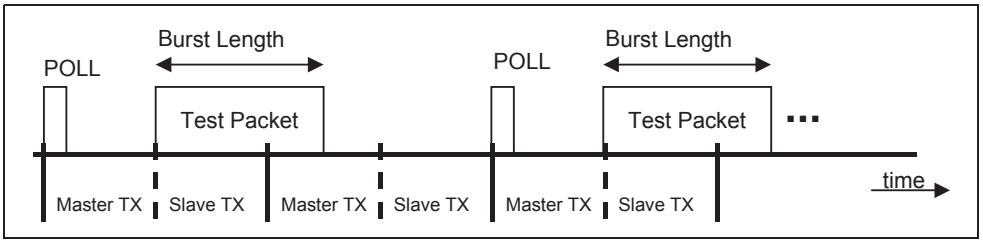



*Figure 1.2: Timing for Transmitter Test*



LifeScan, Inc. Ex. 1019
Page 1326 of 1420
IPR2025-00103

### 1.1.2.1 Packet Format

The test packet is a normal Bluetooth packet, see Figure 1.3. For the payload itself see below.

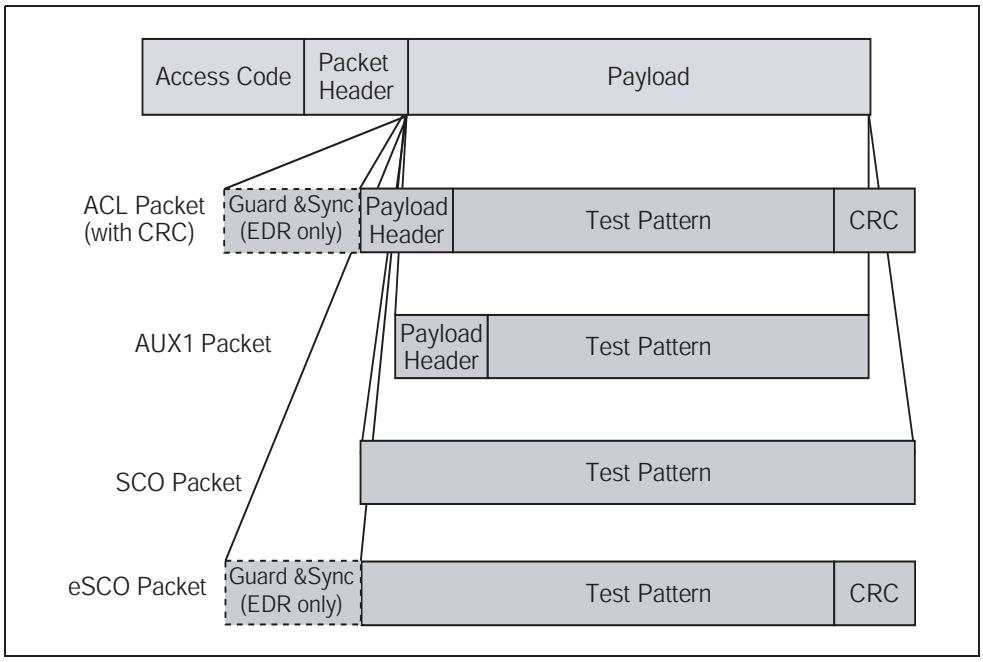



*Figure 1.3: General Format of TX Packet*

During configuration the tester defines:

- the packet type to be used
- payload length

For the payload length, the restrictions from the baseband specification shall apply (see "Baseband Specification" on page 53[vol. 2]). In case of ACL, SCO and eSCO packets the payload structure defined in the baseband specification is preserved as well, see Figure 1.3 on page 251.

For the transmitter test mode, only packets without FEC should be used; i.e. HV3, EV3, EV5, DH1, DH3, DH5, 2-EV3, 2-EV5, 3-EV3, 3-EV5, 2-DH1, 2-DH3, 2-DH5, 3-DH1, 3-DH3, 3-DH5 and AUX1 packets.

In transmitter test mode, the packets exchanged between the tester and the DUT shall not be scrambled with the whitening sequence. Whitening shall be turned off when the DUT has accepted to enter the transmitter test mode, and shall be turned on when the DUT has accepted to exit the transmitter test mode, see Figure 1.4 on page 252. Implementations shall ensure that retransmissions of the LMP_accepted messages use the same whitening status as used in the original LMP_accepted.



LifeScan, Inc. Ex. 1019
Page 1327 of 1420
IPR2025-00103

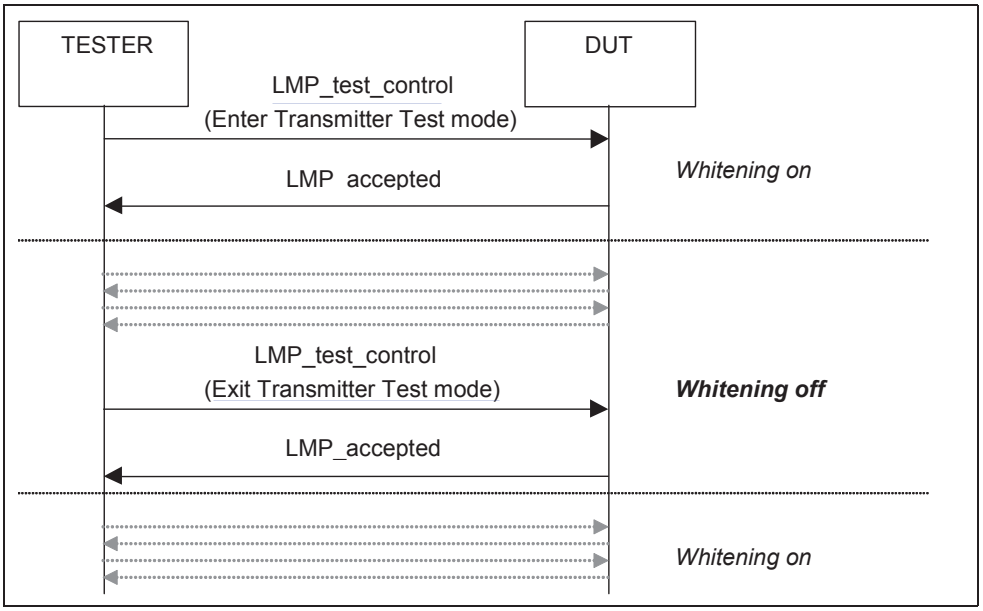



*Figure 1.4: Use of whitening in Transmitter mode*

### *1.1.2.2 Pseudorandom Sequence*

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated). A PRBS-9 Sequence is used, see [2] and [3].

The properties of this sequence are as follows (see [3]). The sequence may be generated in a nine-stage shift register whose 5th and 9th stage outputs are added in a modulo-two addition stage (see Figure 1.5), and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONEs; i.e. the shift register is initialized with nine ones.

- Number of shift register stages: 9
- Length of pseudo-random sequence: $2^9$-1 = 511 bits
- Longest sequence of zeros: 8 (non-inverted signal)

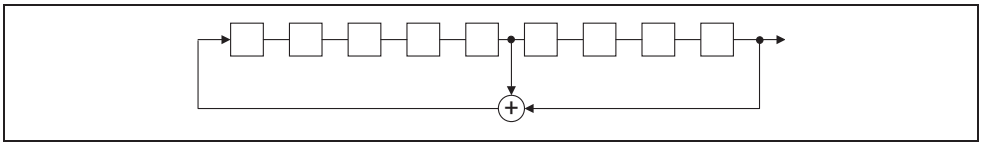


*Figure 1.5: Linear Feedback Shift Register for Generation of the PRBS sequence*



LifeScan, Inc. Ex. 1019
Page 1328 of 1420
IPR2025-00103

#### *1.1.2.3 Control of Transmit Parameters*

The following parameters can be set to configure the transmitter test:

1. Bit pattern:
   - Constant zero
   - Constant one
   - Alternating 1010...[1]
   - Alternating 1111 0000 1111 0000...[4]
   - Pseudorandom bit pattern
   - Transmission off
2. Frequency selection:
   - Single frequency
   - Normal hopping
3. TX frequency
   - k $\Rightarrow$ f := (2402 + k) MHz
4. Default poll period in TDD frames (n * 1.25 ms)
5. Packet Type
6. Length of Test Sequence (user data of packet definition in "Baseband Specification" on page 53[vol. 2])

#### *1.1.2.4 Power Control*

If adaptive power control is tested, the normal LMP commands will be used. The DUT shall start transmitting at the maximum power and shall reduce/increase its power by one step on every LMP_incr_power_req or LMP_decr_power_req command received.

#### *1.1.2.5 Switch Between Different Frequency Settings*

A change in the frequency selection becomes effective when the LMP procedure is completed:

When the tester receives the LMP_accepted it shall then transmit POLL packets containing the Ack for at least 8 slots (4 transmissions). When these transmissions have been completed the tester shall change to the new frequency hop and whitening settings.

After sending LMP_accepted the DUT shall wait for the LC level Ack for the LMP_accepted. When this is received it shall change to the new frequency hop and whitening settings.

1. It is recommended that the sequence starts with a one; but, as this is irrelevant for measurements, it is also allowed to start with a zero.

There will be an implementation defined delay after sending the LMP_accepted before the TX or loopback test starts. Testers shall be able to cope with this.

*Note: Loss of the LMP_Accepted packet will eventually lead to a loss of frequency synchronization that cannot be recovered. Similar problems occur in normal operation, when the hopping pattern changes.*

#### *1.1.2.6 Adaptive Frequency Hopping*

Adaptive Frequency Hopping (AFH) shall only be used when the Hopping Mode is set to 79 channels (e.g. Hopping Mode = 1) in the LMP_test_control PDU. If AFH is used, the normal LMP commands and procedures shall be used. When AFH is enabled prior to entering test mode it shall continue to be used with the same parameters if Hopping Mode = 1 until the AFH parameters are changed by the LMP_set_AFH PDU.

The channel classification reporting state shall be retained upon entering or exiting Test Mode. The DUT shall change the channel classification reporting state in Test Mode based on control messages from the tester (LMP_channel_classification_req) and from the Host (HCI Write_AFH_Channel_Classification_Mode).



LifeScan, Inc. Ex. 1019
Page 1330 of 1420
IPR2025-00103

### 1.1.3 LoopBack test

In loopback, the device under test receives normal baseband packets containing payload *Accepted* from the tester. The received packets shall be decoded in the DUT, and the payload shall be sent back using the same packet type. The return packet shall be sent back in either the slave-to-master transmission slot directly following the transmission of the tester, or it is delayed and sent back in the slave-to-master transmission slot after the next transmission of the tester (see Figure 1.7 to Figure 1.9 on page 257).

There is no signaling to determine or control the mode. The device behavior shall be fixed or adjusted by other means, and shall not change randomly.

The tester can select, whether whitening is on or off. This setting holds for both up- and downlink. For switching the whitening status, the same rules as in Section 1.1.2 on page 250 (Figure 1.4) shall apply.

The following rules apply (for illustration see Figure 1.6 on page 256):

- If the synch word was not detected, the DUT shall not reply.
- If the header error check (HEC) fails, the DUT shall either reply with a NULL packet with the ARQN bit set to NAK or send nothing.
- If the packet contains an LMP message relating to the control of the test mode this command shall be executed and the packet shall not be returned, though ACK or NAK shall be returned as per the usual procedure. Other LMP commands are ignored and no packet is returned.
- The payload FEC is decoded and the payload shall be encoded again for transmission. This allows testing of the FEC handling. If the pure bit error rate shall be determined the tester chooses a packet type without FEC.
- The CRC is shall be evaluated. In the case of a failure, ARQN=NAK shall be returned. The payload shall be returned as received.
  A new CRC for the return packet shall be calculated for the returned payload regardless of whether the CRC was valid or not.
- If the CRC fails for a packet with a CRC and a payload header, the number of bytes as indicated in the (possibly erroneous) payload header shall be looped back.



LifeScan, Inc. Ex. 1019
Page 1331 of 1420
IPR2025-00103

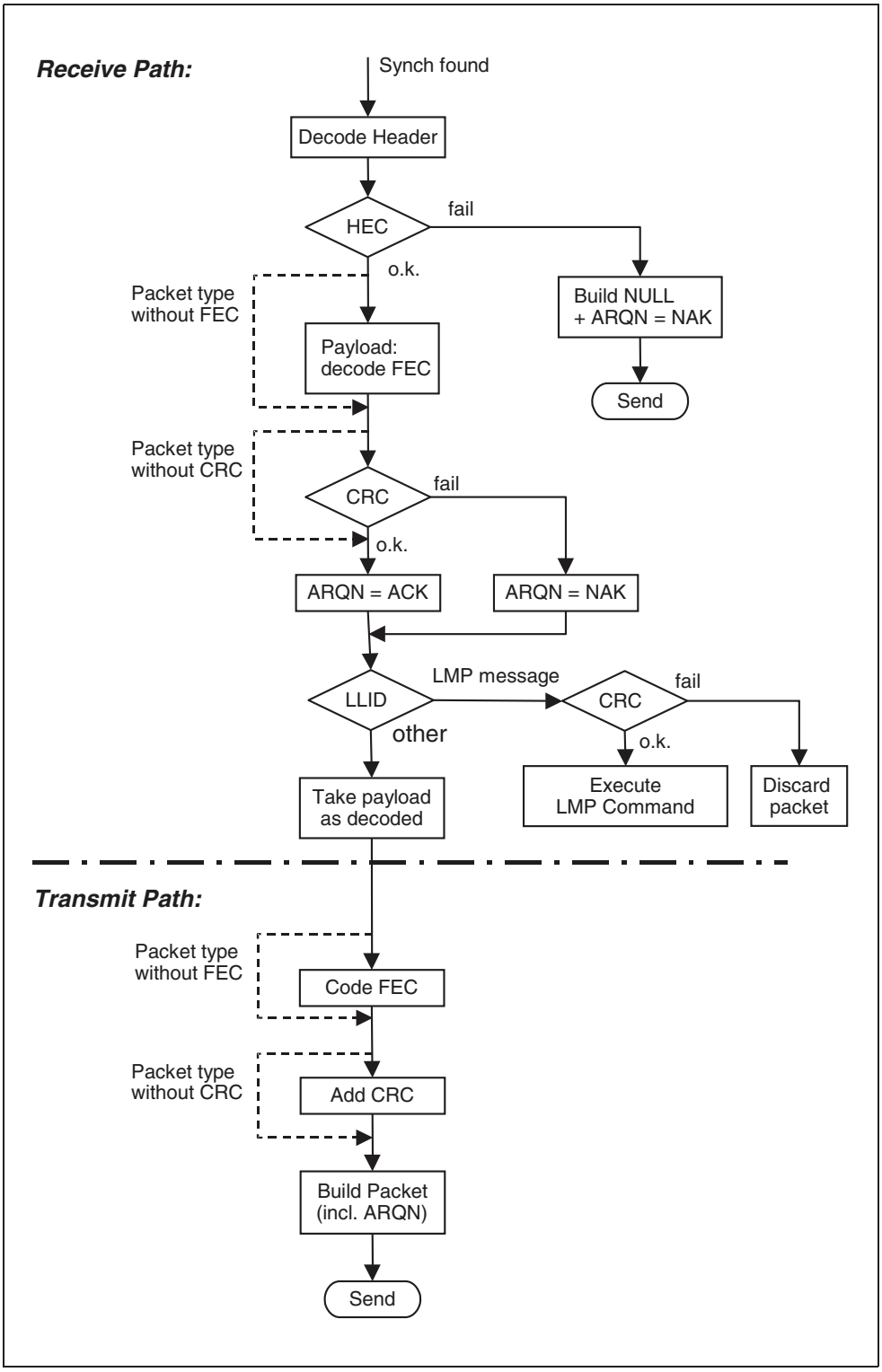



*Figure 1.6: DUT Packet Handling in Loop Back Test*



LifeScan, Inc. Ex. 1019
Page 1332 of 1420
IPR2025-00103

The timing for normal and delayed loopback is illustrated in Figure 1.7 to Figure 1.9:

