The BlueNRG-MS Bluetooth® LE stack application command interface (ACI)

Introduction

The purpose of this document is to describe the design of the application command interface (ACI) of the BlueNRG-MS Bluetooth® low energy (LE) stack. This document specifies the list of commands supported by the BlueNRG-MS ACI.

1 Overview

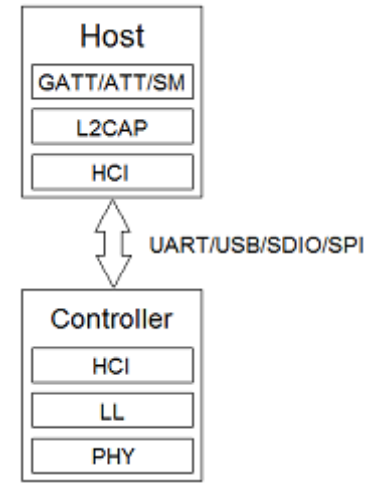
Bluetooth LE technology was adopted by the Bluetooth SIG starting from the Bluetooth core specification v4.1. Bluetooth LE technology was designed to enable products that require lower power consumption, lower complexity and lower cost compared to the Bluetooth classic or Bluetooth high-speed systems.

A typical BLE system consists of an LE controller and a host. The LE controller consists of a physical layer (PHY) including the radio, a link layer (LL) and a standard host controller interface (HCI). The host consists of a HCI and other higher protocol layers, e.g. L2CAP, SM, ATT/GATT, GAP etc.

In many designs the LE controller and the host reside in two separate silicon chips and are controlled by 2 different microcontrollers. Communication between the two is transmitted via a hardware connection, e.g. UART, SPI, USB, etc. The host can send HCI commands to control the LE controller. The HCI interface and the HCI commands are standardized by the Bluetooth core specification. Please refer to the official document for more information.

The HCI interface provides a significant benefit. Any Bluetooth tester can easily connect to the controller via the hardware connection, e.g. SPI, to test the controller by sending HCI commands. This removes the need to involve the host if the only interest is to test the controller.

Figure 1. Bluetooth LE system with separate host and controller

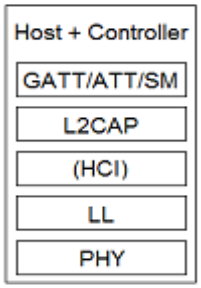


GAMS20150917EC-0945

Sometimes the LE controller and the host can be combined into a single chipset solution. This lowers the cost of hardware, first because it reduces the number of silicon chips to one rather than two, and second because the hardware connection between the host and the controller is no longer required.

Under this scenario, the host may directly control the LL/PHY. In this way, there is no need to generate the standard HCI commands, transmit the commands to the LL, and then process them. As a result, the execution time is improved and the calculation load on the CPU is reduced. However, if the HCI is removed completely, it will be more difficult to test only the controller with an external Bluetooth tester.

Figure 2. **Bluetooth LE system with combined host and controller**



GAMS20150917EC-1011

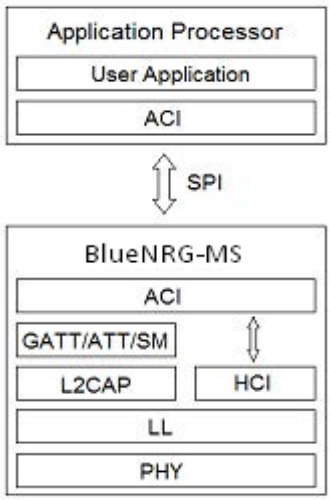
1.1 BlueNRG-MS ACI architecture

The BlueNRG-MS is implemented using a combined host and controller solution. An application command interface (ACI) is provided for accessing the BlueNRG-MS host and controller.

User applications, i.e. programs running in another silicon chip, can send ACI commands to control the BlueNRG-MS. The ACI commands should be sent over an SPI connection. The SPI protocol is described in a later section.

The ACI interface also supports HCI commands. If a command is received the ACI will check whether the command is for the host or for the controller. If the command is a HCI command, i.e. a command for the controller, the ACI will forward directly the command to the controller, bypassing it from the host. This implementation has two advantages: (1) normally the host can control the LL/PHY without using the HCI commands, which improves performance, and (2) user applications can still test the controller individually or set up some low-level hardware parameters with the HCI commands, without going through the host.

Figure 3. **BlueNRG-MS ACI architecture**



GAMS20150917EC-1038

2 ACI command data format

When sending an ACI command to the BlueNRG-MS, the command must be formatted as described in this chapter.

2.1 Overview

The BlueNRG-MS ACI commands utilize and extend the standard HCI data format. The standard HCI data format is a part of the Bluetooth core specification. To avoid redundancy and inconsistency, the texts are not copied into this document. Please refer to the official Bluetooth specification, Volume 2, Part E, Section 5 for detailed information.

According to the Bluetooth specification, a standard HCI packet can be an:

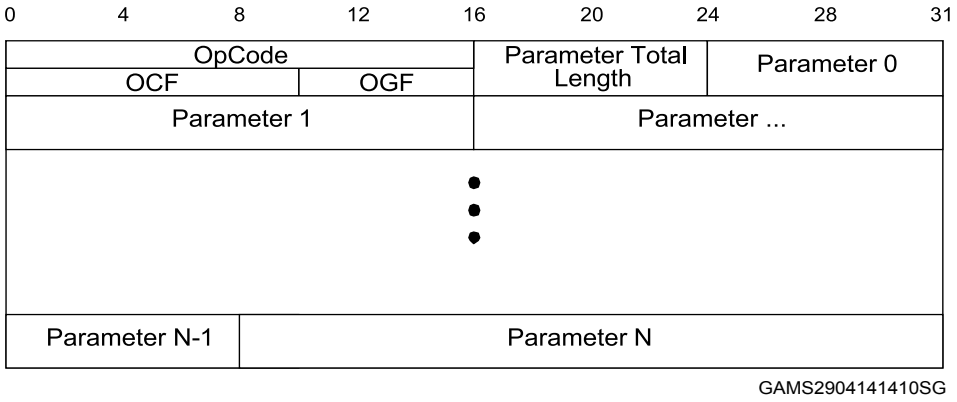
1. HCI command packet
2. HCI ACL data packet
3. HCI synchronous data packet
4. HCI event packet

In the BlueNRG-MS, the HCI synchronous data packet is not supported, but the other three are.

2.2 HCI command packet

When an external device gives a command to the BlueNRG-MS, the command must be formatted in a HCI command packet. The BlueNRG-MS only receives the HCI command packets, but does not transmit.

Figure 4. HCI command packet



Each HCI command uses a 2 byte OpCode to uniquely identify different types of commands. The OpCode is divided into two fields: the OpCode Group Field (OGF) and the OpCode Command Field (OCF). The OGF is the upper 6 bits and the remaining lower 10 bits are used by the OCF.

All HCI commands are grouped into logical groups by the Bluetooth specification and each group is assigned a unique OGF value.

Table 1. **OGF value**

Group name	OGF value
Link control commands	0x01
Link policy commands	0x02
Controller and baseband commands	0x03
Information parameters	0x04
Status parameters	0x05
Testing commands	0x06
LE controller commands	0x08
Vendor specific commands	0x3F

Depending on the OGF, the commands can be categorized into 3 sets:

1. Standard HCI commands: the commands with an OGF value other than 0x3F. These commands are designed for the controller. All standard HCI command are clearly defined in the Bluetooth specification, so they are not described in this document.
2. Vendor specific (VS) HCI commands: the commands with an OGF value 0x3F and are designed for the controller. Each vendor can define their own VS commands based on the hardware implementation. The VS commands defined for the BlueNRG-MS are described in this document.
3. Vendor specific (VS) ACI commands: the commands also with an OGF value 0x3F, but these commands are designed to control/access the host. The Bluetooth specification does not define any command for the host, so all the ACI commands are naturally vendor specific. The ACI commands for the BlueNRG-MS are described in this document.

To clarify, the commands designed to control the controller are called HCI commands. This name is defined by the Bluetooth specification and we maintain it here in order to comply with the specification. The commands designed to control the host are called ACI commands. We give it a different name to indicate that this command is rather assigned to the host, i.e. to the whole BLE system, and not only to the controller at the lower level.

However, because both ACI and HCI commands are the commands received from the external device via the ACI interface, and both of them share the same data format, sometimes it is not very important to differentiate between the two. So in this document, when referring in general to commands, we use the term “ACI commands”.

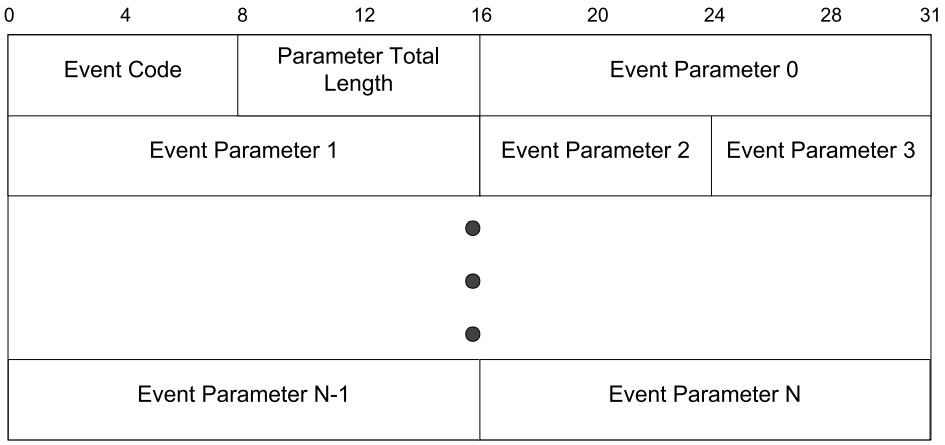
In practice, the value of the OpCode is more important. Each command, regardless of whether it is an HCI or an ACI command, matches only one OpCode value. The user application should guarantee the OpCode value is used correctly.

Please note that the Bluetooth specification uses bit-wise little endian format for the OpCode. So in Figure 4. HCI command packet, the OGF field is placed to the right. But in this document, when writing an OpCode in the format of 0xXXXX, we imply the most significant bit to the left. For example, if an OpCode is 0xFE81, its top 6-bit OGF is “111111”, i.e. 0x3F, so it is a vendor specific command. And its OCF is 10-bit 0x281. For another example, if an OpCode is 0x0406, the 6-bit OGF is 0x01, and its OCF is 0x06. So this is a standard HCI command, HCI_Disconnect.

2.3 HCI event packet

The BlueNRG-MS uses event packets to acknowledge a command or to notify that its status has updated to the user application. The BlueNRG-MS only sends HCI event packets, but does not receive them.

