Bluetooth®
SPECIAL INTEREST GROUP

SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Experience More

Bluetooth Core Specification Addendum 3 rev. 2

Adoption Date: 24 July 2012

Disclaimer and Copyright Notice

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

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

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

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

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

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

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

Addendum 3

This addendum provides an optional update to the Bluetooth® Core Specification. When the addendum is applied to an allowed Core Specification, the following parts of the specification shall be replaced, added, or appended with the revised versions:

Volume 1, Part D
Mixing of Specification Versions

Volume 2, Part E
Host Controller Interface

Coexistence Signaling (HCI)

Volume 3, Part C
Generic Access Profile

GAP Connection Parameters
GAP Authentication and Lost Bond
Private Addressing
Dual Mode Addressing

Volume 3, Part F
Attribute Protocol

Common Profile
Service Error Code Range Change

Volume 7, Part A

Coexistence Signaling (Logical)

Volume 7, Part B

Wireless Coexistence Interface (WCI-1) Transport Specification

Volume 7, Part C

Wireless Coexistence Interface (WCI-2) Transport Specification

Low Energy Errata

Revision 2 Explanation

Republished CSA3 on 19 September 2012 to add Disclaimer and Copyright Notice and correct incorporation of adopted erratum 4139.

TABLE OF CONTENTS

Specification Volume 7

Part A
MWS COEXISTENCE LOGICAL SIGNALING SPECIFICATION

Part B
WIRELESS COEXISTENCE INTERFACE 1 (WCI-1) TRANSPORT SPECIFICATION

Part C
WIRELESS COEXISTENCE INTERFACE 2 (WCI-2) TRANSPORT SPECIFICATION

MIXING OF SPECIFICATION VERSIONS

CONTENTS

1 MIXING OF SPECIFICATION VERSIONS

This part describes how volumes, and parts within volumes, of different versions of the Core Specification may be mixed in Bluetooth implementations. The Core System consists of a BR/EDR Controller Package (see Volume 2), a Low Energy Controller Package (see Volume 6), a Host Package (see Volume 3) and AMP Protocol Adaptation Layers (see Volume 5).

~~A Core Specification Addendum contains one or more parts of a single volume or one or more parts in multiple volumes, changes on one or more parts, or a combination of parts and changes. Addenda are used to supersede a part in a volume or may be used to add a part to a volume according to the rules in Section 1.2.~~

- All parts within a Primary Controller implementation shall be the same version of Volume 2 and Volume 6, and shall support at least one Type 1, ~~2, or 3 feature from~~ Type 2, or Type 3 feature introduced in that version of the Core Specification from Table 1.2.
- All parts within a Host implementation of Volume 3 shall be the same version and shall support at least one Type 3 or Type 4 feature introduced in that version of the Core Specification from Table 1.2.
- An AMP Controller implementation shall contain parts of Volume 2 and Volume 5 from the same version ~~and shall support at least one new Type 3 or Type 4 feature from Table 1.2~~.
- The Primary Controller, AMP Controller, and Host may be different versions within a single implementation.

In order to describe how these volumes and parts within volumes 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/enabled 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

Table 1.1: Feature type definitions

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 or AMP Protocol Adaptation Layer (PAL) 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 or AMP PAL of this version.
- In order to fully use the feature under all conditions, the Host Package, Controller Package, and AMP PAL must be of the same or later version.
- If an implementation contains features of type 3, these features can only function if the Host Package supports this version or a later version and if the Controller Package supports this version or a later version.

See the Bluetooth Brand Book for specification naming requirements.

1.1 FEATURES AND THEIR TYPES

The following table lists the features and their types.

Feature	Version	Type
Basic AFH operation	1.2	1
Enhanced inquiry	1.2	1
Configuration of AFH (setting channels and enabling/disabling channel assessment)	1.2	2
Enhanced synchronization capability	1.2	2
Interlaced inquiry scan	1.2	2
Interlaced page scan	1.2	2
Broadcast encryption	1.2	2
Enhanced flow specification and flush time-out	1.2	3
Extended SCO links	1.2	3
Inquiry Result with RSSI	1.2	3
L2CAP flow and error control	1.2	4
2 Mbps EDR	2.0 + EDR	2
3 Mbps EDR	2.0 + EDR	2
3 slot packets in EDR	2.0 + EDR	2
5 slot packets in EDR	2.0 + EDR	2
2 Mbps eSCO	2.0 + EDR	2^1
3 Mbps eSCO	2.0 + EDR	2^1
3 slot packets for EDR eSCO	2.0 + EDR	2^1
Erroneous Data Reporting	2.1 + EDR	3
Extended Inquiry Response	2.1 + EDR	3

Table 1.2: Features and their types

Feature	Version	Type
Encryption Pause and Resume	2.1 + EDR	1
Link Supervision Timeout Changed Event	2.1 + EDR	3
Non-Flushable Packet Boundary Flag	2.1 + EDR	3
Sniff subrating	2.1+ EDR	3
Secure Simple Pairing	2.1.+ EDR	3
L2CAP Enhanced Retransmission Mode	Addendum 1/ 3.0 + HS	4
L2CAP Streaming Mode	Addendum 1/ 3.0 + HS	4
Enhanced Power Control	3.0 + HS	2
AMP Manager Protocol (A2MP)	3.0 + HS	4
L2CAP Enhancements for AMP	3.0 + HS	4
802.11 PAL	3.0 + HS	3
Generic Test Methodology	3.0 + HS	3
Unicast Connectionless Data	3.0 + HS	4
Low Energy Controller (PHY and LL)~~(up through L2CAP)~~	4.0	3
Low Energy Host (L2CAP and Security Manager)	4.0	~~3~~4
Attribute Protocol and Generic Attribute Profile	4.0	4
~~Generic Attribute Profile~~	~~4.0~~	~~4~~
~~Security Manager~~	~~4.0~~	~~3~~
~~Audio Architecture – HCI Changes~~	~~Addendum 2~~	~~2~~
~~Audio Architecture – USB Changes~~	~~Addendum 2~~	~~2~~
~~Limited Discovery Time~~	~~Addendum 2~~	~~4~~
Appearance Data Type	Addendum 2	4
802.11n Enhancements to the 802.11 PAL	Addendum 2	3
MWS Coexistence Signaling	Addendum 3	2

Table 1.2: Features and their types

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

1.2 CORE SPECIFICATION ADDENDA

~~A Core Specification Addendum contains one or more complete Parts and/or Changes of existing or new Parts.~~

A Core Specification Addendum (CSA) contains one or more parts of a single volume, one or more parts in multiple volumes, changes on one or more parts, or a combination of parts and changes. Addenda are used to supersede a part in a volume or may be used to add a part to a volume according to the rules in Table 1.3.

~~Each part within an addendum is identified by a type indicating whether it may replace a part in an allowed Core Specification version or whether it may add to a package within a Core Specification version.~~

Note: Each Change may contain changes and/or additions to one or more parts of the Core Specification.

~~The following table lists adopted addenda and the Core Specification version they are allowed to be used with.~~

Addendum	Volume and Part or Change Name	~~Type~~Addition/ Changes/ Replacement	Allowed Versions	Mandatory / Optional / Conditional	Type
1	Volume 3, Part A	Replacement	2.0 + EDR, 2.1 + EDR	O	4
2	Audio Architecture HCI Changes	Change	2.1 + EDR, 3.0 + HS, 4.0	O	2
	Audio Architecture USB Changes	Change	2.1 + EDR, 3.0 + HS, 4.0	O	2
	LE Limited Discovery Time Changes	Change	4.0	C.1	4
	EIR and AD Data Types in GAP Changes	Change	4.0	C.1	4
	EIR and AD Data Types Specification	Addition	4.0	C.1	4
	Volume 5, Part A	Replacement	3.0 + HS, 4.0	O	3

Table 1.3: Adopted core specification versions to use with addenda

Addendum	Volume and Part or Change Name	~~Type~~Addition/ Changes/ Replacement	Allowed Versions	Mandatory / Optional / Conditional	Type
3	LE Errata	Change	4.0 with CSA2	C.3	Multiple
	GAP Connection Parameters Changes	Change	4.0 with CSA2	C.2	4
	GAP Authentication and Lost Bond Changes	Change	4.0 with CSA2	C.2	4
	Common Profile and Services Error Code Range Changes	Change	4.0 with CSA2	C.2	4
	Private Addressing Changes	Change	4.0 with CSA2	C.2	4
	Dual Mode Addressing Changes	Change	4.0 with CSA2	C.4	4
	MWS Coexistence Logical Signaling Specification	Addition	2.1 + EDR, 3.0 + HS, 4.0 with CSA2	O	2
	MWS Coexistence HCI	Addition	2.1 + EDR, 3.0 + HS, 4.0 with CSA2	C.5	2
	Wireless Coexistence Interface 1 (WCI-1) Transport Layer Specification	Addition	2.1 + EDR, 3.0 + HS, 4.0 with CSA2	C.5	2
	Wireless Coexistence Interface 2 (WCI-2) Transport Layer Specification	Addition	2.1 + EDR, 3.0 + HS, 4.0 with CSA2	C.5	2

Table 1.3: Adopted core specification versions to use with addenda

C.1: Mandatory if either the Host Part of the Low Energy Core Configuration or the Host Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.2: Mandatory if either the Host Part of the Low Energy Core Configuration or the Host Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.3: Mandatory if either the Host Part of the Low Energy Core Configuration, Controller Part of the Low Energy Core Configuration, Host Part of the Basic Rate and Low Energy Combined Core Configuration, or Controller Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.4: Mandatory if the Host Part of the Basic Rate and Low Energy Combined Core Configuration is supported, otherwise Excluded.

C.5: Optional if MWS Coexistence Logical Signaling is supported, otherwise Excluded.

MWS COEXISTENCE HCI CHANGES

CONTENTS

1 CHANGE INSTANCES

1.1 CHANGE #1 - VOLUME 2, PART E (HCI), SECTION 3

[Insert the following rows to Table 3.5 at the end]

Name	Ver	Summary Description	Supported Controllers
Set MWS Channel Parameters Command	CSA3	The Set MWS Channel Parameters command enables an MWS device to inform the Controller about the MWS channel configuration.	BR/EDR, LE and AMP
Set External Frame Configuration Command	CSA3	The Set External Frame Configuration command enables an external device to describe a frame structure to the Controller.	BR/EDR, LE and AMP
Set MWS Signaling Command	CSA3	The Set MWS Signaling command enables a MWS device to inform the Controller about the timing parameters for the MWS coexistence interface.	BR/EDR, LE and AMP
Set Transport Layer Command	CSA3	The Set Transport Layer command selects the MWS coexistence signaling transport layer in the Controller.	BR/EDR, LE and AMP
Get Transport Layer Configuration Command	CSA3	The Get Transport Layer Configuration command reads the supported baud rates from the Controller.	BR/EDR, LE and AMP
Set MWS Scan Frequency Table Command	CSA3	The Set MWS Scan Frequency Table Command specifies the frequencies represented by the frequency index supplied by the MWS_SCAN_FREQUENCY signal.	BR/EDR, LE and AMP
Set MWS PATTERN Configuration Command	CSA3	The Set MWS PATTERN Command specifies the configuration of the pattern indicated over the MWS Coexistence Transport Layer.	BR/EDR, LE and AMP

[Add the following row to Table 3.19 after the “Encryption Key Refresh Complete Event” row.]

Commands / Events	Group
Set MWS Channel Parameters Command	Controller Configuration
Set External Frame Configuration Command	Controller Configuration
Set MWS Signaling Command	Controller Configuration
Set Transport Layer Command	Controller Configuration
Get Transport Layer Configuration Command	Controller Configuration

Commands / Events	Group
Set MWS Scan Frequency Table Command	Controller Configuration
Set MWS PATTERN Command	Controller Configuration

[Add the following rows to the table in Section 6.27 in order]

Octet	Bit	Command Supported
29	6	Set MWS Channel Parameters Command
29	7	Set External Frame Configuration Command
30	0	Set MWS Signaling Command
30	1	Set Transport Layer Command
30	2	Set MWS Scan Frequency Table Command
30	3	Get Transport Layer Configuration Command
30	4	Set MWS PATTERN Configuration Command

1.2 CHANGE #2 - VOLUME 2, PART E (HCI), SECTION 7

7.3.80 Set MWS Channel Parameters Command

Command	OCF	Command Parameters	Return Parameters
HCI_ Set_MWS_Channel_ Parameters	0x006E	MWS_Channel_Enable MWS_RX_Center_Frequency, MWS_TX_Center_Frequency, MWS_RX_Channel_Bandwidth, MWS_TX_Channel_Bandwidth, MWS_Channel_Type	Status

Description:

The Set_MWS_Channel_Parameters command is used to inform the Controller of the MWS channel parameters.

The MWS_Channel_Enable parameter is used to enable or disable the MWS channel.

The MWS_RX_Center_Frequency and MWS_TX_Center_Frequency parameters are used to indicate the center frequency of the MWS device's uplink (TX) and downlink (RX) channels. Note that the uplink and downlink channel centers may be the same value or different values.

The MWS_RX_Channel_Bandwidth and MWS_TX_Channel_Bandwidth parameters are used to indicate the bandwidth, in kHz, of the MWS device's uplink and downlink channels.

The MWS_Channel_Type parameter describes the type of channel. The types are defined in the Bluetooth Assigned Numbers.

Command Parameters:

MWS_Channel_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	MWS channel is disabled.
0x01	MWS channel is enabled.
0x02 – 0xFF	Reserved.

MWS_RX_Center_Frequency: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS RX center frequency in MHz.

MWS_TX_Center_Frequency: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS TX center frequency in MHz

MWS_RX_Channel_Bandwidth: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS RX channel bandwidth in kHz.

MWS_TX_Channel_Bandwidth: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS TX channel bandwidth in kHz.

MWS_Channel_Type: *Size: 1 Octet*

Value	Parameter Description
0xXX	See Bluetooth Assigned numbers.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_MWS_Channel_Parameters command succeeded.
0x01-0xFF	Set_MWS_Channel_Parameters command failed. See Part D, Error Codes, for error codes and descriptions

Event(s) generated (unless masked away):

When the Set_MWS_Channel_Parameters command has completed, a Command Complete event shall be generated.

7.3.81 Set External Frame Configuration Command

Command	OCF	Command Parameters	Return Parameters
HCI_ Set_ External_Frame _Configuration	0x006F	Ext_Frame_Duration, Ext_Frame_Sync_Assert_Offset, Ext_Frame_Sync_Assert_Jitter, Ext_Num_Periods, Period_Duration[i], Period_Type[i]	Status

Description:

The Set_External_Frame_Configuration command allows the Host to specify a frame configuration for an external collocated system. This frame configuration persists until overwritten with a subsequent Set_External_Frame_Configuration or until the Primary Controller is reset.

The start of the external frame structure is defined as an offset from an external frame synchronization signal. This offset is defined by the Ext_Frame_Sync_Assert_Offset parameter. The offset is represented as the time (in microseconds) from the start of the next MWS frame to the FRAME_SYNC signal.

An external frame consists of downlink periods, uplink periods and guard periods. Downlink means the collocated MWS system is receiving, thus may be interfered with by Bluetooth transmissions. Uplink means the collocated MWS system is transmitting, thus may cause interference to Bluetooth receptions. A guard period may be used by the MWS system to compensate for propagation delays; in this case it should be regarded as split equally between downlink and uplink durations.

The number of specified periods is given by Ext_Num_Periods.

The duration in microseconds of each period is defined by the Period_Duration[i] parameters.

The Period_Type[i] parameter indicates if the specified period is an uplink, downlink, bi-directional or guard period.

The sum of all Period_Duration[i] parameters shall be equal to the Ext_Frame_Duration parameter.

Command Parameters:

Ext_Frame_Duration: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	External frame duration in microseconds (unsigned integer)

Ext_Frame_Sync_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	External frame offset in microseconds (signed integer)

Ext_Frame_Sync_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	External frame sync jitter in microseconds (unsigned integer)

Ext_Num_Periods: *Size: 1 Octet*

Value	Parameter Description
N	Number of specified periods in an external frame. Valid range: 1 to 32 (unsigned integer)

Period_Duration[i]: *Size: Ext_Num_Periods * 2 Octets*

Value	Parameter Description
0xXXXX	Duration of the [i] period in microseconds (unsigned integer)

Period_Type[i]: *Size: Ext_Num_Periods * 1 Octet*

Value	Parameter Description
0x00	Downlink
0x01	Uplink
0x02	Bi-Directional
0x03	Guard Period
0x04-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_External_Frame_Configuration command succeeded.
0x01-0xFF	Set_External_Frame_Configuration command failed. See Part D, Error Codes, for error codes and descriptions

Event(s) generated (unless masked away):

When the Set_External_Frame_Configuration command has completed, a Command Complete event shall be generated.

7.3.82 Set MWS Signaling Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_MWS_Signaling	0x0070	MWS_RX_Assert_Offset, MWS_RX_Assert_Jitter, MWS_RX_Deassert_Offset, MWS_RX_Deassert_Jitter, MWS_TX_Assert_Offset, MWS_TX_Assert_Jitter, MWS_TX_Deassert_Offset, MWS_TX_Deassert_Jitter, MWS_Pattern_Assert_Offset, MWS_Pattern_Assert_Jitter, MWS_Inactivity_Duration_Assert_Offset, MWS_Inactivity_Duration_Assert_Jitter, MWS_Scan_Frequency_Assert_Offset, MWS_Scan_Frequency_Assert_Jitter, MWS_Priority_Assert_Offset_Request	Status, Bluetooth_Rx_Priority_Assert_Offset, Bluetooth_Rx_Priority_Assert_Jitter, Bluetooth_Rx_Priority_Deassert_Offset, Bluetooth_Rx_Priority_Deassert_Jitter, 802_Rx_Priority_Assert_Offset, 802_Rx_Priority_Assert_Jitter, 802_Rx_Priority_Deassert_Offset, 802_Rx_Priority_Deassert_Jitter, Bluetooth_Tx_On_Assert_Offset, Bluetooth_Tx_On_Assert_Jitter,Bluetooth_Tx_On_Deassert_Offset, Bluetooth_Tx_On_Deassert_Jitter, 802_Tx_On_Assert_Offset, 802_Tx_On_Assert_Jitter, 802_Tx_On_Deassert_Offset, 802_Tx_On_Deassert_Jitter

Description:

The Set_MWS_Signaling command is used to inform the Bluetooth Controller of the MWS signaling interface logical layer parameters.

All signals are defined in Volume 7, Part A.

Command Parameters:

MWS_RX_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_RX signal assert offset in microseconds (signed integer).

MWS_RX_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_RX signal assert jitter in microseconds (unsigned integer).

MWS_RX_Deassert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_RX signal de-assert offset in microseconds (signed integer).

MWS_RX_Deassert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_RX signal de-assert jitter in microseconds (unsigned integer).

MWS_TX_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_TX signal assert offset in microseconds (signed integer).

MWS_TX_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_TX signal assert jitter in microseconds (unsigned integer).

MWS_TX_Deassert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_TX signal de-assert offset in microseconds (signed integer).

MWS_TX_Deassert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_TX signal de-assert jitter in microseconds (unsigned integer).

MWS_Pattern_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_PATTERN signal assert offset in microseconds (signed integer).

MWS_Pattern_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_PATTERN signal assert jitter in microseconds (unsigned integer).

MWS_Inactivity_Duration_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_INACTIVITY_DURATION signal assert offset in microseconds (signed integer).

MWS_Inactivity_Duration_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_INACTIVITY_DURATION signal assert jitter in microseconds (unsigned integer).

MWS_Scan_Frequency_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_SCAN_FREQUENCY signal assert offset in microseconds (signed integer).

MWS_Scan_Frequency_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	MWS_SCAN_FREQUENCY signal assert jitter in microseconds (unsigned integer).

MWS_Priority_Assert_Offset_Request: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Minimum advance notification from the beginning of an MWS Uplink period in microseconds (unsigned integer) before which the BLUETOOTH_RX_PRI or 802_RX_PRI signal shall be asserted to be recognized by the MWS.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_MWS_Signaling command succeeded.
0x01-0xFF	Set_MWS_Signaling command failed. See Part D, Error Codes, for error codes and descriptions

Bluetooth_RX_Priority_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_RX_PRI signal assert offset in microseconds (signed integer).

Bluetooth_RX_Priority_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_RX_PRI signal assert jitter in microseconds (unsigned integer).

Bluetooth_RX_Priority_Deassert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_RX_PRI signal de-assert offset in microseconds (signed integer).

Bluetooth_RX_Priority_Deassert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_RX_PRI signal de-assert jitter in microseconds (unsigned integer).

802_RX_Priority_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_RX_PRI signal assert offset in microseconds (signed integer).

802_RX_Priority_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_RX_PRI signal assert jitter in microseconds (unsigned integer).

802_RX_Priority_Deassert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_RX_PRI signal de-assert offset in microseconds (signed integer).

802_RX_Priority_Deassert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_RX_PRI signal de-assert jitter in microseconds (unsigned integer).

Bluetooth_TX_On_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_TX_ON signal assert offset in microseconds (signed integer).

Bluetooth_TX_On_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_TX_ON signal assert jitter in microseconds (unsigned integer).

Bluetooth_TX_On_Deassert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_TX_ON signal de-assert offset in microseconds (signed integer).

Bluetooth_TX_On_Deassert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	BLUETOOTH_TX_ON signal de-assert jitter in microseconds (unsigned integer).

802_TX_On_Assert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_TX_ON signal assert offset in microseconds (signed integer).

802_TX_On_Assert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_TX_ON signal assert jitter in microseconds (unsigned integer).

802_TX_On_Deassert_Offset: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_TX_ON signal de-assert offset in microseconds (signed integer).

802_TX_On_Deassert_Jitter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	802_TX_ON signal de-assert jitter in microseconds (unsigned integer).

Event(s) generated (unless masked away):

When the Set_MWS_Signaling command has completed, a Command Complete event shall be generated.

7.3.83 Set MWS Transport Layer Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_MWS_ Transport_Layer	0x0071	Transport_Layer, To_MWS_Baud_Rate, From_MWS_Baud_Rate	Status

Description:

The Set_MWS_Transport_Layer command configures the transport layer between the Bluetooth Controller and MWS device.

Command Parameters:

Transport_Layer: *Size: 1 Octet*

Value	Parameter Description
0xXX	See Assigned Numbers.