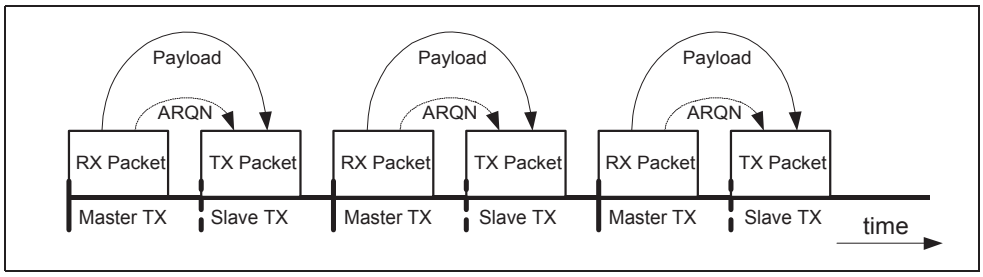



*Figure 1.7: Payload & ARQN handling in normal loopback.*

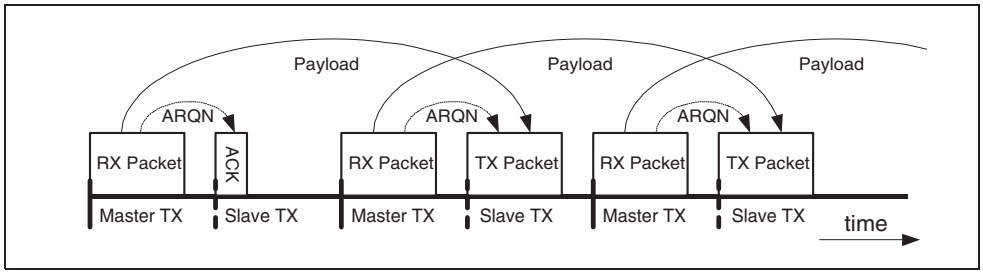



*Figure 1.8: Payload & ARQN handling in delayed loopback - start.*

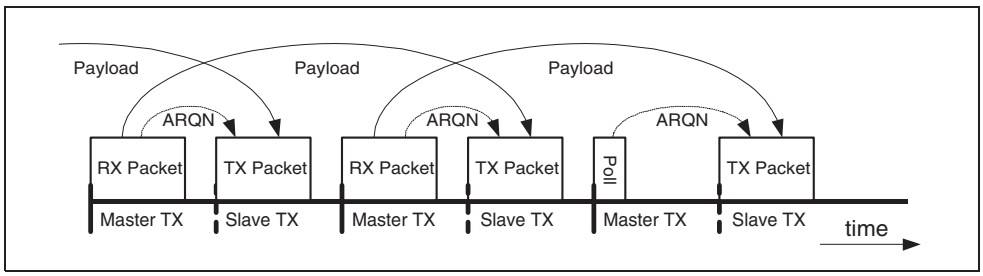



*Figure 1.9: Payload & ARQN handling in delayed loopback - end.*

The whitening is performed in the same way as it is used in normal active mode.

The following parameters can be set to configure the loop back test:

1. Packet Class[1]
   - ACL Packets
   - SCO Packets
   - eSCO Packets
   - ACL Packets without whitening

1. This is included because, in the future, the packet type numbering may not remain unambiguous.



LifeScan, Inc. Ex. 1019
Page 1333 of 1420
IPR2025-00103

- SCO Packets without whitening
- eSCO Packets without whitening

2. Frequency Selection
   - Single frequency (independent for RX and TX)
   - Normal hopping
3. Power level: (To be used according radio specification requirements)
   - power control or fixed TX power

The switch of the frequency setting is done exactly as for the transmitter test (see Section 1.1.2.5 on page 253).

### 1.1.4 Pause test

Pause test is used by testers to put the device under test into Pause Test mode from either the loopback or transmitter test modes.

When an LMP_test_control PDU that specifies Pause Test is received the DUT shall stop the current test and enter Pause Test mode. In the case of a transmitter test this means that no more packets shall be transmitted. While in Pause Test mode the DUT shall respond normally to POLL packets (i.e. responds with a NULL packet). The DUT shall also respond normally to all the LMP packets that are allowed in test mode.

When the test scenario is set to Pause Test all the other fields in the LMP_test_control PDU shall be ignored. There shall be no change in hopping scheme or whitening as a result of a request to pause test.

## 1.2 REFERENCES

[1] Bluetooth Link Manager Protocol.

[2] CCITT Recommendation O.153 (1992), Basic parameters for the measurement of error performance at bit rates below the primary rate.

[3] ITU-T Recommendation O.150 (1996), General requirements for instrumentation for performance measurements on digital transmission equipment.

[4] Bluetooth Baseband Specification.



LifeScan, Inc. Ex. 1019
Page 1334 of 1420
IPR2025-00103

# 2 TEST CONTROL INTERFACE (TCI)

This section describes the Bluetooth Test Control Interface (TCI). The TCI provides a uniform method of accessing the upper interface of the implementation being tested. This facilitates the use of a standardized interface on test equipment used for formal Qualification of implementations.

## 2.1 INTRODUCTION

### 2.1.1 Terms used

| | |
|---|---|
| Conformance testing | Testing according to the applicable procedures given in the Bluetooth Protocol Test Specifications and the Bluetooth Profile Conformance Test Specification when tested against a test system. |
| HCI | Host Controller Interface |
| IUT | Implementation Under Test: An implementation of one or more Bluetooth protocols and profiles which is to be studied by testing. This term is used when describing the test concept for products and components equipped with Bluetooth wireless technology as defined in the PRD. |
| PRD | Bluetooth Qualification Program Reference Document: This document is maintained by the Bluetooth Qualification Review Board and is the reference to specify the functions, organization and processes inside the Bluetooth Qualification program. |
| TCI | Test Control Interface: The interface and protocol used by the test equipment to send and receive messages to and from the upper interface of the IUT. |

### 2.1.2 Usage of the interface

For all products and components equipped with Bluetooth wireless technology, conformance testing is used to verify the implemented functionality in the lower layers. Conformance testing of the lowest layers requires an upper tester to test the implementation sufficiently well.

In order to avoid that the tester will have to adapt to each and every product and component equipped with Bluetooth wireless technology, the use of the standardized TCI is mandated. This concept puts some burden upon the manufacturer of the IUT in terms of supplying an adapter providing the necessary conversion from/to the IUT's specific interface to the TCI. The adapter can consist of hardware, firmware and software. After qualification testing has been performed the TCI may be removed from the product or component equipped with Bluetooth wireless technology. It is the manufacturer's option to remove it from the qualified product or component equipped with Bluetooth wireless technology.



LifeScan, Inc. Ex. 1019
Page 1335 of 1420
IPR2025-00103

The TCI is used when qualifying the implemented functionality of the:

- Baseband layer, BB
- Link Manager layer, LM

If support of the Host Controller Interface is claimed by the manufacturer the TCI is used to qualify it.

## 2.2 TCI CONFIGURATIONS

This section describes the test configurations used when verifying the different Bluetooth requirements. Each layer in the Bluetooth stack is qualified using the procedures described in the layer specific test specification.

### 2.2.1 Bluetooth RF requirements

For qualification of the Bluetooth Radio Frequency requirements the defined Test Mode is used, see Section 1 on page 249.

Similar to TCI, the specific test mode functionality may be removed from the product or component after qualification, at the discretion of the manufacturer.

#### *2.2.1.1 Required interfaces*

For RF qualification only the air interface is required, see Figure 2.1. Depending on the physical design of the IUT it might be necessary to temporarily attach an RF connector for executing the RF tests. As stated in Section 1 on page 249, the Test Mode shall be locally enabled on the IUT for security reasons. The implementation of this local enabling is not subject to standardization.



LifeScan, Inc. Ex. 1019
Page 1336 of 1420
IPR2025-00103

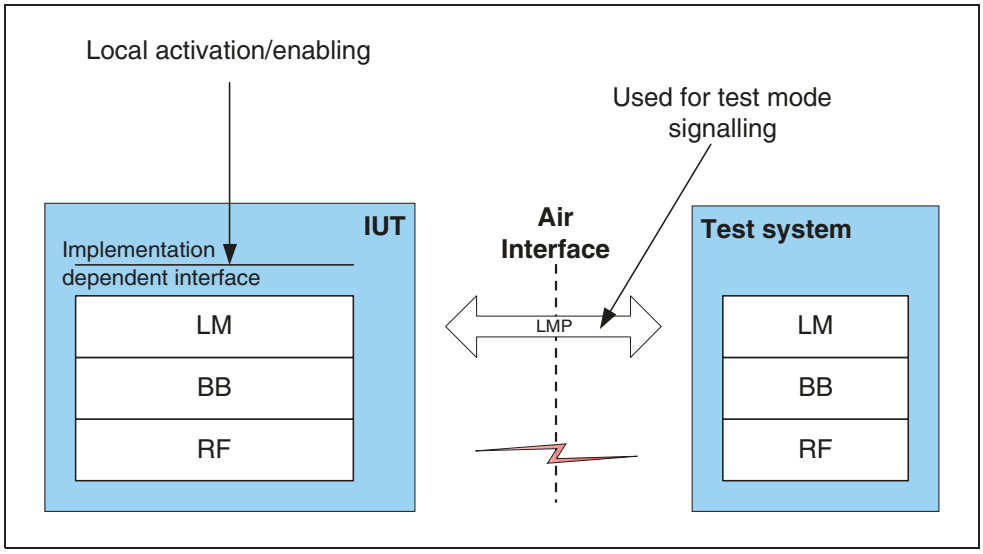



*Figure 2.1: General test set-up for RF qualification*

### 2.2.2 Bluetooth protocol requirements

Depending on which of the Bluetooth layers BB, LM or HCI are implemented in the product subject to qualification, the amount of testing needed to verify the Bluetooth protocol requirements differs. Also, the TCI used during the qualification is slightly different. The commands and events necessary for qualification are detailed in the test specifications and only those commands indicated in the test cases to be executed need be implemented.

For other protocols in the Bluetooth stack the TCI is not used. An implementation specific user interface is used to interact with the IUT's upper interface.

#### *2.2.2.1 Required interfaces*

For BB, LM and HCI qualification both the air interface of the IUT and the TCI are required. For other protocols both the air interface and the user interface are used as described in the test specification.

### 2.2.3 Bluetooth profile requirements

For each Bluetooth profile for which conformance is claimed, profile qualification testing is performed to verify the Bluetooth profile requirements. With higher layer protocols the TCI is not used during testing. A user interface specific to the implementation is used to interact with the IUT's upper interface.

#### *2.2.3.1 Required interfaces*

For this type of qualification both the air interface and the user interface of the IUT are used as described in the test specification.

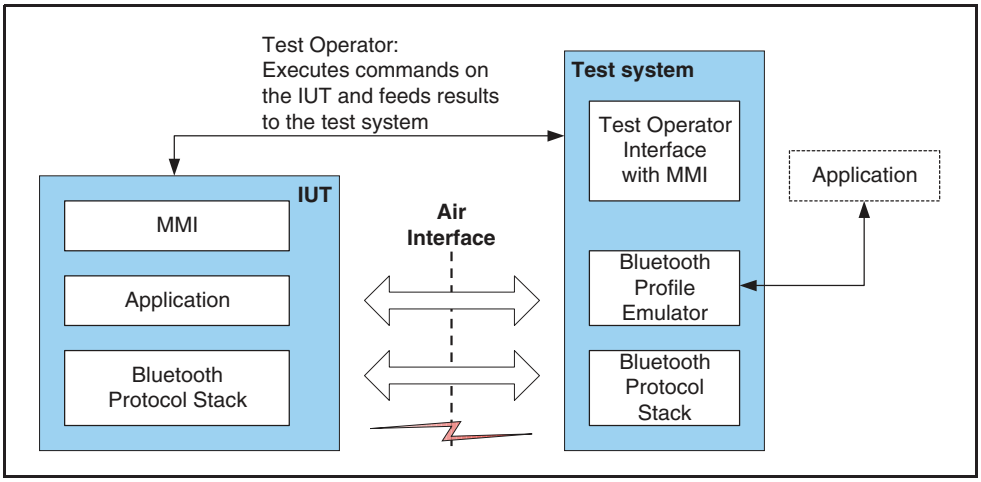



*Figure 2.2: General test set-up for profile qualification*



LifeScan, Inc. Ex. 1019
Page 1338 of 1420
IPR2025-00103

## 2.3 TCI CONFIGURATION AND USAGE

This interface is semantically and syntactically identical to the HCI interface described in “Host Controller Interface Functional Specification” on page 347[vol. 2]. The complete HCI interface is not required in the TCI, but the subset of HCI commands and events necessary for verifying the functionality of the IUT. The exact set of commands and events is specified in the test specifications for the layers subject to testing.

It is worth emphasizing again the TCI is an adapter, logically attached to the upper interface of the IUT. As such the TCI adapts the standardized signaling described here to the implementation specific interface of the IUT.

### 2.3.1 Transport layers

The method used to convey commands and events between the tester and the IUT’s upper interface can be either physical bearer or “software bearer”.

#### *2.3.1.1 Physical bearer*

It is recommended to use one of the transport layers specified for the HCI in. However, other physical bearers are not excluded and may be provided on test equipment. The use of a physical bearer is required for test cases active in category A. Please see the PRD for the definitions of test case categories. Please see the current active Test Case Reference List for the categorization of specific test cases.

#### *2.3.1.2 Software bearer*

There is no physical connection between the tester and the IUT’s upper interface. In this case, the manufacturer of the IUT shall supply test software and hardware that can be operated by a test operator. The operator will receive instructions from the tester and will execute them on the IUT. The “software bearer” shall support the same functionality as if using the TCI with a physical bearer. Use of the “software bearer” shall be agreed upon between the manufacturer of the IUT and the test facility that performs the qualification tests. The test facilities can themselves specify requirements placed on such a “software bearer”. Furthermore, the use of a “software bearer” is restricted to test cases active in one of the three lower categories B, C and D.



LifeScan, Inc. Ex. 1019
Page 1339 of 1420
IPR2025-00103

### 2.3.2 Baseband and link manager qualification

For the qualification of the link control part of the Baseband layer and for the Link Manager layer, the TCI is used as the interface between the test system and the upper interface of the IUT. The test system accesses the upper interface of the IUT by sending HCI commands and receiving HCI events from the IUT as described in the HCI specification. The required functionality on the TCI depends on the IUT's implemented functionality of the BB and LM layers, and therefore which test cases are executed.

A schematic example in Figure 2.3 shows the test configuration for BB and LM qualification of Bluetooth products which do not support HCI, and use a physical bearer for the TCI. In this example the Test Control (TC) Software represents what the manufacturer has to supply with the IUT when using an external test facility for qualification. The function of the TC Software is to adapt the implementation dependent interface to the TCI.