Figure 5. HCI event packet

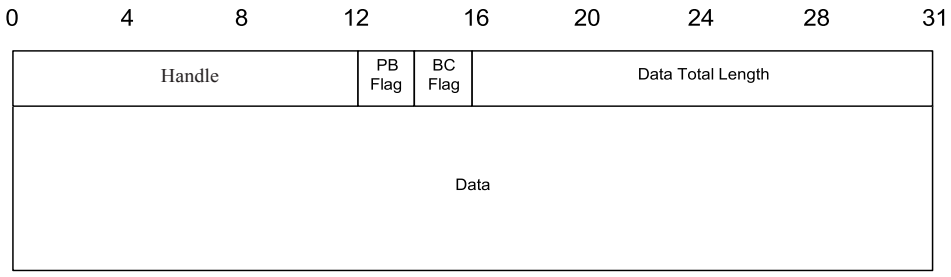


GAMS2904141415SG

2.4 HCI ACL data packet

The ACL data packets are used to exchange data.

Figure 6. HCI ACL data packet



GAMS2904141420SG

3 Standard HCI commands

3.1 Supported HCI commands

The table below lists the standard Bluetooth HCI commands which are supported by the BlueNRG-MS. For the details of each command, e.g. command descriptions, parameters, etc. please refer to the Bluetooth core specification, Volume 2, Part E, Chapter 7.

Table 2. HCI commands

Item	Command	OGF	OCF	OpCode
1	HCI_Disconnect	0x01	0x06	0x0406
2	HCI_Read_Remote_Version_Information	0x01	0x1D	0x041D
3	HCI_Set_Event_Mask	0x03	0x01	0x0C01
4	HCI_Reset	0x03	0x03	0x0C03
5	HCI_Read_Transmit_Power_Level	0x03	0x2D	0x0C2D
6	HCI_Read_Local_Version_Information	0x04	0x01	0x1001
7	HCI_Read_Local_Supported_Commands	0x04	0x02	0x1002
8	HCI_Read_Local_Supported_Features	0x04	0x03	0x1003
9	HCI_Read_BD_ADDR	0x04	0x09	0x1009
10	HCI_Read_RSSI	0x05	0x05	0x1405
11	HCI_LE_Set_Event_Mask	0x08	0x01	0x2001
12	HCI_LE_Read_Buffer_Size	0x08	0x02	0x2002
13	HCI_LE_Read_Local_Supported_Feature	0x08	0x03	0x2003
14	HCI_LE_Set_Random_Address	0x08	0x05	0x2005
15	HCI_LE_Set_Advertising_Parameters	0x08	0x06	0x2006
16	HCI_LE_Read_Advertizing_Channel_Tx_Power	0x08	0x07	0x2007
17	HCI_LE_Set_Advertising_Data	0x08	0x08	0x2008
18	HCI_LE_Set_Scan_Response_Data	0x08	0x09	0x2009
19	HCI_LE_Set_Advertise_Enable	0x08	0x0A	0x200A
20	HCI_LE_Set_Scan_Parameters	0x08	0x0B	0x200B
21	HCI_LE_Set_Scan_Enable	0x08	0x0C	0x200C
22	HCI_LE_Create_Connection	0x08	0x0D	0x200D
23	HCI_LE_Create_Connection_Cancel	0x08	0x0E	0x200E
24	HCI_LE_Read_White_List_Size	0x08	0x0F	0x200F
25	HCI_LE_Clear_While_List	0x08	0x10	0x2010
26	HCI_LE_Add_Device_To_While_List	0x08	0x11	0x2011
27	HCI_LE_Remove_Device_From_While_List	0x08	0x12	0x2012
28	HCI_LE_Connection_Update	0x08	0x13	0x2013
29	HCI_LE_Set_Host_Channel_Classification	0x08	0x14	0x2014
30	HCI_LE_Read_Channel_Map	0x08	0x15	0x2015
31	HCI_LE_Read_Remote_Used_Features	0x08	0x16	0x2016

Item	Command	OGF	OCF	OpCode
32	HCI_LE_Encrypt	0x08	0x17	0x2017
33	HCI_LE_Rand	0x08	0x18	0x2018
34	HCI_LE_Start_Encryption	0x08	0x19	0x2019
35	HCI_LE_Long_Term_Key_Request_Reply	0x08	0x1A	0x201A
36	HCI_LE_Long_Term_Key_Requested_Negative_Reply	0x08	0x1B	0x201B
37	HCI_LE_Read_Supported_States	0x08	0x1C	0x201C
38	HCI_LE_Receiver_Test	0x08	0x1D	0x201D
39	HCI_LE_Transmitter_Test	0x08	0x1E	0x201E
40	HCI_LE_Test_End	0x08	0x1F	0x201F

3.2 Supported HCI events

The table below lists the HCI events supported by the BlueNRG-MS.

Table 3. HCI events

Item	Event	Event code	Sub event code
1	Evt_Disconnect_Complete	0x05	-
2	Evt_Encryption_Change	0x08	-
3	Evt_Read_Remote_Version_Information_Complete	0x0C	-
4	Evt_Command_Complete	0x0E	-
5	Evt_Command_Status	0x0F	-
6	Evt_Hardware_Error	0x10	-
7	Evt_Number_Of_Completed_Packets	0x13	-
8	Evt_Data_Buffer_Overflow	0x1A	-
9	Evt_Encryption_Key_Refresh_Complete	0x30	-
10	Evt_LE_Connection_Complete	0x3E	0x01
11	Evt_LE_Advertising_Report	0x3E	0x02
12	Evt_LE_Connection_Update_Complete	0x3E	0x03
13	Evt_LE_Read_Remote_Used_Features_Complete	0x3E	0x04
14	Evt_LE_Long_Term_Key_Request	0x3E	0x05

3.3 Correct use of HCI commands

The use HCI commands for advertising needs to follow a particular sequence to ensure correct outcome.
The sequence is in case of starting advertising.
When using the commands at HCI level, the correct sequence has to be followed as below:

1. HCI_LE_Set_Advertising_Parameters - Used to set the advertising parameters.
2. HCI_LE_Set_Advertising_Data - Used to set the data used in advertising packets that have a data field.
3. HCI_LE_Set_Scan_Response_Data - Used to set the data used in scanning packets that have a data field.
4. HCI_LE_Set_Advertise_Enable - Turn on Advertising.

4 Vendor specific commands

The VS commands can be either ACI VS commands to access the host of the BlueNRG-MS, or the HCI VS commands to access the LE controller. Both types of the commands use the OGF value of 0x3F.

4.1 VS command and VS event format

The OCF field of the OpCode of the VS commands is further divided into two fields: Command Group ID and Command ID.

Figure 7. OpCode format for VS commands

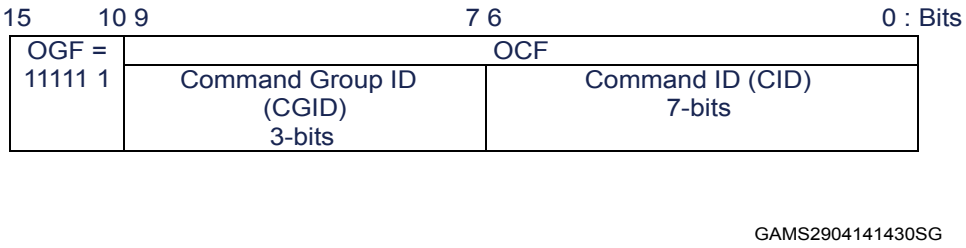


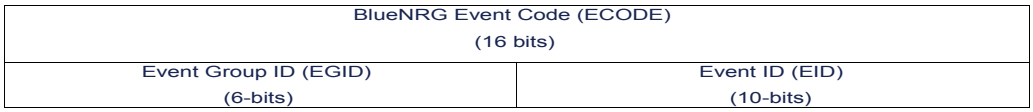
Figure 7. OpCode format for VS commands above gives the 16-bit OpCode format (see also Section 2.2 HCI command packet). The OGF field is always 0x3F for VS commands. The 10-bit OCF field is split into two parts: 3-bit Command Group ID (CGID) and 7-bit Command ID (CID). The CGID is used by the BlueNRG-MS ACI interface to route the commands to different logical layers, e.g. L2CAP, GAP, GATT, etc. It also helps to categorize the VS commands with a more clear structure. The CID determines the ID of each command. Each CGID group can have up to 128 VS commands.

Table 4. CGID group

Command group	Description	CGID
HCI	HCI extension commands	0x0
GAP	Generic access profile commands	0x1
GATT	Generic attribute profile commands	0x2
L2CAP	L2CAP commands	0x3
Reserved		0x4 – 0x7

The VS event also has a slightly different format than the standard HCI event (see also Section 2.3 HCI event packet). (1) The 8-bit event code of the VS event always has the value of 0xFF. (2) The 16-bit event parameter 0 has a different format.

Figure 8. **Event parameter 0 format for VS events**



GAMS2904141435SG

The event parameter 0 is the first return parameter in the HCI event packets, following the Parameter Length field. These 2 bytes together are defined as the BlueNRG-MS Event Code (ECODE). ECODE is further divided into 2 fields: Event Group ID (EGIO) and Event ID (EID). BlueNRG-MS combines the events into logical groups using the EGID. And the EID is used to specify an event in the group. The EGID occupies 6-bits in the ECODE field while the EID occupies the remaining 10 bits.

Table 5. **EGID group**

Event Group	Description	EGID
HCI	HCI extension events	0x0
GAP	Generic access profile commands	0x1
L2CAP	L2CAP events	0x2
GATT	Generic attribute profile events	0x3

4.2 L2CAP VS commands

Table 6. **L2CAP VS commands**

Item	Command	CGID	CID	OpCode
1	Aci_L2CAP_Connection_Parameter_Update_Request	0x03	0x01	0xFD81
2	Aci_L2CAP_Connection_Parameter_Update_Response	0x03	0x02	0xFD82

4.2.1 Aci_L2CAP_Connection_Parameter_Update_Request

Table 7. **Aci_L2CAP_Connection_Parameter_Update_Request**

Command name	Parameters	Return
Aci_L2CAP_Connection_Parameter_Update_Request (0xFD81)	Connection_handle Interval_min Interval_max Slave_latency Timeout_multiplier	Status

Description:

Send an L2CAP connection parameter update request from the slave to the master.