To_MWS_Baud_Rate: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Baud rate in the Bluetooth to MWS direction in Baud.

From_MWS_Baud_Rate: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Baud rate in the MWS to Bluetooth direction in Baud.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_MWS_Transport_Layer command succeeded.
0x01-0xFF	Set_MWS_Transport_Layer command failed. See Part D, Error Codes, for error codes and descriptions

Event(s) generated (unless masked away):

When the Set_MWS_Transport_Layer command has completed, a Command Complete event shall be generated.

7.3.84 Set MWS Scan Frequency Table Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_MWS_Scan_Frequency_Table	0x0072	Num_Scan_Frequencies, Scan_Frequency_Low[i], Scan_Frequency_High[i]	Status

Description:

The Set_MWS_Scan_Frequency_Table command configures the MWS scan frequency table in the Controller.

The Num_Scan_Frequencies parameter indicates the number of MWS scan frequencies to be set. A Controller shall support at least 8 table entries.

The Scan_Frequency_Low[i] and Scan_Frequency_High[i] parameters indicate the lower and upper edges for each Scan Frequency.

Command Parameters:

Num_Scan_Frequencies: *Size: 1 Octet*

Value	Parameter Description
N	Number of MWS scan frequencies to be set in the table.

Scan_Frequency_Low[i]: *Size: Num_Scan_Frequencies * 2 Octets*

Value	Parameter Description
0xXXXX, ..., 0xXXXX	Lower edge of the MWS scan frequency in MHz (unsigned integer).

Scan_Frequency_High[i]: *Size: Num_Scan_Frequencies * 2 Octets*

Value	Parameter Description
0xXXXX, ..., 0xXXXX	Upper edge of the MWS scan frequency in MHz (unsigned integer).

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_MWS_Scan_Frequency_Table command succeeded.
0x01-0xFF	Set_MWS_Scan_Frequency_Table command failed. See Part D, Error Codes, for error codes and descriptions

Event(s) generated (unless masked away):

When the Set_MWS_Scan_Frequency_Table command has completed, a Command Complete event shall be generated.

7.3.85 HCI Set MWS_PATTERN Configuration Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_MWS_PATTERN_Configuration	0x0073	MWS_PATTERN_Index, MWS_PATTERN_NumIntervals, MWS_PATTERN_IntervalDuration[i], MWS_PATTERN_IntervalType[i]	Status

Description:

The Set_MWS_PATTERN_Configuration command is used by the Host to specify, in conjunction with the Set_External_Frame_Configuration command, local MWS_PATTERN parameters for an external collocated system.

An MWS_PATTERN configuration shall persist until overwritten by a subsequent Set_MWS_PATTERN_Configuration or Set_External_Frame_Configuration command, or until the Primary Controller is reset.

Command Parameters:

MWS_PATTERN_Index: *Size: 1 Octet*

Value	Parameter Description
0xXX	Index of the MWS_PATTERN instance to be configured. Range is 0-2.

MWS_PATTERN_NumIntervals: *Size: 1 Octet*

Value	Parameter Description
0xXX	The number of intervals in the following array.

MWS_PATTERN_IntervalDuration[i]: *Size: MWS_PATTERN_NumInterval * 2 Octets*

Value	Parameter Description
0xXXXX, ..., 0xXXXX	An array specifying the duration of Bluetooth activity intervals in microseconds.

MWS_PATTERN_IntervalType[i]: *Size: MWS_PATTERN_NumInterval * 1 Octet*

Value	Parameter Description
0xXX, ...0xXX	An array specifying the types of permitted Bluetooth activities in each interval: 0. Neither transmission nor reception is allowed. 1. Transmission is allowed. 2. Reception is allowed. 3. Both transmission and reception are allowed. 4. Interval for the MWS frame as defined by the Set_External_Frame_Configuration command.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_MWS_PATTERN_Configuration command succeeded
0x01-0xFF	Set_MWS_PATTERN_Configuration 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_MWS_PATTERN_Configuration command has completed, a Command Complete event shall be generated.

7.5.11 Get MWS Transport Layer Configuration Command

Command	OCF	Command Parameters	Return Parameters
HCI_Get_MWS_Transport Layer_Configuration	0x000C		Status, Num_Transports, Transport_Layer[i], Num_Baud_Rates[i], To_MWS_Baud_Rate[k], From_MWS_Baud_Rate[k]

Description:

The Get_MWS_Transport_Layer_Configuration command is used to inform the Host of the Baud rates supported by the Controller for the transport layer.

The Num_Transports parameter is used to indicate the number of MWS coexistence transport interfaces supported by the Controller. The Num_Baud_Rates parameter indicates the number of supported baud rates for each transport.

The To_MWS_Baud_Rate parameters indicate the supported baud rates in the direction from Bluetooth to MWS for each transport.

The From_MWS_Baud_Rate parameters indicate the supported baud rates in the direction from MWS to Bluetooth for each transport.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Get_Transport_Layer_Configuration command succeeded.
0x01-0xFF	Get_Transport_Layer_Configuration command failed. See Part D, Error Codes, for error codes and descriptions

Num_Transports: *Size: 1 Octet*

Value	Parameter Description
0xXX	Number of supported MWS coexistence transport layers.

Transport_Layer[i]: *Size: Num_ Transports * 1 Octet*

Value	Parameter Description
0xXX, ..., 0xXX	See Assigned Numbers.

Num_Baud_Rates[i]: *Size: Num_Transports * 1 Octet*

Value	Parameter Description
0xXX, ..., 0xXX	Number of different baud rates supported for one transport.

To_MWS_Baud_Rate[k]: *Size: SUM(Num_Baud_Rates [i])* 4 Octets*

Value	Parameter Description
0xXXXXXXXX, 0xXXXXXXXX, ... 0xXXXXXXXX	List of supported Baud rates in the Bluetooth Controller to MWS Device direction in Baud. Each element of the list shall have the format 0xXXXXXXXX. The list shall start with the first baud rate for the first transport, followed by the remaining baud rates for the first transport, followed by the baud rates for the second transport (if any), followed by baud rates for subsequent transports (if any).

From_MWS_Baud_Rate[k]: *Size: SUM (Num_Baud_Rates[i]) * 4 Octets*

Value	Parameter Description
0xXXXXXXXX, 0xXXXXXXXX, ... 0xXXXXXXXX	List of supported Baud rates in the Bluetooth Controller for signals in the MWS to Bluetooth Controller Device direction in Baud. Each element of the list shall have the format 0xXXXXXXXX. The list shall start with the first baud rate for the first transport, followed by the remaining baud rates for the first transport, followed by the baud rates for the second transport (if any), followed by baud rates for subsequent transports (if any).

Event(s) generated (unless masked away):

When the Get_MWS_Transport Layer_Configuration command has completed, a Command Complete event shall be generated.

GAP CONNECTION PARAMETERS CHANGES

CONTENTS

1 CHANGE INSTANCES

1.1 CHANGE #1 – VOLUME 3, PART C (GAP), SECTION 9.2.3.2, LIMITED DISCOVERABLE MODE

While a device is in limited discoverable mode the Host configures the Controller as follows:

- The Host shall set the advertising filter policy to ‘process scan and connection requests from all devices’ .
- ~~The Host should set the minimum advertising interval to TGAP(lim_disc_adv_int_min)~~
- ~~The Host should set the maximum advertising interval to TGAP(lim_disc_adv_int_max).~~
- The Host should set the advertising intervals as defined in Section 9.3.11.

1.2 CHANGE #2 – VOLUME 3, PART C (GAP), SECTION 9.2.4.2 GENERAL DISCOVERABLE MODE

While a device is in general discoverable mode the Host configures the Controller as follows:

- The Host shall set the advertising filter policy to ‘process scan and connection requests from all devices’.
- ~~The Host should set the minimum advertising interval to TGAP(gen_disc_adv_int_min)~~
- ~~The Host should set the maximum advertising interval to TGAP(gen_disc_adv_int_max).~~
- The Host should set the advertising intervals as defined in Section 9.3.11.

1.3 CHANGE #3 – VOLUME 3, PART C (GAP), SECTION 9.2.5.2, LIMITED DISCOVERY PROCEDURE

When a Host performs the limited discovery procedure, the Host configures the Controller as follows:

1. The Host shall set the scanner filter policy to ‘process all advertising packets’.

~~2. The Host should set the scan interval of T_{GAP}(lim_disc_scan_int)~~

~~3. The Host should set the scan window of T_{GAP}(lim_disc_scan_wind)~~

2. The Host should set the scan interval and scan window as defined in Section 9.3.11.

1.4 CHANGE #4 – VOLUME 3, PART C (GAP), SECTION 9.2.6.2, GENERAL DISCOVERY PROCEDURE

When a Host performs the general discovery procedure, the Host configures the Controller as follows:

1. The Host shall set the scanner filter policy to 'process all advertising packets'.

~~2.The Host should set the scan interval of T_{GAP}(gen_disc_scan_int)~~

~~3. The Host should set the scan window of T_{GAP}(gen_disc_scan_wind)~~

2. The Host should configure the Controller to use active scanning.
3. The Host should set the scan interval and scan window as defined in Section 9.3.11.

1.5 CHANGE #5 – VOLUME 3, PART C (GAP), SECTION 9.3.3.2, DIRECTED CONNECTABLE MODE

6. ~~The Host should configure the Controller with the minimum advertising interval set to T_{GAP}(dir_conn_adv_int_min) and the maximum advertising interval set to T_{GAP}(dir_conn_adv_int_max).~~ The Host shall configure the Controller to send directed connectable advertising events.

1.6 CHANGE #6 – VOLUME 3, PART C (GAP), SECTION 9.3.4.2, UNDIRECTED CONNECTABLE MODE

3. The Host shall generate a resolvable private address using the 'resolvable private address generation procedure' as defined in Section 10.8.2.2. The Host shall set a timer equal to or greater than T_{GAP}(private_addr_int). The Host shall configure the Controller to use the resolvable private address as the advertiser's device address in the connectable undirected advertising events. The Host should set the advertising filter policy to either 'process scan and connection requests only from devices in the White List' or 'process scan and connection requests from all devices'. The Host shall configure the Controller to send undirected connectable advertising events ~~with the minimum advertising interval set to T_{GAP}(undir_conn_adv_int_min) and the maximum advertising interval set to T_{GAP}(undir_conn_adv_int_max)~~as defined in Section 9.3.11. If privacy is disabled then the Host shall disable the T_{GAP}(private_addr_int) timer and proceed to step 4. When the T_{GAP}(private_addr_int) timer expires, the Host shall configure the Controller to stop advertising and repeat step 3.
4. The Host uses the static or the public address of the local device as the advertiser's device address. The Host should configure the Controller ~~with the minimum advertising interval set to T_{GAP}(undir_conn_adv_int_min) and the maximum advertising interval set to T_{GAP}(undir_conn_adv_int_max)~~as

defined in Section 9.3.11. The Host shall configure the Controller to send undirected connectable advertising events.

1.7 CHANGE #7 - VOLUME 3, PART C (GAP), SECTION 9.3.5.2, AUTO CONNECTION ESTABLISHMENT PROCEDURE

1. The Host shall write the list of device addresses that are to be auto connected to into the White List.
2. The Host shall set the initiator filter policy to 'process connectable advertising packets from all devices in the White List'.
3. ~~The Host should set the minimum connection interval to be T_{GAP}(conn_est_int_min).~~
4 ~~The Host should set the maximum connection interval to be T_{GAP}(conn_est_int_max)~~
5. ~~The Host should set the slave latency to be T_{GAP}(conn_est_latency)~~
3. The Host should set the scan interval ~~of T_{GAP}(auto_conn_est_scan_int) and the scan window of T_{GAP}(auto_conn_est_scan_wind)~~and scan window as defined in Section 9.3.11.
4. The Host shall initiate a connection. For a privacy enabled device as defined in Section 10.7, the Host shall use a non-resolvable private address.
5. The Host should set connection parameters as defined in Section 9.3.12.

1.8 CHANGE #8 – VOLUME 3, PART C (GAP), SECTION 9.3.6.2, GENERAL CONNECTION ESTABLISHMENT PROCEDURE

Host configures the Controller as follows:

1. The Host shall set the scanner filter policy to 'process all advertising packets'.
2. The Host should set the scan interval ~~to T_{GAP}(gen_conn_est_scan_int)~~as defined in Section 9.3.11.
3. The Host should set the scan window t~~o T_{GAP}(gen_conn_est_scan_wind)~~as defined in Section 9.3.11.
4. The Host shall start scanning. For a privacy enabled device as defined in Section 10.7, the Host shall use a non-resolvable private address if the Host is in active scanning.
5. The Host should set connection parameters as defined in Section 9.3.12.

1.9 CHANGE #9 – VOLUME 3, PART C (GAP), SECTION 9.3.7.2, SELECTIVE CONNECTION ESTABLISHMENT PROCEDURE

When a Host performs the selective connection establishment procedure, the Host configures the Controller as follows:

1. The Host shall write the list of device addresses that are to be selectively connected to into the White List.
2. The Host shall set the scanner filter policy to 'process advertising packets only from devices in the White List'.
3. The Host should set the scan interval ~~to T_{GAP}(sel_conn_est_scan_int)~~as defined in Section 9.3.11.
4. The Host should set the scan window ~~to T_{GAP}(sel_conn_est_scan_wind)~~as defined in Section 9.3.11.
5. The Host shall start active scanning or passive scanning.

1.10 CHANGE #10 – VOLUME 3, PART C (GAP), SECTION 9.3.8.2, DIRECTED CONNECTION ESTABLISHMENT PROCEDURE

1. The Host shall set the initiator filter policy to 'ignore the White List and process connectable advertising packets from a specific single device specified by the Host'.
2. The Host shall set the peer address to the device address of the specific single device.
3. ~~The Host shall set the connection configuration parameters for the peer device.~~The Host should set connection parameters as defined in Section 9.3.12.

1.11 CHANGE #11 – VOLUME 3, PART C (GAP), SECTION 9.3.11 (NEW SECTION), CONNECTION ESTABLISHMENT TIMING PARAMETERS

[Add the following section into GAP, Volume 3, Part C, Section 9.3.11.]

9.3.11 Connection Establishment Timing Parameters

9.3.11.1 Description

The connection establishment timing parameters are used during initial connection establishment between a Central and a Peripheral device.

A Central device should use one of the GAP connection establishment procedures to initiate a connection to a Peripheral device in a connectable mode. The procedures and modes that may use these timing parameters are defined in Section 9.3.4 through Section 9.3.8.

9.3.11.2 Conditions

A Central device starting a GAP connection establishment procedure should use the recommended scan interval T_{GAP}(scan_fast_interval) and scan window T_{GAP}(scan_fast_window) for T_{GAP}(scan_fast_period).

A Central device that is background scanning should use the recommended scan interval T_{GAP}(scan_slow_interval1) and scan window T_{GAP}(scan_slow_window1). Alternatively the central may use T_{GAP}(scan_slow_interval2) and scan window T_{GAP}(scan_slow_window2).

A Peripheral device starting a GAP connection establishment mode should use the recommended advertisement interval T_{GAP}(adv_fast_interval) for T_{GAP}(adv_fast_period).

A Peripheral device that is background advertising should use the recommended advertisement interval T_{GAP}(adv_slow_interval).

1.12 CHANGE #12 – VOLUME 3, PART C (GAP), SECTION 9.3.12 (NEW SECTION), CONNECTION INTERVAL TIMING PARAMETERS

9.3.12 Connection Interval Timing Parameters

9.3.12.1 Description

The connection interval timing parameters are used within a connection. Initial connection interval is used to ensure procedures such as bonding, encryption setup and service discovery are completed quickly. The connection interval should be changed to the Peripheral Preferred Connection Parameters after initiating procedures are complete.

9.3.12.2 Conditions

The Central device should either read the Peripheral Preferred Connection Parameters characteristic (see Section 12.5) or retrieve the parameters from advertising data (see Section 12.5).

The connection interval should be set to T_{GAP}(initial_conn_interval) and slave latency should be set to zero. These parameters should be used until the Central device has no further pending actions to perform or until the Peripheral device performs a Connection Parameter Update procedure (see Section 9.3.9).

After the Central device has no further pending actions to perform and the Peripheral device has not initiated any other actions within T_{GAP}(conn_pause_central), then the Central device should invoke the Connection Parameter Update procedure (see Section 9.3.9) and change the connection intervals as defined in the Peripheral Preferred Connection Parameters.

If the Central device does not have the Peripheral Preferred Connection Parameters, then the Central device may choose the connection parameters to be used.

After the Peripheral device has no further pending actions to perform and the Central device has not initiated any other actions within T_{GAP}(conn_pause_central), then the Peripheral device may perform a Connection Parameter Update procedure (see Section 9.3.9). The Peripheral device should not perform a Connection Parameter Update procedure within T_{GAP}(conn_pause_peripheral) after establishing a connection.

At any time a key refresh or encryption setup procedure is required and the current connection interval is greater than T_{GAP}(initial_conn_interval), the key refresh or encryption setup procedure should be preceded with a Connection Parameter Update procedure (see Section 9.3.9). The connection interval should be set to T_{GAP}(initial_conn_interval) and slave latency should be set to zero. This fast connection interval should be maintained until the key refresh or encryption setup procedure is complete. It should then switch to the Peripheral Preferred Connection Parameters.

1.13 CHANGE #13 – VOLUME 3, PART C (GAP), APPENDIX A (NORMATIVE) TIMERS AND CONSTANTS

The following timers are required by this profile.

Timer name	Value	Description	Requirement or Recommendation
T_{GAP}(lim_adv_timeout)	180 s	Maximum time to remain advertising when in the limited discoverable mode	Required value
T_{GAP}(gen_disc_scan_min)	10.24 s	Minimum time to perform scanning when performing the general discovery procedure	Recommended value
~~T_{GAP}(gen_disc_scan_int)~~	~~11.25 ms~~	~~Scan interval used in the general discovery procedure~~	~~Recommended value~~
~~T_{GAP}(gen_disc_scan_wind)~~	~~11.25 ms~~	~~Scan window used in the general discovery procedure~~	~~Recommended value~~
T_{GAP}(lim_disc_scan_min)	10.24 s	Minimum time to perform scanning when performing the limited discovery procedure	Recommended value

Table 1.1: Defined GAP timers

Timer name	Value	Description	Requirement or Recommendation
T_{GAP}(lim_disc_scan_int)	11.25 ms	Scan interval used in the limited discovery procedure	Recommended value
~~T_{GAP}(lim_disc_scan_wind)~~	~~11.25 ms~~	~~Scan window used in the limited discovery procedure~~	~~Recommended value~~
T_{GAP}(conn_param_timeout)	30 s	Connection parameter update notification timer when performing the connection parameter update procedure	Recommended value
T_{GAP}(scan_fast_period)	30.72 s	Minimum time to perform scanning when user initiated	Recommended value
T_{GAP}(scan_fast_interval)	30 ms to 60 ms	Scan interval in any discovery or connection establishment procedure when user initiated	Recommended value
T_{GAP}(scan_fast_window)	30 ms	Scan window in any discovery or connection establishment procedure when user initiated	Recommended value
T_{GAP}(scan_slow_interval1)	1.28 s	Scan interval in any discovery or connection establishment procedure when background scanning	Recommended value
T_{GAP}(scan_slow_window1)	11.25 ms	Scan window in any discovery or connection establishment procedure when background scanning	Recommended value
T_{GAP}(scan_slow_interval2)	2.56 s	Scan interval in any discovery or connection establishment procedure when background scanning	Recommended value
T_{GAP}(scan_slow_window2)	22.5 ms	Scan window in any discovery or connection establishment procedure when background scanning	Recommended value
T_{GAP}(adv_fast_period)	30 s	Minimum time to perform advertising when user initiated	Recommended value

Table 1.1: Defined GAP timers

Timer name	Value	Description	Requirement or Recommendation
T_{GAP}(adv_fast_interval)	30 ms to 60 ms	Minimum to maximum advertisement interval in any discoverable or connectable mode when user initiated	Recommended value
T_{GAP}(adv_slow_interval)	1 s to 1.2 s	Minimum to maximum advertisement interval in any discoverable or connectable mode when background advertising	Recommended value
T_{GAP}(initial_conn_interval)	30 ms to 50 ms	Minimum to maximum connection interval upon any connection establishment	Recommended value
T_{GAP}(conn_pause_central)	1 s	Central idle timer	Recommended value
T_{GAP}(conn_pause_peripheral)	5 s	Minimum time upon connection establishment before the peripheral starts a connection update procedure	Recommended value
~~T_{GAP}(conn_est_latency)~~	~~0~~	~~Connection slave latency in any connection establishment procedure~~	~~Recommended value~~
T_{GAP}(private_addr_int)	15 mins	Minimum time interval between private address change	Recommended value
~~T_{GAP}(lim_disc_adv_int_min)~~	~~250 ms~~	~~Minimum advertising interval when in the limited discoverable mode~~	~~Recommended value~~
~~T_{GAP}(lim_disc_adv_int_max)~~	~~500 ms~~	~~Maximum advertising interval when in the limited discoverable mode~~	~~Recommended value~~
~~T_{GAP}(gen_disc_adv_int_min)~~	~~1.28 s~~	~~Minimum advertising interval when in the general discoverable mode~~	~~Recommended value~~
~~T_{GAP}(gen_disc_adv_int_max)~~	~~2.56 s~~	~~Maximum advertising interval when in the general discoverable mode~~	~~Recommended value~~
~~T_{GAP}(dir_conn_adv_int_min)~~	~~250 ms~~	~~Minimum advertising interval when in the directed connectable mode~~	~~Recommended value~~

Table 1.1: Defined GAP timers

Timer name	Value	Description	Requirement or Recommendation
~~T_{GAP}(dir_conn_adv_int_max)~~	~~500 ms~~	~~Maximum advertising interval when in the directed connectable mode~~	~~Recommended value~~
~~T_{GAP}(undir_conn_adv_int_min)~~	~~1.28 s~~	~~Minimum advertising interval when in the undirected connectable mode~~	~~Recommended value~~
~~T_{GAP}(undir_conn_adv_int_max)~~	~~2.56 s~~	~~Maximum advertising interval when in the undirected connectable mode~~	~~Recommended value~~
~~T_{GAP}(auto_conn_est_scan_int)~~	~~1.28 s~~	~~Scan interval used in the auto connection establishment procedure~~	~~Recommended value~~
~~T_{GAP}(auto_conn_est_scan_wind)~~	~~11.25 ms~~	~~Scan window used in the auto connection establishment procedure~~	~~Recommended value~~
~~T_{GAP}(gen_conn_est_scan_int)~~	~~1.28 s~~	~~Scan interval used in the general connection establishment procedure~~	~~Recommended value~~
~~T_{GAP}(gen_conn_est_scan_wind)~~	~~11.25 ms~~	~~Scan window used in the general connection establishment procedure~~	~~Recommended value~~
~~T_{GAP}(sel_conn_est_scan_int)~~	~~1.28 s~~	~~Scan interval used in the selective connection establishment procedure~~	~~Recommended value~~
~~T_{GAP}(sel_conn_est_scan_wind)~~	~~11.25 ms~~	~~Scan window used in the selective connection establishment procedure~~	~~Recommended value~~
T_{GAP}(conn_param_timeout)	30 s	Timer used in connection parameter update procedure	Recommended value

Table 1.1: Defined GAP timers

GAP AUTHENTICATION AND LOST BOND

CONTENTS

1 CHANGE INSTANCES

1.1 CHANGE #1: VOLUME 3, PART C (GAP), SECTION 10.3

10.3 AUTHENTICATION PROCEDURE

The authentication procedure describes how the required security is established when a device initiates a service request to a remote device and when a device receives a service request from a remote device. A service request is a request to perform a GATT procedure.The authentication procedure covers ~~both~~ LE security mode 1~~ and LE security mode 2~~. The authentication procedure shall only be initiated after a connection has been established.