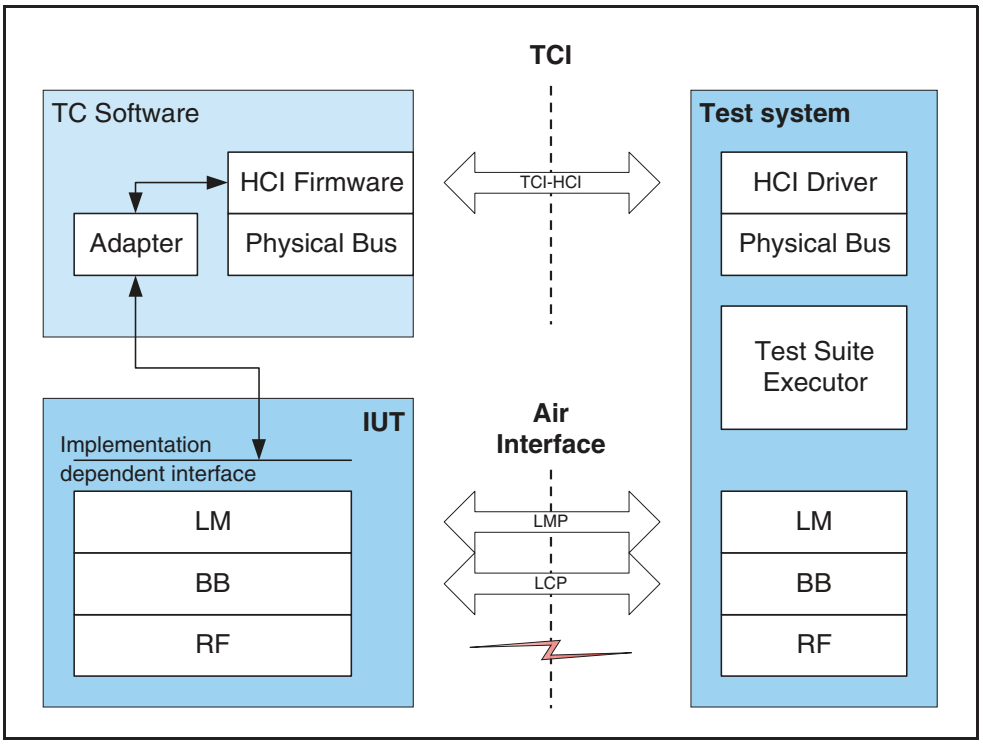



*Figure 2.3: BB and LM qualification with TCI physical bearer*



LifeScan, Inc. Ex. 1019
Page 1340 of 1420
IPR2025-00103

Figure 2.4 shows a schematic example of the test configuration for the same Bluetooth product using a “software bearer” for the TCI. Here the function of the Test Control Software is to represent the application that can be controlled by the test operator.

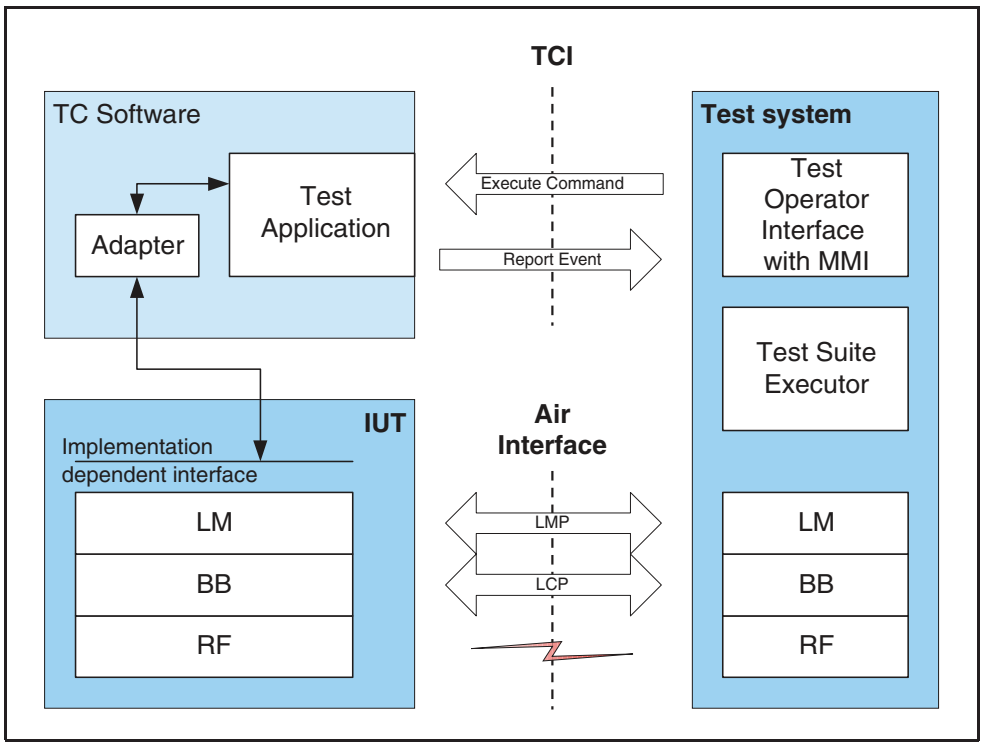



*Figure 2.4: BB and LM qualification with “software bearer”*



LifeScan, Inc. Ex. 1019
Page 1341 of 1420
IPR2025-00103

### 2.3.3 HCI qualification

The TCI may also be used for HCI signaling verification and qualification. The HCI signaling is only verified if support of the HCI functionality is claimed by the manufacturer.

A schematic example in Figure 2.5 shows the test configuration for HCI qualification of Bluetooth products. As can be seen in the figure the implemented HCI is used as the interface to the tester.

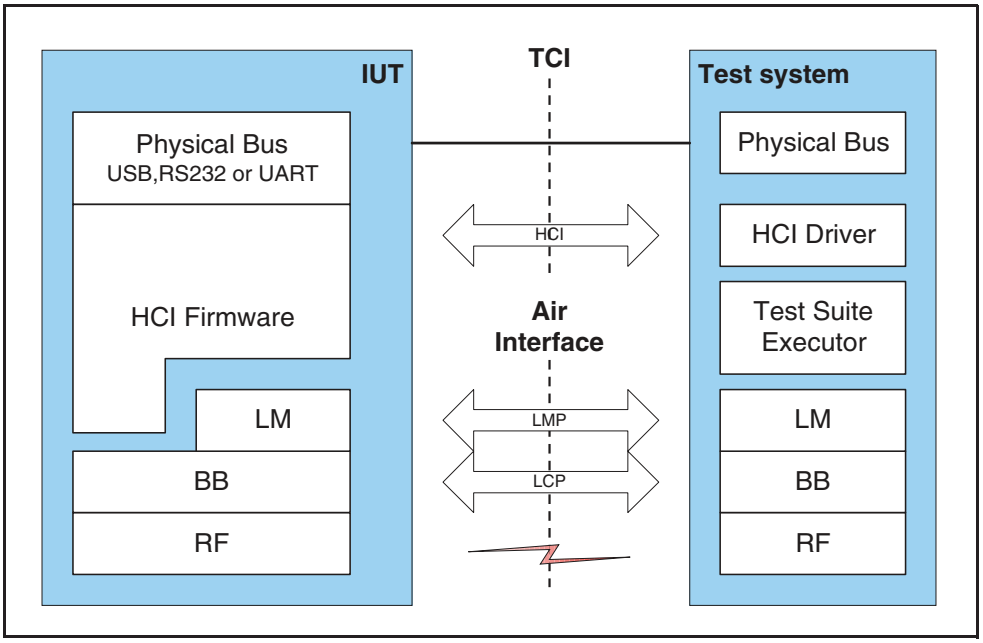



*Figure 2.5: General test set-up for HCI qualification*



LifeScan, Inc. Ex. 1019
Page 1342 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1343 of 1420
IPR2025-00103



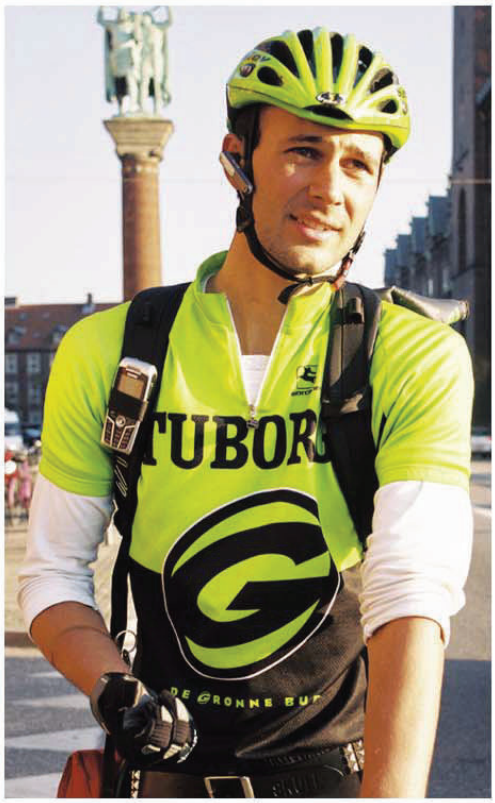


LifeScan, Inc. Ex. 1019
Page 1344 of 1420
IPR2025-00103

Specification Volume 4

# SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Wireless Connections Made Easy



# Host Controller Interface

## [Transport Layer]

**Covered Core Package version:**
**2.1 + EDR**

Bluetooth®

1345

LifeScan, Inc. Ex. 1019
Page 1345 of 1420
IPR2025-00103

## Revision History

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

## Contributors

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

## Web Site

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

## Disclaimer and Copyright Notice

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

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

Any use of the Specification not in compliance with the terms of the applicable Membership Agreement, Early Adopters Agreement or Promoters Agreement is prohibited and any such prohibited use may result in termination of the applicable Membership Agreement or Early Adopters Agreement and other liability permitted by the applicable agreement or by applicable law to Bluetooth SIG or any of its members for patent, copyright and/or trademark infringement.



LifeScan, Inc. Ex. 1019
Page 1346 of 1420
IPR2025-00103



THE SPECIFICATION IS PROVIDED “AS IS” WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NONINFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, SATISFACTORY QUALITY, OR REASONABLE SKILL OR CARE, OR ANY WARRANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

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

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

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

Copyright © 1999, 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007

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

*Third-party brands and names are the property of their respective owners.

LifeScan, Inc. Ex. 1019
Page 1348 of 1420
IPR2025-00103

# TABLE OF CONTENTS

**Part A**
**UART TRANSPORT LAYER**

1 General ..... 13
2 Protocol ..... 14
3 RS232 Settings ..... 15
4 Error Recovery ..... 16

**Part B**
**USB TRANSPORT LAYER**

1 Overview ..... 21
2 USB Endpoint Expectations ..... 23
2.1 Descriptor Overview ..... 23
2.2 Control Endpoint Expectations ..... 28
2.3 Bulk Endpoints Expectations ..... 28
2.4 Interrupt Endpoint Expectations ..... 28
2.5 Isochronous Endpoints Expectations ..... 29
3 Class Code ..... 30
4 Device Firmware Upgrade ..... 31
5 Limitations ..... 32
5.1 Power Specific Limitations ..... 32
5.2 Other Limitations ..... 32

**Part C**
**SECURE DIGITAL (SD)**
**TRANSPORT LAYER**

1 Introduction ..... 37
2 Goals ..... 38
2.1 Hardware Goals ..... 38
2.2 Software Goals ..... 38
3 Physical Interface Documents ..... 39
4 Communication ..... 40
4.1 Overview ..... 40
5 Appendix A - Acronyms and Abbreviations ..... 41
6 Appendix B - Related Documents ..... 42
7 Appendix C - Tests ..... 43



LifeScan, Inc. Ex. 1019
Page 1349 of 1420
IPR2025-00103

7.1 Test Suite Structure....43

**Part D**
**THREE-WIRE UART TRANSPORT LAYER**

**1 General....49**
**2 Overview....50**
**3 Slip Layer....51**
3.1 Encoding a Packet....51
3.2 Decoding a Packet....51
**4 Packet Header....53**
4.1 Sequence Number....53
4.2 Acknowledge Number....54
4.3 Data Integrity Check Present....54
4.4 Reliable Packet....54
4.5 Packet Type....54
4.6 Payload Length....55
4.7 Packet Header Checksum....55
**5 Data Integrity Check....56**
5.1 16 Bit CCITT-CRC....56
**6 Reliable Packets....57**
6.1 Header Checksum Error....57
6.2 Slip Payload Length Error....57
6.3 Data Integrity Check Error....57
6.4 Out Of Sequence Packet Error....57
6.5 Acknowledgement....58
6.6 Resending Packets....58
6.7 Example Reliable Packet Flow....58
**7 Unreliable Packets....61**
7.1 Unreliable Packet Header....61
7.2 Unreliable Packet Error....61
**8 Link Establishment....62**
8.1 Uninitialized State....63
8.2 Initialized State....63
8.3 Active State....63
8.4 Sync Message....64
8.5 Sync Response Message....64
8.6 Config Message....64
8.7 Config Response Message....65
8.8 Configuration Field....65
8.8.1 Configuration Messages....66



LifeScan, Inc. Ex. 1019
Page 1350 of 1420
IPR2025-00103

8.8.2 Sliding Window Size ..... 66
8.8.3 Level of Data Integrity Check ..... 66
8.8.4 Out of Frame Software Flow Control ..... 67
8.8.5 Version Number ..... 67
**9 LOW POWER ..... 68**
9.1 Wakeup Message ..... 68
9.2 Woken Message ..... 68
9.3 Sleep Message ..... 69
**10 Out of Frame Control ..... 70**
10.1 Software Flow Control ..... 70
**11 Hardware Configuration ..... 71**
11.1 Wires ..... 71
11.1.1 Transmit & Receive ..... 71
11.1.2 Ground ..... 71
11.2 Hardware Flow ..... 71
11.2.1 RTS & CTS ..... 71
**12 Recommended Parameters ..... 72**
12.1 Timing Parameters ..... 72
12.1.1 Acknowledgement of Packets ..... 72
12.1.2 Resending Reliable Packets ..... 72
**13 References ..... 73**



LifeScan, Inc. Ex. 1019
Page 1351 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1352 of 1420
IPR2025-00103

# UART TRANSPORT LAYER

***This document describes the UART transport layer (between the Host and the Host Controller). HCI command, event, and data packets flow through this layer, but the layer does not decode them.***



LifeScan, Inc. Ex. 1019
Page 1353 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1354 of 1420
IPR2025-00103

# CONTENTS

1 General ..... 13
2 Protocol ..... 14
3 RS232 Settings ..... 15
4 Error Recovery ..... 16



LifeScan, Inc. Ex. 1019
Page 1355 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1356 of 1420
IPR2025-00103

# 1 GENERAL

The objective of this HCI UART Transport Layer is to make it possible to use the Bluetooth HCI over a serial interface between two UARTs on the same PCB. The HCI UART Transport Layer assumes that the UART communication is free from line errors.