Table 8. Aci_L2CAP_Connection_Parameter_Update_Requests command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle of the link which the connection parameter update request has to be sent
Interval_min	2 bytes	Defines minimum value for the connection event interval in the following manner: connIntervalMin = Interval Min * 1.25 ms
Interval_max	2 bytes	Defines maximum value for the connection event interval in the following manner: connIntervalMax = Interval Max * 1.25 ms
Slave_latency	2 bytes	Defines the slave latency parameter (as number of LL connection events)
Timeout_multiplier	2 bytes	Defines connection timeout parameter in the following manner: Timeout Multiplier * 10 ms

Table 9. Aci_L2CAP_Connection_Parameter_Update_Requests return parameters

Parameter	Size	Description
Status	1 byte	0x0: connection parameter accepted 0x1: connection parameter rejected

Event(s) generated:

A command status event on the receipt of the command and a Evt_Blue_L2CAP_Conn_Upd_Resp event when the master responds to the request (accepts or rejects).

4.2.2 Aci_L2CAP_Connection_Parameter_Update_Response

Table 10. Aci_L2CAP_Connection_Parameter_Update_Response

Command name	Parameters	Return
Aci_L2CAP_Connection_Parameter_Update_Response (0xFD82)	Conn_Handle Conn_Interval_Min Conn_Interval_Max Conn_Latency Timeout_Multiplier Min_CE_length Max_CE_length Identifier Accept	Status

Description:

This command should be sent in response to the Evt_Blue_L2CAP_Connection_Update_Req event from the controller. The accept parameter has to be set to1 if the connection parameters given in the event are acceptable.

Table 11. **Aci_L2CAP_Connection_Parameter_Update_Response command parameters**

Parameter	Size	Description
Conn_Handle	2 bytes	Handle received in Evt_Blue_L2CAP_Connection_Update_Req event.
Min connection interval	2 bytes	The connection interval parameter as received in the l2cap connection update request event
Max connection interval	2 bytes	The maximum connection interval parameter as received in the l2cap connection update request event.
latency	2 bytes	The slave latency parameter as received in the l2cap connection update request event.
Timeout_Multiplier	2 bytes	The supervision connection timeout parameter as received in the l2cap connection update request event.
Min_CE length	2 bytes	Minimum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Max_CE_length	2 bytes	Maximum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Identifier	1 byte	Identifier received in Evt_Blue_L2CAP_Connection_Update_Req event.
Accept	1 byte	0x00: The connection update parameters are not acceptable. 0x01: The connection update parameters are acceptable.

Table 12. **Aci_L2CAP_Connection_Parameter_Update_Response return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command complete event is generated. If the status is success, then the link parameters are updated with the new set values as received in the connection update complete event.

4.3 L2CAP VS events

Table 13. **L2CAP VS events**

Item	Event	EGID	EID	ECODE
1	Evt_Blue_L2CAP_Connection_Update_Resp	0x02	0x00	0x0800
2	Evt_Blue_L2CAP_Procedure_Timeout	0x02	0x01	0x0801
3	Evt_Blue_L2CAP_Connection_Update_Request	0x02	0x02	0x0802

4.3.1 Evt_Blue_L2CAP_Connection_Update_Resp

This event is generated when the master responds to the connection update request packet with a connection update response packet or a command reject packet.

Table 14. Evt_Blue_L2CAP_Connection_Update_Resp

Parameter	Size	Description
Event code	2 bytes	The event code for connection update response event
Conn_handle	2 bytes	The connection handle related to the event
Event_data_length	1 byte	Length of following data
Code	1 byte	0x13 in case of valid L2CAP Connection Parameter Update Response packet. 0x01 in case of command reject.
Identifier	1 byte	Identifier of the response. It is equal to the request.
L2cap_length	2 bytes	Length of following data. It should always be 2
Result	2 bytes	Result code (parameters accepted or rejected) in case of Connection Parameter Update Response (code=0x13) or reason code for rejection in case of Command Reject (code=0x01).

4.3.2 Evt_Blue_L2CAP_Procedure_Timeout

This event is generated when the master does not respond to the connection update request packet with a connection update response packet or a command reject packet within 30 seconds.

Table 15. Evt_Blue_L2CAP_Procedure_Timeout

Parameter	Size	Description
Event code	2 bytes	The event code for L2CAP procedure timeout event
Connection_handle	2 bytes	Connection handle for which the command is given
Datalen	1 byte	Is always one

4.3.3 Evt_Blue_L2CAP_Connection_Update_Request

The event is given by the L2CAP layer when a connection update request is received from the slave. The upper layer which receives this event has to respond by sending a Aci_L2CAP_Connection_Update_Resp command to the BlueNRG-MS.

Table 16. Evt_Blue_L2CAP_Connection_Update_Request

Parameter	Size	Description
Event code	2 bytes	The event code for connection update request event.
Conn_Handle	2 bytes	Handle of the connection for which the connection update request has been received. The same handle has to be returned while responding to the event with the command Aci_L2CAP_Connection_Update_Resp.
DataLen	1 byte	Length of the data to follow. The data will be the L2CAP connection update request received
Identifier	1 byte	This is the identifier which associates the request to the response. The same identifier has to be returned by the upper layer in the command Aci_L2CAP_Connection_Update_Resp
Length	2 bytes	Length of the L2CAP connection update request
Interval_Min	2 bytes	Value as defined in Bluetooth 4.1 spec, Volume 3, Part A 4.20
Interval_Max	2 bytes	Value as defined in Bluetooth 4.1 spec, Volume 3, Part A 4.20
Slave_Latency	2 bytes	Value as defined in Bluetooth 4.1 spec, Volume 3, Part A 4.20
Timeout_Multiplier	2 bytes	Value as defined in Bluetooth 4.1 spec, Volume 3, Part A 4.20

4.4 GAP VS commands and events

4.4.1 Overview

There are 4 GAP roles defined for LE devices:

- Broadcaster: Broadcaster is a device that sends advertising events. Broadcaster shall have a transmitter and can optionally have a receiver.
- Observer: In observer role device receives the advertising events. An observer shall have a receiver and can optionally have a transmitter.
- Peripheral: Any device that accepts the establishment of LE physical link is referred as peripheral. A device operating in peripheral role will be slave in LL connection state. A peripheral shall have both a transmitter and receiver.
- Central: A device that initiates the establishment of LE physical link is referred as central device. A device operating in central role will be master in LL connection state. A central shall have both a transmitter and receiver.

BlueNRG-MS is capable of being the peripheral or central in the GAP profile context because:

- The BlueNRG-MS controller is capable of acting as a slave in a connection, i.e. it can accept the LL connection request from a central device and can support all the mandatory requirements of the GAP peripheral role as dictated in Table 2.1, Part C, Generic Access Profile of Bluetooth core specification 4.1.
- The BlueNRG-MS controller is capable of acting as a master in a connection, i.e. it can issue the LL connect request to a peripheral device and can support all the mandatory requirements of the GAP central role as dictated in Table 2.1, Part C, Generic Access Profile of Bluetooth core specification 4.1.

4.4.2 GAP VS commands

Table 17. GAP VS commands

Item	Command	CGID	CID	OpCode
1	Aci_Gap_Set_Non_Discoverable	0x01	0x01	0xFC81
2	Aci_Gap_Set_Limited_Discoverable	0x01	0x02	0xFC82
3	Aci_Gap_Set_Discoverable	0x01	0x03	0xFC83
4	Aci_Gap_Set_Direct_Connectable	0x01	0x04	0xFC84
5	Aci_Gap_Set_IO_Capability	0x01	0x05	0xFC85
6	Aci_Gap_Set_Auth_Requirement	0x01	0x06	0xFC86
7	Aci_Gap_Set_Author_Requirement	0x01	0x07	0xFC87
8	Aci_Gap_Pass_Key_Response	0x01	0x08	0xFC88
9	Aci_Gap_Authorization_Response	0x01	0x09	0xFC89
10	Aci_Gap_Init	0x01	0x0A	0xFC8A
11	Aci_Gap_Set_Non_Connectable	0x01	0x0B	0xFC8B
12	Aci_Gap_Set_Undirected_Connectable	0x01	0x0C	0xFC8C
13	Aci_Gap_Slave_Security_request	0x01	0x0D	0xFC8D
14	Aci_Gap_Update_Adv_Data	0x01	0x0E	0xFC8E
15	Aci_Gap_Delete_AD_Type	0x01	0x0F	0xFC8F
16	Aci_Gap_Get_Security_Level	0x01	0x10	0xFC90
17	Aci_Gap_Set_Event_Mask	0x01	0x11	0xFC91
18	Aci_Gap_Configure_WhiteList	0x01	0x12	0xFC92
19	Aci_Gap_Terminate	0x01	0x13	0xFC93
20	Aci_Gap_Clear_Security_Database	0x01	0x14	0xFC94

Item	Command	CGID	CID	OpCode
21	Aci_Gap_Allow_Rebond	0x01	0x15	0xFC95
22	Aci_Gap_Start_Limited_Discovery_Proc	0x01	0x16	0xFC96
23	Aci_Gap_Start_General_Discovery_Proc	0x01	0x17	0xFC97
24	Aci_Gap_Start_Name_Discovery_Proc	0x01	0x18	0xFC98
25	Aci_Gap_Start_Auto_Conn_Establishment	0x01	0x19	0xFC99
26	Aci_Gap_Start_General_Conn_Establishment	0x01	0x1A	0xFC9A
27	Aci_Gap_Start_Selective_Conn_Establishment	0x01	0x1B	0xFC9B
28	Aci_Gap_Create_Connection	0x01	0x1C	0xFC9C
29	Aci_Gap_Terminate_Gap_Procedure	0x01	0x1D	0xFC9D
30	Aci_Gap_Start_Connection_Update	0x01	0x1E	0xFC9E
31	Aci_Gap_Send_Pairing_Request	0x01	0x1F	0xFC9F
32	Aci_Gap_Resolve_Private_Address	0x01	0x20	0xFCA0
33	Aci_Gap_Get_Bonded_Devices	0x01	0x23	0xFCA3
34	Aci_Gap_Set_Broadcast_Mode	0x01	0x21	0xFCA1
35	Aci_Gap_Start_Observation_Proc	0x01	0x22	0xFCA2
36	Aci_Gap_Is_Device_Bonded	0x01	0x24	0xFCA4

4.4.3 Aci_Gap_Set_Non_Discoverable

Table 18. Aci_gap_set_non_discoverable

Command name	Parameters	Return
Aci_Gap_Set_Non_Discoverable (0xFC81)		Status

Description:

Set the device in non-discoverable mode. This command will disable the LL advertising and put the device in standby state.

Table 19. Aci_gap_set_non_discoverable return parameters

Parameter	Size	Description
Status	1 byte	0x0: status success 0xC: Command Disallowed

Event(s) generated:

When the Aci_Gap_Set_Non_Discoverable command has completed, the controller will generate a command complete event.