LE security mode 2 pertains to the use of data signing and is covered in Section 10.4.

Authentication in LE security mode 1 is achieved by enabling encryption as defined in Section 10.6. The security of the encryption is impacted by the type of pairing performed. There are two types of pairing: authenticated pairing or unauthenticated pairing. Authenticated pairing involves performing the pairing procedure defined in [Vol. 3], Part H, Section 2.1 with the authentication set to 'MITM protection'. Unauthenticated pairing involves performing the pairing procedure with authentication set to 'No MITM protection'. Note: In this section, the terms "authenticated pairing" and "unauthenticated pairing" refer to the security method used to perform pairing and are not related to the authentication of previously bonded devices during a reconnection.

Section 10.3.1 specifies how service requests are to be handled by a GATT server, and 10.3.2 specifies how service requests are made by a GATT client.

10.3.1 Receiving a Service Request

~~A service request is a request to perform a GATT procedure. A service request may also be a request to perform a procedure using other higher layer protocols on top of L2CAP.~~

When a GATT server receives a service request, it shall respond with an error code if the service request is denied. The error code is dependent on whether the current connection is encrypted or not and on the type of pairing that was performed to create the LTK to be used.

When ~~a device~~a GATT server receives a service request ~~from a remote device~~ it shall behave ~~as follows~~according to the following rules:

- The ~~S~~GATT server's security database specifies the security settings required to accept a service request~~ from a remote device~~. If no ~~security~~encryption ~~is~~and ~~no data signing are~~ required, the service request shall be accepted. ~~; otherwise, pairing is required before the service request can be accepted~~If encryption is required the server must send an error code as

defined in Table 10.2. If no encryption is required, but data signing is required, then the server behavior is as defined in Section 10.4.

- If ~~pairing has not occurred or an authenticated pairing is required but only an unauthenticated pairing has occurred~~an LTK is not available, the service request shall be rejected with the error code “Insufficient Authentication”.

  Note: When the link is not encrypted, the error code “Insufficient Authentication” does not indicate that MITM protection is required.

- If ~~the required pairing has occurred~~an LTK is available and encryption is required (LE security mode 1) but encryption is not enabled, the service request shall be rejected with the error code “Insufficient Encryption”. If the encryption is enabled with insufficient key size then the service request shall be rejected with the error code “Insufficient Encryption Key Size.”

- If an authenticated pairing is required but only an unauthenticated pairing has occurred and the link is currently encrypted, the service request shall be rejected with the error code “Insufficient Authentication.”

  Note: When unauthenticated pairing has occurred and the link is currently encrypted, the error code “Insufficient Authentication” indicates that MITM protection is required.

- ~~If authenticated pairing has occurred and encryption is not required (LE security mode 2) then Data Signing shall be used if CSRK has been exchanged, else the Host shall re-pair and exchange CSRK before proceeding with Data Signing.~~

The server will respond with the minimum security level required for access to its services. If the client has no security requirement it should default to the minimum security level that the server is capable of as defined in pairing phase 1, (see Vol 3, Part H, Section 2.1).

A server shall not require an authenticated pairing (MITM) if the server does not support the required IO capabilities or OOB data[1].

1. Note that if an OOB mechanism is used, the level of MITM protection is dependent upon the properties of the OOB communications channel. See Volume 3, Part H, Section 2.3.5.1 for more information.

- The server responses to a service request from a client are summarized in Table 10.2.

		Server Pairing Status		
Link Encryption State	**Server's Access Requirement for Service**	**No LTK**	**Unauthenticated LTK**	**Authenticated LTK**
Unencrypted	**None**	Request succeeds	Request succeeds	Request succeeds
	Encryption, No MITM Protection	Error Response = Insufficient Authentication	Error Response = Insufficient Encryption	Error Response = Insufficient Encryption
	Encryption, MITM Protection	Error Response = Insufficient Authentication	Error Response = Insufficient Encryption	Error Response = Insufficient Encryption
Encrypted	**None**	N/A (Not possible to be encrypted without LTK)	Request succeeds	Request succeeds
	Encryption, No MITM Protection		Requestsucceeds	Request succeeds
	Encryption, MITM Protection		Error Response = Insufficient Authentication	Request succeeds

Table 10.2: Server responses to a service request

[Add new section 10.3.1.1]

10.3.1.1 Handling of Indications and Notifications

A client "requests" a server to send indications and notifications by appropriately configuring the server via a Client Characteristic Configuration Descriptor. Since the configuration is persistent across a disconnection and reconnection, security requirements must be checked upon a reconnection before sending indications or notifications. When a server reconnects to a client to send an indication or notification for which security is required, the server shall initiate or request encryption with the client prior to sending an indication or notification. If the client does not have an LTK indicating that the client has lost the bond, enabling encryption will fail.

10.3.2 Initiating a Service Request

When ~~a device~~a GATT client initiates a service request to a remote device it shall behave ~~as follows~~according to the following rules:

- The ~~C~~client's security database specifies the security required to initiate a service request. If no ~~security~~encryption is required by the client then the service request ~~shall~~may proceed ~~otherwise pairing is required. If security is~~

~~required but pairing has not occurred in a previous bonding procedure then pairing shall occur during the current connection~~without encryption or pairing.

- If ~~pairing has not occurred or an authenticated pairing is required but only an unauthenticated pairing has occurred~~an LTK is not available but encryption is required, the pairing procedure shall be ~~executed~~initiated with the client's required authentication settings. If the pairing procedure fails then the service request shall be aborted.

  Note: When encryption is not enabled, the error code "Insufficient Authentication" does not indicate to the client that MITM protection is required.

  Note: If the client is a Peripheral then it may send a Slave Initiated Security Request as defined in Vol3, Part H, Section 2.4.6.

- If ~~authenticated~~ pairing has occurred but the encryption key size is insufficient the pairing procedure shall be executed with the required encryption key size. If the pairing procedure fails then the service request shall be aborted.

- If ~~the required pairing has occurred~~an LTK is available and encryption is required (LE security mode 1) then encryption shall be enabled before the service request proceeds~~. If encryption is required but encryption is not enabled and the remote device supports encryption then encryption shall be enabled using the encryption procedure~~ as defined in Section 10.6. Once encryption is enabled the service request shall proceed. ~~If encryption is required but not supported by the remote device then the service request shall be aborted.~~If encryption fails either the bond no longer exists on the server, or the wrong device has been connected. The client must, after user interaction to confirm the remote device, re-bond, perform service discovery and re-configure the server. If the client had previously determined that the server did not have the «Service Changed» characteristic then service discovery may be skipped.~~-~~

- ~~If encryption is not required and authenticated pairing has occurred (LE security mode 2) then the data signing procedure shall be used when making the service request~~If an authenticated pairing is required but only an unauthenticated pairing has occurred and the link is currently encrypted, the pairing procedure should be executed with the required authentication settings. If the pairing procedure fails, or an authenticated pairing cannot be performed with the IO capabilities of the client and server, then the service request shall be aborted.

  When a bond has been created between two devices, any reconnection should result in the client enabling or requesting encryption with the server before initiating any service request.

  If a client does not enable encryption before initiating a service request and relies on the error codes to determine the security requirements, the client shall not request pairing with MITM protection in response to receiving an "Insufficient Authentication" error code from the server

while the link is unencrypted. The client shall only set the MITM protection required flag if the client itself requires MITM protection.

- If encryption is not enabled at the time of the service request, the error code Insufficient Authentication is received, and the client currently has an LTK, then the encryption procedure should be started (see Section 10.6). If this fails (likely indicating that the server has lost the bond and no longer has the LTK) or the client does not have the correct LTK, then the pairing procedure should be started. IO capabilities are exchanged in pairing phase 1, (see Vol 3, Part H, Section 2.1) and the security level shall be determined by the device's IO capabilities and MITM requirements.
- If encryption is not enabled at the time of the service request, the error code Insufficient Encryption is received, and the client currently has an LTK, then the encryption procedure shall be started (see Section 10.6). If starting encryption fails (likely indicating that the server has lost the bond and no longer has the LTK) or the client does not have the correct LTK, then the pairing procedure should be started.

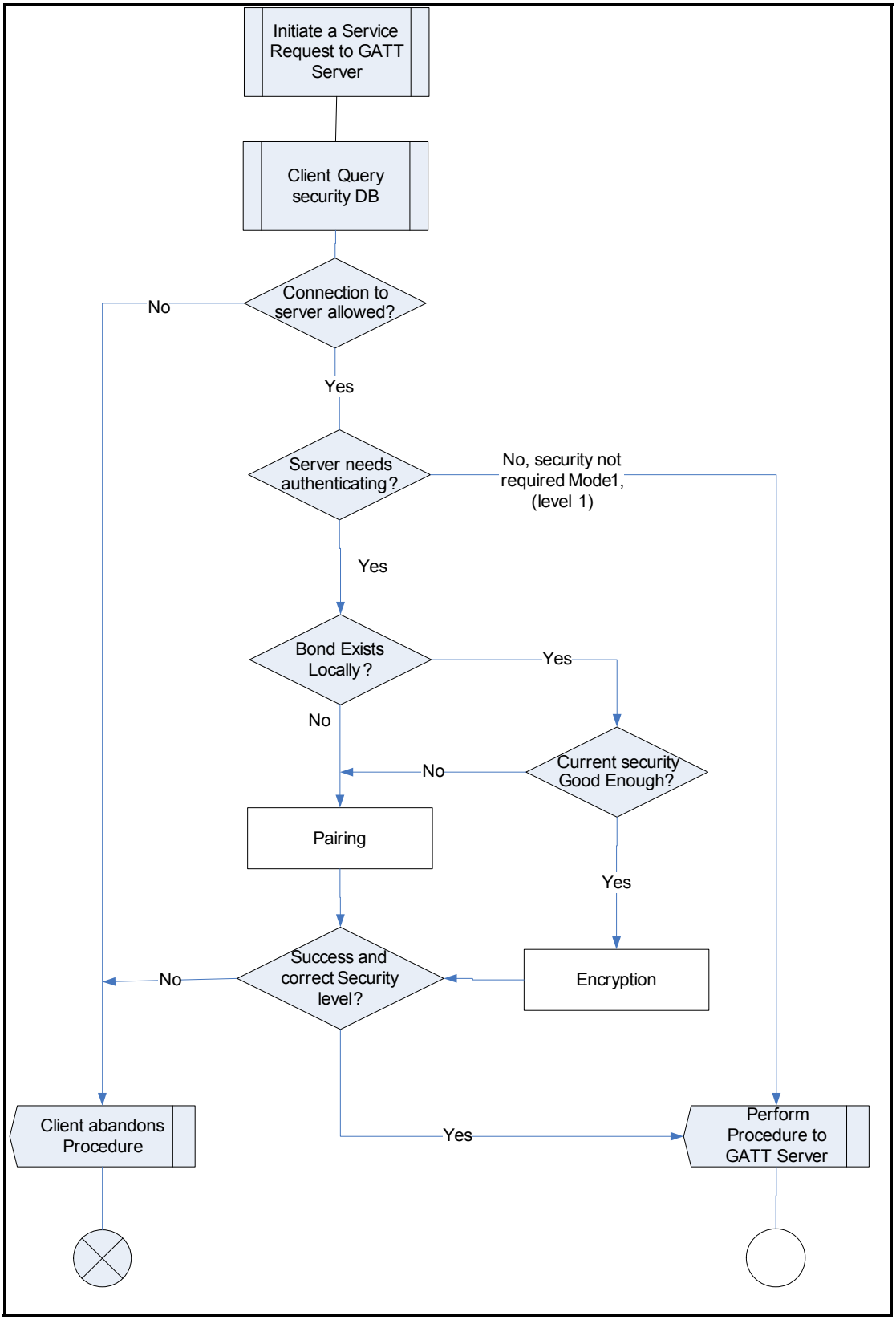


Figure 10.2: Flow chart for a client issuing a service request

1.2 CHANGE #2: VOLUME 3, PART C (GAP), SECTION 10.4.2

10.4.2. Authenticate Signed Data Procedure

If encryption is not required and CSRK is available (LE security mode 2) then the data signing procedure shall be used when making a service request involving a write operation.

Note: The existence of the bond on the server can be determined by successfully enabling encryption with the server using the encryption procedure defined in Section 10.6. A higher layer profile may allow a client to not perform the authentication procedure. Alternatively, a higher layer protocol may signal the client that the signature check has failed due to a lost bond, and the client may then take action to notify the user or attempt to pair again to reestablish the bond.

~~A device connected to a peer device sending signed data can authenticate the signed data by using the stored CSRK.~~

A device receiving signed data shall authenticate it by performing the Signing Algorithm.The signed data shall be authenticated by performing the Signing Algorithm where m is the Data PDU to be authenticated, k is the stored CSRK and the SignCounter is the received counter value. If the MAC computed by the Signing Algorithm does not match the received MAC, the verification fails and the Host shall ignore the received Data PDU.

Note that since the server does not respond to a Signed Write Command, the higher layer application may not be notified of the ignored request. Hence, it is recommended that the server disconnect the link in case the client is a malicious device attempting to mount a security attack.

If the server has no stored CSRK upon receiving a signed write command, it shall ignore the received Data PDU. Note that since the server does not respond to a Signed Write Command, the higher layer application may not be notified of the ignored request. Although the disconnection may be an adequate indication to the user that the devices need to be paired, it is recommended that implementers consider providing a more informative indication to the user that the devices need to be paired to establish a CSRK.

If the server receives a request from a client for a write operation that requires a response (i.e. other than a Signed Write Command or Write Command), and encryption is not enabled, then the server shall respond with the error code “Insufficient Authentication”.

If the link is encrypted and the server receives a request from a client for which the server requires data signing but does not require encryption, then the server shall complete the request if it is otherwise valid as the encrypted state of the link is considered to satisfy the signing requirement.

As required by Section 10.2.2, for a given link, signed data is not used at the same time as encryption. Therefore, if the client wishes to test that the server is still bonded, and thus enables encryption, further data transfer must occur without signing, assuming the server does not disconnect the link as recommended above.

If a higher layer determines the bond no longer exists on the server, the client must, after user interaction to confirm the remote device, re-bond, perform service discovery and re-configure the server. If the client had previously determined that the server did not have the «Service Changed» characteristic then service discovery may be skipped.

DUAL MODE ADDRESSING

CONTENTS

1 CHANGE INSTANCES

1.1 CHANGE #1 - VOLUME 3, PART C (GAP), SECTION 3.2.1.1

3.2.1.1 Definition

~~A device shall be identified by a Bluetooth device address. When the device is operating over a BR/EDR physical channel the device address shall be the BD_ADDR (see [Vol. 2], Part B Section 1.2). The device address is obtained from a remote device during the device discovery procedure (see Section 6.4 and Section 13.2.3).~~

~~The BD_ADDR shall be created in accordance with Section 9.2 (“48-bit universal LAN MAC addresses”) of the IEEE 802-2001 standard (http://standards.ieee.org/getieee802/download/802-2001.pdf) and using a valid Organizationally Unique Identifier (OUI) obtained from the IEEE Registration Authority (see http://standards.ieee.org/regauth/oui/forms/ and sections 9 and 9.1 of the IEEE 802-2001 specification).~~

~~*3.2.1.1.1 Bluetooth Device Address in an LE-only Device Type*~~

~~An LE-only device type shall use either a public or a random device address. The public device address shall be set to the BD_ADDR. A random device address is defined in Section 10.8. An LE-only device type may support the privacy feature as defined in Section 10.7 which requires the support of private device addresses as defined in Section 10.8.2.~~

~~*3.2.1.1.2 Bluetooth Device Address in a BR/EDR/LE Device Type*~~

~~A BR/EDR/LE device type shall use the BD_ADDR on the BR/EDR physical channel and a public device address over the LE physical channel. The public device address shall be set to the BD_ADDR. A BR/EDR/LE device type shall be capable of connecting to an LE-only or another BR/EDR/LE device type that is identified by a random device address.~~

A BD_ADDR is the address used by a Bluetooth device as defined in Section 15.1. It is received from a remote device during the device discovery procedure.

1.2 CHANGE #2 - VOLUME 3, PART C (GAP), SECTION 15

Note: Subsequent sections in GAP will be renumbered.

[New Section]

15 Bluetooth Device Requirements

15.1 Bluetooth Device Address

All Bluetooth devices shall have a Bluetooth Device Address (BD_ADDR) that uniquely identifies the device to another Bluetooth device. The specific Bluetooth Device Address requirements depend on the type of Bluetooth device.

15.1.1 Bluetooth Device Address Types

15.1.1.1 Public Bluetooth Address

A Bluetooth public address used as the BD_ADDR on either the BR/EDR or LE physical channel is an address that complies with Section 9.2 (“48-bit universal LAN MAC addresses”) of the IEEE 802-2001 standard (http://standards.ieee.org/getieee802/download/802-2001.pdf) and uses a valid Organizationally Unique Identifier (OUI) obtained from the IEEE Registration Authority (see http://standards.ieee.org/regauth/oui/forms/ and sections 9 and 9.1 of the IEEE 802-2001 specification).

15.1.1.2 Random Bluetooth Address

A random device address used as the BD_ADDR on the LE physical channel is defined in Section 10.8.

PRIVATE ADDRESS CHANGES

CONTENTS

1 CHANGE INSTANCES

1.1 CHANGE #1 - VOLUME 3, PART C (GAP), SECTION 10.8

Random device addresses are defined in the Link Layer section in [Vol. 6], Part B Section 1.3.

For the purposes of this profile, the random device address may be of either of the following two sub-types:

- Static address
- Private address

The term random device address refers to both static and private address types.

The transmission of a random device address is optional. A device shall accept the reception of a random device address from a remote device.

The private address may be of either of the following two sub-types:

- Non-resolvable private address
- Resolvable private address

A bonded device shall process a resolvable private address as defined in Volume 3, Part C, Section 10.8.2.3 or by establishing a connection and then performing the authentication procedure as defined in Volume 3, Part C, Section 10.3. A device that uses resolvable private address shall always request to distribute its IRK value as defined in Volume 3, Part H, Section 3.6.4 if both sides are bondable, unless keys have been pre-distributed. If the request to distribute the IRK fails then the peer device may authenticate re-connections using the authentication procedure as defined in Volume 3, Part C, Section 10.3 or terminate the current connection.

COMMON PROFILE AND SERVICE ERROR CODE CHANGES

CONTENTS

1 CHANGE INSTANCES

CHANGE #1 – VOLUME 3, PART F (ATT), SECTION 3.4.1.1

Name	Error Code	Description
Invalid Handle	0x01	The attribute handle given was not valid on this server.
Read Not Permitted	0x02	The attribute cannot be read.
Write Not Permitted	0x03	The attribute cannot be written.
Invalid PDU	0x04	The attribute PDU was invalid.
Insufficient Authentication	0x05	The attribute requires authentication before it can be read or written.
Request Not Supported	0x06	Attribute server does not support the request received from the client.
Invalid Offset	0x07	Offset specified was past the end of the attribute.
Insufficient Authorization	0x08	The attribute requires authorization before it can be read or written.
Prepare Queue Full	0x09	Too many prepare writes have been queued.
Attribute Not Found	0x0A	No attribute found within the given attribute handle range.
Attribute Not Long	0x0B	The attribute cannot be read or written using the Read Blob Request
Insufficient Encryption Key Size	0x0C	The Encryption Key Size used for encrypting this link is insufficient.
Invalid Attribute Value Length	0x0D	The attribute value length is invalid for the operation.
Unlikely Error	0x0E	The attribute request that was requested has encountered an error that was unlikely, and therefore could not be completed as requested.
Insufficient Encryption	0x0F	The attribute requires encryption before it can be read or written.
Unsupported Group Type	0x10	The attribute type is not a supported grouping attribute as defined by a higher layer specification.
Insufficient Resources	0x11	Insufficient Resources to complete the request
Reserved	0x12 – 0x7F	Reserved for future use.