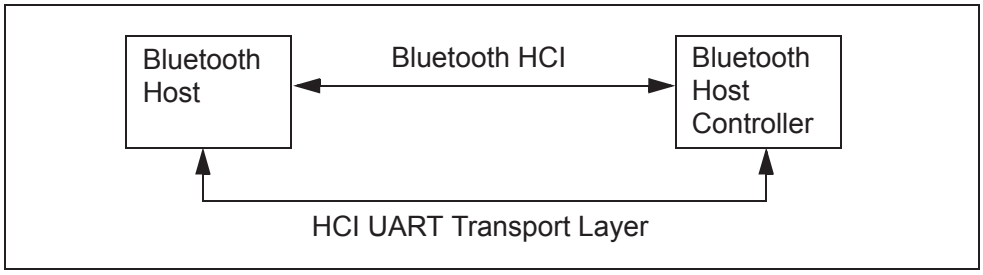



*Figure 1.1: HCI UART Transport Layer*



LifeScan, Inc. Ex. 1019
Page 1357 of 1420
IPR2025-00103

# 2 PROTOCOL

There are four kinds of HCI packets that can be sent via the UART Transport Layer; i.e. HCI Command Packet, HCI Event Packet, HCI ACL Data Packet and HCI Synchronous Data Packet (see "Host Controller Interface Functional Specification" in Volume 2, Part E). HCI Command Packets can only be sent to the Bluetooth Host Controller, HCI Event Packets can only be sent from the Bluetooth Host Controller, and HCI ACL/Synchronous Data Packets can be sent both to and from the Bluetooth Host Controller.

HCI does not provide the ability to differentiate the four HCI packet types. Therefore, if the HCI packets are sent via a common physical interface, a HCI packet indicator has to be added according to Table 2.1 below.

| HCI packet type | HCI packet indicator |
|---|---|
| HCI Command Packet | 0x01 |
| HCI ACL Data Packet | 0x02 |
| HCI Synchronous Data Packet | 0x03 |
| HCI Event Packet | 0x04 |

*Table 2.1: HCI packet indicators*

The HCI packet indicator shall be sent immediately before the HCI packet. All four kinds of HCI packets have a length field, which is used to determine how many bytes are expected for the HCI packet. When an entire HCI packet has been received, the next HCI packet indicator is expected for the next HCI packet. Over the UART Transport Layer, only HCI packet indicators followed by HCI packets are allowed.



LifeScan, Inc. Ex. 1019
Page 1358 of 1420
IPR2025-00103

# 3 RS232 SETTINGS

The HCI UART Transport Layer uses the following settings for RS232:

| **Baud rate:** | **manufacturer-specific** |
|---|---|
| Number of data bits: | 8 |
| Parity bit: | no parity |
| Stop bit: | 1 stop bit |
| Flow control: | RTS/CTS |
| Flow-off response time: | manufacturer specific |

*Table 3.1:*

Flow control with RTS/CTS is used to prevent temporary UART buffer overrun. It should not be used for flow control of HCI, since HCI has its own flow control mechanisms for HCI commands, HCI events and HCI data.

If CTS is 1, then the Host/Host Controller is allowed to send.
If CTS is 0, then the Host/Host Controller is not allowed to send.

The flow-off response time defines the maximum time from setting RTS to 0 until the byte flow actually stops.

The RS232 signals should be connected in a null-modem fashion; i.e. the local TXD should be connected to the remote RXD and the local RTS should be connected to the remote CTS and vice versa.

# 4 ERROR RECOVERY

If the Host or the Host Controller lose synchronization in the communication over RS232, then a reset is needed. A loss of synchronization means that an incorrect HCI packet indicator has been detected, or that the length field in an HCI packet is out of range.

If the UART synchronization is lost in the communication from Host to Host Controller, then the Host Controller shall send a Hardware Error Event to tell the Host about the synchronization error. The Host Controller will then expect to receive an HCI_Reset command from the Host in order to perform a reset. The Host Controller will also use the HCI_Reset command in the byte stream from Host to Host Controller to re-synchronize.

If the UART synchronization is lost in the communication from Host Controller to Host, then the Host shall send the HCI_Reset command in order to reset the Host Controller. The Host shall then re-synchronize by looking for the HCI Command Complete event for the HCI_Reset command in the byte stream from Host Controller to Host.

See "Host Controller Interface Functional Specification" for HCI commands and HCI events in the Bluetooth Specification v1.2 or later.



LifeScan, Inc. Ex. 1019
Page 1360 of 1420
IPR2025-00103

# USB TRANSPORT LAYER

***This document describes the USB transport layer (between a host and the host controller). HCI commands flow through this layer, but the layer does not decode the commands.***



LifeScan, Inc. Ex. 1019
Page 1361 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1362 of 1420
IPR2025-00103

# CONTENTS

**1 Overview ..... 21**

**2 USB Endpoint Expectations ..... 23**

2.1 Descriptor Overview ..... 23

2.2 Control Endpoint Expectations ..... 28

2.3 Bulk Endpoints Expectations ..... 28

2.4 Interrupt Endpoint Expectations ..... 28

2.5 Isochronous Endpoints Expectations ..... 29

**3 Class Code ..... 30**

**4 Device Firmware Upgrade ..... 31**

**5 Limitations ..... 32**

5.1 Power Specific Limitations ..... 32

5.2 Other Limitations ..... 32



LifeScan, Inc. Ex. 1019
Page 1363 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1364 of 1420
IPR2025-00103

# 1 OVERVIEW

This document discusses the requirements of the Universal Serial Bus (USB) interface for Bluetooth hardware. Readers should be familiar with USB, USB design issues, Advanced Configuration Power Interface (ACPI), the overall Bluetooth architecture, and the basics of the radio interface.

The reader should also be familiar with the Bluetooth Host Controller Interface.

Referring to Figure 1.1 below, notice that this document discusses the implementation details of the two-way arrow labelled 'USB Function'.

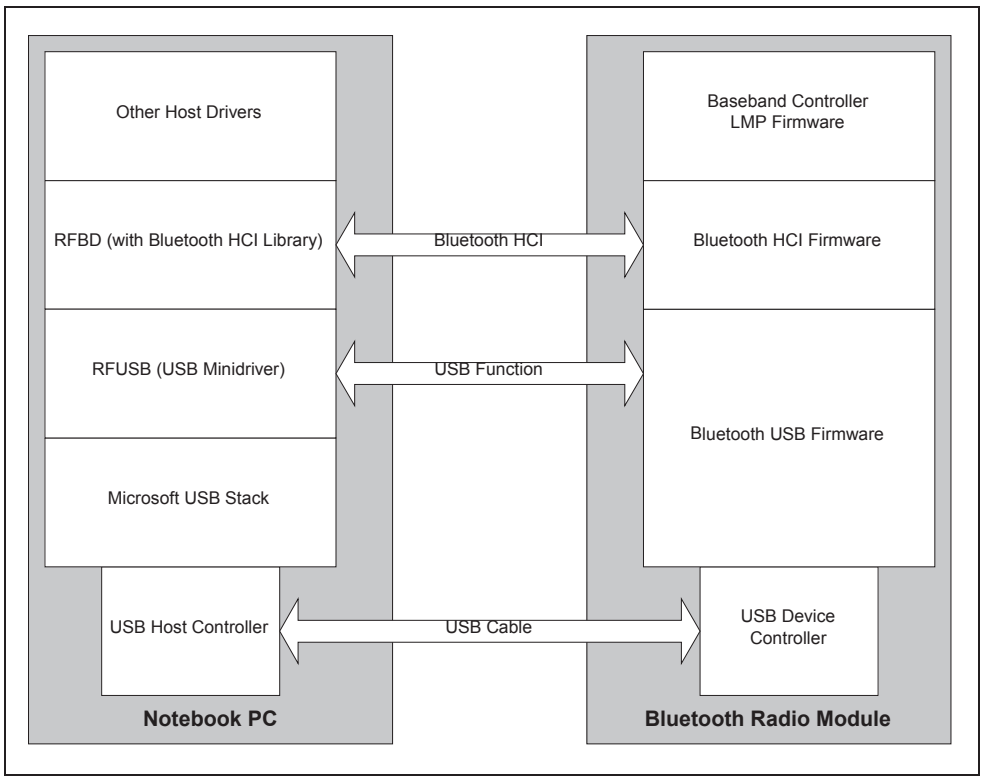



*Figure 1.1: Relationship between the host and the Bluetooth Radio Module*

The USB hardware can be embodied in one of two ways:

1. As a USB dongle, and
2. Integrated onto the motherboard of a notebook PC.

Finally, for an overview of the connection that is established between two Bluetooth devices, reference Figure 1.2, below.



LifeScan, Inc. Ex. 1019
Page 1365 of 1420
IPR2025-00103

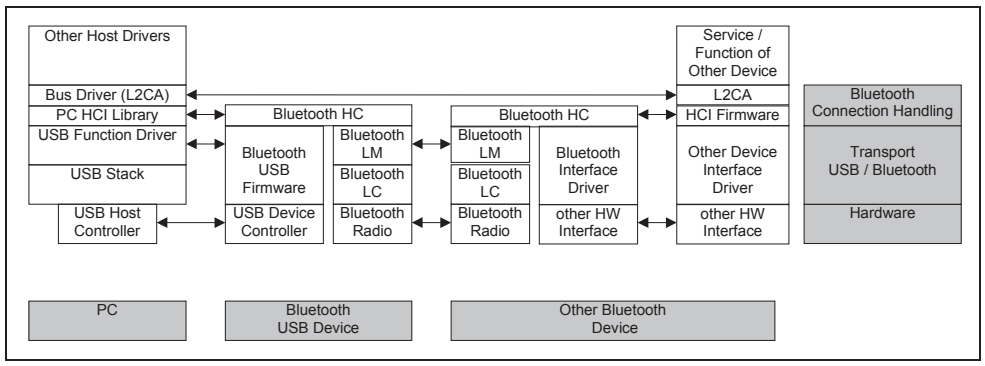



*Figure 1.2: Flow of data from one Bluetooth device to another*



LifeScan, Inc. Ex. 1019
Page 1366 of 1420
IPR2025-00103

# 2 USB ENDPOINT EXPECTATIONS

This section outlines specific USB endpoints that are required in order to function properly with the host. This section assumes a basic familiarity with USB. The endpoint numbers (labelled 'Suggested Endpoint Address' below) may be dynamically recognized upon driver initialization – this depends on the implementation.

## 2.1 DESCRIPTOR OVERVIEW

The USB device is intended for high speed. The firmware configuration consists of two interfaces. The first interface (interface zero) has no alternate settings and contains the bulk and interrupt endpoints. The second interface (interface one) provides scalable isochronous bandwidth consumption. The second interface has four alternate settings that provide different consumption based on the required isochronous bandwidth. The default interface is empty so that the device is capable of scaling down to no isochronous bandwidth.

An HCI frame, consisting of an HCI header and HCI data, should be contained in one USB transaction. A USB transaction is defined as one or more USB frames that contain the data from one IO request. For example, an ACL data packet containing 256 bytes (both HCI header and HCI data) would be sent over the bulk endpoint in one IO request. That IO request will require four 64-byte USB frames, and forms a transaction.

The endpoints are spread across two interfaces so that when adjusting isochronous bandwidth consumption (via select interface calls), any pending bulk and/or interrupt transactions do not have to be terminated and resubmitted.



LifeScan, Inc. Ex. 1019
Page 1367 of 1420
IPR2025-00103

The following table outlines the required configuration.

| Interface Number | Alternate Setting | Suggested Endpoint Address | Endpoint Type | Suggested Max Packet Size |
|---|---|---|---|---|
| **HCI Commands** | | | | |
| 0 | 0 | 0x00 | Control | 8/16/32/64 |
| **HCI Events** | | | | |
| 0 | 0 | 0x81 | Interrupt (IN) | 16 |
| **ACL Data** | | | | |
| 0 | 0 | 0x82 | Bulk (IN) | 32/64 |
| 0 | 0 | 0x02 | Bulk (OUT) | 32/64 |
| **No active voice channels (for USB compliance)** | | | | |
| 1 | 0 | 0x83 | Isoch (IN) | 0 |
| 1 | 0 | 0x03 | Isoch (OUT) | 0 |
| **One voice channel with 8-bit encoding** | | | | |
| 1 | 1 | 0x83 | Isoch (IN) | 9 |
| 1 | 1 | 0x03 | Isoch (OUT) | 9 |
| **Two voice channels with 8-bit encoding & One voice channel with 16-bit encoding** | | | | |
| 1 | 2 | 0x83 | Isoch (IN) | 17 |
| 1 | 2 | 0x03 | Isoch (OUT) | 17 |
| **Three voice channels with 8-bit encoding** | | | | |
| 1 | 3 | 0x83 | Isoch (IN) | 25 |
| 1 | 3 | 0x03 | Isoch (OUT) | 25 |
| **Two voice channels with 16-bit encoding** | | | | |
| 1 | 4 | 0x83 | Isoch (IN) | 33 |
| 1 | 4 | 0x03 | Isoch (OUT) | 33 |
| **Three voice channels with 16-bit encoding** | | | | |
| 1 | 5 | 0x83 | Isoch (IN) | 49 |
| 1 | 5 | 0x03 | Isoch (OUT) | 49 |

*Table 2.1: USB firmware interface and endpoint settings*

The following two examples are used to demonstrate the flow of data given the describe endpoints.

| Number of voice channels | Duration of voice data | Encoding |
|---|---|---|
| One | 3 ms per IO Request | 8-bit |

| Time (ms | USB data (header refers to HCI header) (Receive & Send from the host) | Queued data (read / write) | Time (ms) | Air data | Amount Received/ Sent (ms) |
|---|---|---|---|---|---|
| 0 | Receive 0 bytes<br>Send 9 bytes (3 header, 6 data) | 0 / 6 | 0 | Send 0 | 0 / 0 |
| | | 10 / 6 | 0.625 | Receive 10 | 1.25 / 0 |
| 1 | Receive 0 bytes<br>Send 9 bytes (9 bytes HCI data) | 10 / 15 | 1.25 | Send 0 | 1.25 / 0 |
| | | 20 / 15 | 1.875 | Receive 10 | 2.50 / 0 |
| 2 | Receive 0 bytes<br>Send 9 bytes (9 bytes HCI data) | 20 / 24 | 2.50 | Send 0 | 2.50 / 0 |
| | | 30 / 24 | 3.125 | Receive 10 | 3.75 / 0 |
| 3 | Receive 9 bytes (3 header, 6 data)<br>Send 9 bytes (3 header, 6 data) | 24 / 20 | 3.75 | Send 10 | 3.75 / 1.25 |
| 4 | Receive 9 bytes (9 bytes data)<br>Send 9 bytes (9 bytes HCI data) | 25 / 29 | 4.375 | Receive 10 | 5.0 / 1.25 |
| 5 | Receive 9 bytes (9 bytes data)<br>Send 9 bytes (9 bytes HCI data) | 16 / 28 | 5.0 | Send 10 | 5.0 / 2.50 |
| | | 26 / 28 | 5.625 | Receive 10 | 6.25 / 2.50 |
| 6 | Receive 9 bytes (3 header, 6 data)<br>Send 9 bytes (3 header, 6 data) | 20 / 24 | 6.25 | Send 10 | 6.25 / 3.75 |
| | | 30 / 24 | 6.875 | Receive 10 | 7.5 / 3.75 |
| 7 | Receive 9 bytes (9 bytes data)<br>Send 9 bytes (9 bytes HCI data) | 21 / 23 | 7.5 | Send 10 | 7.5 / 5.0 |
| 8 | Receive 9 bytes (9 bytes data)<br>Send 9 bytes (9 bytes HCI data) | 22 / 32 | 8.125 | Receive 10 | 8.75 / 5.0 |
| | | 22 / 22 | 8.75 | Send 10 | 8.75 / 6.25 |
| 9 | Receive 9 bytes (3 header, 6 data)<br>Send 9 bytes (3 header, 6 data) | 26 / 28 | 9.375 | Receive 10 | 10.0 / 6.25 |