4.4.4 Aci_Gap_Set_Limited_Discoverable

Table 20. Aci_Gap_Set_Limited_Discoverable

Command name	Parameters	Return
Aci_Gap_Set_Limited_Discoverable (0xFC82)	Advertising_Event_Type Adv_Interval_Min Adv_Interval_Max Address_Type Adv_Filter_Policy Local_Name_Length Local_Name Service_Uuid_Length Service_Uuid_List Slave_Conn_Interval_Min Slave_Conn_Interval_Max	Status

Description:

Set the device in limited discoverable mode (as defined in GAP specification volume 3, section 9.2.3). The device will be discoverable for maximum period of TGAP (lim_adv_timeout) = 180 seconds (from errata). The advertising can be disabled at any time by issuing Aci_Gap_Set_Non_Discoverable command.

The Adv_Interval_Min and Adv_Interval_Max parameters are optional. If both are set to 0, the GAP will use default values for adv intervals for limited discoverable mode.

To allow a fast connection, the host can set Local_Name, Service_Uuid_List, Slave_Conn_Interval_Min and Slave_Conn_Interval_Max. These parameters are optional in this command. These values can be set in advertised data using GAP_Update_Adv_Data command separately.

Table 21. Aci_Gap_Set_limited_Discoverable command parameters

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x00: Connectable undirected advertising (default) 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising
Adv_Interval_Min	2 bytes	Minimum advertising interval for non-directed advertising. Range: 0x0020 to 0x4000 Default: N = 0x0800 (1.28 second) Time = N * 0.625 ms Time Range: 20 ms to 10.24 s
Adv_Interval_Max	2 bytes	Maximum advertising interval for non-directed advertising. Range: 0x0020 to 0x4000 Default: N = 0x0800 (1.28 seconds) Time = N * 0.625 ms Time Range: 20 ms to 10.24 s
Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Adv_Filter_Policy	1 byte	0x00: Allow scan request from any, allow connect request from any (default). 0x01: Allow scan request from white list only, allow connect request from Any. 0x02: Allow scan request from any, allow connect request from white list only. 0x03: Allow scan request from white list only, allow connect request from white list only.
Local_Name_Length	1 byte	Length of the local name string in octets. If length is set to 0x00, Local_Name parameter should not be used.

Parameter	Size	Description
Local_Name	0-N bytes	Local name of the device. This is an ASCII string without NULL character. First byte is the AD type: AD_TYPE_SHORTENED_LOCAL_NAME or AD_TYPE_COMPLETE_LOCAL_NAME.
Service_UUID_Length	1 byte	Length of the service UUID list in octets. If there is no service to be advertised, set this field to 0x00.
Service_UUID_List	0-N bytes	This is the list of the UUID's AD types as defined in Volume 3, Section 11.1.1 of GAP Specification
Slave_Conn_Interval_Min	2 bytes	Slave connection internal minimum value. Connection interval is defined in the following manner: connIntervalmin = Slave_Conn_Interval_Min * 1.25 ms Slave_Conn_Interval_Min range: 0x0006 to 0x0C80 Value of 0xFFFF indicates no specific minimum.
Slave_Conn_Interval_Max	2 bytes	Slave connection internal maximum value. ConnIntervalmax = Slave_Conn_Interval_Max * 1.25 ms Slave_Conn_Interval_Max range: 0x0006 to 0x0C80 Slave_ Conn_Interval_Max shall be equal to or greater than the Slave_Conn_Interval_Min. Value of 0xFFFF indicates no specific maximum.

Table 22. Aci_Gap_Set_Limited_Discoverable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

When the controller receives the command, it will generate a command status event with return parameter. Controller starts the advertising after this and when advertising timeout happens (i.e. limited discovery period has elapsed), controller generates Evt_Blue_Gap_Limited_Discoverable_Complete event.

4.4.5 Aci_Gap_Set_Discoverable

Table 23. Aci_Gap_Set_Discoverable

Command name	Parameters	Return
Aci_Gap_Set_Discoverable (0xFC83)	Advertising_Event_Type Adv_Interval_Min Adv_Interval_Max Address_Type Adv_Filter_Policy Local_Name_Length Local_Name Service_Uuid_Length Service_Uuid_List Slave_Conn_Interval_Min Slave_Conn_Interval_Max	Status

Description:

Set the device in general discoverable mode (as defined in GAP specification volume 3, section 9.2.4). The device will be discoverable until the host issues the Aci_Gap_Set_Non_Discoverable command. The Adv_Interval_Min and Adv_Interval_Max parameters are optional. If both are set to 0, the GAP uses the default values for adv intervals for general discoverable mode.

Table 24. Aci_Gap_Set_Discoverable command parameters

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x00: Connectable undirected advertising (default) 0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising
Adv_Interval_Min	2 bytes	Minimum advertising interval for non-directed advertising. Range: 0x0020 to 0x4000 Default: N = 0x0800 (1.28 second) Time = N * 0.625 ms Time Range: 20 ms to 10.24 s
Adv_Interval_Max	2 bytes	Maximum advertising interval for non-directed advertising. Range: 0x0020 to 0x4000 Default: N = 0x0800 (1.28 seconds) Time = N * 0.625 ms Time Range: 20 ms to 10.24 s
Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Adv_Filter_Policy	1 byte	0x00: Allow scan request from any, allow connect request from any (default). 0x01: Allow scan request from white list only, allow connect request from any. 0x02: Allow scan request from any, allow connect request from white list only. 0x03: Allow scan request from white list only, allow connect request from white list only.
Local_Name_Length	1 byte	Length of the local name string in octets. If length is set to 0x00, Local_Name parameter should not be used.
Local_Name	0-N bytes	Local name of the device. This is an ASCII string without NULL character. First byte is the AD type: AD_TYPE_SHORTENED_LOCAL_NAME or AD_TYPE_COMPLETE_LOCAL_NAME.

Parameter	Size	Description
Service_UUID_Length	1 byte	Length of the service UUID list in octets. If there is no service to be advertised, set this field to 0x00.
Service_UUID_List	0-N bytes	This is the list of the UUIDs AD types as defined in Volume 3, Section 11.1.1 of GAP Specification
Slave_Conn_Interval_Min	2 bytes	Slave connection internal minimum value. Connection interval is defined in the following manner: connIntervalmin = Slave_Conn_Interval_Min * 1.25 ms Slave_Conn_Interval_Min range: 0x0006 to 0x0C80 Value of 0xFFFF indicates no specific minimum.
Slave_Conn_Interval_Max	2 bytes	Slave connection internal maximum value. ConnIntervalmax = Slave_Conn_Interval_Max * 1.25 ms Slave_Conn_Interval_Max range: 0x0006 to 0x0C80 Slave_ Conn_Interval_Max shall be equal to or greater than the Slave_Conn_Interval_Min. Value of 0xFFFF indicates no specific maximum.

Table 25. Aci_Gap_Set_Discoverable return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

When the Aci_Gap_Set_Discoverable command has completed, the controller will generate a command complete event.

4.4.6 Aci_Gap_Set_Direct_Connectable

Table 26. Aci_Gap_Set_Direct_Connectable

Command name	Parameters	Return
Aci_Gap_Set_Direct_Connectable (0xFC84)	Own_Address_Type Advertising Type Initiator_Address_Type Initiator_Direct_Address	Status

Description:

Set the device in direct connectable mode (as defined in GAP specification Volume 3, Section 9.3.3). Device uses direct connectable mode to advertise using either High Duty cycle advertisement events or Low Duty cycle advertisement events and the address as what is specified in the Own Address Type parameter. The Advertising Type parameter in the command specifies the type of the advertising used.

The device will be in directed connectable mode only for 1.28 seconds. If no connection is established within this duration, the device enters non discoverable mode and advertising will have to be again enabled explicitly.

Table 27. **Aci_Gap_Set_Direct_Connectable command parameters**

Parameter	Size	Description
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Advertising Type	1 byte	The advertising type 0x01 : High Duty Cycle directed Advertising 0x04 : Low Duty Cycle directed Advertising
Initiator_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Initiator_Direct_Address	6 bytes	Initiator Bluetooth address
Adv_Interval_min	2 bytes	Minimum advertising interval for low duty cycle directed advertising. Range: 0x0020 to 0x4000. Time = N * 0.625 ms. Time Range: 20 ms to 10.24 s.
Adv_Interval_max	2 bytes	Maximum advertising interval for low duty cycle directed advertising. Range: 0x0020 to 0x4000. Time = N * 0.625 ms. Time Range: 20 ms to 10.24 s.

Table 28. **Aci_Gap_Set_Direct_Connectable return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:

When the command has completed, the controller will generate a command complete event. The controller also generates a LE_Connection_Complete event with the status set to directed_advertising_timeout if the connection was not established and 0x00 if the connection was successfully established.

4.4.7 Aci_Gap_Set_IO_Capability

Table 29. **Aci_Gap_Set_IO_Capability**

Command name	Parameters	Return
Aci_Gap_Set_IO_Capability (0xFC85)	IO_Capability	Status

Description:

Set the IO capabilities of the device. This command has to be given only when the device is not in a connected state.

Table 30. **Aci_Gap_Set_IO_Capability command parameters**

Parameter	Size	Description
IO_Capability	1 byte	0x00: Display only 0x01: Display yes/no 0x02: Keyboard only 0x03: No input, no output 0x04: Keyboard display

Table 31. **Aci_Gap_Set_IO_Capability return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.4.8 Aci_Gap_Set_Auth_Requirement

Table 32. **Aci_Gap_Set_Auth_Requirement**

Command name	Parameters	Return
Aci_Gap_Set_Auth_Requirement (0xFC86)	MIMT_mode OOB_enable OOB_data Min_encryption_key_size Max_encryption_key_size Use_fixed_pin Fixed_pin Bonding_mode	Status

Description:

Set the authentication requirements for the device. If the OOB_Enable is set to 0, the following 16 octets of OOB_Data will be ignored on reception. This command has to be given only when the device is not in a connected state.