Table 3.3 Error Codes

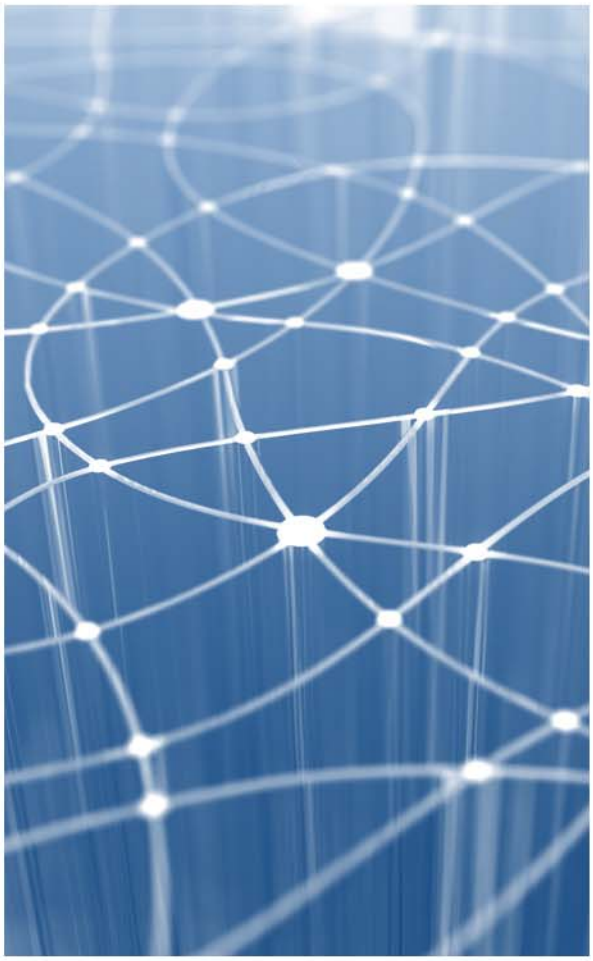
Name	Error Code	Description
Application Error	0x80 – 0x9F	Application error code defined by a higher layer specification.
Reserved	0xA0 – 0xDF	Reserved for future use.
Common Profile and Service Error Codes	0xE0 – 0xFF	Common profile and service error codes defined in Core Specification Supplement, Part B.

Table 3.3 Error Codes

Specification Volume 7

SPECIFICATION OF THE *BLUETOOTH* SYSTEM

Experience More



Core System Package [Wireless Coexistence volume]

Covered Core Package version: 4.0
Current Master TOC
Publication date: 24 July 2012

Bluetooth®

Revision History

The Revision History is shown in the [Vol 0] Part C, Section on page 55.

Contributors

The persons who contributed to this specification are listed in the [Vol 0] Part C, Section on page 55.

Web Site

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

Disclaimer and Copyright Notice

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.

Core System Package [Wireless Coexistence volume]

Part A

MWS COEXISTENCE LOGICAL SIGNALING SPECIFICATION

This document specifies the MWS coexistence logical interface between the host controller and an MWS device.

CONTENTS

1 INTRODUCTION

This part of the Bluetooth Core Specification describes the MWS Coexistence Logical Signaling. A Bluetooth Controller may incorporate a real-time transport interface to transport the logical signals defined in this section between the Bluetooth Controller and an MWS device.

Figure 1.1 depicts the signaling and messaging architecture.

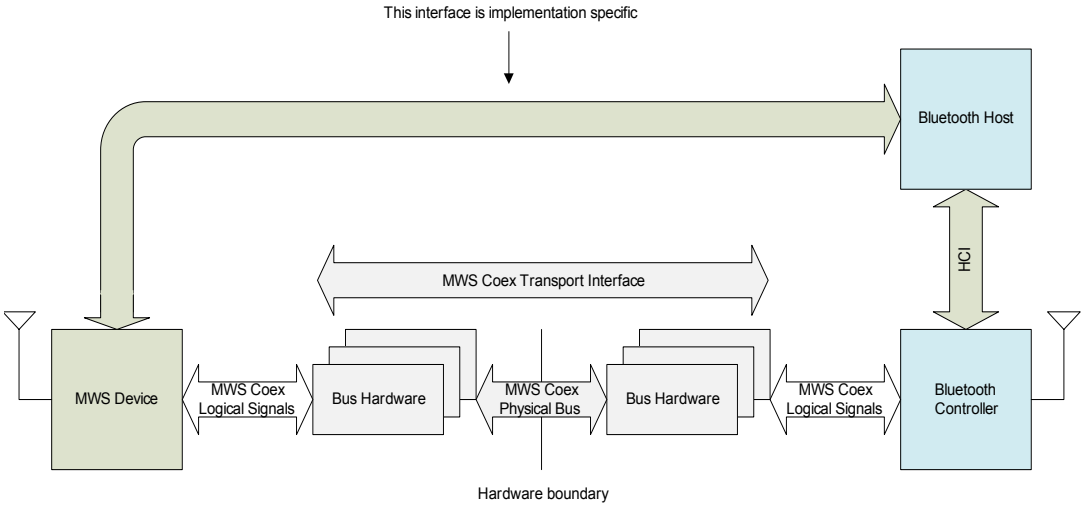


Figure 1.1: Coexistence signaling/messaging architecture

MWS Coexistence logical signaling is designed to enable a standard interface to allow an MWS device and a Bluetooth Controller to exchange information and support cooperative coexistence.

MWS Coexistence logical signaling defines a set of signals between the collocated Bluetooth Controller and MWS device. Those signals carry time critical information such as the start point of an MWS frame. The coexistence logical signaling architecture also includes a transparent data messaging mechanism to enable passing of information between the MWS device and Bluetooth Controller when such information cannot tolerate the long latency (tens of milliseconds) of the signaling path via the Bluetooth Host.

2 LOGICAL INTERFACE

2.1 COEXISTENCE SIGNALS CONVEYING TIME CRITICAL INFORMATION

Table 2.1 defines the logical signals. These logical signals assist in time alignment, protecting the MWS device and the Bluetooth Controller from mutual interference, thus maximizing the usability of the Bluetooth radio.

Name	Direction	Description
FRAME_SYNC	MWS → Bluetooth	See Section 2.1.1
MWS_RX	MWS → Bluetooth	See Section 2.1.2
BLUETOOTH_RX_PRI	Bluetooth → MWS	See Section 2.1.3
BLUETOOTH_TX_ON	Bluetooth → MWS	See Section 2.1.4
MWS_PATTERN	MWS → Bluetooth	See Section 2.1.5
MWS_TX	MWS → Bluetooth	See Section 2.1.6
802_RX_PRI	Bluetooth → MWS	See Section 2.1.7
802_TX_ON	Bluetooth → MWS	See Section 2.1.8
MWS_INACTIVITY_DURATION	MWS → Bluetooth	See Section 2.1.9
MWS_SCAN_FREQUENCY	MWS → Bluetooth	See Section 2.1.10

Table 2.1: Time Critical Coexistence Signals

Many of the signals have associated parameters, which are configured by the Bluetooth Host using HCI commands. There is no requirement for signals that are used internally to be connected to an external interface, although testing requires external control of the FRAME_SYNC signal. For example, a combo-device that integrates a Bluetooth Controller and an MWS radio together does not need to bring out the coexistence signals.

Section 2.2 provides recommended values for the defined offset and jitter parameters.

2.1.1 FRAME_SYNC

The FRAME_SYNC signal is sent by the MWS device to indicate the time of the beginning of MWS frames according to MWS network timing. It provides the anchor point for the Bluetooth Controller to properly align Bluetooth transmission and reception activity with the MWS network frame structure. The time when FRAME_SYNC is sent over the transport, adjusted for Ext_Frame_Sync_Assert_Offset, indicates the start time of the first Period_Duration parameter in the HCI_Set_External_Frame_Configuration command.

The MWS device will inform the Bluetooth Controller about changes to the lay-out and timing of the MWS frame by issuing new HCI_Set_External_Frame_Configuration commands.

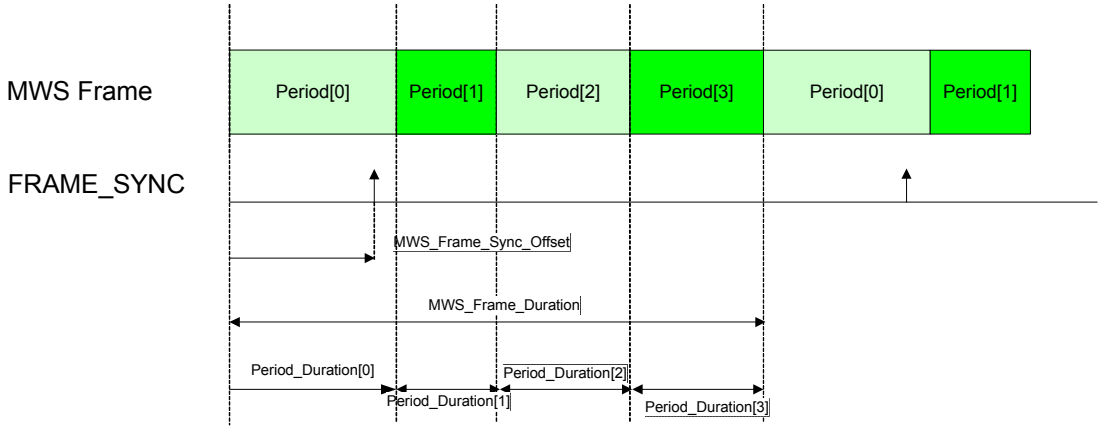


Figure 2.1: Illustration of FRAME_SYNC

2.1.2 MWS_RX

The MWS_RX signal is sent by the MWS device to indicate that an MWS reception is occurring and to request that the Bluetooth Controller cease ongoing transmission and not start a new transmission. The Bluetooth Controller can occasionally disregard the MWS_RX signal for critical transmissions.

The MWS RX signal should be de-asserted at the time an MWS device stops actively receiving. If there are multiple distinct periods of reception within an MWS downlink duration, the signal may stay asserted until the last period has finished receiving.

2.1.3 BLUETOOTH_RX_PRI

The BLUETOOTH_RX_PRI signal is used by the Bluetooth Controller to request that the MWS device cease its transmission and/or refrain from starting a transmission because the Bluetooth Controller is expecting a high priority reception.

The signal should be used minimally by the Bluetooth system so as not to adversely affect the MWS system. There is no guarantee that the MWS device will honor the signal and not transmit or abort an ongoing transmission.

2.1.4 BLUETOOTH_TX_ON

The BLUETOOTH_TX_ON signal is sent by the Bluetooth Controller to indicate that it is actively transmitting.

2.1.5 MWS_PATTERN

The MWS_PATTERN signal is sent by the MWS device to inform the Bluetooth Controller which MWS_PATTERN is in use.

MWS_PATTERN changes take effect at the start of the next MWS Frame as defined by the FRAME_SYNC signal plus the MWS_Frame_Sync_Offset.

Up to three different MWS_PATTERNs can be selected: 0, 1, and 2. The MWS device may indicate that the MWS_PATTERN has not changed by setting it to 3. The definition of the patterns is communicated to the Bluetooth Controller using the HCI_Set_MWS_PATTERN_Configuration command.

The MWS_PATTERN will be retransmitted by the MWS device every pattern interval.

2.1.6 MWS_TX

The MWS_TX signal is sent by the MWS device to indicate its transmission state.

The signal should be asserted at the beginning of an MWS transmission and de-asserted at the end of a transmission. If there are multiple transmission periods during an uplink frame, the signal may stay asserted until the end of the last transmission period.

2.1.7 802_TX_ON

Bluetooth technology and 802.11 may be collocated and share an interface to coordinate access to the 2.4 GHz ISM band.

The 802_TX_ON is used by the 802.11 device to indicate the state of the 802.11 transmission.

Interference generated by 802.11 to the MWS device will not necessarily be distinguishable from interference created by Bluetooth transmissions.

This signal allows the MWS device to distinguish the interference generated by 802.11 transmissions from the interference generated by the Bluetooth transmissions. The MWS device can use that information to optimize its channel access.

2.1.8 802_RX_PRI

This signal requests that the MWS device stop or refrain from any transmission because the WLAN device collocated with the Bluetooth Controller is expecting a high priority reception.

The signal should be used minimally by the 802.11 system so as not to adversely affect the MWS system. There is no guarantee that the MWS device will honor the signal and not transmit or abort an ongoing transmission.

2.1.9 MWS_INACTIVITY_DURATION

The MWS_INACTIVITY_DURATION signal provides the time duration until the MWS device is active again. Subsequent MWS_INACTIVITY_DURATION signals override previously sent time durations.

MWS_INACTIVITY_DURATION may be set to zero (i.e. cancel), infinite, or a positive finite duration. The transport layer defines the value for infinite duration and the set of finite durations available.

2.1.10 MWS_SCAN_FREQUENCY

The MWS_SCAN_FREQUENCY signal provides an index to a table of RF frequencies during MWS scan operation.

The MWS device signals MWS_SCAN_FREQUENCY if it starts an inter-frequency scan.

Setting MWS_SCAN_FREQUENCY to a non-zero value indicates the start of the MWS scan period. Setting MWS_SCAN_FREQUENCY to zero indicates the end of the scan period.

The Bluetooth Controller should avoid any transmissions that can interfere with the scan while a scan is active. The Bluetooth Controller can occasionally disregard the signal for critical transmissions.

2.2 TOLERANCES FOR OFFSETS AND JITTER

This section lists the recommended tolerances for the signal assertion and de-assertion offsets and jitter for those signals. Note: The transmitting side should use a value within the range and the receiving side should accept any value within the range and may accept other values.

An offset is a static advance notification or delay between the real physical event and the time when the corresponding signal is issued.

Jitter is variations in the timing of each signal from ideal timing.

Signals that are turned on and off around a period of time have specified values for both assertion and de-assertion offsets and jitter. Signals that represent a single event at a single instant in time have only assertion offset and jitter timing specified.

All jitter values are given as an unsigned value representing the maximum allowable jitter in positive and negative directions. De-assertion and

MWS_Frame_Sync_Assert_Offset may be negative (i.e., signal is asserted before the event) or positive (i.e., signal is asserted after the event). All other assertion signals should be negative (i.e., signal is issued before the event).

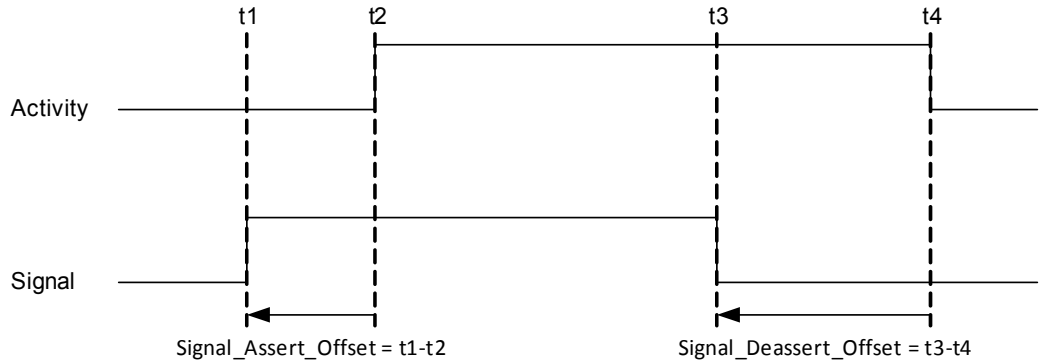


Figure 2.2: Signal with negative Assert Offset and negative De-assert Offset

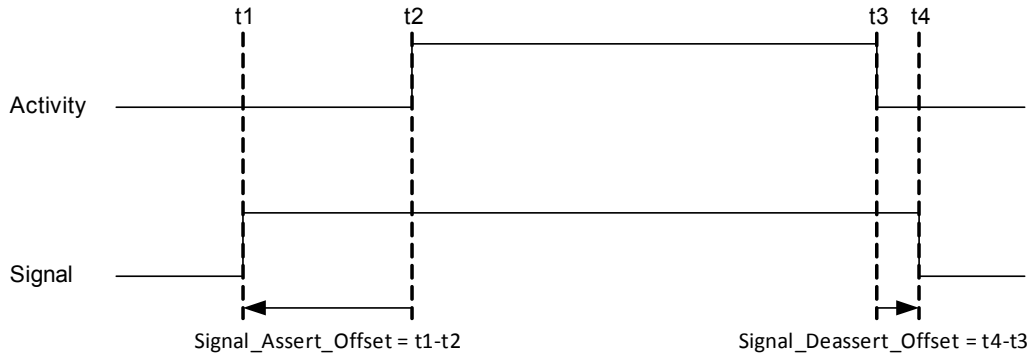


Figure 2.3: Signal with negative Assert Offset and positive De-assert Offset

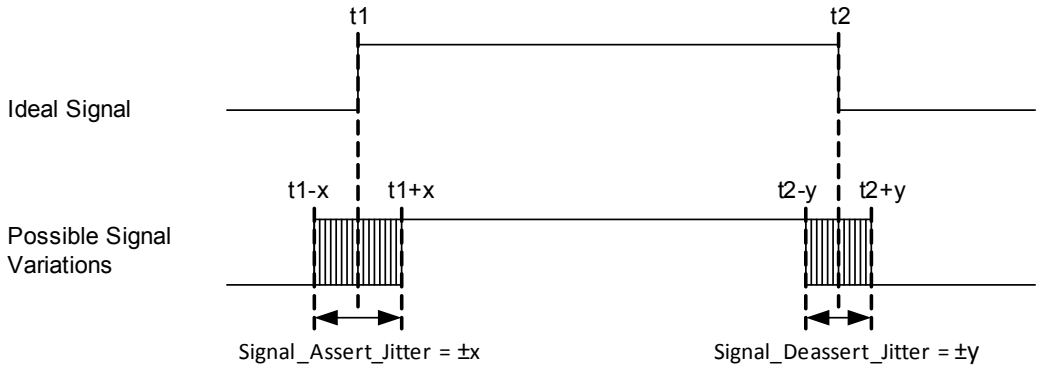


Figure 2.4: Signal Assertion and De-assertion Jitter

Parameter	Earliest	Latest
MWS_Frame_Sync_Assert_Offset	-Ext_Frame_Duration	+Ext_Frame_Duration
MWS_Frame_Sync_Assert_Jitter	N/A	±3 usec
MWS_Rx_Assert_Offset	-2 ms	-20 usec
MWS_Rx_Assert_Jitter	N/A	±5 usec
MWS_Rx_Deassert_Offset	-2 ms	0
MWS_Rx_Deassert_Jitter	N/A	±5 usec
MWS_Tx_Assert_Offset	-2 ms	0
MWS_Tx_Assert_Jitter	N/A	±5 usec
MWS_Tx_Deassert_Offset	-2 ms	0
MWS_Tx_ Deassert_Jitter	N/A	±5 usec
MWS_Pattern_Assert_Offset	0	+Ext_Frame_Duration
MWS_Pattern_ Assert_Jitter	N/A	±5 usec
MWS_Inactivity_Duration_Assert_Offset	0	+Ext_Frame_Duration
MWS_Inactivity_Duration_Assert_Jitter	N/A	±5 usec
MWS_Scan_Frequency_Assert_Offset	-2 ms	-20 usec
MWS_Scan_Frequency_Assert_Jitter	N/A	±5 usec
Bluetooth_Rx_Priority_Assert_Offset	-1 ms	0
Bluetooth_Rx_ Priority _Assert_Jitter	N/A	±5 usec
Bluetooth_Rx_Priority_Deassert_Offset	-1 ms	0
Bluetooth_Rx Priority Deassert_Jitter	N/A	±5 usec
802_Rx_Priority Assert_Offset	-1 ms	0
802_Rx_Priority Assert_Jitter	N/A	±5 usec
802_Rx_Priority_Deassert_Offset	-1 ms	0
802_Rx_Priority_Deassert_Jitter	N/A	±5 usec
Bluetooth_Tx_On_Assert_Offset	-100 usec	0
Bluetooth_Tx_On_Assert_Jitter	N/A	±5 usec
Bluetooth_Tx_On_Deassert_Offset	0	100 usec
Bluetooth_Tx_On_Deassert_Jitter	N/A	±5 usec
802_Tx_On_Assert_Offset	-100 usec	0
802_Tx_On_Assert_Jitter	N/A	±5 usec
802_Tx_On_Deassert_Offset	0	100 usec
802_Tx_On_Deassert_Jitter	N/A	±5 usec

Table 2.2: Tolerances for Offsets and Jitter

Parameter	Earliest	Latest
MWS_Priority_Assert_Offset_Request	-1 ms	-200 usec

Table 2.2: Tolerances for Offsets and Jitter

WIRELESS COEXISTENCE INTERFACE 1 (WCI-1) TRANSPORT SPECIFICATION

CONTENTS

1 INTRODUCTION

This part of the Bluetooth Core Specification describes the MWS Wireless Coexistence Interface 1 (WCI-1) Transport for a Bluetooth Controller. It provides a half-duplex UART carrying logical signals framed as UART characters. Only the TXD and RXD UART signals are used.

2 PHYSICAL LAYER

The WCI-1 physical layer multiplexes the UART TXD and RXD onto a single wire, using drive strengths to resolve contention. The MWS device uses direct drive to transmit its signals, while the Bluetooth Controller uses pull up / pull down to transmit its signals. The configuration is illustrated in Figure 2.1.

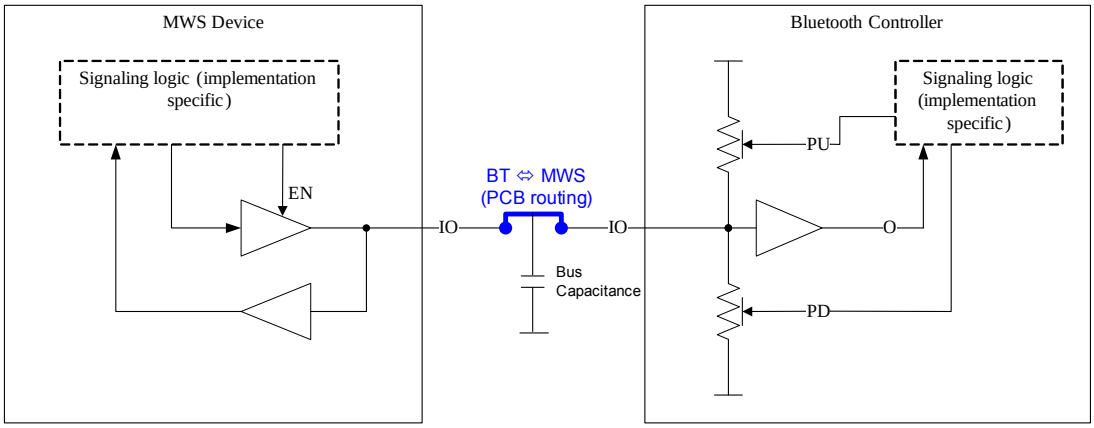


Figure 2.1: WCI-1 physical interface

A high voltage on the wire shall be interpreted as a logical ‘1’ and a low voltage shall be interpreted as a logical ‘0’. The actual voltage levels are vendor specific.

The MWS device output buffer shall be in high impedance state when it is not transmitting. The Bluetooth Controller shall be in pulled up state when it is not transmitting.

The MWS device may transmit at any time, using the waveform illustrated in Figure 2.2.