*Table 2.2: USB transactions*

| Time (ms | USB data (header refers to HCI header) (Receive & Send from the host) | Queued data (read / write) | Time (ms) | Air data | Amount Received/ Sent (ms) |
|---|---|---|---|---|---|
| 10 | Receive 9 bytes (9 bytes data)<br>Send 9 bytes (9 bytes HCI data) | 17 / 27 | 10 | Send 10 | 10.0 / 7.5 |
| | | 27 / 27 | 10.625 | Receive 10 | 11.25 / 7.5 |
| 11 | Receive 9 bytes (9 bytes data)<br>Send 9 bytes (9 bytes HCI data) | 18 / 26 | 11.25 | Send 10 | 11.25 / 8.75 |

*Table 2.2: USB transactions*

Convergence is expected because the radio is sending out an average of eight bytes of voice data every one ms and USB is sending eight bytes of voice data every one ms.

| Number of voice channels | Duration of voice data | Encoding |
|---|---|---|
| Two | 3 ms per IO Request | 8-bit |

| Time (ms | USB data (header refers to HCI header) (Receive & Send from the host) | Queued data (read / write) | Time (ms) | Air data | Amount Received / Sent (ms) |
|---|---|---|---|---|---|
| 0 | Receive 0 bytes for Channel #1<br>Send 17 bytes (3 header, 14 data) for Channel #1 | C1- 0/14<br>C2- 0/0 | 0 | Send 0 for C1 | C1- 0/0<br>C2- 0/0 |
| | | C1- 20/14<br>C2- 0/0 | 0.625 | Receive 20 for C1 | C1- 2.5/0<br>C2- 0/0 |
| 1 | Receive 0 bytes for Channel #1<br>Send 17 bytes (17 bytes HCI data) for Channel #1 | C1- 20/31<br>C2- 0/0 | 1.25 | Send 0 for C2 | C1- 2.5/0<br>C2- 0/0 |
| | | C1- 20/31<br>C2- 20/0 | 1.875 | Receive 20 for C2 | C1- 2.5/0<br>C2- 2.5/0 |
| 2 | Receive 0 bytes for Channel #1<br>Send 17 bytes (17 bytes HCI data) for Channel #1 | C1- 20/28<br>C2- 20/0 | 2.50 | Send 20 for C1 | C1- 2.5/2.5<br>C2- 2.5/0 |
| | | C1- 40/28<br>C2- 0/0 | 3.125 | Receive 20 for C1 | C1- 5.0/2.5<br>C2- 2.5/0 |

*Table 2.3: Convergence of radio and USB data*

| Time (ms | USB data (header refers to HCI header) (Receive & Send from the host) | Queued data (read / write) | Time (ms) | Air data | Amount Received / Sent (ms) |
|---|---|---|---|---|---|
| 3 | Receive 0 bytes for Channel #2<br>Send 17 bytes (3 header, 14 data) for Channel #2 | C1- 40/28<br>C2- 20/14 | 3.75 | Send 0 for C2 | C1- 5.0/2.5<br>C2- 2.5/0 |
| 4 | Receive 0 bytes for Channel #2<br>Send 17 bytes (17 bytes HCI data) for Channel #2 | C1- 40/28<br>C2- 40/31 | 4.375 | Receive 20 for C2 | C1- 5.0/2.5<br>C2- 5.0/0 |
| 5 | Receive 0 bytes for Channel #2<br>Send 17 bytes (17 bytes HCI data) for Channel #2 | C1- 40/8<br>C2- 40/48 | 5.0 | Send 20 for C1 | C1- 5.0/5.0<br>C2- 5.0/0 |
| | | C1- 60/8<br>C2- 40/48 | 5.625 | Receive 20 for C1 | C1- 7.5/5.0<br>C2- 5.0/0 |
| 6 | Receive 17 bytes (3 header, 14 data) for Channel #1<br>Send 17 bytes (3 header, 14 data) for Channel #1 | C1- 46/22<br>C2- 40/28 | 6.25 | Send 20 for C2 | C1- 7.5/5.0<br>C2- 5.0/2.5 |
| | | C1- 46/22<br>C2- 60/28 | 6.875 | Receive 20 for C2 | C1- 7.5/5.0<br>C2- 7.5/2.5 |
| 7 | Receive 17 bytes (17 bytes data) for Channel #1<br>Send 17 bytes (17 bytes HCI data) for Channel #1 | C1- 29/19<br>C2- 60/28 | 7.5 | Send 20 for C1 | C1- 7.5/7.5<br>C2- 7.5/2.5 |
| 8 | Receive 17 bytes (17 bytes data) for Channel #1<br>Send 17 bytes (17 bytes HCI data) for Channel #1 | C1- 32/36<br>C2- 60/28 | 8.125 | Receive 20 for C1 | C1- 10/7.5<br>C2- 7.5/2.5 |
| | | C1- 32/36<br>C2- 60/8 | 8.75 | Send 20 for C2 | C1- 10/7.5<br>C2- 7.5/5.0 |
| 9 | Receive 17 bytes (3 header, 14 data) for Channel #2<br>Send 17 bytes (3 header, 14 data) for Channel #2 | C1- 32/36<br>C2- 54/22 | 9.375 | Receive 20 for C2 | C1- 10/7.5<br>C2- 10/5.0 |
| 10 | Receive 17 bytes (17 bytes data) for Channel #2<br>Send 17 bytes (17 bytes HCI data) for Channel #2 | C1- 32/16<br>C2- 37/39 | 10 | Send 20 for C1 | C1- 10/10<br>C2- 10/5.0 |
| | | C1- 52/16<br>C2- 37/39 | 10.625 | Receive 20 for C1 | C1- 12.5/10<br>C2- 10/5.0 |

*Table 2.3: Convergence of radio and USB data*



LifeScan, Inc. Ex. 1019
Page 1371 of 1420
IPR2025-00103

| Time (ms | USB data (header refers to HCI header) (Receive & Send from the host) | Queued data (read / write) | Time (ms) | Air data | Amount Received / Sent (ms) |
|---|---|---|---|---|---|
| 11 | Receive 17 bytes (17 bytes data) for Channel #2<br>Send 17 bytes (17 bytes HCI data) for Channel #2 | C1- 52/16<br>C2- 20/36 | 11.25 | Send 20 for C2 | C1- 12.5/10<br>C2- 10/7.5 |

*Table 2.3: Convergence of radio and USB data*

## 2.2 CONTROL ENDPOINT EXPECTATIONS

Endpoint 0 is used to configure and control the USB device. Endpoint 0 will also be used to allow the host to send HCI-specific commands to the host controller. When the USB firmware receives a packet over this endpoint that has the Bluetooth class code, it should treat the packet as an HCI command packet.

## 2.3 BULK ENDPOINTS EXPECTATIONS

Data integrity is a critical aspect for ACL data. This, in combination with bandwidth requirements, is the reason for using a bulk endpoint. Multiple 64-byte packets can be shipped, per millisecond, across the bus.

Suggested bulk max packet size is 64 bytes. Bulk has the ability to transfer multiple 64-byte buffers per one millisecond frame, depending on available bus bandwidth.

Bulk has the ability to detect errors and correct them. Data flowing through this pipe might be destined for several different slaves. In order to avoid starvation, a flow control model similar to the shared endpoint model is recommended for the host controller.

## 2.4 INTERRUPT ENDPOINT EXPECTATIONS

An interrupt endpoint is necessary to ensure that events are delivered in a predictable and timely manner. Event packets can be sent across USB with a guaranteed latency.

The interrupt endpoint should have an interval of 1 ms.

The USB software and firmware requires no intimate knowledge of the events passed to the host controller.



LifeScan, Inc. Ex. 1019
Page 1372 of 1420
IPR2025-00103

## 2.5 ISOCHRONOUS ENDPOINTS EXPECTATIONS

These isochronous endpoints transfer synchronous data to and from the host controller of the radio.

Time is the critical aspect for this type of data. The USB firmware should transfer the contents of the data to the host controllers' synchronous FIFOs. If the FIFOs are full, the data should be overwritten with new data.

These endpoints have a one (1) ms interval, as required by Chapter 9 of the USB Specification, Versions 1.0 and 1.1.

The radio is capable of three (3) 64Kb/s voice channels (and can receive the data coded in different ways – 16-bit linear audio coding is the method that requires the most data). A suggested max packet size for this endpoint would be at least 64 bytes. (It is recommended that max packet sizes be on power of 2 boundaries for optimum throughput.) However, if it is not necessary to support three voice channels with 16-bit coding, 32 bytes could also be considered an acceptable max packet size.

# 3 CLASS CODE

A class code will be used that is specific to all USB Bluetooth devices. This will allow the proper driver stack to load, regardless of which vendor built the device. It also allows HCI commands to be differentiated from USB commands across the control endpoint.

The class code (bDeviceClass) is 0xE0 – Wireless Controller.

The SubClass code (bDeviceSubClass) is 0x01 – RF Controller.

The Protocol code (bDeviceProtocol) is 0x01 – Bluetooth programming.



LifeScan, Inc. Ex. 1019
Page 1374 of 1420
IPR2025-00103

# 4 DEVICE FIRMWARE UPGRADE

Firmware upgrade capability is not a required feature. But if implemented, the firmware upgrade shall be compliant with the "Universal Serial Bus Device Class Specification for Device Firmware Upgrade" (version 1.0 dated May 13, 1999) available on the USB Forum web site at http://www.usb.org.



LifeScan, Inc. Ex. 1019
Page 1375 of 1420
IPR2025-00103

# 5 LIMITATIONS

## 5.1 POWER SPECIFIC LIMITATIONS

Today, the host controller of USB-capable machines resides inside a chip known as PIIX4. Unfortunately, because of errata, the USB host controller will not receive power while the system is in S3 or S4. This means that a USB wake-up can only occur when the system is in S1 or S2.

Another issue with the USB host controller is that, while a device is attached, it continually snoops memory to see if there is any work that needs to be done. The frequency that it checks memory is 1ms. This prevents the processor from dropping into a low power state known as C3. Because the notebook processor is not able to enter the C3 state, significant power loss will occur. This is a real issue for business users – as a typical business user will spend almost 90% of their time in the C3 state.

## 5.2 OTHER LIMITATIONS

Data corruption may occur across isochronous endpoints. Endpoints one and two may suffer from data corruption.

USB provides 16-CRC on all data transfers. The USB has a bit error rate of $10^{-13}$.

*Note that when a dongle is removed from the system, the radio will lose power (assuming this is a bus-powered device). This means that devices will lose connection.*



LifeScan, Inc. Ex. 1019
Page 1376 of 1420
IPR2025-00103

# SECURE DIGITAL (SD) TRANSPORT LAYER

***This document describes the SD transport layer (between the Host and Controller). HCI command, event and data packets flow through this layer, but the layer does not decode them. The Bluetooth SD Transport layer is defined in a document owned and maintained by the Secure Digital Association. Information regarding that document is described herein.***



LifeScan, Inc. Ex. 1019
Page 1377 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1378 of 1420
IPR2025-00103

# CONTENTS

**1 Introduction ..... 37**

**2 Goals ..... 38**

2.1 Hardware Goals ..... 38

2.2 Software Goals ..... 38

**3 Physical Interface Documents ..... 39**

**4 Communication ..... 40**

4.1 Overview ..... 40

**5 Appendix A - Acronyms and Abbreviations ..... 41**

**6 Appendix B - Related Documents ..... 42**

**7 Appendix C - Tests ..... 43**

7.1 Test Suite Structure ..... 43



LifeScan, Inc. Ex. 1019
Page 1379 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1380 of 1420
IPR2025-00103

# 1 INTRODUCTION

This document discusses the requirements of the Secure Digital (SD) interface for Bluetooth hardware. Readers should be familiar with SD, SD design issues, and the overall Bluetooth architecture. The reader should also be familiar with the Bluetooth Host Controller Interface.

The SD Bluetooth Protocol is documented in the SDIO Card Type-A Specification for Bluetooth, which is owned and maintained by the Secure Digital Association (SDA). The full specification is available to members of the SDA that have signed all appropriate SD NDA and license requirements. The SDA also makes a Non-NDA version available, the Simplified Version of: SDIO Card Type-A Specification for Bluetooth. There are no changes to the SDA document to comply with the requirements of the Bluetooth SIG.



LifeScan, Inc. Ex. 1019
Page 1381 of 1420
IPR2025-00103

# 2 GOALS

## 2.1 HARDWARE GOALS

The Bluetooth SD transport interface specification is designed to take advantage of both the SD Physical Transport bus and the packet orientation of the Bluetooth HCI protocol. Thus, all data is transferred in blocks as packets. Since the block size used on the SD bus may be smaller than the HCI packet, a segmentation and recombination protocol is defined.

## 2.2 SOFTWARE GOALS

The Bluetooth SD transport interface specification is designed for non-embedded solutions. It is assumed that the host software does not necessarily have a priori knowledge of the SD Bluetooth device.

The interface is not designed for embedded applications where much of the information passed via the interface is known in advance.

The SDA also defines a Bluetooth interface for embedded applications where the Controller contains protocol layers above HCI (RFComm, SDP etc.). This specification is called SDIO Card Type-B Specification for Bluetooth. Information about this specification can be obtained from the SDA (http://www.sdcard.org).



LifeScan, Inc. Ex. 1019
Page 1382 of 1420
IPR2025-00103

# 3 PHYSICAL INTERFACE DOCUMENTS

This specification references the SD SDIO Card Type-A Specification for Bluetooth. This SDA document defines the Bluetooth HCI for all SD devices that support an HCI level interface. Any SD Bluetooth device claiming compliance with the SD Bluetooth Transport must support this interface and additionally adhere to its device type specification, which is set by the Secure Digital Association. The SDIO Card Type-A Specification for Bluetooth document is based on the SDIO Card Specification, which in turn is based on the SD Memory Card Specification: Part 1 Physical Layer Specification. All of these documents are copyrighted by the SDA and are available ONLY to SDA member companies that have signed the appropriate NDA documents with the SDA. As an introduction to the SD Bluetooth Type A specification, the SDA has created 'Simplified' versions of each of these documents. The simplified versions do not contain enough information to fully implement a device, however they do contain enough information to convey the structure and intent of the specifications.

Applicable SDA Documents available to members of the SDA:

**SD Memory Card Specification: Part 1 Physical Layer Specification**

**SDIO Card Specification**

**SDIO Card Type-A Specification for Bluetooth.**

Applicable Simplified SDA Documents available to non-members and members of the SDA:

**Simplified Version of: SD Memory Card Specification: Part 1 Physical Layer Specification**

**Simplified Version of: SDIO Card Specification:**

**Simplified Version of: SDIO Card Type-A Specification for Bluetooth**

More information on the Secure Digital Association and the SD specifications can be found at the SDA website at. http://www.sdcard.org.