Table 33. **Aci_Gap_Set_Auth_Requirement command parameters**

Parameter	Size	Description
MITM_mode	1 byte	0x00: MITM not required 0x01: MITM required
OOB_enable	1 byte	0x00: Out Of bound authentication not enabled 0x01: Out Of bound authentication enabled
OOB_data	16 bytes	OOB data
Min_encryption_key_size	1 byte	Minimum size of the encryption key to be used during the pairing process
Max_encryption_key_size	1 byte	Maximum size of the encryption key to be used during the pairing process
Use_fixed_pin	1 byte	0x00: Use fixed pin during the pairing process 0x01: Do not use fixed pin during the pairing process. In this case, GAP will generate Evt_Blue_Pin_Code_Request event to the host.
Fixed_pin	4 bytes	Fixed pin to be used during pairing if MIMT protection is enabled. Any random value between 0 to 999999
Bonding_mode	1 byte	0x00: Bonding not required 0x01: Bonding required

Table 34. **Aci_Gap_Set_Auth_Requirement return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command Disallowed 0x11: Unsupported Feature

Event(s) generated:
When the command has completed, the controller will generate a command complete event.

4.4.9 Aci_Gap_Set_Author_Requirement

Table 35. **Aci_Gap_Set_Author_Requirement**

Command name	Parameters	Return
Aci_Gap_Set_Author_Requirement (0xFC87)	Connection_handle Authorization_enable	Status

Description:
Set the authorization requirements of the device. This command has to be given when connected to a device if authorization is required to access services which require authorization.

Table 36. **Aci_Gap_Set_Author_Requirement command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Handle of the connection.

Parameter	Size	Description
Authorization_enable	1 byte	0x00: Authorization not required (default) 0x01: Authorization required. This enables the authorization in the device and when a remote device tries to read/write a characteristic with authorization requirements, the stack will send back an error response with "Insufficient authorization" error code. After pairing is complete a Evt_Blue_Gap_Authorization_Request event will be sent to the Host

Table 37. Aci_Gap_Set_Author_Requirement return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When the command has completed, the controller will generate a command complete event.

4.4.10 Aci_Gap_Pass_Key_Response

Table 38. Aci_Gap_Pass_Key_Response

Command name	Parameters	Return
Aci_Gap_Pass_Key_Response (0xFC88)	Conn_handle Pass_Key	Status

Description:

This command should be send by the host in response to Evt_Blue_Pass_Key_Request event. The command parameter contains the pass key which will be used during the pairing process.

Table 39. Aci_Gap_Pass_Key_Response command parameters

Parameter	Size	Description
Conn_handle	2 bytes	Connection_handle
Pass_Key	4 byte	Pass key, value range: 0 - 999999 (decimal)

Table 40. Aci_Gap_Pass_Key_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:

When controller receives this command, it will generate a command complete event with return parameter. When paring process completes, it will generate Evt_Blue_Gap_Pairing_Cmplt event.

4.4.11 Aci_Gap_Authorization_Response

Table 41. Aci_Gap_Authorization_Response

Command name	Parameters	Return
Aci_Gap_Authorization_Response (0xFC89)	Connection_handle Authorize	Status

Description:
This command should be sent by the host in response to Evt_Blue_Gap_Authorization_Request event.

Table 42. Aci_Gap_Authorization_Response command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection_handle
Authorize	1 byte	0x01: Authorize (accept connection) 0x02: Reject (reject connection)

Table 43. Aci_Gap_Authorization_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed

Event(s) generated:
When the command has completed, the controller will generate a command complete event.

4.4.12 Aci_Gap_Init

Table 44. Aci_Gap_Init

Command name	Parameters	Return
Aci_Gap_Init (0xFC8A)	Role enable_Privacy device_name_char_len	Service_handle Dev_name_char_handle Appearance_Char_handle

Description:
Register the GAP service with the GATT. The device name characteristic and appearance characteristic are added by default and the handles of these characteristics are returned in the event data. The role parameter can be a bitwise OR of any of the values mentioned below.

Table 45. **Aci_Gap_Init command parameters**

Parameter	Size	Description
Role	1 byte	0x01:Peripheral 0x02: Broadcaster 0x04: Central 0x08: Observer
enable_Privacy	1 byte	0x00: Privacy is disabled 0x01: Privacy is enabled
device_name_char_len	1 byte	Length of the device name characteristic

Table 46. **Aci_Gap_Init return parameters**

Parameter	Size	Description
Status	1 byte	0x00 : Success 0x47 : Error. Does not have sufficient memory to add the required gatt characteristics 0x12 : HCI_Invalid Parameters
Service_handle	2 bytes	Handle for the GAP service
Dev_name_char_handle	2 bytes	Handle for the device name characteristic added to the GAP service
Appearance_Char_handle	2 bytes	Handle for the appearance characteristic added to the GAP service

Event(s) generated:

On the completion of the command, a command complete event is generated. The status indicates success or failure. If the registration of GAP service is successful, the event data contains the handle for the GAP service registered with GATT and also the attribute handles for the device name and appearance characteristics.

4.4.13 Aci_Gap_Set_Non_Connectable

Table 47. **Aci_Gap_Set_Non_Connectable**

Command name	Parameters	Return
Aci_Gap_Set_Non_Connectable (0xFC8B)	Advertising_Event_Type own_address_Type	Status

Description:

Put the device into non connectable mode. This mode does not support connection. The privacy setting done in the gap_init command plays a role in deciding the valid parameters for this command.

Table 48. **Aci_Gap_Set_Non_Connectable command parameters**

Parameter	Size	Description
Advertising_Event_Type	1 byte	0x02: Scannable undirected advertising 0x03: Non connectable undirected advertising

Parameter	Size	Description
own_address_type	1 byte	If Privacy is disabled: 0x00 : Public device address type 0x01 : static random address type If Privacy is enabled: 0x02 : Private resolvable address type 0x03 : Private non resolvable address type

Table 49. Aci_Gap_Set_Non_Connectable return parameters

Parameter	Size	Description
Status	1 byte	0x00 : Success 0x41 : BLE_Status_Failed 0x42 : BLE_Status_Invalid Parameter 0x0C: Command Disallowed 0x12 : Invalid HCI parameters

Event(s) generated:

A command complete event is generated upon the completion of the command.

4.4.14 Aci_Gap_Set_Undirected_Connectable

Table 50. Aci_Gap_Set_Undirected_Connectable

Command name	Parameters	Return
Aci_Gap_Set_Undirected_Connectable (0xFC8C)	Advertising_filtering_policy Own_address_type	Status

Description:

Put the device into undirected connectable mode. The privacy setting done in the gap_init command plays a role in deciding the valid parameters for this command.

Table 51. Aci_Gap_Set_Undirected_Connectable command parameters

Parameter	Size	Description
Advertising_filtering_policy	1 byte	0x00: Allow scan request from any, allow connect request from any (default). 0x03: Allow scan request from white list only, allow connect request from White List Only.
Own_Address_Type	1 byte	If privacy is disabled 0x00: Public device address (default) 0x01: Random device address If privacy is enabled 0x02: Private resolvable address 0x03: Non resolvable address

Table 52. **Aci_Gap_Set_Undirected_Connectable return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameter 0x0C: Command disallowed 0x11: Unsupported feature

Event(s) generated:
A command complete event is generated on the completion of the command.

4.4.15 Aci_Gap_Slave_Security_Request

Table 53. **Aci_Gap_Slave_Security_Request**

Command name	Parameters	Return
Aci_Gap_Slave_Security_Request (0xFC8D)	Connection_handle Bonding MITM_Protection	Status

Description:
This command has to be issued to notify the master of the security requirements of the slave.

Table 54. **Aci_Gap_Slave_Security_Request command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Handle of the connection on which the slave security request is sent (ignored in slave-only role)
Bonding	1 byte	0x00: No bonding 0x01: Bonding required
MITM_protection	1 byte	0x00: MITM protection not required 0x01: MITM protection required

Table 55. **Aci_Gap_Slave_Security_Request return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x1F: Out of memory 0x0C: Command disallowed

Event(s) generated:
A command status event will be generated when a valid command is received. On completion of the command, i.e. when the security request is successfully transmitted to the master, a Evt_Blue_Slave_Security_Initiated vendor specific event will be generated.

4.4.16 Aci_Gap_Update_Adv_Data

Table 56. Aci_Gap_Update_Adv_Data

Command name	Parameters	Return
Aci_Gap_Update_Adv_Data (0xFC8E)	Adv_Data_Length Adv_Data	Status

Description:

This command can be used to update the advertising data for a particular AD type. If the AD type specified does not exist, then it is added to the advertising data. If the overall advertising data length is more than 31 octets after the update, then the command is rejected and the old data is retained.

Table 57. Aci_Gap_Update_Adv_Data command parameters

Parameter	Size	Description
Adv_Data_Length	1 byte	The number of bytes of data that is to follow in the advData parameter
Adv_Data	0-N bytes	The advData has to be formatted as specified in Bluetooth specification (volume 3, Part C, 11), including data length

Table 58. Aci_Gap_Update_Adv_Data return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x41: Failed

Event(s) generated:

A command complete event is generated when the command has completed with the status indicating success or failure.

4.4.17 Aci_Gap_Delete_AD_Type

Table 59. Aci_Gap_Delete_AD_Type

Command name	Parameters	Return
Aci_Gap_Delete_AD_Type (0xFC8F)	AD_type	Status

Description:

This command can be used to delete the specified AD type from the advertisement data if present.

Table 60. **Aci_Gap_Delete_AD_Type command parameters**

Parameter	Size	Description
AD_type	1 byte	0x01: AD_TYPE_FLAGS 0x02: AD_TYPE_16_BIT_SERV_UUID 0x03: AD_TYPE_16_BIT_SERV_UUID_CMPLT_LIST 0x04: AD_TYPE_32_BIT_SERV_UUID 0x05: AD_TYPE_32_BIT_SERV_UUID_CMPLT_LIST 0x06: AD_TYPE_128_BIT_SERV_UUID 0x07: AD_TYPE_128_BIT_SERV_UUID_CMPLT_LIST 0x08: AD_TYPE_SHORTENED_LOCAL_NAME 0x09: AD_TYPE_COMPLETE_LOCAL_NAME 0x0A: AD_TYPE_TX_POWER_LEVEL 0x10: AD_TYPE_SEC_MGR_TK_VALUE 0x11: AD_TYPE_SEC_MGR_OOB_FLAGS 0x12: AD_TYPE_SLAVE_CONN_INTERVAL 0x14: AD_TYPE_SERV_SOLICIT_16_BIT_UUID_LIST 0x15: AD_TYPE_SERV_SOLICIT_32_BIT_UUID_LIST 0x16: AD_TYPE_SERVICE_DATA 0xFF: AD_TYPE_MANUFACTURER_SPECIFIC_DATA

Table 61. **Aci_Gap_Delete_AD_Type return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x1F: AD Type not found

Event(s) generated:
A command complete event is generated and the status indicates success or failure.

4.4.18 Aci_Gap_Get_Security_Level

Table 62. **Aci_Gap_Get_Security_Level**

Command name	Parameters	Return
Aci_Gap_Get_Security_Level (0xFC90)		Status MIMT_protection_required Bonding_required OOB_data_present Passkey_required

Description:
This command can be used to get the current security settings of the device.