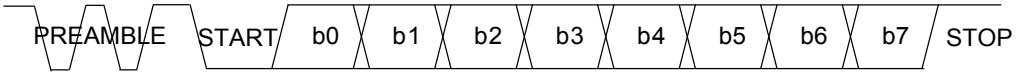


Figure 2.2: UART waveform for MWS-to-Bluetooth signals

Every MWS-to-Bluetooth message shall be preceded by a preamble, which consists of 5 bits 01011 (the left-most bit is transmitted first). The preamble is sent at a baud rate that is at least twice the baud rate of Bluetooth-to-MWS signals. The nominal drive strength for the preamble should be targeted at no more than 250Ω equivalent output resistance. The Bluetooth Controller shall be able to detect the preamble and go into the reception mode to receive the message that follows. If the Bluetooth Controller detects a preamble while it is transmitting, it shall stop the transmission and go into the reception mode. After completion of the reception, it may retransmit the message that was interrupted.

When the Bluetooth Controller is not in the reception mode, it may transmit a message using the pull up / pull down mechanism. The nominal pull strength should be targeted at 4 kΩ ± 1 kΩ equivalent resistance for both pull up and pull down. The Bluetooth-to-MWS waveform is illustrated in Figure 2.3.

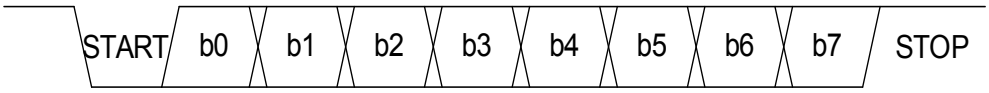


Figure 2.3: UART waveform for Bluetooth-to-MWS signals

2.1 PHYSICAL SIGNAL SPECIFICATIONS (INFORMATIVE)

The tables below provide more complete specifications for the physical signals. They are provided as a reference to device makers:

Symbol	Parameter	Condition	Value	
			Min	Max
V_{IL}	Low level input voltage	V_{DD}=1.8~2.5V[1]	-0.5V	0.2V_{DD}
V_{IH}	High level input voltage	V_{DD} =1.8~2.5V[1]	0.8 V_{DD}	V_{DD}+0.5V
V_{OL}	Low level output voltage (with extra pull high R_B)	V_{DD} =1.8~2.5V[1] R_B=2.5kΩ	0	0.1V_{DD}
V_{OH}	High level output voltage (with extra pull low R_B)	V_{DD}=1.8~2.5V[1] R_B=2.5kΩ	0.9V_{DD}	V_{DD}
C_{IO}	Capacitance for I/O		--	5pF
CB	Capacitive load for bus		--	10pF
R_{ON}	Turn on impedance			250Ω
T_R	Rise time (10% to 90% swing time, with extra pull low R_B and capacitive load C_L = C_B + other IO)	R_B=2.5kΩ C_L=15pF	--	50 ns
T_F	Fall time (90% to 10% swing time, with extra pull high R_B and capacitive load C_L = C_B + other IO)	R_B=2.5kΩ C_L=15pF	--	50 ns
F	Frequency		(1-1%)*4MHz[2]	(1+1%)*4MHz[2]
CLK_j	Clock jitter			1%

Table 2.1: WCI-1 UART Physical Signal Specification for the MWS device

Symbol	Parameter	Condition	Value	
			Min	Max
V_{IL}	Low level input voltage	V_{DD} =1.8~2.5V[1]	-0.5V	$0.2V_{DD}$
V_{IH}	High level input voltage	V_{DD} =1.8~2.5V[1]	$0.8V_{DD}$	V_{DD}+0.5V
V_{OL}	Low level output voltage	V_{DD} =1.8~2.5V[1]	0	$0.1V_{DD}$
V_{OL}	High level output voltage	V_{DD} =1.8~2.5V[1]	$0.9V_{DD}$	V_{DD}
C_{IO}	Capacitance for I/O		--	5 pF
CB	Capacitive load for bus		--	10 pF
R_P	Pull up/pull down resistance		3kΩ	5kΩ
T_R	Rise time (10% to 90% swing time, with capacitive load $C_L = C_B$ + other IO)	C_L=15pF	--	220 ns
T_F	Fall time (90% to 10% swing time, with capacitive load $C_L = C_B$ + other IO)	C_L =15pF	--	220 ns
F	Frequency		(1-1%)*1MHz[2]	(1+1%)*1MHz[2]
CLK_j	Clock jitter			1%

Table 2.2: WCI-1 UART Physical Signal Specification for the Bluetooth Controller

Notes:

1. The voltage levels are vendor specific. The tables in this section do not cover all possible ranges of voltage for all devices, nor is it required that a single device be able to operate in the full range indicated here.
2. The frequencies are vendor specific. The values provided in this section are for reference only.

3 TRANSPORT LAYER

The transport layer defines the mapping of the logical signals (see Vol 7, Part A) onto the physical transport channel.

The 8-bit UART character is divided into two portions with three bits for the message type indicator and five bits for the message body. The bit with index 0 is the LSB and shall be transmitted first.

All bits marked RFU shall be set to zero on transmission and ignored on reception.

b0	b1	b2	b3	b4	b5	b6	b7
Type[0]	Type[1]	Type[2]	MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]

Table 3.1: Transport Layer Format

3.1 MESSAGE TYPES

This section describes the message formats for the logical coexistence signals. The message types are listed in Table 3.2.

Message Type Indicator	Direction	Message Type
0	MWS ↔ Bluetooth	Real Time Signal Message
1	MWS ↔ Bluetooth	Transport Control Message
2	MWS ↔ Bluetooth	Transparent Data Message
3	MWS → Bluetooth	MWS Inactivity Duration Message
	BT → MWS	RFU
4	MWS -> Bluetooth	MWS Scan Frequency Message
	Bluetooth → MWS	RFU
5	MWS → Bluetooth	RFU
	Bluetooth → MWS	RFU
6		Vendor Specific
7		Vendor Specific

Table 3.2: Message Type Indicator

The logical coexistence signals are listed in Table 3.3.

Signal Name	Description
FRAME_SYNC	See Volume 7, Part A, Section 2.1.1
MWS_RX	See Volume 7, Part A, Section 2.1.2
MWS_TX	See Volume 7, Part A, Section 2.1.6
PATTERN[1,0]	See Volume 7, Part A, Section 2.1.5
BLUETOOTH_RX_PRI	See Volume 7, Part A, Section 2.1.3
BLUETOOTH_TX_ON	See Volume 7, Part A, Section 2.1.4
802_RX_PRI	See Volume 7, Part A, Section 2.1.8
802_TX_ON	See Volume 7, Part A, Section 2.1.7
MWS_INACTIVITY_DURATION	See Volume 7, Part A, Section 2.1.9
MWS_SCAN_FREQUENCY_OFFSET	See Volume 7, Part A, Section 2.1.10

Table 3.3: Coexistence Signals

3.1.1 Real Time Signaling Message (Type 0)

The Real Time Signaling message is used to transport the real time signals over the WCI-1 transport interface.

The Real Time Signaling message conveys all the real time signals (see Vol 7, Part A) in one message. The time reference point for Real Time Signaling message is end of MSG[4] (transition to STOP bit).

Two Real Time Signaling messages are defined, one from the Bluetooth Controller to the MWS device and another from the MWS device to the Bluetooth Controller.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
FRAME_SYNC	MWS_RX	MWS_TX	MWS_PATTERN[0]	MWS_PATTERN[1]

Table 3.4: Real Time Message from MWS Device to Bluetooth Controller

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
BLUETOOTH_RX_PRI	BLUETOOTH_TX_ON	802_RX_PRI	802_TX_ON	RFU

Table 3.5: Real Time Message from Bluetooth Controller to MWS Device

3.1.2 Transport Control Messages (Type 1)

The Transport Control messages can request state information of the MWS device's coexistence interface.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
RESEND_REAL_TIME	RFU	RFU	RFU	RFU

Table 3.6: Transport Control Message

Signal Name	Description
RESEND_REAL_TIME	This bit is set if a device wants to get a status update of the real time signals. The signal is usually used after wake-up from sleep of the transport interface. If the receiving device's transport interface is awake it shall send a Real Time Message with the current status of the real time signals within 4 UART character periods. If the signal is not received within 4 UART character periods the device is considered asleep.

Table 3.7: Transport Control Signals

3.1.3 Transparent Data Message (Type 2)

The Transparent Data message can be used to exchange non-time critical signals between the MWS device and the Bluetooth Controller. The interface does not guarantee the delivery of a message. Protocol and content of the message is vendor specific.

The least significant nibble of each octet shall be transmitted first.

A least significant nibble shall be discarded if the next nibble is a least significant nibble. A most significant nibble shall only be accepted if the preceding nibble was a least significant nibble.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
NIBBLE_POSITION	DATA[0] / DATA[4]	DATA[1] / DATA[5]	DATA[2] / DATA[6]	DATA[3] / DATA[7]

Table 3.8: Transparent Data Message

Signal Name	Description
NIBBLE_POSITION	0 – Least Significant Nibble 1 – Most Significant Nibble
DATA[n]; n = 0..7	Data bits of the message octet

Table 3.9: Transparent Data Bits

3.1.4 MWS Inactivity Duration Message (Type 3)

The MWS Inactivity Duration message is used to send the MWS_INACTIVITY_DURATION signal from the MWS device to the Bluetooth Controller.

The message is sent at the beginning of an MWS inactivity period.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
DURATION[0]	DURATION[1]	DURATION[2]	DURATION[3]	DURATION[4]

Table 3.10: MWS Inactivity Duration Message

The MWS Inactivity Duration is encoded in 5 bits. DURATION is unsigned.

When DURATION = 0, MWS_INACTIVITY_DURATION is cancelled. When DURATION = 31, MWS_INACTIVITY_DURATION is infinite. Otherwise, MWS_INACTIVITY_DURATION is given by the formula:

MWS_INACTIVITY_DURATION = DURATION * 5 ms

3.1.5 MWS Scan Frequency Message (Type 4)

The MWS Scan Frequency message is used to send the MWS_SCAN_FREQUENCY signal from the MWS device to the Bluetooth Controller.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
FREQ[0]	FREQ[1]	FREQ[2]	FREQ[3]	FREQ[4]

Table 3.11: MWS Scan Frequency Message

The MWS Scan Frequency index is encoded in 5 bits. FREQ is unsigned.

Core System Package [Wireless Coexistence volume]

Part C

WIRELESS COEXISTENCE INTERFACE 2 (WCI-2) TRANSPORT SPECIFICATION

This document specifies the MWS Wireless Coexistence Interface 2 (WCI-2) Transport Interface between the Bluetooth Controller and an MWS device.

CONTENTS

1 INTRODUCTION

This part of the Bluetooth Core Specification describes the MWS Wireless Coexistence Interface 2 (WCI-2) Transport Interface for a Bluetooth Controller.

2 PHYSICAL LAYER

The WCI-2 Transport is based on a standard full duplex UART carrying logical signals framed as UART characters. Only the TXD and RXD UART signals are used. The interface supports multiple logical channels.

The messaging is based on a standard UART format.

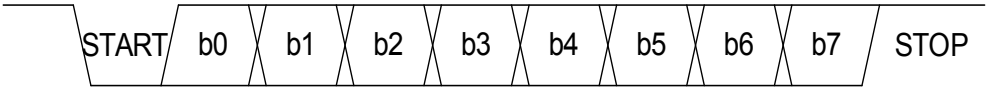


Figure 2.1: UART Waveform

The UART signals shall be connected in a null-modem fashion; i.e. the local TXD shall be connected to the remote RXD and vice versa.

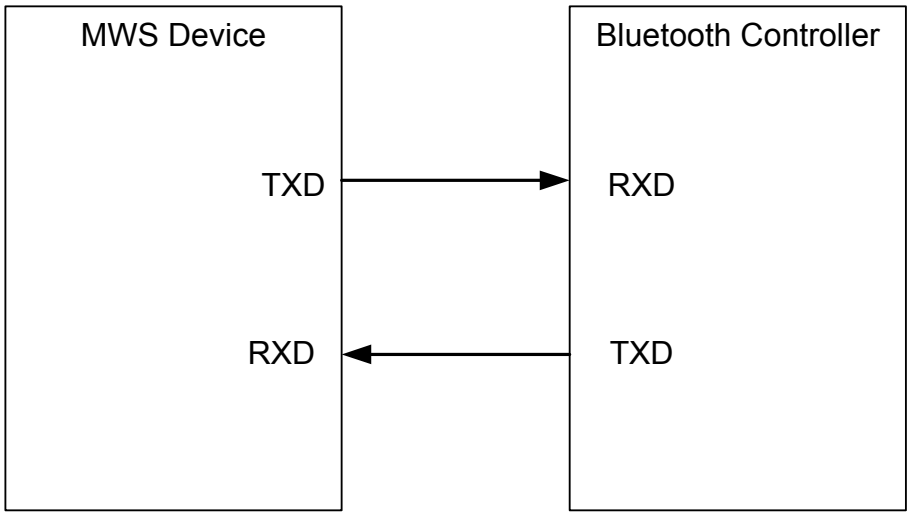


Figure 2.2: WCI-2 Physical Interface

3 TRANSPORT LAYER

The transport layer defines the mapping of the logical signals (see Vol 7, Part A) onto the physical transport channel.

The 8 bit UART character is divided into two portions with three bits for the message type indicator and five bits for the message body. The bit with index 0 is the LSB and shall be transmitted first.

All bits marked RFU shall be set to zero on transmission and ignored on receipt.

b0	b1	b2	b3	b4	b5	b6	b7
Type[0]	Type[1]	Type[2]	MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]

Table 3.1: Transport Layer Format

3.1 MESSAGE TYPES

This section describes the message formats for the logical coexistence signals. The message types are listed in Table 3.2.

Message Type Indicator	Direction	Message Type
0	MWS ↔ BT	Real Time Signal Message
1	MWS ↔ BT	Transport Control Message
2	MWS ↔ BT	Transparent Data Message
3	MWS → BT	MWS Inactivity Duration Message
	BT → MWS	RFU
4	MWS → BT	MWS Scan Frequency Message
	BT → MWS	RFU
5	MWS → BT	RFU
	BT→ MWS	RFU
6		Vendor Specific
7		Vendor Specific

Table 3.2: Message Types

The logical coexistence signals are listed in Table 3.3.

Logical Signal Name	Description
FRAME_SYNC	See Volume 7, Part A, Section 2.1.1

Table 3.3: Logical Coexistence Signals

Logical Signal Name	Description
MWS_RX	See Volume 7, Part A, Section 2.1.2
MWS_TX	See Volume 7, Part A, Section 2.1.6
PATTERN[1:0]	See Volume 7, Part A, Section 2.1.5
BLUETOOTH_RX_PRI	See Volume 7, Part A, Section 2.1.3
BLUETOOTH_TX_ON	See Volume 7, Part A, Section 2.1.4
802_RX_PRI	See Volume 7, Part A, Section 2.1.8
802_TX_ON	See Volume 7, Part A, Section 2.1.7
MWS_INACTIVITY_DURATION	See Volume 7, Part A, Section 2.1.9
MWS_SCAN_FREQUENCY_OFFSET	See Volume 7, Part A, Section 2.1.10

Table 3.3: Logical Coexistence Signals

3.1.1 Real Time Signaling Message (Type 0)

The Real Time Signaling message is used to transport the real time signals over the WCI-2 transport.

The Real Time Signaling message conveys all the real time signals (see Vol 7, Part A) in one message. The time reference point for real time signaling message is the end of MSG[4] (transition to Stop bit).

Two Real Time Signaling messages are defined, one from the Bluetooth Controller to the MWS device and another from the MWS device to the Bluetooth Controller.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
FRAME_SYNC	MWS_RX	MWS_TX	MWS_PATTERN[0]	MWS_PATTERN [1]

Table 3.4: Real Time Message from MWS Device to Bluetooth Controller

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
BLUETOOTH_RX_PRI	BLUETOOTH_TX_ON	802_RX_PRI	802_TX_ON	RFU

Table 3.5: Real Time Message from Bluetooth Controller to MWS Device

3.1.2 Transport Control Messages (Type 1)

The Transport Control messages can request state information of the MWS device's coexistence interface.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
RESEND_REAL_TIME	RFU	RFU	RFU	RFU

Table 3.6: Transport Control Message

Signal Name	Description
RESEND_REAL_TIME	This bit is set if a device wants to get a status update of the real time signals. The signal is usually used after wake-up from sleep of the transport interface. If the receiving device's transport interface is awake it shall send a Real Time Message with the current status of the real time signals within 4 UART character periods. If the signal is not received within 4 UART character periods the device is considered asleep.

Table 3.7: Transport Control Signals

3.1.3 Transparent Data Message (Type 2)

The Transparent Data message can be used to exchange non-time critical messages between the MWS device and the Bluetooth Controller. The interface does not guarantee the delivery of a message. Protocol and content of the message is vendor specific.

The least significant nibble of each octet shall be transmitted first.

A least significant nibble shall be discarded if the next nibble is a least significant nibble. A most significant nibble shall only be accepted if the preceding nibble was a least significant nibble.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
NIBBLE_ POSITION	DATA[0]/ DATA[4]	DATA[1]/ DATA[5]	DATA[2]/ DATA[6]	DATA[3]/ DATA[7]

Table 3.8: Transparent Data Message

Signal Name	Description
NIBBLE_POSITION	0 – Least Significant Nibble 1 – Most Significant Nibble
DATA[n]; n=0 .. 7	Data bits of the message octet

Table 3.9: Transparent Data Bits

3.1.4 MWS Inactivity Duration Message (Type 3)

The MWS Inactivity Duration message is used to send the MWS_INACTIVITY_DURATION signal from the MWS device to the Bluetooth Controller.

The message is sent at the beginning of the MWS inactivity period.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
DURATION[0]	DURATION[1]	DURATION[2]	DURATION[3]	DURATION[4]

Table 3.10: MWS Inactivity Duration Message

The MWS Inactivity Duration is encoded in 5 bits. DURATION is unsigned.

When DURATION = 0, MWS_INACTIVITY_DURATION is cancelled. When DURATION = 31, MWS_INACTIVITY_DURATION is infinite. Otherwise, MWS_INACTIVITY_DURATION is given by the formula:

MWS_INACTIVITY_DURATION = DURATION * 5 ms

3.1.5 MWS Scan Frequency Message (Type 4)

The MWS Scan Frequency message is used to send the MWS_SCAN_FREQUENCY signal from the MWS device to the Bluetooth Controller.

MSG[0]	MSG[1]	MSG[2]	MSG[3]	MSG[4]
FREQ[0]	FREQ[1]	FREQ[2]	FREQ[3]	FREQ[4]

Table 3.11: MWS Scan Frequency Message

The MWS Scan Frequency index is encoded in 5 bits. FREQ is unsigned.

LE ERRATA

CONTENTS

1 OVERVIEW

This part of CSA3 contains text changes derived from errata contained in Errata Service Release 5 (ESR05) and later approved errata for parts of the Bluetooth Core Specification noted in this document. All changes contained in this part of CSA 3 are mandatory for any applicable parts of the Core Specification claimed.

1.1 ESR05 ERRATA

Normative text for errata listed in Table 1.1 is in ESR05.

Errata	ESR05 Reference	Description
3813	5.3.2	LE piconet channel supported layers
3891	5.3.3	Typos in text referring to devices B and C incorrectly. Should be A and C
3892	5.3.4	Typos referring to devices H, I and J
3890	5.3.5	Typo in the text: used the term
3540	5.7.2	Advertiser filter policy text does not match Link Layer
3514	5.7.3	Missing reference in description
3815	5.7.4	BR/EDR/LE controller instead of BR/ED
3893	5.7.5	Incorrect reference; missing a Volume link
3895	5.7.6	Incorrect ref to Volume 2 (own volume) instead of Volume 6
3896	5.7.7	parameter Data[i]: is referring to the scan response data; and pointing to BR/EDR instead of LE
3879	5.7.8 and 5.8.1	Slave latency range inconsistency
3897	5.7.9	Inconsistency between the LE link layer and the HCI spec
3925	5.9.1	One of the parameters in Table 3.12 is missing starting range
3924	5.9.2	A typo at Section 3.3.1 Attribute PDU Format
4062	5.9.3	Sentence is not possible to ever be considered
3933	5.10.1	In Configured Broadcast, the case if BR/EDR link is not stated
3881	5.10.2	Clarify that BR-only GATT services will be reported in GATT Discover Primary Services executed on LE
3929	5.10.3	Service Changed Client Configuration Characteristic Descriptor
3915	5.10.4	Client characteristic configuration descriptor

Table 1.1: ESR05 Errata

Errata	ESR05 Reference	Description
3833	5.10.5	Service Changed
3932	5.10.6	Attribute Cashing
3821	5.10.7	Read Long Characteristic Values
3951	5.10.8	Clarify what occurs when a service changes and the Service Changed indication/notification is not enabled by the client
4120	5.10.9	PublicBrowseGroup should be PublicBrowseRoot
4067	5.10.10	Are prepared values written multiple times
3948	5.11.1	Reference to IETF RFC 4493 should be removed due to conflict in terminology with NIST 800-38B
4150	5.11.2	Typo in value field
3938	5.14.1	Device Address
3931	5.14.2	Missing Pause Encryption LLC requirements when Encryption is supported
3751	5.14.3	Is instant an absolute or relative time?
3636	5.14.4	Incorrect use of the NESN bit
3565	5.14.5	No spec for behavior when remote sends non-empty Data Channel Data PDU during encryption pause
3869	5.14.6	Typo ADV_DISCOVER-IND in footnote 1 for ADV_SCAN_IND
4149	5.14.7	Typo
4137	5.14.8	Inconsistent session key identifier
4092	5.14.9	In LL_CONNECTION_UPDATE_REQ, a typo minus sign is in front of connSupervisionTimeout.
4095	5.14.10	Clarify if the Link layer of the Master could initiate autonomously a channel map update
3904	5.14.11	When does the instant occur? Unclear when connEventCount is incremented? Beginning of new event or end of last event
3828	5.15.1	Incorrect parsed length field in the LL_START_ENC_REQ

Table 1.1: ESR05 Errata

1.2 ERRATA SINCE ESR05

Normative text for errata listed in the following table is in Section 2.