LifeScan, Inc. Ex. 1019
Page 1383 of 1420
IPR2025-00103

# 4 COMMUNICATION

## 4.1 OVERVIEW

Figure 4.1 below is a diagram of the communication interface between a Bluetooth SD device and the Bluetooth host protocol stack. Modifications to this diagram might be needed for operating systems that do not support a miniport model:

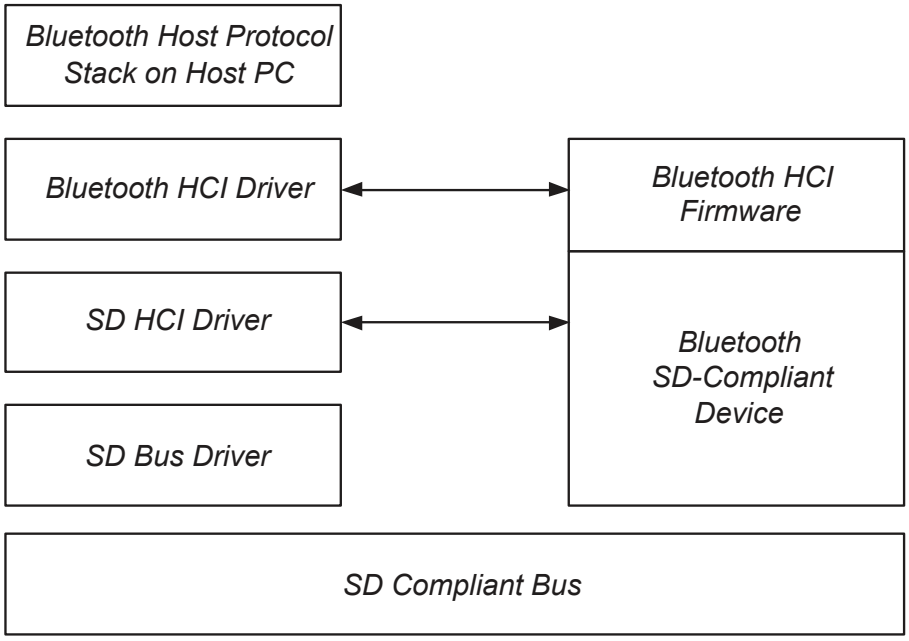



*Figure 4.1: SD Communication Diagram*



LifeScan, Inc. Ex. 1019
Page 1384 of 1420
IPR2025-00103

# 5 APPENDIX A - ACRONYMS AND ABBREVIATIONS

| Acronym | Description |
|---|---|
| HCI | Host Controller Interface |
| NDA | Non-Disclosure Agreement |
| OS | Operating System |
| SD | Secure Digital |
| SDA | Secure Digital Association |
| SDIO | Secure Digital Input/Output |
| SDP | Service Discovery Protocol |
| SIG | Special Interest Group |

*Table 5.1: Acronyms and Abbreviations*



LifeScan, Inc. Ex. 1019
Page 1385 of 1420
IPR2025-00103

# 6 APPENDIX B - RELATED DOCUMENTS

A) Bluetooth Core Specification v1.2 or later.

B) Applicable SDA Documents available to members of the SDA:

B.1) SD Memory Card Specification: Part 1 Physical Layer Specification

B.2) SDIO Card Specification

B.3) SDIO Card Type-A Specification for Bluetooth.

B.4) SDIO Card Type-B Specification for Bluetooth.

B.5) SDIO Card Physical Test Specification

B.5) SDIO Host Physical Test Specification

B.6) SD Bluetooth Type A Test Specification

**These documents are available to members of the SDA in the "Members Only" section of the SDA web site (http://www.sdcard.org/access.htm). See http://www.sdcard.org/join.htm for information on joining the SDA.**

C) Applicable Simplified SDA Documents available to non-members and members of the SDA:

C.1) Simplified Version of: SD Memory Card Specification: Part 1 Physical Layer Specification
http://www.sdcard.org/sdphysical_simplifyed_Ver101.pdf

C.2) Simplified Version of: SDIO Card Specification http://www.sdcard.org/SDIO-SimpleSpec-1.00_A.pdf

C.3) Simplified Version of: SDIO Card Type-A Specification for Bluetooth - http://www.sdcard.org/Simple_of_SDIO_Card_Type_A_Spec_v1.0_Draft_C_20020524.pdf

# 7 APPENDIX C - TESTS

The SDA has defined formal test procedures for SDIO Type A Bluetooth cards (Controller) and Hosts. It is expected that both Controllers and Hosts will comply with all test requirements set forth by the SDA in accordance with the rules of the SDA. The Bluetooth SIG does not require any formal testing to comply with SIG requirements. The test document names are listed in Appendix B.

## 7.1 TEST SUITE STRUCTURE

There are two types of tests defined for the HCI SD Transport Layer:

1. Functional Tests
2. Protocol Tests

Tests of both types are defined for both the Host and Controller.

The purpose of the functional tests is to verify that the SD Bluetooth Type A Specification, SDIO Standard and SD Physical Standard have been implemented according to the specifications. These tests and the test environment for these tests are defined in documents provided by the SDA.

The purpose of the protocol tests are to verify that the Bluetooth Controller SD implementation or the Host implementation are according to the SD Bluetooth Type A specification.

The test environment for the protocol tests consists of the tester and the Device Under Test (DUT) as illustrated in Figure 7.1 below.

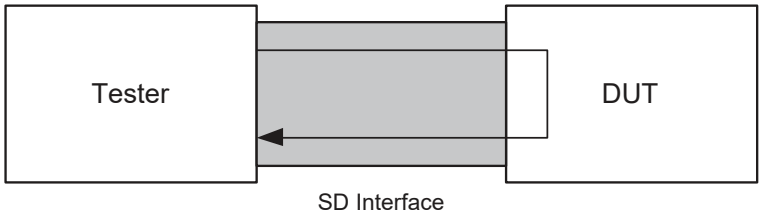



*Figure 7.1: Protocol Test Environment*

The tester is typically a PC with an SD interface. The DUT is placed into local loopback mode and standard HCI commands are used to drive the tests. The test results are verified in the tester.



LifeScan, Inc. Ex. 1019
Page 1387 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1388 of 1420
IPR2025-00103

# THREE-WIRE UART TRANSPORT LAYER

***This document describes the Three-Wire UART transport layer (between the Host and Controller). HCI command, event and data packets flow through this layer, but the layer does not decode them.***



LifeScan, Inc. Ex. 1019
Page 1389 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1390 of 1420
IPR2025-00103

# CONTENTS

**1 General ..... 49**

**2 Overview ..... 50**

**3 Slip Layer ..... 51**

3.1 Encoding a Packet ..... 51

3.2 Decoding a Packet ..... 51

**4 Packet Header ..... 53**

4.1 Sequence Number ..... 53

4.2 Acknowledge Number ..... 54

4.3 Data Integrity Check Present ..... 54

4.4 Reliable Packet ..... 54

4.5 Packet Type ..... 54

4.6 Payload Length ..... 55

4.7 Packet Header Checksum ..... 55

**5 Data Integrity Check ..... 56**

5.1 16 Bit CCITT-CRC ..... 56

**6 Reliable Packets ..... 57**

6.1 Header Checksum Error ..... 57

6.2 Slip Payload Length Error ..... 57

6.3 Data Integrity Check Error ..... 57

6.4 Out Of Sequence Packet Error ..... 57

6.5 Acknowledgement ..... 58

6.6 Resending Packets ..... 58

6.7 Example Reliable Packet Flow ..... 58

**7 Unreliable Packets ..... 61**

7.1 Unreliable Packet Header ..... 61

7.2 Unreliable Packet Error ..... 61

**8 Link Establishment ..... 62**

8.1 Uninitialized State ..... 63

8.2 Initialized State ..... 63

8.3 Active State ..... 63

8.4 Sync Message ..... 64

8.5 Sync Response Message ..... 64

8.6 Config Message ..... 64

8.7 Config Response Message ..... 65

8.8 Configuration Field ..... 65

8.8.1 Configuration Messages ..... 66

8.8.2 Sliding Window Size ..... 66

8.8.3 Level of Data Integrity Check ..... 66



LifeScan, Inc. Ex. 1019
Page 1391 of 1420
IPR2025-00103

8.8.4 Out of Frame Software Flow Control....67
8.8.5 Version Number....67
**9 LOW POWER....68**
9.1 Wakeup Message....68
9.2 Woken Message....68
9.3 Sleep Message....69
**10 Out of Frame Control....70**
10.1 Software Flow Control....70
**11 Hardware Configuration....71**
11.1 Wires....71
11.1.1 Transmit & Receive....71
11.1.2 Ground....71
11.2 Hardware Flow....71
11.2.1 RTS & CTS....71
**12 Recommended Parameters....72**
12.1 Timing Parameters....72
12.1.1 Acknowledgement of Packets....72
12.1.2 Resending Reliable Packets....72
**13 References....73**

# 1 GENERAL

The HCI Three-Wire UART Transport Layer makes it possible to use the Bluetooth HCI over a serial interface between two UARTs. The HCI Three-Wire UART Transport Layer assumes that the UART communication may have bit errors, overrun errors or burst errors. See also “UART Transport Layer” on page 9[vol. 4].



LifeScan, Inc. Ex. 1019
Page 1393 of 1420
IPR2025-00103

# 2 OVERVIEW

The HCI Three-Wire UART Transport Layer is a connection based protocol that transports HCI commands, events, ACL and Synchronous packets between the Bluetooth Host and the Bluetooth Controller. Packet construction is in done in two steps. First, it adds a packet header onto the front of every HCI Packet which describes the payload. Second, it frames the packets using a SLIP protocol. Finally, it sends this packet over the UART interface.

The SLIP layer converts an unreliable octet stream into an unreliable packet stream. The SLIP layer places start and end octets around the packet. It then changes all occurrences of the frame start or end octet in the packet to an escaped version.

The packet header describes the contents of the packet, and if this packet needs to be reliably transferred, a way of identifying the packet uniquely, allowing for retransmission of erroneous packets.

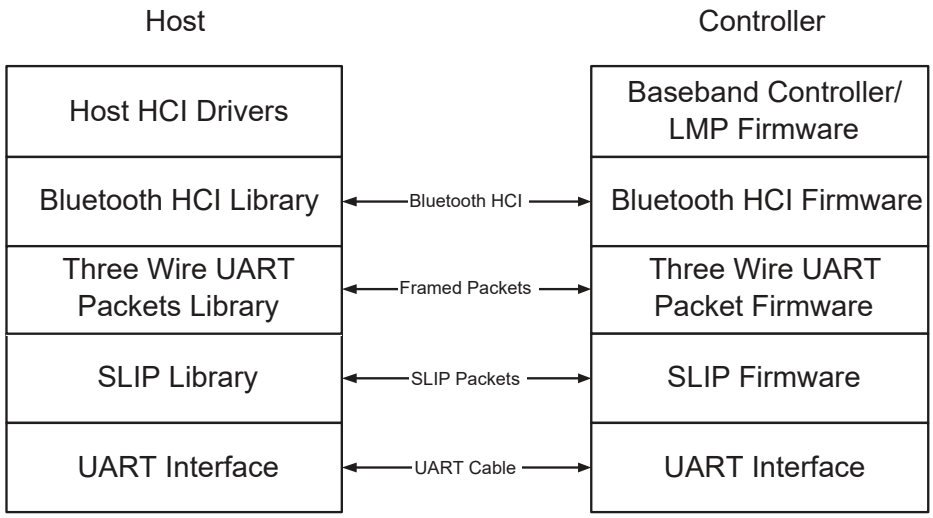



*Figure 2.1: The Relationship Between the Bluetooth Host and the Bluetooth Controller*



LifeScan, Inc. Ex. 1019
Page 1394 of 1420
IPR2025-00103

# 3 SLIP LAYER

The SLIP layer places packet framing octets around each packet being transmitted over the Three-Wire UART Transport Layer. This delimits the packets and allows packet boundaries to be detected if the receiver loses synchronization. The SLIP layer is based upon the RFC 1055 Standard [1].

## 3.1 ENCODING A PACKET

The SLIP layer performs octet stuffing on the octets entering the layer so that specific octet codes which may occur in the original data do not occur in the resultant stream.

The SLIP layer places octet 0xC0 at the start and end of every packet it transmits. Any occurrence of 0xC0 in the original packet is changed to the sequence 0xDB 0xDC before being transmitted. Any occurrence of 0xDB in the original packet is changed to the sequence 0xDB 0xDD before being transmitted. These sequences, 0xDB 0xDC and 0xDB 0xDD are SLIP escape sequences. All SLIP escape sequences start with 0xDB. All SLIP escape sequences are listed in Table 3.1.

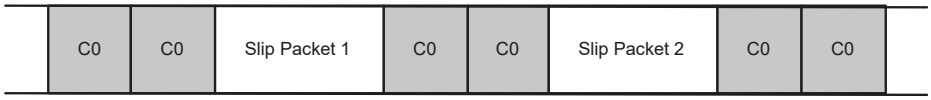



*Figure 3.1: SLIP Packets with 0xC0 at the Start and End of Each Packet*

## 3.2 DECODING A PACKET

When decoding a SLIP stream, a device will first be in an unknown state, not knowing if it is at the start of a packet or in the middle of a packet. The device must therefore discard all octets until it finds a 0xC0. If the 0xC0 is followed immediately by a second 0xC0, then the device will discard the first 0xC0 as it was presumably the end of the last packet, and the second 0xC0 was the start of the next packet. The device will then be in the decoding packet state. It can then decode the octets directly changing any SLIP escape sequences back into their unencoded form. When the device decodes the 0xC0 at the end of the packet, it will calculate the length of the SLIP packet, and pass the packet data into the packet decoder. The device will then seek the next packet. If the device does not receive an 0xC0 for the start of the next packet, then all octets up to and including the next 0xC0 will be discarded.



LifeScan, Inc. Ex. 1019
Page 1395 of 1420
IPR2025-00103

| SLIP Escape Sequence | Unencoded form | Notes |
|---|---|---|
| 0xDB 0xDC | 0xC0 | |
| 0xDB 0xDD | 0xDB | |
| 0xDB 0xDE | 0x11 | Only valid when OOF Software Flow Control is enabled |
| 0xDB 0xDF | 0x13 | Only valid when OOF Software Flow Control is enabled |

*Table 3.1: SLIP Escape Sequences*



LifeScan, Inc. Ex. 1019
Page 1396 of 1420
IPR2025-00103

# 4 PACKET HEADER

Every packet that is sent over the Three-Wire UART Transport Layer has a packet header. It also has an optional Data Integrity Check at the end of the payload. The Transport Layer does not support packet segmentation and reassembly. Each transport packet will contain at most one higher layer packet.

A packet consists of a Packet Header of 4 octets, a Payload of 0 to 4095 octets, and an optional Data Integrity Check of 2 octets. See Figure 4.1.

The Packet header consists of a Sequence Number of 3 bits, an Acknowledge Number of 3 bits, a Data Integrity Check Present bit, a Reliable Packet bit, a Packet Type of 4 bits, a Payload Length of 12 bits and an 8 bit Header Checksum. See Figure 4.2.

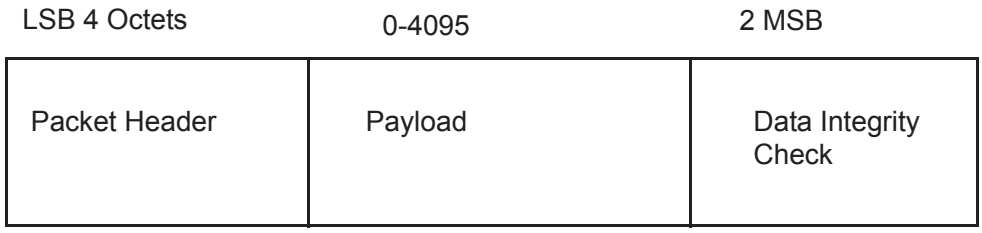



*Figure 4.1: Packet Format*

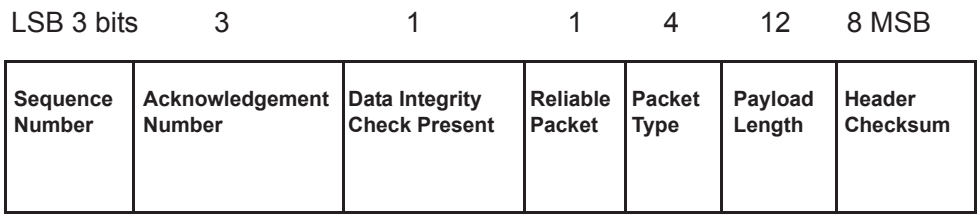



*Figure 4.2: Packet Header Format*

## 4.1 SEQUENCE NUMBER

For unreliable packets this field will be set to 0 on transmit and ignored on receive.