Table 63. **Aci_Gap_Get_Security_Level return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success
MIMT_protection_required	1 byte	0x00: Not required 0x01: Required
Bonding_required	1 byte	0x00: Not required 0x01: Required
OOB_data_present	1 byte	0x00: Not present 0x01: Present
Passkey_required	1 byte	0x00: Not required 0x01: Fixed pin is present which is being used 0x2: Passkey required for pairing. An event will be generated when required.

Event(s) generated:

The security parameters is returned in a command complete event. The first byte indicates the status of the security manager (channel opened or closed).

4.4.19 Aci_Gap_Set_Event_Mask

Table 64. **Aci_Gap_Set_Event_Mask**

Command name	Parameters	Return
Aci_Gap_Set_Event_Mask (0xFC91)	Event_mask	Event_mask status

Description:

It allows masking events from the GAP. The default configuration is all the events masked.

Table 65. **Aci_Gap_Set_Event_Mask command parameters**

Parameter	Size	Description
Event_mask	2 bytes	0x0001: Evt_Blue_Gap_Limited_Discoverable 0x0002: Evt_Blue_Pairing_Complete 0x0004: Evt_Blue_Pass_Key_Request 0x0008: Evt_Blue_Authorization_Request 0x0010: Evt_Blue_Slave_Security_Initiated 0x0020: Evt_Blue_Gap_Bond_Lost

Table 66. **Aci_Gap_Set_Event_Mask return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid HCI command parameter

Event(s) generated:

A command complete event is generated on the completion of the command.

4.4.20 Aci_Gap_Configure_WhiteList

Table 67. Aci_Gap_Configure_WhiteList

Command name	Parameters	Return
Aci_Gap_Configure_WhiteList (0xFC92)		Status

Description:

Configure the controller's white list with devices that are present in the security database.

Table 68. Aci_Gap_Configure_WhiteList return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid HCI command parameter 0x47: Error

Event(s) generated:

The controller generates a command complete event on the completion of the command and the status indicates whether the white list was configured or not. The command returns an error if there are no devices in the database or it was unable to add to the white list.

4.4.21 Aci_Gap_Terminate

Table 69. Aci_Gap_Terminate

Command name	Parameters	Return
Aci_Gap_Terminate (0xFC93)	Conn_handle Reason	Status

Description:

Command the controller to terminate the connection.

Table 70. Aci_Gap_Terminate command parameters

Parameter	Size	Description
Conn_handle	2 bytes	Handle of the connection to be terminated
Reason	1 byte	Reason for requesting disconnection. The error code can be any of ones as specified for the disconnected command in the HCI specification

Table 71. **Aci_Gap_Terminate return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x0C: Command disallowed

Event(s) generated:

The controller generates a command status event when the command is received and a disconnection complete event is generated when the link is disconnected.

4.4.22 Aci_Gap_Clear_Security_Database

Table 72. **Aci_Gap_Clear_Security_Database**

Command name	Parameters	Return
Aci_Gap_Clear_Security_Database (0xFC94)		Status

Description:

Clear the security database. All the devices in the security database are removed.

Table 73. **Aci_Gap_Clear_Security_Database return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x4B: Flash erase failed

Event(s) generated:

The controller generates a command complete event on the completion of the command.

4.4.23 Aci_Gap_Allow_Rebond

Table 74. **Aci_Gap_Allow_Rebond**

Command name	Parameters	Return
Aci_Gap_Allow_Rebond (0xFC95)	Connection_ handle	Status

Table 75. **Aci_Gap_Allow_Rebond command parameters**

Connection_ Handle	2 bytes	Handle of the connection with which re-bonding is required

Description:

This command should be given by the application when it receives the EVT_BLUE_BOND_LOST if it wants the re-bonding to happen successfully. If this command is not given on receiving the event, the bonding procedure is timeout.

Table 76. **Aci_Gap_Allow_Rebond return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success

Event(s) generated:

The controller generates a command complete event on the completion of the command. Even if the command is given when it is not valid, success is returned but internally it has no effect.

4.4.24 Aci_Gap_Start_Limited_Discovery_Proc

Table 77. **Aci_Gap_Start_Limited_Discovery_Proc**

Command name	Parameters	Return
Aci_Gap_Start_Limited_Discovery_Proc (0xFC96)	Scan_Interval Scan_Window Own_Address_Type filterDuplicates	Status

Description:

Start the limited discovery procedure. The controller is commanded to start active scanning. When this procedure is started, only the devices in limited discoverable mode are returned to the upper layers.

Table 78. **Aci_Gap_Start_Limited_Discovery_Proc command parameters**

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates

Table 79. **Aci_Gap_Start_Limited_Discovery_Proc return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, the procedure is terminated when either the upper layers issue a command to terminate the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x01 or a timeout happens. When the procedure is terminated due to any of the above reasons, Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x01. The device found when the procedure is ongoing is returned to the upper layers through the event LE_Advertising_Report.

4.4.25 Aci_Gap_Start_General_Discovery_Proc

Table 80. Aci_Gap_Start_General_Discovery_Proc

Command name	Parameters	Return
Aci_Gap_Start_General_Discovery_Proc (0xFC97)	Scan_Interval Scan_Window Own_Address_Type filterDuplicates	Status

Description:

Start the general discovery procedure. The controller is commanded to start active scanning.

Table 81. Aci_Gap_Start_General_Discovery_Proc command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates

Table 82. Aci_Gap_Start_General_Discovery_Proc return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, the procedure is terminated when either the upper layers issue a command to terminate the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x02 or a timeout happens. When the procedure is terminated due to any of the above reasons, Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x02. The device found when the procedure is ongoing is returned to the upper layers through the event LE_Advertising_Report.

4.4.26 Aci_Gap_Start_Name_Discovery_Proc

Table 83. **Aci_Gap_Start_Name_Discovery_Proc**

Command name	Parameters	Return
Aci_Gap_Start_Name_Discovery_Proc (0xFC98)	Scan_Interval Scan_Window Peer_Address_Type Peer_Address Own_Address_Type Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max	Status

Description:

Start the name discovery procedure. A LE_Create_Connection call is made to the controller by GAP with the initiator filter policy set to “ignore whitelist and process connectable advertising packets only for the specified device”. Once a connection is established, GATT procedure is started to read the device name characteristic. When the read is completed (successfully or unsuccessfully), a Evt_Blue_Gap_Procedure_Complete event is given to the upper layer. The event also contains the name of the device if the device name was read successfully.

Table 84. **Aci_Gap_Start_Name_Discovery_Proc command parameters**

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms.
Peer_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Peer_Address	6 bytes	Address of the peer device with which a connection has to be established.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This is less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds

Parameter	Size	Description
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This is greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE link. Range: 0x000A to 0x0C80 Time = N * 10 ms Time range: 100 ms to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Conn_Len_Max	2 bytes	Maximum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.

Table 85. Aci_Gap_Start_Name_Discovery_Proc return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, on completion of the procedure, a Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x04.

4.4.27 Aci_Gap_Start_Auto_Conn_Establishment

Table 86. Aci_Gap_Start_Auto_Conn_Establishment

Command name	Parameters	Return
Aci_Gap_Start_Auto_Conn_Establishment (0xFC99)	Scan_Interval Scan_Window Own_Address_Type Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max Num_WhiteList_Entries Addr_Array	Status

Description:

Start the auto connection establishment procedure. The devices specified are added to the white list of the controller and a LE_Create_Connection call will be made to the controller by GAP with the initiator filter policy set to “use whitelist to determine which advertiser to connect to”. When a command is issued to terminate the procedure by upper layer, a LE_Create_Connection_Cancel call will be made to the controller by GAP.

Table 87. Aci_Gap_Start_Auto_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This is less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This is greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4

Parameter	Size	Description
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 ms Time range: 100 ms to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Conn_Len_Max	2 bytes	Maximum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Num_WhiteList_Entries	1 byte	Number of devices that have to be added to the whitelist.
Addr_Array	N * 7 bytes	The addr_array contains Num_White_List_Entries number of addresses and their address types that have to be added into the whitelist. The format of the addr_array should be address type followed by address.

Table 88. Aci_Gap_Start_Auto_Conn_Establishment return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, the procedure is terminated when either a connection is successfully established with one of the specified devices in the white list or the procedure is explicitly terminated by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x08. A Evt_Blue_Gap_Procedure_Complete event is returned with the procedure code set to 0x08.

4.4.28 Aci_Gap_Start_General_Conn_Establishment

Table 89. Aci_Gap_Start_General_Conn_Establishment

Command name	Parameters	Return
Aci_Gap_Start_General_Conn_Establishment (0xFC9A)	Scan_Type Scan_Interval Scan_Window Own_Address_Type filterDuplicates	Status

Start a general connection establishment procedure. The host enables scanning in the controller with the scanner filter policy set to “accept all advertising packets” and from the scanning results, all the devices are sent to the upper layer using the event LE_Advertising_Report. The upper layer then has to select one of the devices to which it wants to connect by issuing the command Aci_Gap_Create_Connection. If privacy is enabled, then either a private resolvable address or a non resolvable address, based on the address type specified in the command is

set as the scanner address but the gap create connection always uses a private resolvable address if the general connection establishment procedure is active.

Table 90. Aci_Gap_Start_General_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Type	1 byte	0x00: Passive scanning 0x01: Active scanning
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Own_Address_Type	1 byte	If Privacy is disabled, then the scanner address should be set as 0x00: Public Device address (default) 0x01: Random Device Address If Privacy is enabled, then the scanner address should be set as- 0x02 : Private Resolvable address 0x03 : Non Resolvable Address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates

Table 91. Aci_Gap_Start_General_Conn_Establishment return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated when the command is issued. If the status is returned as BLE_STATUS_SUCCESS, then on completion of the procedure a Evt_Blue_Gap_Procedure_Completed event is generated with the procedure code set to 0x10. The procedure is terminated when a connection is established or the upper layer terminates the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x10.

4.4.29 Aci_Gap_Start_Selective_Conn_Establishment

Table 92. Aci_Gap_Start_Selective_Conn_Establishment

Command name	Parameters	Return
Aci_Gap_Start_Selective_Conn_Establishment (0xFC9B)	Scan_Type Scan_Interval Scan_Window Own_Address_Type filterDuplicates Num_WhiteList_Entries Addr_Array	Status

Description:

Start a selective connection establishment procedure. The GAP adds the specified device addresses into white list and enables scanning in the controller with the scanner filter policy set to “accept packets only from devices in whitelist”. All the devices found are sent to the upper layer by the event LE_Advertising_Report. The upper layer then has to select one of the devices to which it wants to connect by issuing the command Aci_Gap_Create_Connection.