Errata	Part and Section Number	Description
4173	ATT 3.4.6.3	Errcode not clear enough & typo
4195	ATT 3.2.9	Typos
4229	ATT 3.4.5.4	Attribute Value size wrong in Table 3.29
4242	ATT 3.3.1	Clarification of authentication signature
4616	ATT 3.2.5	Encryption Permission
4063	GATT 4.4.1	Additional Ending Conditions
4065	GATT 4.4.1	Optimization for primary service discovery
4067	GATT 4.9.5	Are prepared values written multiple times
4100	GATT 9	Incorrect caption for Table 9.1 and missing BluetoothProfileDescriptorList
4119	GATT 9	PublicBrowseGroup should be PublicBrowseRoot
4169	GATT 11	Attribute Opcodes Supported shall be removed
4248	GATT 4.9.2	Specifies use of Write Command instead of Signed Write Command
4254	GATT 8.1	Discovery without security
4255	GATT 8.1	Error Code not suitable
4256	GATT 4.4	3rd Paragraph unclarity
4284	GATT 4.9.5	Conflict to ATT spec 3.4.6.1 for Error checking of Wrong Size or Invalid value for Prepare Write Request
4441	GATT 3.3.3.3	Clarify the requirement that a characteristic's property shall be set by the server before the client can configure indications, notifications or broadcasts
4163	SM 5.3.5.3	Figure 5.12 issue
4230	SM 2.3.5.5	typo
4238	SM 5.3.2.2	Figure not correct
4241	SM 2.4.5	Clarification of sample data
4249	SM 2.3.5.1	Clarification of STK generation
4407	SM 2.4.2.3	Slave store eDIV, Rand, LTK mapping is not necessary at all.
4418	SM 2.4.5	CSRK and signing need to be clarified
4648	SM 2.3.5.5	Incorrect ra value in Sconfirm calculation description
4481	PHY 1	Clarify Frequency Hopping
4217	LL 4.4.2.3	Connectable Undirected Event Type

Table 1.2: Post-ESR05 Errata

Errata	Part and Section Number	Description
4246	LL 5.1.2	Channel Map and connection update text difference
4664	LL 5.1.3.1	Paragraph 5.1.3.1 is not consistent with MSC's
4683	LL 5.1.3.1	Calculating session key diversifier and initialization vector
4212	HCI 7.8.2	Clarify of return parameters for implementations using separate buffers for LE
4215	HCI 4.5	LE connection complete event missing exception
4216	HCI 7.8.1	Typo
4310	HCI 7.7.65.3	Correction of range values
4311	HCI 6.27 HCI 7.8.26	Typo in command name
4397	HCI 7.8.5	Change ”non-directed” to ”undirected”
4644	HCI 7.8.12 HCI 7.8.18	Change ”connection” to ”connection event”
4101	HCI 7.1.6	Add another Reason code to Disconnect Command
4139	HCI 7.8.12 HCI 7.8.18	Correct Connection_Interval_Max value
3870	GAP 13.4	Missing specification text for Bonding Procedure
3928	GAP 2.3.1	Inheritance of security
4061	GAP 13.1.3.2	Missing space
4202	GAP 12.1 GAP 12.2 GAP 12.3 GAP 12.4 GAP 12.5	Typo and need to add permission
4205	GAP 9.1.2 GAP 10.7	It should be possible to use resolvable private addresses for active scanning
4206	GAP 9.1.1.2	The peer device cannot resolve the resolvable private address of a device in Broadcaster role, when privacy is enabled.
4233	GAP 9.3.9.2	Clarification text for central device when receiving a connection update req from slave

Table 1.2: Post-ESR05 Errata

Errata	Part and Section Number	Description
4208	GAP 9.3.4.2 GAP 9.3.4.3 GAP 9.3.4.4 GAP 9.3.4.5 GAP 9.3.4.6 GAP 9.3.4.7 GAP 9.3.4.8	Figure 9.4 flow chart is a copy paste mistake except for the first rectangular box that says
4359	GAP 10.4.2	Signing
4364	CSS V1, Section 1.1.1	Clarification of Service UUIDs contents
4638	GAP 13.2.4	Wrong reference of LE name discovery

Table 1.2: Post-ESR05 Errata

2 POST-ESR05 ERRATA TEXT

2.1 VOLUME 2 PART E: HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

2.1.1 Erratum 4310 - Typo in Supervision_Timeout parameter description in 7.7.65.3

Section: 4 7.7.65.3 LE Connection Update Complete Event, in the table titled "Supervision_Timeout:", page 804

[Original text in column "Parameter Description" states]

"Supervision timeout for this connection.

Range: 0x0006 to 0x000A

Time = N * 10 msec

Time Range: 100 msec to 32 msec."

[Replace with]

"Supervision timeout for this connection.

Range: 0x000A to 0x0C80

Time = N * 10 msec

Time Range: 100 msec to 32000 msec."

[End of changes for Erratum 4310]

2.1.2 Erratum 4215 - "LE Connection Complete event" is missing as exception in section "4.5 COMMAND ERROR HANDLING"

Section: 4.5 COMMAND ERROR HANDLING, 2nd paragraph, page 423

[Original text states]

"The above also applies to commands that have associated command specific completion events with a status parameter in their completion event, with three exceptions. The first two 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 third exception is the Logical Link Complete event. On failure, for this event, the second parameter (Logical_Link_Handle) is not valid but the third parameter (Physical_Link_Handle) is valid for identification purposes. The validity of other parameters is likewise implementation specific for failed commands in this group."

[Replace with]

"The above also applies to commands that have associated command specific completion events with a status parameter in their completion event, with four exceptions. The first two

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 third exception is the LE Connection Complete event. On failure, for this event, all other parameters are not valid. The fourth exception is the Logical Link completion event. On failure, for this event, the second parameter (Logical_Link_Handle) is not valid but the third parameter (Physical_Link_Handle) is valid for identification purposes. The validity of other parameters is likewise implementation specific for failed commands in this group, but they shall be sent in any case."

[End of changes for Erratum 4215]

2.1.3 Erratum 4216 - Typo in second paragraph

Section: 7.8.1 LE Set Event Mask Command, 2nd paragraph, page 806

[Original text states]

"For LE events to be generated, the LE Meta-Event bit in the Event_Mask shall also be set. If that bit is not set, then LE events not shall be generated, regardless of how the LE_Event_Mask is set."

[Replace with]

"For LE events to be generated, the LE Meta-Event bit in the Event_Mask shall also be set. If that bit is not set, then LE events shall not be generated, regardless of how the LE_Event_Mask is set."

[End of changes for Erratum 4216]

2.1.4 Erratum 4644 - Typo in parameter descriptions for Minimum_CE_Length and Maximum_CE_Length

Section: 7.8.12 LE Create Connection Command, Command Parameters Section, "Minimum_CE_Length", page 828

[Original table]

Minimum_CE_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Information parameter about the minimum length of connection needed for this LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

[Replace with]

Minimum_CE_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Information parameter about the minimum length of connection event needed for this LE connection. Range: 0x0000 to 0xFFFF Time = N * 0.625 msec.

Section: 7.8.12 LE Create Connection Command, Command Parameters Section, "Maximum_CE_Length", page 828

[Original table]

Maximum_CE_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Information parameter about the maximum length of connection needed for this LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

[Replace with]

Maximum_CE_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Information parameter about the maximum length of connection event needed for this LE connection. Range: 0x0000 to 0xFFFF Time = N * 0.625 msec.

Section: 7.8.18 LE Connection Update Command, Command Parameters Section, "Minimum_CE_Length", page 837

[Original table]

Value	Parameter Description
N = 0xXXXX	Information parameter about the minimum length of connection needed for this LE connection. How this value is used is outside the scope of this specification. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

[Replace with]

Minimum_CE_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Information parameter about the minimum length of connection event needed for this LE connection. How this value is used is outside the scope of this specification. Range: 0x0000 to 0xFFFF Time = N * 0.625 msec.

Section: 7.8.18 LE Connection Update Command, Command Parameters Section, "Maximum_CE_Length", page 837

[Original table]

Value	Parameter Description
N = 0xXXXX	Information parameter about the minimum length of connection needed for this LE connection. How this value is used is outside the scope of this specification. Range: 0x0000 – 0xFFFF Time = N * 0.625 msec.

[Replace with]

Maximum_CE_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Information parameter about the maximum length of connection event needed for this LE connection. How this value is used is outside the scope of this specification. Range: 0x0000 to 0xFFFF Time = N * 0.625 msec.

[End of changes for Erratum 4644]

2.1.5 Erratum 4311 –Typo

Section: 7.8.26 LE Long Term Key Request Negative Reply Command, 1st table, Column "Command", page 849

[Original text in Command column states]

"HCI_LE_Long_Term_Key_Requested_Negative_Reply"

[Replace with]

"HCI_LE_Long_Term_Key_Request_Negative_Reply"

Section: 6.27 Supported Commands, in table, Octet "28", Bit "2", column "Command Supported", page 454

[Original text]

"LE Long Term Key Requested Negative Reply"

[Replace with]

"LE Long Term Key Request Negative Reply"

[End of changes for Erratum 4311]

2.1.6 Erratum 4397 - Change non-directed to undirected

Section: 7.8.5 LE Set Advertising Parameters Command, subsection: "Command Parameters:" table "Advertising_Interval_Min", page 813

[Original text in "Parameter Description" column states]

"Minimum advertising interval for non-directed advertising."

[Replace with]

"Minimum advertising interval for undirected advertising."

Section: 7.8.5 LE Set Advertising Parameters Command, subsection: "Command Parameters:" table "Advertising_Interval_Max", page 813

[Original text in "Parameter Description" column states]

"Maximum advertising interval for non-directed advertising."

[Replace with]

"Maximum advertising interval for undirected advertising."

[End of changes for Erratum 4397]

2.1.7 Erratum 4139 – Supervision_Timeout min value

Section: 7.8.12 LE Create Connection Command, in paragraph labeled "Description:", bottom of page 825

[Original text states]

"The Supervision_Timeout parameter defines the link supervision timeout for the connection. The Supervision_Timeout in milliseconds shall be larger than the Conn_Interval_Max in milliseconds. (See [Vol 6] Part B, Section 4.5.2)."

[Replace with]

"The Supervision_Timeout parameter defines the link supervision timeout for the connection. The Supervision_Timeout in milliseconds shall be larger than (1 + Conn_Latency) * Conn_Interval_Max * 2, where Conn_Interval_Max is given in milliseconds. (See [Vol 6] Part B, Section 4.5.2.)"

Section: 7.8.18 LE Connection Update Command, 4th paragraph labeled "Description:", page 835

[Original text states]

"The Supervision_Timeout parameter shall define the link supervision timeout for the LE link. The Supervision_Timeout shall be larger than the Conn_Interval_Max."

[Replace with]

"The Supervision_Timeout parameter shall define the link supervision timeout for the LE link. The Supervision_Timeout in milliseconds shall be larger than (1 + Conn_Latency) * Conn_Interval_Max * 2, where Conn_Interval_Max is given in milliseconds. (See [Vol 6] Part B, Section 4.5.2.)"

[End of changes for Erratum 4139]

2.1.8 Erratum 4101 – Disc reason Unacceptable Connection Interval not allowed in Disconnect command

Section: 7.1.6 Disconnect Command, in table labeled "Reason:", page 469

[Original table]

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 Part D, Error Codes on page 339 for a list of error codes and descriptions.

[Replace with]

Reason: *Size: 1 Octet*

Value	Parameter Description
0x05, 0x13-0x15, 0x1A, 0x29, 0x3B	Authentication Failure error code (0x05), Other End Terminated Connection error codes (0x13-0x15), Unsupported Remote Feature error code (0x1A), Pairing with Unit Key Not Supported error code (0x29) and Unacceptable Connection Interval error code (0x3B), see Part D, Error Codes on page 339 for list of error codes and descriptions.

[End of changes for Erratum 4101]

2.1.9 Erratum 4212 – HC_Total_Num_LE_ACL_Data_Packets not clear for zero value

Section: 7.8.2 LE Read Buffer Size Command, 3rd paragraph, page 808

[Original text states]

"Note: Both the Read_Buffer_Size and LE_Read_Buffer_Size commands may return buffer length parameter values that are non-zero. This allows a Controller to offer different sized buffers for BR/EDR data packets and LE data packets."

[Replace with]

"Note: Both the Read_Buffer_Size and LE_Read_Buffer_Size commands may return buffer length and number of packets parameter values that are non-zero. This allows a Controller to offer different buffers and number of buffers for BR/EDR data packets and LE data packets."

[End of changes for Erratum 4212]

2.2 VOLUME 3 PART C: GENERIC ACCESS PROFILE

2.2.1 Erratum 4359 - Authenticated signed procedure

Section: 10.4.2 Authenticate Signed Data Procedure, 1st paragraph, page 369

[Original text states]

"…peer device. If the SignCounter was previously used then the receiving device shall ignore the Data PDU and respond with error code "Signature Authentication Failed", if permitted. The receiving device should protect from repeated failures by implementing a timer similar to the pairing timer as defined in [Vol.3], Part H Section 2.3.6."

[Replace with]

"…peer device. If the SignCounter was previously used then the receiving device shall ignore the Data PDU."

[End of changes for Erratum 4359]

2.2.2 Erratum 4233 - Clarification text for central device when receiving a connection update request from slave

Section: 9.3.9.2 Conditions 3rd paragraph, page 358

[Original text states]

"A Peripheral initiating the connection parameter update procedure shall use the L2CAP Connection Parameter Update Request command defined in [Vol.1], Part A Section 4.2 with the required connection parameters. The Peripheral shall not send an L2CAP Connection Parameter Update Request command within T_{GAP}(conn_param_timeout) of an L2CAP Connection Parameter Update Response being received. If the updated connection parameters are unacceptable to the Peripheral then it may disconnect the connection with the

error code «Unacceptable Connection Interval». Peripherals should be tolerant of connection intervals given to it from the Central."

[Replace with]

"A Peripheral initiating the connection parameter update procedure shall use the L2CAP Connection Parameter Update Request command defined in [Vol.1], Part A Section 4.2 with the required connection parameters. The Peripheral shall not send an L2CAP Connection Parameter Update Request command within T_{GAP}(conn_param_timeout) of an L2CAP Connection Parameter Update Response being received. If the updated connection parameters are unacceptable to the Peripheral then it may disconnect the connection with the error code «Unacceptable Connection Interval». Peripherals should be tolerant of connection intervals given to it from the Central. When the Central accepts the Peripheral initiated Connection Parameter Update, the Central should initiate the Link Layer Connection Update procedure defined in Vol 6, Part B Section 5.1.1 with the required connection parameters within T_{GAP}(conn_param_timeout)."

[End of changes for Erratum 4233]

2.2.3 Erratum 4061- Missing space

Section: 13.1.3.2 Bondable Mode, page 385

[Original text states]

"For a device that is operating in the LE Central role, a device in the bondable mode shall follow the requirements for the bondable mode for BR/EDR as defined inSection 4.3.2 and it shall follow the requirements for the bondable mode for LE as defined in Section 9.4.3."

[Replace with]

"For a device that is operating in the LE Central role, a device in the bondable mode shall follow the requirements for the bondable mode for BR/EDR as defined in Section 4.3.2 and it shall follow the requirements for the bondable mode for LE as defined in Section 9.4.3."

[End of changes for Erratum 4061]

2.2.4 Erratum 4202 - Typo and need to add permission

Section: 12.1 Device Name Characteristic, last sentence of 1st paragraph, page 379

[Original text states]

"The Device Name characteristic shall contain the name of the device as an UTF-8 string as defined in Section 3.2.2. The Device Name characteristic value shall be readable without authentication or authorization. The Device Name characteristic value may be writable."

[Replace with]

"The Device Name characteristic shall contain the name of the device as an UTF-8 string as defined in Section 3.2.2. The Device Name characteristic value shall be readable without authentication or authorization. The Device Name characteristic value may be writable. If

writeable, authentication and authorization may be defined by a higher layer specification or is implementation specific."

Section: 12.1 Device Name Characteristic, Table 12.2, page 379

[Original table]

Attribute Handle	Attribute Type	Attribute Value
0xMMMM	0x2A00 – UUID for «Device Name»	Device Name

Table 12.2: Device Name characteristic

[Add new column to end of Table 12.2]

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xMMMM	0x2A00 – UUID for «Device Name»	Device Name	Readable without authentication or authorization Optionally writeable, authentication and authorization may be defined by a higher layer specification or is implementation specific

Section: 12.2 Appearance Characteristic, last sentence of 1st paragraph, page 380

[Original text states]

"…The Appearance characteristic value shall be readable without authentication or authorization. The Appearance characteristic value may be writable."

[Replace with]

"…The Appearance characteristic value shall be readable without authentication or authorization. The Appearance characteristic value may be writable. If writeable, authentication and authorization may be defined by a higher layer specification or is implementation specific."

Section: 12.2 Appearance Characteristic, Table 12.3, page 380

[Original table]

Attribute Handle	Attribute Type	Attribute Value
0xMMMM	0x2A01 – UUID for «Appearance»	Appearance

Table 12.3: Appearance characteristic

[Add new column to end of Table 12.3]

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xMMMM	0x2A01 – UUID for «Appearance»	Appearance	Readable without authentication or authorization Optionally writeable, authentication and authorization may be defined by a higher layer specification or is implementation specific

Section: 12.3 Peripheral Privacy Flag Characteristic, end of 1st paragraph, page 380

[Original text]

"The Peripheral Privacy Flag characteristic defines whether privacy is currently in use within this device. See Table 10.7.2."

[Replace with]

"The Peripheral Privacy Flag characteristic defines whether privacy is currently in use within this device. See Table 10.7.2.
The Privacy Flag characteristic shall be readable. Authentication and authorization may be defined by a higher layer specification or is implementation specific.

The Privacy Flag characteristic may be writable and shall require authentication. Authorization may be defined by a higher layer specification or is implementation specific."

Section: 12.3 Peripheral Privacy Flag Characteristic, Table 12.4, page 380

[Original table]

Attribute Handle	Attribute Type	Attribute Value
0xMMMM	0x2A02 – UUID for «Peripheral Privacy Flag»	Peripheral Privacy Flag

Table 12.4: Peripheral Privacy Flag characteristic

[Add new column to end of Table 12.4]

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xMMMM	0x2A02 – UUID for «Peripheral Privacy Flag»	Peripheral Privacy Flag	Readable, authentication and authorization may be defined by a higher layer specification or is implementation specific Optionally writeable with authentication. Authorization may be defined by a higher layer specification or is implementation specific

Section: 12.4 Reconnection Address Characteristic, end of 1st paragraph, page 380

[Original text]

“The Reconnection Address characteristic defines the reconnection address. See Table 10.7.2. The reconnection address is a non-resolvable private address.”

[Replace with]

“The Reconnection Address characteristic defines the reconnection address. See Table 10.7.2. The reconnection address is a non-resolvable private address.

The Reconnection Address characteristic value shall not be readable.

The Reconnection Address characteristic value shall be writable and shall require authentication. Authorization may be defined by a higher layer specification or is implementation specific.”

Section: 12.4 Reconnection Address Characteristic, Table 12.5, page 381

[Original table]

Attribute Handle	Attribute Type	Attribute Value
0xMMMM	0x2A03 – UUID for «Recon-nection Address»	Reconnection Address

Table 12.5: Reconnection Address characteristic

[Add new column to end of Table 12.5]

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xMMMM	0x2A03 – UUID for «Reconnection Address»	Reconnection Address	Not Readable Writeable with authentication. Authorization may be defined by a higher layer specification or is implementation specific

Section: 12.5 Peripheral Preferred Connection Parameters Characteristic, end of 1st paragraph, page 381

[Original text]

“The Peripheral Preferred Connection Parameters (PPCP) characteristic contains the preferred connection parameters of the Peripheral.”

[Replace with]

“The Peripheral Preferred Connection Parameters (PPCP) characteristic contains the preferred connection parameters of the Peripheral.

The Peripheral Preferred Connection Parameters characteristic value shall be readable. Authentication and authorization may be defined by a higher layer specification or is implementation specific.

The Peripheral Preferred Connection Parameters characteristic value shall not be writable.”

Section: 12.5 Peripheral Preferred Connection Parameters Characteristic, Table 12.6, page 381

[Original table]

Attribute Handle	Attribute Type	Attribute Value
0xMMMM	0x0x2A04 – UUID for «Peripheral Preferred Connection Parameters»	Peripheral Preferred Connection Parameter

Table 12.6: Peripheral Preferred Connection Parameters characteristic

[Add new column to end of Table 12.6 and correct typo under "Attribute Type"]

Attribute Handle	Attribute Type	Attribute Value	Attribute Permissions
0xMMMM	0x2A04 – UUID for «Peripheral Preferred Connection Parameters»	Peripheral Preferred Connection Parameter	Readable, authentication and authorization may be defined by a higher layer specification or is implementation specific Not writeable

[End of changes for Erratum 4202]

2.2.5 Erratum 4205 - It should be possible to use resolvable private addresses for active scanning.

Section: 9.1.2 Observation Procedure, 3rd paragraph, page 337

[Original text states]

"A privacy enabled device as defined in Section 10.7 performing the observation procedure shall use a non-resolvable private address as the device address when performing active scanning. The generation of a non-resolvable address is defined in Section 10.8.2.1."

[Replace with]

"A privacy enabled device as defined in Section 10.7 performing the observation procedure shall use either a non-resolvable or resolvable private address as the device address when performing active scanning. The generation of a non-resolvable address is defined in Section 10.8.2.1. The generation of a resolvable private address is defined in Section 10.8.2.2.
Note: It is not expected that a Broadcaster will resolve resolvable private addresses."

Section: 10.7 Privacy Feature, Table 10.2, Column "Observer", Row "Resolvable private address generation procedure", page 369

[Original text states]

“E”

[Replace with]

“C4”

Section: 10.7 Privacy Feature, Table 10.2, Column “Central”, Row “Resolvable private address generation procedure”, page 369

[Original text states]

“E”

[Replace with]

“C4”

Section: 10.7 Privacy Feature, Table 10.2, page 369

[Original text states]

“C1: Mandatory if privacy feature and Active scanning are supported, else excluded.”

[Replace with]

“C1: Mandatory if privacy feature and active scanning are supported and resolvable private address generation procedure is not supported, else excluded.”

Section: 10.7 Privacy Feature, Table 10.2, page 369

[Append text After C3 in the last row of Table 10.2]

“C4: Mandatory if privacy feature and active scanning are supported and non-resolvable private address generation procedure is not supported, else excluded.”