Each new reliable packet will be assigned a sequence number which will be equal to the sequence number of the previous reliable packet plus one modulo eight. A packet will use the same sequence number each time it is retransmitted.



LifeScan, Inc. Ex. 1019
Page 1397 of 1420
IPR2025-00103

## 4.2 ACKNOWLEDGE NUMBER

The acknowledge number must be set to the sequence number of the next reliable packet this device expects to receive. See Section 6.4 on page 57.

## 4.3 DATA INTEGRITY CHECK PRESENT

If a 16 bit CCITT-CRC Data Integrity Check is appended to the end of the payload, this bit shall be set to 1.

## 4.4 RELIABLE PACKET

If this bit it set to 1, then this packet is reliable. This means that the sequence number field is valid, and the receiving end must acknowledge its receipt. If this bit is set to 0, then this packet is unreliable.

## 4.5 PACKET TYPE

There are four kinds of HCI packets that can be sent via the Three-Wire UART Transport Layer; these are HCI Command Packet, HCI Event Packet, HCI ACL Data Packet and HCI Synchronous Data Packet (see “Host Controller Interface Functional Specification” in the Bluetooth Core Specification v1.2 or later). HCI Command Packets can be sent only to the Bluetooth Controller, HCI Event Packets can be sent only from the Bluetooth Controller, and HCI ACL/ Synchronous Data Packets can be sent both to and from the Bluetooth Controller.

HCI packet coding does not provide the ability to differentiate the four HCI packet types. Therefore, the Packet Type field is used to distinguish the different packets. The acceptable values for this Packet Type field are given in Table 4.1.

| HCI Packet Type | Packet Type |
|---|---|
| Acknowledgement Packets | 0 |
| HCI Command Packet | 1 |
| HCI ACL Data Packet | 2 |
| HCI Synchronous Data Packet | 3 |
| HCI Event Packet | 4 |
| Reserve | 5-13 |
| Vendor Specific | 14 |
| Link Control Packet | 15 |

*Table 4.1: Three-Wire UART Packet Type*



LifeScan, Inc. Ex. 1019
Page 1398 of 1420
IPR2025-00103

HCI Command Packets, HCI ACL Data Packets and HCI Event Packets are always sent as reliable packets. HCI Synchronous Data Packets are sent as unreliable packets unless HCI Synchronous Flow Control is enabled, in which case they are sent as reliable packets.

In addition to the four HCI packet types, other packet types are defined. One packet type is defined for pure Acknowledgement Packets, and one additional packet type is to support link control. One packet type is made available to vendors for their own use. All other Three-Wire UART Packet Types are reserved for future use.

## 4.6 PAYLOAD LENGTH

The payload length is the number of octets in the payload data. This does not include the length of the packet header, or the length of the optional data integrity check.

## 4.7 PACKET HEADER CHECKSUM

The packet header checksum validates the contents of the packet header against corruption. This is calculated by setting the Packet Header Checksum to a value such that the 2's complement sum modulo 256 of the four octets of the Packet Header including the Packet Header Checksum is 0xFF.



LifeScan, Inc. Ex. 1019
Page 1399 of 1420
IPR2025-00103

# 5 DATA INTEGRITY CHECK

The Data Integrity Check field is optional. It can be used to ensure that the packet is valid. The Data Integrity Check field is appended onto the end of the packet. Each octet of the Packet Header and Packet Payload is used to compute the Data Integrity Check.

## 5.1 16 BIT CCITT-CRC

The CRC is defined using the CRC-CCITT generator polynomial

$g(D) = D^{16} + D^{12} + D^5 + 1$

(see Figure 5.1)

The CRC shift register is filled with 1s before calculating the CRC for each packet. Octets are fed through the CRC generator least significant bit first.

The most significant parity octet is transmitted first (where the CRC shift register is viewed as shifting from the least significant bit towards the most significant bit). Therefore, the transmission order of the parity octets within the CRC shift register is as follows:

x[8] (first), x[9],..., x[15], x[0], x[1],..., x[7] (last)

where x[15] corresponds to the highest power CRC coefficient and x[0] corresponds to the lowest power coefficient.

The switch S shall be set in position 1 while the data is shifted in. After the last bit has entered the LFSR, the switch shall be set in position 2, and the registers contents shall be read out for transmission.

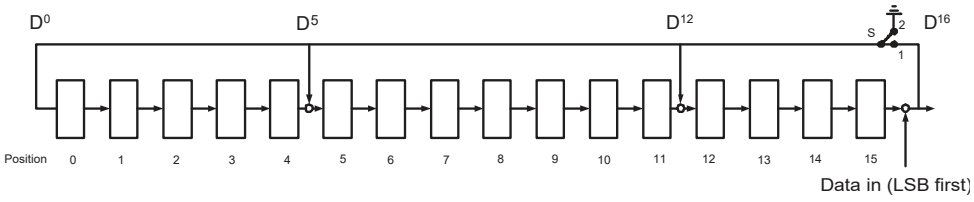



*Figure 5.1: The LFSR Circuit Generating the CRC*



LifeScan, Inc. Ex. 1019
Page 1400 of 1420
IPR2025-00103

# 6 RELIABLE PACKETS

To allow the reliable transmission of packets through the transport, a method needs to be defined to recover from packet errors. The Host or Controller can detect a number of different errors in the packet.

## 6.1 HEADER CHECKSUM ERROR

The header of the packet is protected by a Packet Header Checksum. If the 2's complement sum modulo 256 of the four octets of the header is not 0xFF, then the packet has an unrecoverable error and all information contained in the packet shall be discarded.

## 6.2 SLIP PAYLOAD LENGTH ERROR

The length of the SLIP packet shall be checked against the Packet Payload Length. If the Data Integrity Check Present bit is set to 1, then the SLIP packet length should be 6 + Packet Payload Length. If the Data Integrity Check Present bit is set to 0, then the SLIP packet length should be 4 + Packet Payload Length. If this check fails, then all information contained in the packet shall be discarded. The SLIP packet length is the length of the data received from the SLIP layer after the SLIP framing, and SLIP escape codes have been processed.

## 6.3 DATA INTEGRITY CHECK ERROR

The packet may have a Data Integrity Check at the end of the payload. This is controlled by the Data Integrity Check Present bit in the header. If this is set to 1, then the Data Integrity Check at the end of the payload is checked. If this is different from the value expected, then the packet shall be discarded. If the link is configured to not use data integrity checks, and a packet is received with the Data Integrity Check Present bit set to 1, then the packet shall be discarded.

## 6.4 OUT OF SEQUENCE PACKET ERROR

Each device keeps track of the sequence number it expects to receive next. This will be one more than the sequence number of the last successfully received reliable packet, modulo eight. If a reliable packet is received which has the expected sequence number, then this packet shall be accepted.

If a reliable packet is received which does not have the expected sequence number, then the packet shall be discarded.



LifeScan, Inc. Ex. 1019
Page 1401 of 1420
IPR2025-00103

## 6.5 ACKNOWLEDGEMENT

Whenever a reliable packet is received, an acknowledgement shall be generated.

If a packet is available to be sent, the Acknowledgement Number of that packet shall be updated to the latest expected sequence number.

If a requirement to send an acknowledgement value is pending, but there are no other packets available to be sent, the device can send a pure Acknowledgement Packet. This is an Unreliable Packet, with the Packet Type set to 0, Payload Length set to 0, and the Sequence Number set to 0. The Acknowledge Number must be set correctly.

The maximum number of reliable packets that can be sent without acknowledgement defines the sliding window size of the link. This is configured during link establishment. See Sections 8.6, 8.7 and 8.8.

## 6.6 RESENDING PACKETS

A Reliable Packet shall be resent until it is acknowledged. Devices should refrain from resending packets too quickly to avoid saturating the link with retransmits. See Section 12.1.2 on page 72.

## 6.7 EXAMPLE RELIABLE PACKET FLOW

Figure 6.1 shows the transmission of reliable packets between two devices. Device A sends a packet with a Sequence Number of 6, and an Acknowledgement Number of 3. Device B receives this packet correctly, so needs to generate an acknowledgement. Device B then sends a packet with Sequence Number 3 with its Acknowledgement Number set to the next expected packet Sequence Number from Device A of 7.



LifeScan, Inc. Ex. 1019
Page 1402 of 1420
IPR2025-00103

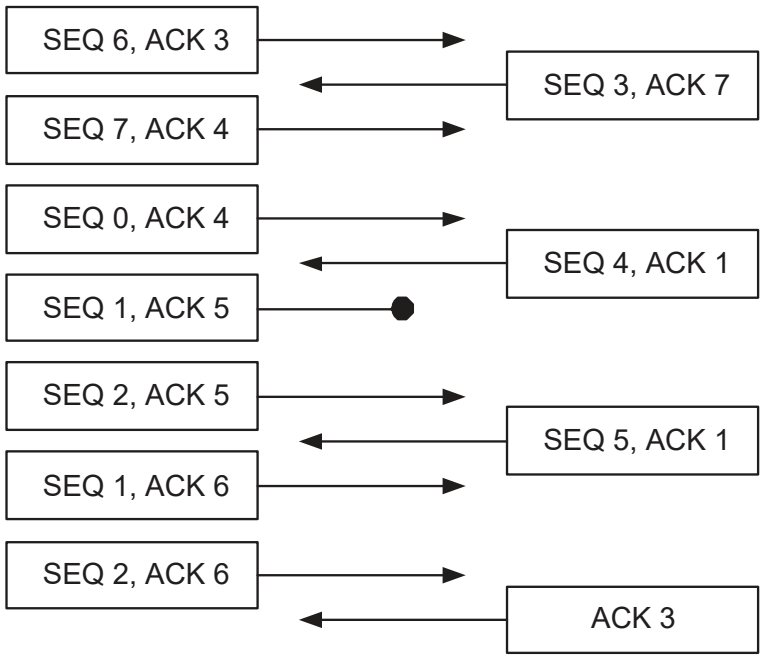



*Figure 6.1: Message Diagram Showing Transmission of Reliable Packets*

Device A receives a packet with Sequence Number 3 and an Acknowledgement Number of 7. Device A was expecting this sequence number so needs to generate an acknowledgement. The Acknowledgement Number of 7 is one greater than the last Sequence Number that was sent, meaning that this packet was received correctly (see Section 6.6 on page 58).

Device A sends two packets, Sequence Numbers 7 and 0. Both packets have the Acknowledgement Number of 4, the next sequence number it expects from Device B. Device B receives the first correctly, and increments its next expected sequence number to 0. It then receives the second packet correctly, and increments the next expected sequence number to 1.

Device B sends a packet with Sequence Number 4, and the Acknowledgement Number of 1. This will acknowledge both of the previous two packets sent by Device A.

Device A now sends two more packets, Sequence Numbers 1 and 2. Unfortunately, the first packet is corrupted. Device B receives the first packet, and discovers the error, so discards this packet (see Section 6.1 on page 57, Section 6.2 on page 57 or Section 6.3 on page 57). It must generate an acknowledgement of this erroneously received reliable packet. Device B then receives the second packet. This is received out of sequence, as it is currently expecting Sequence Number 1, but has received Sequence Number 2 (see 6.4). Again, it must generate an acknowledgement.

Device B sends another packet with Sequence Number 5. It is still expecting a packet with Sequence Number 1 next, so the Acknowledgement Number is set to 1. Device A receives this, and accepts this packet.

Device A has not had either of its last two packets acknowledged, so it must resend them (see 6.6). It does this, but must update the Acknowledgement Number of the original packets that were sent (see Section 6.5 on page 58). The Sequence Numbers of these packets must stay the same (see Section 4.1 on page 53).

Device B receives these packets correctly, and schedules the sending of an acknowledgement. Because Device B doesn't have any data packets that need to be sent, it sends a pure Acknowledgement Packet (see Section 6.5 on page 58).



LifeScan, Inc. Ex. 1019
Page 1404 of 1420
IPR2025-00103

# 7 UNRELIABLE PACKETS

To allow the transmission of unreliable packets through the transport, the following method shall be used.

## 7.1 UNRELIABLE PACKET HEADER

An unreliable packet header always has the Reliable Packet bit set to 0. The sequence number shall be set to 0. The Data Integrity Check Present, Acknowledgement Number, Packet Type, Payload Length and Packet Header Checksum shall all be set the same as a Reliable Packet.

## 7.2 UNRELIABLE PACKET ERROR

If a packet that is marked as unreliable and the packet has an error, then the packet shall be discarded.



LifeScan, Inc. Ex. 1019
Page 1405 of 1420
IPR2025-00103

# 8 LINK ESTABLISHMENT

Before any packets except Link Control Packets can be sent, the Link Establishment procedure must be performed. This ensures that the sequence numbers are initialized correctly, it also ensures that the two sides are using the same baud rate, allow detection of peer reset, and allows the device to be configured.

Link Establishment is defined by a state machine with three states: Uninitialized, Initialized and Active. When the transport is first started, the link is in the Uninitialized State. There are four messages that are defined: SYNC, SYNC RESPONSE, CONFIG and CONFIG RESPONSE. All four link establishment messages shall be sent with the Data Integrity Present flag set to 0.

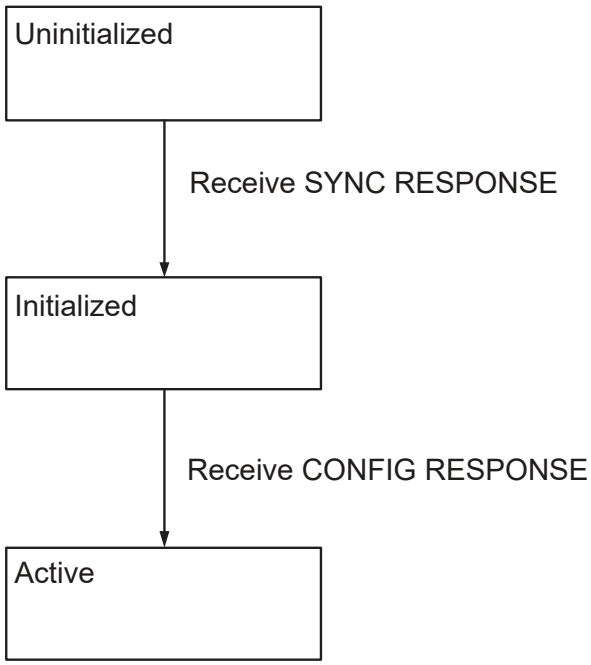



*Figure 8.1: Link Establishment State Diagram*



LifeScan, Inc. Ex. 1019
Page 1406 of 1420
IPR2025-00103

## 8.1 UNINITIALIZED STATE

In the Uninitialized State a device periodically[1] sends SYNC messages. If a SYNC message is received, the device shall respond with a SYNC RESPONSE message. If a SYNC RESPONSE message is received, the device shall move to the Initialized State. In the Initialized State only SYNC and SYNC RESPONSE messages are valid, all other messages that are received must be discarded. If an invalid packet is received, the device shall respond with a SYNC message. The device shall not send any acknowledgement packets in the Uninitialized State[2].