Table 93. Aci_Gap_Start_Selective_Conn_Establishment command parameters

Parameter	Size	Description
Scan_Type	1 byte	0x00: Passive scanning 0x01: Active scanning
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
filterDuplicates	1 byte	0x00: Do not filter the duplicates (default) 0x01: Filter duplicates
Num_WhiteList_Entries	1 byte	Number of devices that have to be added to the whitelist.
Addr_Array	7 bytes	The addr_array will contain Num_White_List_Entries number of addresses and their address types that have to be added into the whitelist. The format of the addr_array should be address type followed by address.

Table 94. Aci_Gap_Start_General_Conn_Establishment return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated when the command is issued. If the status is returned as BLE_STATUS_SUCCESS, then on completion of the procedure a Evt_Blue_Gap_Procedure_Completed event is generated with the procedure code set to 0x20. The procedure is terminated when a connection is established or the upper layer terminates the procedure by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure code set to 0x20.

4.4.30 Aci_Gap_Create_Connection

Table 95. Aci_Gap_Create_Connection

Command name	Parameters	Return
Aci_Gap_Create_Connection (0xFC9C)	Scan_Interval Scan_Window Peer_Address_Type Peer_Address Own_Address_Type Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max	Status

Description:

Start the direct connection establishment procedure. A LE_Create_Connection call is made to the controller by GAP with the initiator filter policy set to “ignore whitelist and process connectable advertising packets only for the specified device”. The procedure can be terminated explicitly by the upper layer by issuing the command Aci_Gap_Terminate_Gap_Procedure. When a command is issued to terminate the procedure by upper layer, a LE_Create_Connection_Cancel call is made to the controller by GAP.

Table 96. Aci_Gap_Create_Connection command parameters

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, Time = N * 0.625 ms.
Peer_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address
Peer_Address	6 bytes	Address of the peer device with which a connection has to be established.
Own_Address_Type	1 byte	0x00: Public device address (default) 0x01: Random device address

Parameter	Size	Description
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This shall be less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This shall be greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 ms Time range: 100 ms to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Conn_Len_Max	2 bytes	Maximum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.

Table 97. Aci_Gap_Create_Connection return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, on termination of the procedure, a LE_Connection_Complete event is returned. The procedure can be explicitly terminated by the upper layer by issuing the command Aci_Gap_Terminate_Gap_Procedure with the procedure_code set to 0x40.

4.4.31 Aci_Gap_Terminate_Gap_Procedure

Table 98. Aci_Gap_Terminate_Gap_Procedure

Command name	Parameters	Return
Aci_Gap_Terminate_Gap_Procedure (0xFC9D)	Procedure_Code	Status

Description:
The gap procedure specified is terminated.

Table 99. Aci_Gap_Terminate_Gap_Procedure command parameters

Parameter	Size	Description
Procedure_Code	1 byte	0x01: LIMITED_DISCOVERY_PROC 0x02: GENERAL_DISCOVERY_PROC 0x04: NAME_DISCOVERY_PROC 0x08: AUTO_CONNECTION_ESTABLISHMENT_PROC 0x10: GENERAL_CONNECTION_ESTABLISHMENT_PROC 0x20: SELECTIVE_CONNECTION_ESTABLISHMENT_PROC 0x40: DIRECT_CONNECTION_ESTABLISHMENT_PROC 0x80: OBSERVATION_PROC

Table 100. Aci_Gap_Terminate_Gap_Procedure return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER

Event(s) generated:
A command complete event is generated for this command. If the command was successfully processed, the status field will have the value BLE_STATUS_SUCCESS and Evt_Blue_Gap_Procedure_Completed event is returned with the procedure code set to the corresponding procedure.

4.4.32 Aci_Gap_Start_Connection_Update

Table 101. Aci_Gap_Start_Connection_Update

Command name	Parameters	Return
Aci_Gap_Start_Connection_Update (0xFC9E)	Conn_Handle Conn_Interval_Min Conn_Interval_Max Conn_Latency Supervision_Timeout Conn_Len_Min Conn_Len_Max	Status

Description:

Start the connection update procedure. A LE_Connection_Update call is be made to the controller by GAP.

Table 102. Aci_Gap_Start_Connection_Update command parameters

Parameter	Size	Description
Conn_Handle	2 bytes	Handle of the connection for which the update procedure has to be started.
Conn_Interval_Min	2 bytes	Minimum value for the connection event interval. This shall be less than or equal to Conn_Interval_Max. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Interval_Max	2 bytes	Maximum value for the connection event interval. This shall be greater than or equal to Conn_Interval_Min. Range: 0x0006 to 0x0C80 Time = N * 1.25 ms Time range: 7.5 ms to 4 seconds
Conn_Latency	2 bytes	Slave latency for the connection in number of connection events. Range: 0x0000 to 0x01F4
Supervision_Timeout	2 bytes	Supervision timeout for the LE Link. Range: 0x000A to 0x0C80 Time = N * 10 ms Time range: 100 ms to 32 seconds
Conn_Len_Min	2 bytes	Minimum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.
Conn_Len_Max	2 bytes	Maximum length of connection event needed for the LE connection. Range: 0x0000 – 0xFFFF Time = N * 0.625 ms.

Table 103. Aci_Gap_Start_Connection_Update return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated as soon as the command is given. If BLE_STATUS_SUCCESS is returned, on completion of connection update, a LE_Connection_Update_Complete event is returned to the upper layer.

4.4.33 Aci_Gap_Send_Pairing_Request

Table 104. Aci_Gap_Send_Pairing_Request

Command name	Parameters	Return
Aci_Gap_Send_Pairing_Request (0xFC9F)	Conn_Handle Force_Rebond	Status

Description:

Send the SM pairing request to start a pairing process. The authentication requirements and IO capabilities should be set before issuing this command using the Aci_Set_IO_Capabilities and Aci_Set_Authentication_Requirements commands.

Table 105. Aci_Gap_Send_Pairing_Request command parameters

Parameter	Size	Description
Conn_Handle	2 bytes	Handle of the connection for which the pairing request has to be sent.
Force_Rebond	1 byte	The bit 0 decides whether pairing request has to be sent if the device is previously bonded or not. 0: Pairing request is sent only if the device has not previously bonded 1: Pairing request will be sent even if the device was previously bonded. The bit 1 decides whether the link has to be re-encrypted after the key exchange. 0: The link will not be re-encrypted with the LTK at the end of the pairing 1: The link will be re-encrypted with the LTK at the end of the pairing

Table 106. Aci_Gap_Send_Pairing_Request return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command status event is generated when the command is received. If BLE_STATUS_SUCCESS is returned in the command status event, a Evt_Blue_Pairing_Complete event is returned after the pairing process is completed.

4.4.34 Aci_Gap_Resolve_Private_Address

Table 107. Aci_Gap_Resolve_Private_Address

Command name	Parameters	Return
Aci_Gap_Resolve_Private_Address (0xFCA0)	Peer_address	Status address

Description:

This command tries to resolve the address provided with the IRKs present in its database. If the address is resolved successfully with any one of the IRKs present in the database, it returns success and also the corresponding public/static random address stored with the IRK in the database.

Table 108. Aci_Gap_Resolve_Private_Address command parameters

Parameter	Size	Description
Peer_address	6 bytes	Address to be resolved

Table 109. Aci_Gap_Resolve_Private_Address return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x48: The address was not resolved
Address	6 bytes	The public or static random address of the peer device, distributed during pairing phase

Event(s) generated:

A command complete event is generated. If BLE_STATUS_SUCCESS is returned in the status byte, then the address is also returned in the event.

4.4.35 Aci_Gap_Get_Bonded_Devices

Table 110. Aci_Gap_Get_Bonded_Devices

Command name	Parameters	Return
Aci_Gap_Get_Bonded_Devices(0xFCA3)	-	Status Number_of_addresses_to_follow List_of_addr_type-address

Description: This command gets the list of the devices which are bonded. It returns the number of addresses and the corresponding address types and values.

Table 111. Aci_Gap_Get_Bonded_Devices return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED
Number_of_addresses_to_follow	1 byte	The number of address type-address pairs to follow
List_of_addr_type-address	7 bytes	The first byte will indicate the address type: 0x00- Public address 0x01 Random The remaining 6 bytes contain the address of the bonded device
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

Event(s) generated:

A command complete event is returned where the status indicates whether the command was successful. If successful the list of Address_Type-Address pairs follow as specified above.

4.4.36 Aci_Gap_Set_Broadcast_Mode

Table 112. Aci_Gap_Set_Broadcast_Mode

Parameter	Size	Description
Aci_Gap_Set_Broadcast_Mode (0xFCA1)	Advertising_Interval_Min	Status
	Advertising_Interval_Max	
	Advertising_Type	
	Own_Address_Type	
	adv_data_len	
	adv_data	
	Num_WhiteList_Entries	
	whitelist	

Description: this command puts the device into broadcast mode. A privacy enabled device uses either a resolvable private address or a non-resolvable private address as specified in the Own_Addr_Type parameter of the command.

Table 113. Aci_Gap_Set_Broadcast_Mode command parameters

Parameter	Size	Description
Advertising_Interval_Min	2 bytes	Minimum advertising interval. Range: 0x00A0 to 0x4000. Time = N * 0.625 ms. Time range: 100 ms to 10.24 sec
Advertising_Interval_Max	2 bytes	Maximum advertising interval. Range: 0x00A0 to 0x4000. Time = N * 0.625 ms. Time range: 100 ms to 10.24 s
Advertising_Type	1 byte	0x02: scannable undirected advertising 0x03: non-connectable undirected advertising
Own_Address_Type	1 byte	If privacy disabled: 0x00: public address 0x01: static random address. If privacy enabled: 0x02: resolvable private address 0x03: non-resolvable private address
adv_data_len	1 byte	Length of the advertising data in the advertising packet
adv_data	adv_data_len bytes	Advertising data used by the device while advertising
Num_WhiteList_Entries	1 byte	Number of devices to be added to whitelist
whitelist	Num_WhiteLi st_Entries * 7 bytes	The addr_array contains Num_White_List_Entries number of addresses and their address types that have to be added into the whitelist. The format of the addr_array should be address type followed by address.

Table 114. **Aci_Gap_Set_Broadcast_Mode return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS
		0x12: HCI_INVALID_PARAMETER
		0x0C: COMMAND DISALLOWED

Event(s) generated: a command complete event is returned where the status indicates whether the command was successful.