[End of changes for Erratum 4205]

2.2.6 Erratum 4206 - The peer device cannot resolve the resolvable private address of a device in Broadcaster role, when privacy is enabled.

Section: 9.1.1.2 Conditions, 2nd Note, page 336

[Original text states]

“Note: A device that only supports the broadcast mode cannot support sending encrypted or signed data because the device cannot support the distribution of keys as defined by the Security Manager (see [Vol. 3], Part H). Key distribution may be performed by a device using the Peripheral or Central profile role.”

[Replace with]

“Note: A device that only supports the broadcast mode cannot support sending encrypted or signed data because the device cannot support the distribution of keys as defined by the

Security Manager (see [Vol. 3], Part H). A device may distribute the IRK using an out of band mechanism that is not defined by the Bluetooth SIG (e.g. using NFC).”

[End of changes for Erratum 4206]

2.2.7 Erratum 3870 - Missing specification text for Bonding Procedure

Section: 13 BR/EDR/LE Operation Modes and Procedure, page 389

[Add a new section after section 13.4]

“**13.5 BONDING FOR BR/EDR/LE DEVICE TYPE**

The requirements for a device supporting BR/EDR/LE device type are shown in Table 13.4.

Bonding Requirement	Ref.	Peripheral	Central
Bonding	6.5/9.4.4	M	M

Table 13.4: Requirements for the bonding of a BR/EDR/LE device type

If the remote device supports the BR/EDR/LE device type or the BR/EDR device type, the bonding procedures as defined in Section 6.5 shall be used.

If the remote device is of the LE-only device type, the bonding procedures as defined in Section 9.4.4 shall be used.

Note: The device type of the remote device can be discovered using the device type discovery procedure as defined in Table 13.2.3.1.”

[End of changes for Erratum 3870]

2.2.8 Erratum 4100 – Incorrect caption for Table 9.1 and missing BluetoothProfileDescriptorList

Section: 12 GAP Characteristics for Low Energy, first sentence, page 379

[Original text states]

“The service UUID shall be the UUID for «Generic Access Profile.»”

[Replace with]

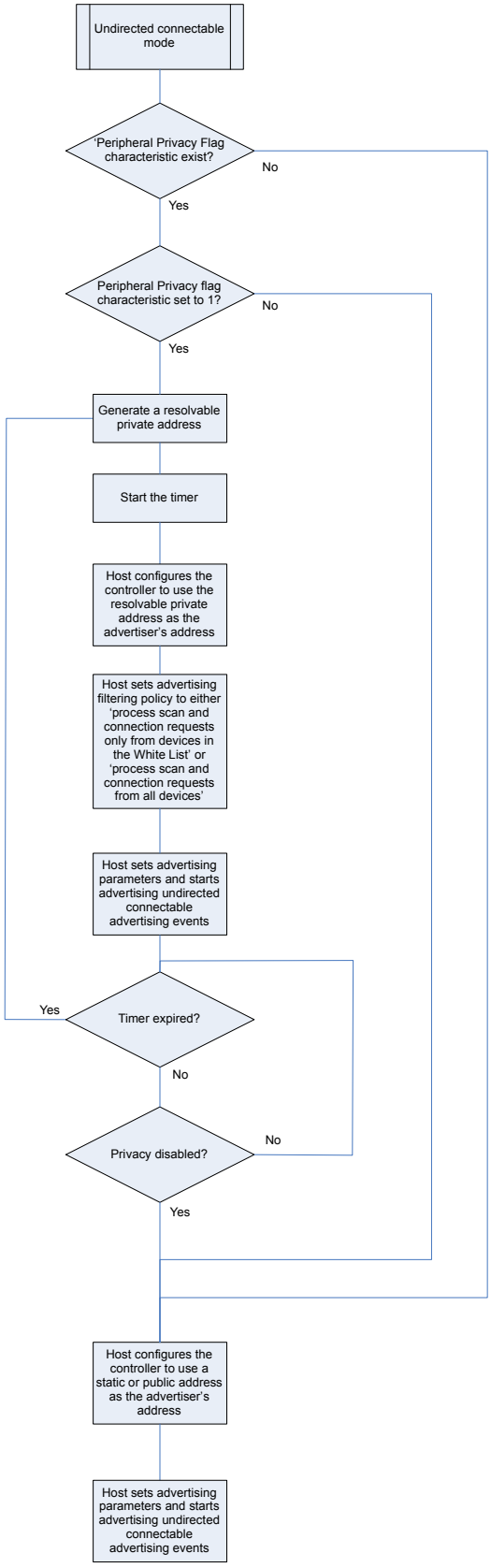
“The service UUID shall be the UUID for «GAP Service».”

[End of changes for Erratum 4100]

2.2.9 Erratum 4208 - Figure 9.4 flow chart is a copy paste mistake except for the first rectangular box that says

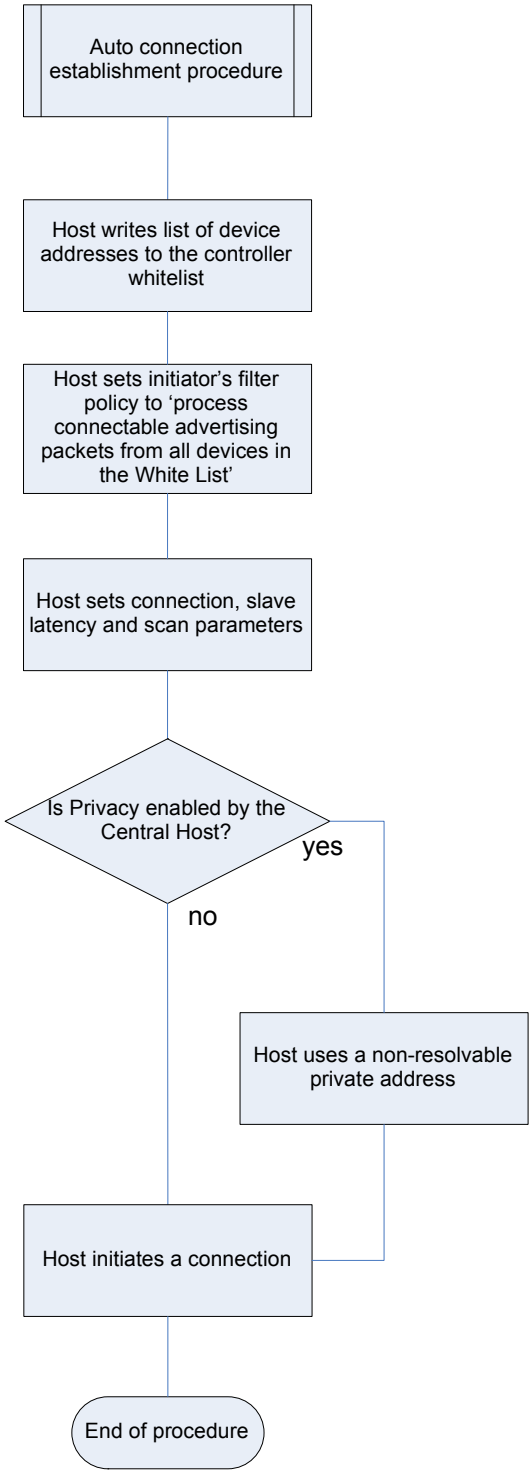
Section: 9.3.4.2 Conditions, Figure 9.4, page 350

[Replace Figure 9.4 with the following figure]


Undirected connectable mode
'Peripheral Privacy Flag characteristic exist?
No
Yes
Peripheral Privacy flag characteristic set to 1?
No
Yes
Generate a resolvable private address
Start the timer
Host configures the controller to use the resolvable private address as the advertiser's address
Host sets advertising filtering policy to either 'process scan and connection requests only from devices in the White List' or 'process scan and connection requests from all devices'
Host sets advertising parameters and starts advertising undirected connectable advertising events
Yes
Timer expired?
No
Privacy disabled?
No
Yes
Host configures the controller to use a static or public address as the advertiser's address
Host sets advertising parameters and starts advertising undirected connectable advertising events

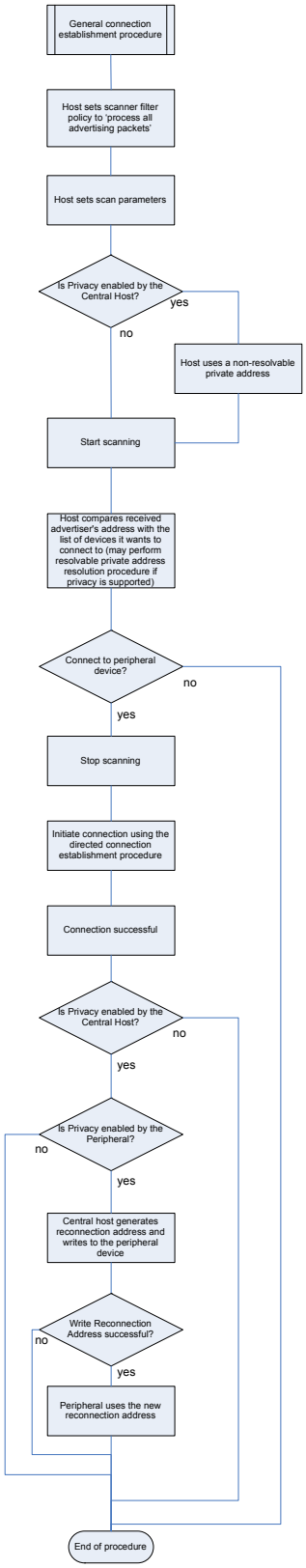
Section: 9.3.5.2 Conditions, Figure 9.5, page 353

[Replace Figure 9.5 with the following figure]



Section: 9.3.6.2 Conditions, Figure 9.6, page 355

[*Replace Figure 9.6 with the following figure]*



Section: 9.3.7.2 Conditions, page 357

[Insert Figure after 1st paragraph]

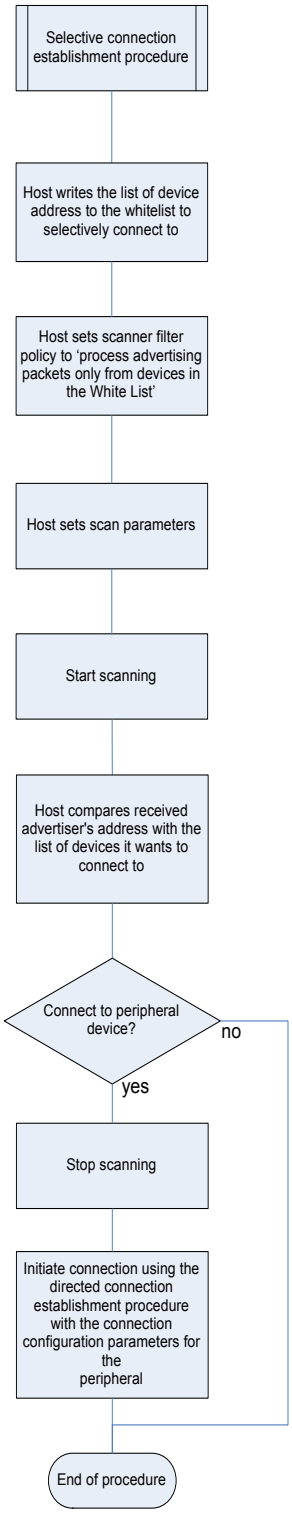


Figure 9.7: Flow chart for a device performing the selective connection establishment procedure

Section: 9.3.8.2 Conditions, page 357

[Insert Figure after 1st paragraph]

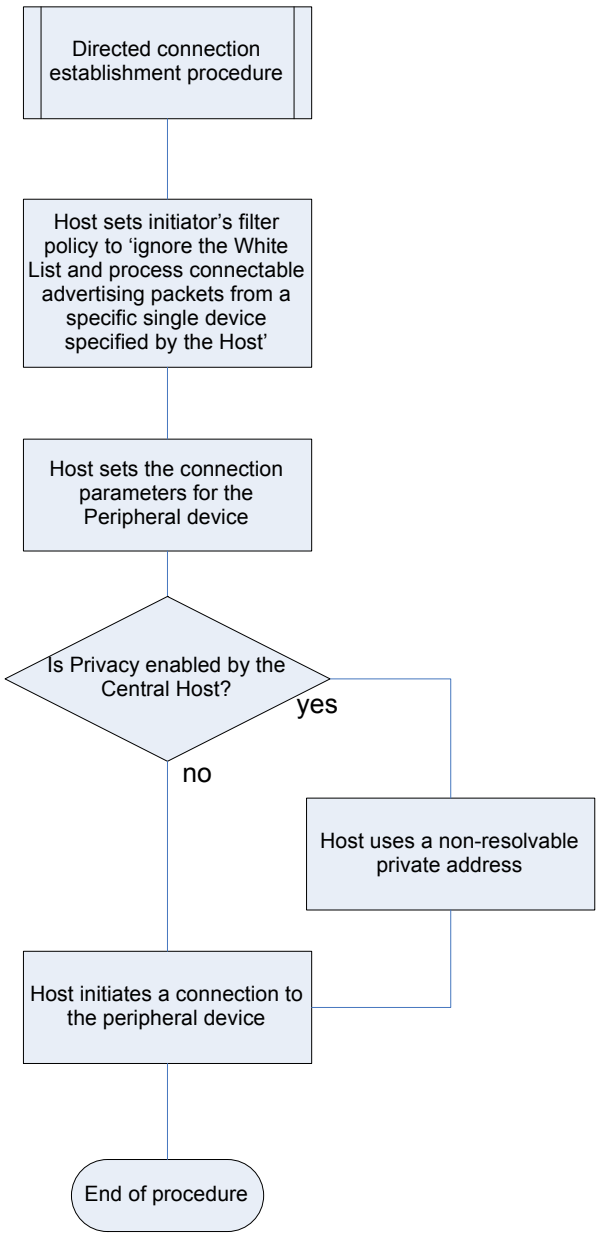


Figure 9.8: Flow chart for a device performing the directed connection establishment procedure

Section: 9.4.4.1 Description, page 360

[Original Figure number states]

"Figure 9.7: The Bonding procedure"

[Replace with]

"Figure 9.9: The Bonding procedure"

[End of changes for Erratum 4208]

2.2.10 Erratum 4364 - Clarification of Service UUIDs contents

Section: 11.1.1 Service UUIDs, page 376

[Append text to end of section 11.1.1]

"GAP and GATT service UUIDs should not be included in a Service UUIDs AD type, for either a complete or incomplete list."

[End of changes for Erratum 4364]

2.2.11 Erratum 4638 - Wrong reference of LE name discovery

Section: 13.2.4 Name Discovery, bottom of page 387

[Original text states]

"...procedure as defined in Section 13.2.4"

[Replace with]

"...procedure as defined in Section 9.2.7"

[End of changes for Erratum 4638]

2.3 VOLUME 3 PART F ATTRIBUTE PROTOCOL (ATT)

2.3.1 Erratum 4229 - Attribute Value size wrong in Table 3.29

Section: 3.4.5.4 Signed Write Command, Table 3.29, Row "Attribute Value", page 504

[Original text states]

"0 to (ATT_MTU - 13)"

[Replace with]

"0 to (ATT_MTU - 15)"

[End of changes for Erratum 4229]

2.3.2 Erratum 4242 - Clarification of authentication signature

Section: 3.3.1 Attribute PDU Format, page 481

[Original text states]

"The Authentication Signature field is calculated as defined in Security Manager (see [Vol. 3] Part H, Section 2.4.5). This value provides an Authentication Signature for the following values in this order: Attribute Opcode, Attribute Parameters."

[Replace with]

"The Authentication Signature field is calculated as defined in Security Manager (see [Vol. 3] Part H, Section 2.4.5). This value provides an Authentication Signature for the variable length message (m) consisting of the following values in this order: Attribute Opcode, Attribute Parameters."

[End of changes for Erratum 4242]

2.3.3 Erratum 4616 - Encryption Permissions

Section: 3.2.5 Attribute Permissions, 1st paragraph, page 478

[Original text states]

"Attribute permissions are a combination of access permissions, authentication permissions and authorization permissions."

[Replace with]

"Attribute permissions are a combination of access permissions, encryption permissions, authentication permissions and authorization permissions."

Section: 3.2.5 Attribute Permissions, 3rd paragraph, page 478

[Original text states]

"The following authentication permissions are possible:

• Authentication Required
• No Authentication Required"

[Replace with]

"The following authentication permissions are possible:

• Authentication Required
• No Authentication Required"

"The following encryption permissions are possible:

• Encryption required
• No encryption required"

[End of changes for Erratum 4616]

2.3.4 Erratum 4195 - Typos

Section: 3.3 Attribute PDU, 3rd bullet, page 480

[Original text states]

- Commands—PDUssent to a server by a client.

[Replace with]

- Commands—PDUs sent to a server by a client.

[End of changes for Erratum 4195]

2.3.5 Erratum 4173 - Errcode not clear enough & typo

Section: 3.4.6.3 Execute Write Response, page 508

[Delete duplicate paragraph]

"If the prepared Attribute Value exceeds the maximum valid length of the attribute value then all pending prepare write values shall be discarded for this client, the queue shall then be cleared, and an Error Response shall be sent with the error code «Invalid Attribute Value Length»."

[End of changes for Erratum 4173]

2.4 VOLUME 3 PART G - GENERIC ATTRIBUTE PROFILE (GATT)

2.4.1 Erratum 4255 – Error code not suitable

Section: 8.1 Authentication Requirements, 3rd paragraph, page 586

[Original text states]

"Over LE, if such a request is issued when the physical link is unauthenticated or unencrypted, the server shall send an *Error Response* with the status code set to *Insufficient Authentication*. The client wanting to read or write this characteristic can then request that the physical link be authenticated using the GAP authentication procedure, and once this has been completed, send the request again."

[Replace with]

"Over LE, if such a request is issued when the physical link is unauthenticated or unencrypted, the server shall send an *Error Response*. The client wanting to read or write this characteristic can then request that the physical link be authenticated using the GAP authentication procedure, and once this has been completed, send the request again."

Section: 8.1 Authentication Requirements, 4th paragraph, page 586

[Original text states]

"Over BR/EDR, if such a request is issued when the physical link is unauthenticated or unencrypted, the server can either send an Error Response with the status code set to Insufficient Authentication or perform the GAP authentication procedure and then upon completion of that procedure send the appropriate response. If the client receives this Error Response it can then request that the physical link be authenticated using the GAP authentication procedure, and once this has been completed send the request again."

[Replace with]

"Over BR/EDR, if such a request is issued when the physical link is unauthenticated or unencrypted, the server can either send an *Error Response* or perform the GAP authentication procedure and then upon completion of that procedure send the appropriate response. If the client receives this *Error Response* it can then request that the physical link be authenticated using the GAP authentication procedure, and once this has been completed send the request again."

[End of changes for Erratum 4255]

2.4.2 Erratum 4256 – 3rd paragraph un-clarity

Section: 4.4 Primary Service Discovery, 3rd paragraph, page 551

[Original text states]

"GATT service discovery over BR/EDR transport will list services that only run over an LE transport and therefore only SDP service discovery shall be used on BR/EDR."

[Replace with]

"GATT service discovery over the BR/EDR transport will list all services independent of the transport over which the services run. In order to discover the services that run over BR/EDR, SDP service discovery shall be used."

[End of changes for Erratum 4256]

2.4.3 Erratum 4254 – Discovery without security

Section: 8.1 Authentication Requirements, 5th paragraph, page 586

[Original text states]

"…This implies that an *Insufficient Authentication* status code shall not be used in an *Error Response* for a *Read Information Request*."

[Replace with]

"…This implies that an *Insufficient Authentication* status code shall not be used in an *Error Response* for a *Find Information Request*."

[End of changes for Erratum 4254]

2.4.4 Erratum – 4284 Conflict to ATT spec 3.4.6.1 for Error checking of Wrong Size or Invalid value for Prepare Write Request

Section: 4.9.5 Reliable Writes, 3rd paragraph, page 568

[Delete text]

"…If the *Characteristic Value* that is written is the wrong size, or has an invalid value as defined by the profile, then the write shall not succeed, and an *Error Response* with the *Error Code* set to *Application Erro*r shall be sent by the server."

Section: 4.9.5 Reliable Writes, 7th paragraph, page 568

[Original text states]

"In the second phase, the Attribute Protocol *Execute Write Request* is used. The Attribute Flags parameter shall be set to 0x01 to immediately write all pending prepared values in the order that they were prepared. The server shall write the prepared writes once it receives this request and shall only send the *Execute Write Response* once all the prepared values have been successfully written. If an application error occurs while writing these Attributes the server shall instead send the *Error Response* with an *Error Code* set to *Application Error* by the server. The state of the Characteristic Values that were prepared is undefined."

[Replace with]

"In the second phase, the Attribute Protocol *Execute Write Request* is used. The Attribute Flags parameter shall be set to 0x01 to immediately write all pending prepared values in the order that they were prepared. The server shall write the prepared writes once it receives this request and shall only send the *Execute Write Response* once all the prepared values have been successfully written. If the *Characteristic Value* that is written is the wrong size, or has an invalid value as defined by the profile, then the write shall not succeed, and an *Error Response* with the *Error Code* set to «Application Error» shall be sent by the server. The state of the characteristic values that were prepared is undefined."

[End of changes for Erratum 4284]

2.4.5 Erratum – 4441 Clarify the requirement that a characteristic's property shall be set by the server before the client can configure indications, notifications or broadcasts.

Section: 3.3.3.3 Client Characteristic Configuration, table 3.11, page 542

[Original text in row "Notification" column "Description"]

"The Characteristic Value shall be notified."

[Replace with]

"The Characteristic Value shall be notified. This value can only be set if the characteristic's property has the notify bit set."

Section: 3.3.3.3 Client Characteristic Configuration, table 3.11, page 542

[Original text in row "Indication" column "Description"]

"The Characteristic Value shall be indicated."

[Replace with]

"The *Characteristic Value* shall be indicated. This value can only be set if the characteristic's property has the indicate bit set."

Section: 3.3.3.3 Client Characteristic Configuration, table 3.13, page 543

[Original text in row "Broadcast" column "Description"]

"The *Characteristic Value* shall be broadcast when the server is in the broadcast procedure if advertising data resources are available."

[Replace with]

"The *Characteristic Value* shall be broadcast when the server is in the broadcast procedure if advertising data resources are available. This value can only be set if the characteristic's property has the broadcast bit set."