In the Uninitialized State the Controller may wait until it receives a SYNC message before sending its first SYNC message. This allows the Host to control when the Controller starts to send data.

The SYNC message can be used for automatic baud rate detection. It is assumed that the Controller shall stay on a single baud rate, while the Host could hunt for the baud rate. Upon receipt of a SYNC RESPONSE message, the Host can assume that the correct baud rate has been detected.

## 8.2 INITIALIZED STATE

In the Initialized State a device periodically sends CONFIG messages. If a SYNC message is received, the device shall respond with a SYNC RESPONSE message. If a CONFIG message is received, the device shall respond with a CONFIG RESPONSE message. If a CONFIG RESPONSE message is received, the device will move to the Active State. All other messages that are received must be ignored.

## 8.3 ACTIVE STATE

In the Active State, a device can transfer higher layer packets through the transport. If a CONFIG message is received, the device shall respond with a CONFIG RESPONSE message. If a CONFIG RESONSE message is received, the device shall discard this message.

If a SYNC message is received while in the Active State, it is assumed that the peer device has reset. The local device should therefore perform a full reset of the upper stack, and start Link Establishment again at the Uninitialized State.

Upon entering the Active State, the first packet sent shall have its SEQ and ACK numbers set to zero.

1. During link establishment, various messages are sent periodically. It is suggested to send 4 messages per second.
2. Any packet that was erroneous would normally be acknowledged, as the recipient does not know if the packet was a reliable packet or not. The recipient cannot do this in the Uninitialized State, as it is possible to receive corrupt data while the Uninitialized state.

## 8.4 SYNC MESSAGE

The SYNC message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2.

The payload is composed of the octet pattern 0x01 0x7E[1].

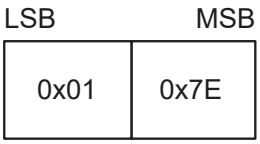



*Figure 8.2: Sync Message Format*

## 8.5 SYNC RESPONSE MESSAGE

The SYNC RESPONSE message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2. The payload is composed of the octet pattern 0x02 0x7D.

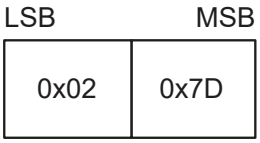



*Figure 8.3: Sync Response Message Format*

## 8.6 CONFIG MESSAGE

The CONFIG message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2 plus the size of the Configuration Field. The payload is composed of the octet pattern 0x03 0xFC and the Configuration Field.

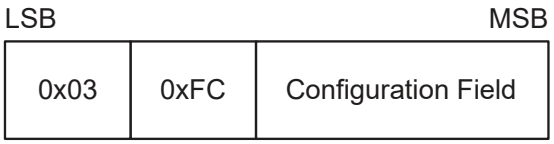



*Figure 8.4: Configuration Message Format*

1. The second octet for all Link Control Packets equals the least significant 7 bits of the first octet, inverted, with the most significant bit set to ensure even parity.

## 8.7 CONFIG RESPONSE MESSAGE

The CONFIG RESPONSE message is an unreliable message sent with the Packet Type of 15 and a Payload Length of 2 plus the size of the Configuration Field. The payload is composed of the octet pattern 0x04 0x7B and the Configuration Field.

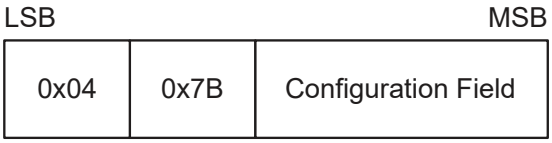



*Figure 8.5: Configuration Response Message Format*

## 8.8 CONFIGURATION FIELD

The Configuration Field contains the Version Number, Sliding Window Size, the Data Integrity Check Type, and if Out Of Frame (OOF) Software Flow Control is allowed.

The Configuration Field in a CONFIG message sent by the Host determines what the Host can transmit and accept. The Configuration Field in a CONFIG RESPONSE message sent by the Controller determines what the Host and Controller shall transmit and can expect to receive.

The Controller sends CONFIG messages without a Configuration Field. The Host sends CONFIG RESPONSE messages without a Configuration Field.

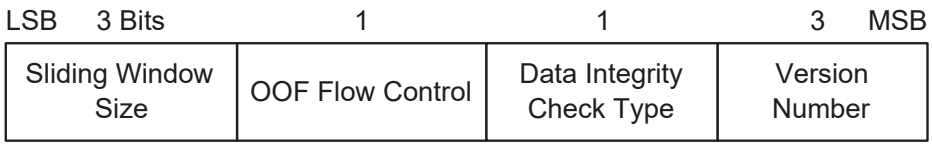



*Figure 8.6: Configuration Field Detail*

To allow for future extension of the Configuration Field, the size of the message determines the number of significant Configuration Octets in the payload. Future versions of this specification may use extra octets. Any bits that are not included in the message shall be set to 0. Any bits that are not defined are reserved and shall be set to 0.

A device shall not change the values if sends in the Configuration Field during Link Establishment.



LifeScan, Inc. Ex. 1019
Page 1409 of 1420
IPR2025-00103

### 8.8.1 Configuration Messages

The CONFIG – CONFIG RESPONSE message sequence configures the link in both directions. Until a CONFIG RESPONSE message is received only unreliable Link Establishment messages may be sent. Once CONFIG RESPONSE message has been received all other packet types may be sent, and received messages passed up to the Host.

The CONFIG and CONFIG RESPONSE messages contain a set of options for both devices on the link. The Host sends a CONFIG message with the set of options that the Host would like to use. The Controller

responds with a CONFIG RESPONSE message with the set of options that the Host and the Controller will use. This means that the Controller is in full control of the set of options that will be used for all messages sent by both the Host and Controller.

### 8.8.2 Sliding Window Size

This is the maximum number of reliable packets a sender of the CONFIG message can send without requiring an acknowledgement. The value of this field shall be in the range one to seven. The value in the CONFIG RESPONSE message shall be less than or equal to the value in the CONFIG message. For example, the Host may suggest a window size of five in its CONFIG message and the Controller may respond with a value of three in its CONFIG RESPONSE message, but not six or seven. Both devices will then use a maximum sliding window size of three.

### 8.8.3 Level of Data Integrity Check

The CONFIG message contains a bit field describing the types of Data Integrity Checks the sender is prepared to transmit. The peer will select the one it is prepared to use and send its choice in the CONFIG RESPONSE message.

If data integrity checks are not required, then the Data Integrity Check Present bit shall be set to 0 by the Host and Controller.

| Level of Data Integrity | Parameter Description for CONFIG Message |
|---|---|
| 0 | No Data Integrity Check is supported. |
| 1 | 16 bit CCITT-CRC may be used. |

*Figure 8.7: Data Integrity Check Type in the CONFIG Message*



LifeScan, Inc. Ex. 1019
Page 1410 of 1420
IPR2025-00103

| Level of Data Integrity | Parameter Description for CONFIG RESPONSE Message |
|---|---|
| 0 | No Data Integrity Check must be used. |
| 1 | 16 bit CCITT-CRC may be used. |

*Figure 8.8: Data Integrity Check Type in the CONFIG RESPONSE Message*

### 8.8.4 Out of Frame Software Flow Control

By default, the transport uses no flow control except that mandated by the HCI Functional Specification and the flow control achieved by not acknowledging reliable Host messages. If Software Flow Control is to be used, this needs to be negotiated.

The CONFIG message specifies whether the sender of the CONFIG message is prepared to receive Out of Frame Software Flow Control messages. The CONFIG RESPONSE message specifies whether the peer can send Out of Frame Software Flow Control messages. The CONFIG RESPONSE message may have the field set to 1 only if the CONFIG message had it set to 1. (See Section 10.1 on page 70)

### 8.8.5 Version Number

The Version Number of this protocol shall determine which facilities are available to be used.

The CONFIG message specifies the Version Number supported by the Host. The CONFIG RESPONSE message specifies the Version Number that shall be used by the Host and Controller when sent by the Controller. The value in the CONFIG RESPONSE message shall be less than or equal to the value in the CONFIG message. The Version Numbers are enumerated in Figure 8.9. This specification is version 1.0 (Version Number = 0).

| Version Number | Parameter Description for CONFIG and CONFIG RESPONSE Message |
|---|---|
| 0 | Version 1.0 of this Protocol |
| 1-7 | Reserved for future use |

*Figure 8.9: Version Number in the CONFIG and CONFIG RESPONSE message*



LifeScan, Inc. Ex. 1019
Page 1411 of 1420
IPR2025-00103

# 9 LOW POWER

After a device is in the Active State, either side of the transport link may wish to enter a low power state. Because recovery from a loss of synchronization is possible, it is allowable to stop listening for incoming packets at any time.

To make the system more responsive after a device has entered a low power state, a system of messages is employed to allow either side to notify the other that they are entering a low power state and to wake a device from that state. These messages are sent as Link Control Packets. It is optional for a device to support the Sleep message. The Wakeup and Woken messages are mandatory.

## 9.1 WAKEUP MESSAGE

The Wakeup message shall be the first message sent whenever the device believes that the other side is asleep. The device shall then repeatedly send the Wakeup message until the Woken message is received. There must be at least a one character gap between the sending of each Wakeup message to allow the UART to resynchronize. The Wakeup message is an unreliable message sent with a Packet Type of 15, and a Payload Length of 2. The payload is composed of the octet pattern 0x05 0xFA. The Wakeup message shall be used after a device has sent a Sleep message. It is mandatory to respond to the Wakeup message.




*Figure 9.1: Wakeup Message Payload Format*

## 9.2 WOKEN MESSAGE

The Woken message shall be sent whenever a Wakeup message is received even if the receiver is currently not asleep. Upon receiving a Woken message, a device can determine that the other device is not in a low power state and can send and receive data. The Woken message is an unreliable message sent with a Packet Type of 15, and a Payload Length of 2. The payload is composed of the octet pattern 0x06 0xF9.



LifeScan, Inc. Ex. 1019
Page 1412 of 1420
IPR2025-00103

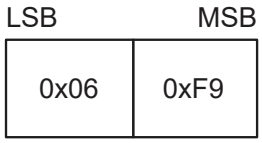



*Figure 9.2: Woken Message Payload Format*

## 9.3 SLEEP MESSAGE

A Sleep message can be sent at any time after Link Establishment has finished. It notifies the other side that this device is going into a low power state, and that it may also go to sleep. If a device sends a Sleep message it shall use the Wakeup / Woken message sequence before sending any data. If a device receives a Sleep message, then it should use the Wakeup / Woken message sequence before sending any data. The Sleep message is an unreliable message sent with a Packet Type of 15, and a Payload Length of 2. The payload is composed of the octet pattern 0x07 0x78.

The sending of this message is optional. The receiver of this message need not go to sleep, but cooperating devices may be able to schedule sleeping more effectively with this message.

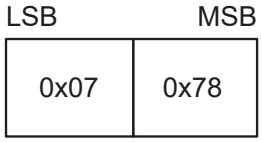



*Figure 9.3: Sleep Message Payload Format*



LifeScan, Inc. Ex. 1019
Page 1413 of 1420
IPR2025-00103

# 10 OUT OF FRAME CONTROL

It is possible to embed information in the SLIP data stream after a SLIP ESCAPE character that can allow for Software Flow Control. This feature is optional and must be negotiated in the Link Establishment configuration messages.

## 10.1 SOFTWARE FLOW CONTROL

If Software Flow Control is enabled, then the standard XON / XOFF (0x11 and 0x13) characters will control the flow of data over the transport. To allow the XON / XOFF characters to be sent in the payload, they shall be escaped as follows: 0x11 shall be changed to 0xDB 0xDE, 0x13 shall be changed to 0xDB DF. This means that the XON / XOFF characters in the data stream are used only by software flow control.

If Software Flow Control is disabled, then the SLIP escape sequences 0xDB 0xDE and 0xDB 0xDF are undefined. In this case, the original octets of 0x11 and 0x13 shall not be changed. Flow control should always be provided by the tunneled protocols, e.g. HCI Flow Control. Flow control is still available using the standard Sequence Number / Acknowledge Number. This can be done by not acknowledging packets until traffic can resume.



LifeScan, Inc. Ex. 1019
Page 1414 of 1420
IPR2025-00103

# 11 HARDWARE CONFIGURATION

The HCI Three-Wire UART Transport uses the following configurations.

## 11.1 WIRES

There are three wires used by the HCI Three-Wire UART Transport. These are Transmit, Receive, and Ground.

### 11.1.1 Transmit & Receive

The transmit line from one device shall be connected to the receive line of the other device.

### 11.1.2 Ground

A common ground reference shall be used.

## 11.2 HARDWARE FLOW

Hardware flow control may be used. The signaling shall be the same as a standard RS232 flow control lines. If used, the signals shall be connected in a null-modem fashion; for example, the local RTS shall be connected to the remote CTS and vice versa.

### 11.2.1 RTS & CTS

Request to Send indicates to the remote side that the local device is able to accept more data.

Clear to Send indicates if the remote side is able to receive data.

(See ITU.T recommendations V.24 [2] and V.28 [3])



LifeScan, Inc. Ex. 1019
Page 1415 of 1420
IPR2025-00103

# 12 RECOMMENDED PARAMETERS

## 12.1 TIMING PARAMETERS

Because this transport protocol can be used with a wide variety of baud rates, it is not possible to specify a single timing value. However, it is possible to specify the time based on the baud rate in use. If $T_{max}$ is defined as the maximum time in seconds it will take to transmit the largest packet over this transport, $T_{max}$ can be expressed as:

$T_{max}$ = maximum size of a packet in bits / baud rate

The maximum size of a packet in bits is either the number of bits in a 4095 octet packet (32,760) or less if required in an embedded system or as determined by the Host or Controller[1]. Thus, at a baud rate of 921,600 and the maximum packet size of 4095 octets, $T_{max}$ is: (4095*10) / 921,600 = 44.434ms.

### 12.1.1 Acknowledgement of Packets

It is not necessary to acknowledge every packet with a pure acknowledgement packet if there is a data packet that will be sent soon. The recommended maximum time before starting to send an acknowledgement is 2 * $T_{max}$.

### 12.1.2 Resending Reliable Packets

A reliable packet must be resent until it is acknowledged. The recommended time between starting to send the same packet is 3 * $T_{max}$.

1. This can be determined using the HCI_Read_Buffer_Size command.

# 13 REFERENCES

[1] [IETF RFC 1055: Nonstandard for transmission of IP datagrams over serial lines: SLIP – http://www.ietf.org/rfc/rfc1055.txt

[2] ITU Recommendation V.24: List of definitions for interchange circuits between data terminal equipment (DTE) and data circuit-terminating equipment (DCE) – http://www.itu.int/rec/recommendation.asp

[3] ITU Recommendation V.28: Electrical characteristics for unbalanced double-current interchange circuits – http://www.itu.int/rec/recommendation.asp



LifeScan, Inc. Ex. 1019
Page 1417 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1418 of 1420
IPR2025-00103

LifeScan, Inc. Ex. 1019
Page 1419 of 1420
IPR2025-00103



1420