[End of changes for Erratum 4441]

2.4.6 Erratum 4248 – Specifies use of Write Command instead of Signed Write Command

Section: 4.9.2 Signed Write Without Response, 3rd paragraph, page 564

[Original text states]

"The Attribute Protocol *Write Command* is used for this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle*. The *Attribute Value* parameter shall be set to the new *Characteristic Value* authenticated by signing the value, as defined in the Security Manager [Vol. 3] Part H, Section 2.4.5."

[Replace with]

"The Attribute Protocol *Signed Write Command* is used for this sub-procedure. The *Attribute Handle* parameter shall be set to the *Characteristic Value Handle*. The *Attribute Value* parameter shall be set to the new *Characteristic Value* authenticated by signing the value, as defined in the Security Manager [Vol. 3] Part H, Section 2.4.5."

[End of changes for Erratum 4248]

2.4.7 Erratum 4063 – Additional Ending Condition

Section: 4.4.1 Discover All Primary Services, 2nd paragraph, page 552

[Original text states]

"The sub-procedure is complete when the Error Response is received and the Error Code is set to Attribute Not Found."

[Replace with]

"This sub-procedure is complete when the *Error Response* is received and the *Error Code* is set to «Attribute Not Found» or when the *End Group Handle* in the *Read by Type Group Response* is 0xFFFF."

Section: 4.4.2 Discover Primary Service by Service UUID, 6th paragraph, page 553

[Original text states]

"The sub-procedure is complete when the Error Response is received and the Error Code is set to Attribute Not Found."

[Replace with]

"This sub-procedure is complete when the *Error Response* is received and the *Error Code* is set to «Attribute Not Found» or when the *End Group Handle* in the *Read by Type Group Response* is 0xFFFF."

[End of changes for Erratum 4063]

2.4.8 Erratum 4169 – Attribute Opcodes Supported shall be removed

Section: 11 APPENDIX: EXAMPLE ATTRIBUTE SERVER ATTRIBUTES, Table 11.1, page 590

[Original text in row 6 "0x0011" column "Attribute Value" states]

"{0x02, 0x0012, «Attribute Opcodes Supported»}"

[Replace with]

"{0x26, 0x0012, «Service Changed»}"

Section: 11 APPENDIX: EXAMPLE ATTRIBUTE SERVER ATTRIBUTES, Table 11.1, page 590

[Original text in row 7 "0x0012" column "Attribute Type" states]

"«Attribute Opcodes Supported»"

[Replace with]

"«Service Changed»"

Section: 11 APPENDIX: EXAMPLE ATTRIBUTE SERVER ATTRIBUTES, Table 11.1, page 590

[Original text in row 7 "0x0012" column "Attribute Value" states]

"0x01FF"

[Replace with]

"0x0000, 0x0000"

[End of changes for Erratum 4169]

2.4.9 Erratum 4100 – Incorrect caption for Table 9.1 and missing BluetoothProfileDescriptorList

Section: 9 SDP Interoperability Requirements, table 9.1, page 588

[Original text in row 3 "Service Class #0" column "Value" states]

"Generic Attribute Profile"

[Replace with]

"GATT Service"

Section: 9 SDP Interoperability Requirements, table 9.1, page 588

[Original text in Table 9.1 caption states]

"*Table 9.1: SDP Record for the Generic Access Profile*"

[Replace with]

"Table 9.1: SDP Record for the Generic Attribute Profile"

Section: 7 DEFINED GENERIC ATTRIBUTE PROFILE SERVICE, page 584

[Original text states]

"All characteristics defined within this section shall be contained in a primary service with the service UUID set to «Generic Attribute Profile» as defined in Section 3.1. Only one instance of the GATT service shall be exposed on a GATT server."

[Replace with]

"All characteristics defined within this section shall be contained in a primary service with the service UUID set to «GATT Service» as defined in Section 3.1. Only one instance of the GATT service shall be exposed on a GATT server."

[End of changes for Erratum 4100]

2.4.10 Erratum 4119 – PublicBrowseGroup should be PublicBrowseRoot

Section: 9 SDP Interoperability Requirements, table 9.1 page 588

[Original text in row 10 "BrowseGroupList" column "Value" states]

"PublicBrowseGroup"

[Replace with]

"PublicBrowseRoot"

Section: 9 SDP Interoperability Requirements, table 9.1 page 588

[Original text in row 10 "BrowseGroupList" column "Item" states]

"BrowseGroupList["

[Replace with]

"BrowseGroupList"

[End of changes for Erratum 4119]

2.4.11 Erratum 4065 - Optimization for primary service discovery

Section: 4.4.1 Optimization for primary service discovery, 1st paragraph, page 552

[Original text states]

"*Read By Group Type Response* returns a list of *Attribute Handle*, *End Group Handle,* and *Attribute Value* tuples corresponding to the services supported by the server. Each *Attribute Value* contained in the response is the Service UUID of a service supported by the server. The *Attribute Handle* is the handle for the service declaration. The *End Group Handle* is the handle of the last attribute within the service definition. The *Read By Group Type Request* shall be called again with the *Starting Handle* set to one greater than the last *End Group Handle* in the *Read By Group Type Response*."

[Replace with]

"*Read By Group Type Response* returns a list of *Attribute Handle*, *End Group Handle,* and *Attribute Value* tuples corresponding to the services supported by the server. Each *Attribute Value* contained in the response is the Service UUID of a service supported by the server. The *Attribute Handle* is the handle for the service declaration. The *End Group Handle* is the handle of the last attribute within the service definition. The *End Group Handle* of the last service in a device can be 0xFFFF. The *Read By Group Type Request* shall be called again with the *Starting Handle* set to one greater than the last *End Group Handle* in the *Read By Group Type Response*."

Section: 4.4.2 Discover Primary Service by Service UUID, 5th paragraph, page 553

[Original text states]

"*Find By Type Value Response* returns a list of *Attribute Handle* ranges. The *Attribute Handle* range is the starting handle and the ending handle of the service definition. If the *Attribute Handle* range for the Service UUID being searched is returned and the End Found Handle is not 0xFFFF, the *Find By*
Type Value Request may be called again with the *Starting Handle* set to one greater than the last *Attribute Handle* range in the Find By Type Value Response."

[Replace with]

"*Find By Type Value Response* returns a list of *Attribute Handle* ranges. The *Attribute Handle* range is the starting handle and the ending handle of the service definition. The *End Group Handle* of the last service in a device can be 0xFFFF. If the *Attribute Handle* range for the Service UUID being searched is returned and the *End Found Handle* is not 0xFFFF, the *Find By Type Value Request* may be called again with the *Starting Handle* set to one greater than the last *Attribute Handle* range in the *Find By Type Value Response*."

[End of changes for Erratum 4065]

2.4.12 Erratum 4067 - Are prepared values written multiple times

Section: 4.9.5 Reliable Writes, page 568

[Original text states]

"If a Characteristic Value is prepared two or more times during this sub-procedure, then all prepared values are written multiple times to the same Characteristic Value in the order that they were prepared."

[Replace with]

"If a Characteristic Value is prepared two or more times during this sub-procedure, then all prepared values are written to the same Characteristic Value in the order that they were prepared."

[End of changes for Erratum 4067]

2.5 VOLUME 3 PART H: SECURITY MANAGER SPECIFICATION

2.5.1 Erratum 3928 - Inheritance of security

Section: 2.3.1 Security Properties, page 605

[Append new text to end of Sections 2.3.1]

"Security properties from the STK generated in phase 2 under which the keys are distributed shall be stored in the security database."

[End of changes for Erratum 3928]

2.5.2 Erratum 4238 - Figure not correct

Section: 5.3 Message Sequence Charts, page 443

[Add a new paragraph before the 1st paragraph]

"This section is informative and 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."

[End of changes for Erratum 4238]

2.5.3 Erratum 4249 - Clarification of STK generation

Section: 2.3.5.1 Selecting STK Generation Method, page 607

[Original text states]

"If both devices have out of band authentication data, then the Authentication Requirements Flags shall be ignored when selecting the pairing method and the Out of Band pairing method shall be used. If both devices have not set the MITM option in the Authentication Requirements Flags, then the IO capabilities shall be ignored and the Just Works association model shall be used. Otherwise the IO capabilities of the devices shall be used to determine the pairing method as defined in Table 2.4."

[Replace with]

"If both devices have out of band authentication data, then the Authentication Requirements Flags shall be ignored when selecting the pairing method and the Out of Band pairing method shall be used. If both devices have not set the MITM option in the Authentication Requirements Flags, then the IO capabilities shall be ignored and the Just Works association model shall be used."

\Initiator Responder\	OOB Set	OOB Not Set	MITM Set	MITM Not Set
OOB Set	Use OOB	Check MITM		
OOB Not Set	Check MITM	Check MITM		
MITM Set			Use IO Capabilities	Use IO Capabilities
MITM Not Set			Use IO Capabilities	Use Just Works

Table 2.4: Rules for using Out-of-Band and MITM flags

Otherwise the IO capabilities of the devices shall be used to determine the pairing method as defined in Table 2.5."

Section: 2.3.5.1 Selecting STK Generation Method, table 2.4, pages 607 and 608

[Original table number]

"*Table 2.4: Mapping of IO Capabilities to STK Generation Method*"

[Replace with]

"*Table 2.5: Mapping of IO Capabilities to STK Generation Method*"

Section: 2.3.5.1 Selecting STK Generation Method, 1st paragraph under table 2.4, page 608

[Original table reference]

"…described in Table 2.4."

[Replace with]

"…described in Table 2.5."

Section: 2.3.5.3 Passkey Entry, 2nd paragraph, page 609

[Original table reference]

"…or if Table 2.4 defines..."

[Replace with]

"…or if Table 2.5 defines..."

Section: 2.4.4.2 Encryption Setup using LTK, table 2.5, page 616

[Original table number]

"*Table 2.5: Action after encryption setup failure*"

[Replace with]

"*Table 2.6: Action after encryption setup failure*"

Section: 2.4.4.2 Encryption Setup using LTK, 1st paragraph, page 616

[Original table reference]

"…information. Table 2.5 defines…"

[Replace with]

"…information. Table 2.6 defines…"

[End of changes for Erratum 4249]

2.5.4 Erratum 4230 - Typo

Section: 2.3.5.5 Confirmation, STK generation, and Encrypt, page 610

[Original text states]

"Mconfirm = c1(TK, Mrand,
Pairing Request command, Pairing Response command,
initiating device address type, initiating device address,
responding device address type, responding devices address)"

[Delete the "s" character from the word "devices"]

"…, responding device address)"

Section: 2.3.5.5 Confirmation, STK generation, and Encrypt, page 610

[Original text states]

"Sconfirm = c1(TK, Srand,
Pairing Request command, Pairing Response command,
initiating device address type, initiating devices address,
responding device address type, responding device address)"

[Delete the "s" character from the word "devices"]

"…type, initiating device address…"

[End of changes for Erratum 4230]

2.5.5 Erratum 4241 - Clarification of sample data

Section: 2.4.5 Signing Algorithm, after 1st paragraph, page 617

[Original text states]

"M = data || SignCounter"

[Replace with]

"M = m || SignCounter"

Section: 2.4.5 Signing Algorithm, after 2nd paragraph, page 617

[Original text states]

"For example, if data..."

[Replace with]

"For example, if m..."

[End of changes for Erratum 4241]

2.5.6 Erratum 4418 - CSRK and signing need to be clarified

Section: 2.4.5 Signing Algorithm, 1st paragraph, page 616

[Original text states]

"An LE device can send signed data without having to establish an encrypted session with a peer device. Data shall be signed using CSRK. A device performing signature verification must have received CSRK from the signing device."

[Replace with]

"An LE device can send signed data without having to establish an encrypted session with a peer device. Data shall be signed using CSRK. A device performing signature verification must have received CSRK from the signing device. The sending device will use its CSRK to sign the transmitted data."

[End of changes for Erratum 4418]

2.5.7 Erratum 4407 - Slave store eDIV, Rand, LTK mapping is not necessary at all

Section: 2.4.2.3 Generation of LTK, EDIV and Rand, 3rd paragraph, page 614

[Original text states]

"The slave device shall store the mapping between EDIV, Rand and LTK in a security database so the correct LTK value is used when the master device requests encryption. Depending upon the LTK generation method additional information may need to be stored, for example the size of the distributed LTK."

[Replace with]

"The slave device can store the mapping between EDIV, Rand and LTK in a security database so the correct LTK value is used when the master device requests encryption. Depending upon the LTK generation method additional information can be stored, for example the size of the distributed LTK."

[End of changes for Erratum 4407]

2.5.8 Erratum 4648 - Incorrect ra value in Sconfirm calculation description

Section: 2.3.5.5 Confirmation, STK generation, and Encryption, 5th paragraph, page 610

[Original text states]

"The responding device calculates the 128-bit confirm value (Sconfirm) using the confirm value generation function c1 (see Section 2.2.3) with the input parameter *k* set to *TK*, the input parameter *r* set to *Srand*,the input parameter *preq* set to Pairing Request command, the input parameter *pres* set to the Pairing Response command, the input parameter *iat* set to the

initiating device address type, *ia* set to the initiating device address, *rat* set to the responding device address type and *ra* set to the initiating devices address:"

[Replace last line with]

"...device address, *rat* set to the responding device address type and *ra* set to the responding devices address:"

[End of changes for Erratum 4648]

2.5.9 Erratum 4163 - Figure 5.12 issue

Section: 5.3.5.3 Slave Rejects Pairing Because of Key Size, page 650

[Replace figure 5.12 with]

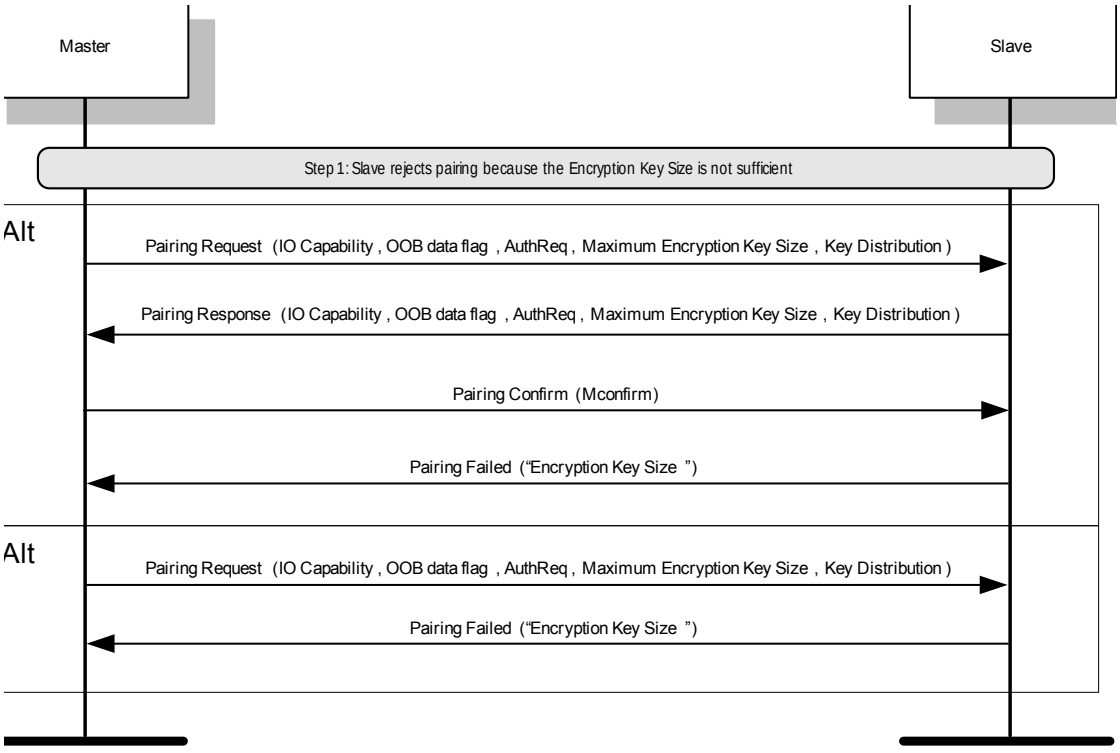


Figure 5.12: Slave rejects pairing because of key size

[End of changes for Erratum 4648]

2.6 VOLUME 6 PART A: PHYSICAL LAYER SPECIFICATION

2.6.1 Erratum 4481- Clarify Frequency Hopping

Section: 1 Scope, page 14

[Append a note to 1st paragraph]

"Note that the transceiver defined in this Part does not meet the requirements for 'frequency hopping' in some governmental regulations. See the Bluetooth Low Energy Regulatory Aspects White Paper for more information."

[End of changes for Erratum 4481]

2.7 VOLUME 6 PART B: LINK LAYER SPECIFICATION

2.7.1 Erratum 4664 - Paragraph 5.1.3.1 is not consistent with MSC's

Section: 5.1.3.1 Encryption Start Procedure, 6th paragraph, page 82

[Delete text]

"...and notify the Host with the Rand and EDIV fields."

Section: 5.1.3.1 Encryption Start Procedure, 8th paragraph, page 83

[Original text states]

"The Link Layer of the slave shall then send an LL_ENC_RSP PDU."

[Replace with]

"The Link Layer of the slave shall then send an LL_ENC_RSP PDU. The Link Layer of the slave shall then notify the Host with the Rand and EDIV fields."

Section: 5.1.3.1 Encryption Start Procedure, 12th paragraph, 3rd bullet, page 83

[Original text states]

"If the Host does provide a Long Term Key, the Link Layer of the slave shall respond to the LL_ENC_REQ PDU from the master with an LL_ENC_RSP PDU. The Link Layer shall also calculate sessionKey using the encryption engine with LTK as the key, and SKD as the plain text input. sessionKey shall be set to the output of the encryption engine."

[Replace with]

"If the Host does provide a Long Term Key, the Link Layer of the slave shall calculate *sessionKey* using the encryption engine with LTK as the key, and SKD as the plain text input. *sessionKey* shall be set to the output of the encryption engine."

[End of changes for Erratum 4664]

2.7.2 Erratum 4683 - Calculating session key diversifier and initialization vector

Section: 5.1.3.1 Encryption Procedure, 2nd paragraph, page 82

[Original text states]

"To start encryption, the Link Layer of the master shall generate the master's part of the initialization vector (IVm) and the master's part of the session key diversifier (SKDm)."

[Replace with]

"To start encryption, the Link Layer of the master shall generate the master's part of the initialization vector (IVm) and the master's part of the session key diversifier (SKDm). IVm shall be a 32 bit random number generated by the Link Layer of the master. SKDm shall be a 64 bit random number generated by the Link Layer of the master. Both IVm and SKDm shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2."

Section: 5.1.3.1 Encryption Start Procedure, 6th paragraph, page 83

[Original text states]

"Otherwise, when the Link Layer of the slave receives an LL_ENC_REQ PDU it shall generate the slave's part of the initialization vector (IVs) and the slave's part of the session key diversifier (SKDs)."

[Replace with, note: Erratum 4664 above deleted a line of text in this same paragraph and still is valid; below is the final change for this paragraph]

"Otherwise, when the Link Layer of the slave receives an LL_ENC_REQ PDU it shall generate the slave's part of the initialization vector (IVs) and the slave's part of the session key diversifier (SKDs) IVs shall be a 32 bit random number generated by the Link Layer of the slave. SKDs shall be a 64 bit random number generated by the Link Layer of the slave. Both IVs and SKDs shall be generated using the requirements for random number generation defined in [Vol 2] Part H, Section 2."

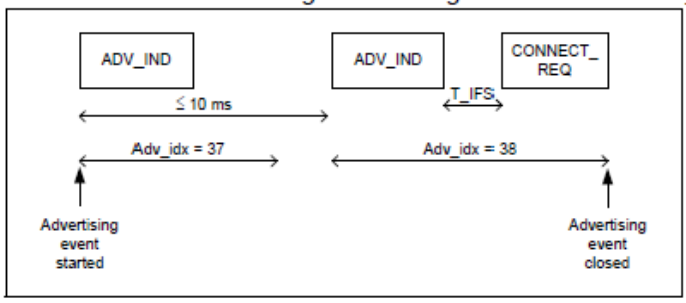
[End of changes for Erratum 4683]

2.7.3 Erratum 4217 - 4.4.2.3 Connectable Undirected Event Type

Section: 4.4.2.3 Connectable Undirected Event Type, figure 4.5, page 61

[Original caption]

Figure 4.5: Connectable undirected advertising event during which a CONNECT_REQ PDU is received



[Move the caption to below figure 4.5]

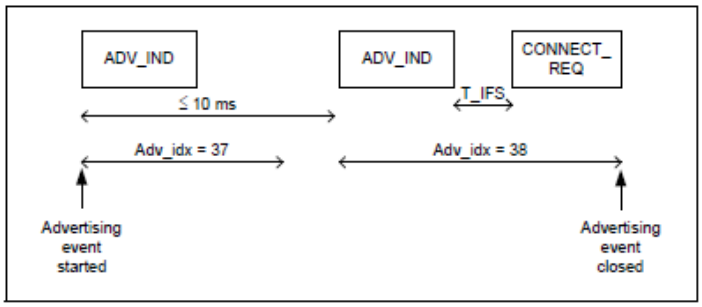


Figure 4.5: Connectable undirected advertising event during which a CONNECT_REQ PDU is received.

[End of changes for Erratum 4217]

2.7.4 Erratum 4246 - E3809

Section: 5.1.2 Channel Map Update Procedure, 4th paragraph, page 81

[Original text states]

"When a slave receives an LL_CHANNEL_MAP_REQ PDU where (Instant – connEventCount) modulo 65536 is less than 32767, the slave shall listen to all the connection events until it has confirmation that the master has received its acknowledgement of the LL_CHANNEL_MAP_REQ PDU or connEventCount equals Instant."

[Replace with]

"When a slave receives an LL_CHANNEL_MAP_REQ PDU where (Instant – *connEventCount*) modulo 65536 is less than 32767 and Instant is not equal to *connEventCount*, the slave shall listen to all the connection events until it has confirmation that the master has received its acknowledgement of the LL_CHANNEL_MAP_REQ PDU or *connEventCount* equals Instant."

[End of changes for Erratum 4246]

3 REFERENCES

[1] ESR05 – Errata Service Release to *Bluetooth*® Specifications.