4.4.37 Aci_Gap_Start_Observation_Proc

Table 115. **Aci_Gap_Start_Observation_Proc**

Command name	Parameters	Return
Aci_Gap_Start_Observation_Proc (0xFCA2)	Scan_Type Scan_Interval Scan_Window Own_Address_Type filterDuplicates	Status

Description:

Starts an observation procedure, when the device is in observer role. The host enables scanning in the controller. The advertising reports are sent to the upper layer using standard LE advertising report event. See Bluetooth Core v4.1, Vol. 2, part E, Ch. 7.7.65.2, LE advertising report event.

Command parameters:

Table 116. **Aci_Gap_Start_Observation_Proc commands parameters**

Parameter	Size	Description
Scan_Interval	2 bytes	Time interval from when the Controller started its last LE scan until it begins the subsequent LE scan. The scan interval should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Scan_Window	2 bytes	Amount of time for the duration of the LE scan. Scan_Window shall be less than or equal to Scan_Interval. The scan window should be a number in the range 0x0004 to 0x4000. This corresponds to a time range 2.5 ms to 10240 ms. For a number N, time = N * 0.625 ms.
Scan_Type	1 byte	0x00: passive scanning 0x01: active scanning
Own_Address_Type	1 byte	If Privacy is disabled, then the scanner address is set as- 0x00: public device address (default) 0x01: random device address If privacy is enabled, then the scanner address is set as- 0x02 : private resolvable address 0x03 : Non resolvable address
filterDuplicates	1 byte	0x00: do not filter the duplicates (default) 0x01: filter duplicates

Table 117. **Aci_Gap_Start_Observation_Proc return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: BLE_STATUS_INVALID_PARAMETER 0x0C: ERR_COMMAND_DISALLOWED

4.4.38 Aci_Gap_Is_Device_Bonded

Table 118. **Aci_Gap_Is_Device_Bonded**

Command name	Parameters	Return
Aci_Gap_Is_Device_Bonded (0xFCA4)	Peer_address_type Peer_address	Status

Description:

The command finds whether the device, whose address is specified in the command, is bonded. If the device is using a resolvable private address and it has been bonded, then the command will return BLE_STATUS_SUCCESS.

Table 119. **Aci_Gap_Is_Device_Bonded command parameters**

Parameter	Size	Description
Peer_address_type	1byte	The address type of the peer device
Peer_address	6 bytes	Address used by the peer device while advertising.

Table 120. **Aci_Gap_Is_Device_Bonded return parameters**

Parameter	Size	Description
Status	1 byte	0x00 : BLE_STATUS_SUCCESS 0x12 : HCI_INVALID_PARAMETER 0x5C: DEV_NOT_FOUND_IN_DATABASE 0x49 : FLASH_READ_FAILED

Event(s) generated:

A command complete event is generated. If BLE_STATUS_SUCCESS is returned in the status byte of the event then the device has been bonded.

4.5 GAP VS events

Table 121. **GAP VS events**

Item	Command	EGID	EID	ECODE
1	Evt_Blue_Gap_Limited_Discoverable	0x01	0x00	0x0400
2	Evt_Blue_Gap_Pairing_Complete	0x01	0x01	0x0401
3	Evt_Blue_Pass_Key_Request	0x01	0x02	0x0402
4	Evt_Blue_Authorization_Request	0x01	0x03	0x0403
5	Evt_Blue_Slave_Security_Initiated	0x01	0x04	0x0404
6	Evt_Blue_Gap_Bond_Lost	0x01	0x05	0x0405
7	Evt_Blue_Gap_Procedure_Complete	0x01	0x07	0x0407
8	Evt_Blue_Gap_Addr_Not_Resolved	0x01	0x08	0x0408

4.5.1 Evt_Blue_Gap_Limited_Discoverable

This event is generated by the controller when the limited discoverable mode ends due to timeout. The timeout is 180 seconds.

4.5.2 Evt_Blue_Gap_Pairing_Complete

This event is generated when the pairing process has completed successfully or a pairing procedure timeout has occurred or the pairing has failed. This is to notify the application that we have paired with a remote device so that it can take further actions or to notify that a timeout has occurred so that the upper layer can decide to disconnect the link.

Table 122. **Evt_Blue_Gap_Pairing_Complt**

Parameter	Size	Description
Event code	2 bytes	0x0401
Connection_handle	2 bytes	Connection handle on which the pairing procedure completed
Status	1 byte	0x00: Pairing success Pairing with a remote device was successful 0x01: pairing timeout The SMP timeout has elapsed and no further SMP commands are processed until reconnection 0x02: pairing failed The pairing failed with the remote device

4.5.3 Evt_Blue_Pass_Key_Request

This event is generated by the security manager to the application when a passkey is required for pairing. When this event is received, the application has to respond with the Aci_Gap_Pass_Key_Response command.

Table 123. **Evt_Blue_Pass_Key_Request**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the passkey has been requested

4.5.4 Evt_Blue_Authorization_Request

This event is generated by the Security manager to the application when the application has set that authorization is required for reading/writing of attributes. This event is generated as soon as the pairing is complete. When this event is received, Aci_Gap_Authorization_Response command should be used to respond by the application.

Table 124. **Evt_Blue_Authorization_Request**

Parameter	Size	Description
Event code	2 bytes	0x0403
Connection_handle	2 bytes	Connection handle for which authorization is being requested

4.5.5 Evt_Blue_Slave_Security_Initiated

This event is generated when the slave security request is successfully sent to the master.

4.5.6 Evt_Blue_Gap_Bond_Lost

This event is generated when a pairing request is issued in response to a slave security request from a master which has previously bonded with the slave. When this event is received, the upper layer has to issue the command Aci_Gap_Allow_Rebond in order to allow the slave to continue the pairing process with the master.

4.5.7 Evt_Blue_Gap_Procedure_Complete

This event is sent by the GAP to the upper layers when a procedure previously started has been terminated by the upper layer or has completed for any other reason

Table 125. **Evt_Blue_Gap_Procedure_Complete**

Parameter	Size	Description
Event code	2 bytes	0x0407
Procedure_Code	1 byte	0x01: LIMITED_DISCOVERY_PROC 0x02: GENERAL_DISCOVERY_PROC 0x04: NAME_DISCOVERY_PROC 0x08: AUTO_CONNECTION_ESTABLISHMENT_PROC 0x10: GENERAL_CONNECTION_ESTABLISHMENT_PROC 0x20: SELECTIVE_CONNECTION_ESTABLISHMENT_PROC 0x40: DIRECT_CONNECTION_ESTABLISHMENT_PROC
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x41: BLE_STATUS_FAILED 0x05: ERR_AUTH_FAILURE, Procedure failed due to authentication requirements
Procedure_Specific_Data	1 or more bytes	Procedure Aci_Gap_Name_Discovery_Proc: The name of the peer device if the procedure completed successfully Procedure Aci_Gap_General_Conn_Establishment_Proc: The reconnection address written to the peripheral device if the peripheral is privacy enabled

4.5.8 Evt_Blue_Gap_Addr_Not_Resolved

This event is sent only by a privacy enabled peripheral. The event is sent to the upper layers when the peripheral is unsuccessful in resolving the resolvable address of the peer device after connecting to it.

Table 126. **Evt_Blue_Gap_Addr_Not_Resolved**

Parameter	Size	Description
Event Code	2 bytes	0x0408
Connection_handle	2 bytes	Connection handle for which the private address could not be resolved with any of the stored IRK's.

4.6 GATT VS commands

Table 127. **GATT VS commands**

Item	Command	CGID	CID	OpCode
1	Aci_Gatt_Init	0x02	0x01	0xFD01
2	Aci_Gatt_Add_Serv	0x02	0x02	0xFD02
3	Aci_Gatt_Include_Service	0x02	0x03	0xFD03
4	Aci_Gatt_Add_Char	0x02	0x04	0xFD04
5	Aci_Gatt_Add_Char_Desc	0x02	0x05	0xFD05
6	Aci_Gatt_Update_Char_Value	0x02	0x06	0xFD06
7	Aci_Gatt_Del_Char	0x02	0x07	0xFD07
8	Aci_Gatt_Del_Service	0x02	0x08	0xFD08
9	Aci_Gatt_Del_Include_Service	0x02	0x09	0xFD09
10	Aci_Gatt_Set_Event_Mask	0x02	0x0A	0xFD0A
11	Aci_Gatt_Exchange_configuration	0x02	0x0B	0xFD0B
12	Aci_Att_Find_Information_Req	0x02	0x0C	0xFD0C
13	Aci_Att_Find_By_Typ_Value_Req	0x02	0x0D	0xFD0D
14	Aci_Att_Read_By_Type_Req	0x02	0x0E	0xFD0E
15	Aci_Att_Read_By_Group_Type_Req	0x02	0x0F	0xFD0F
16	Aci_Att_Prepare_Write_Req	0x02	0x10	0xFD10
17	Aci_Att_Execute_Write_Req	0x02	0x11	0xFD11
18	Aci_Gatt_Disc_All_Prim_Services	0x02	0x12	0xFD12
19	Aci_Gatt_Disc_Prim_serv_By_UUID	0x02	0x13	0xFD13
20	Aci_Gatt_Find_Included_Services	0x02	0x14	0xFD14
21	Aci_Gatt_Disc_All_Charac_Of_Serv	0x02	0x15	0xFD15
22	Aci_Gatt_Disc_Charac_By_UUID	0x02	0x16	0xFD16
23	Aci_Gatt_Disc_All_Charac_Descriptors	0x02	0x17	0xFD17
24	Aci_Gatt_Read_Charac_Val	0x02	0x18	0xFD18
25	Aci_Gatt_Read_Charac_Using_UUID	0x02	0x19	0xFD19
26	Aci_Gatt_Read_Long_Charac_Val	0x02	0x1A	0xFD1A
27	Aci_Gatt_Read_Multiple_Charac_Values	0x02	0x1B	0xFD1B
28	Aci_Gatt_Write_Charac_Val	0x02	0x1C	0xFD1C
29	Aci_Gatt_Write_Long_Charac_Val	0x02	0x1D	0xFD1D
30	Aci_Gatt_Write_Charac_Reliable	0x02	0x1E	0xFD1E
31	Aci_Gatt_Write_Long_Charac_Descriptor	0x02	0x1F	0xFD1F
32	Aci_Gatt_Read_Long_Charac_Descriptor	0x02	0x20	0xFD20
33	Aci_Gatt_Write_Charac_Descriptor	0x02	0x21	0xFD21
34	Aci_Gatt_Read_Charac_Descriptor	0x02	0x22	0xFD22
35	Aci_Gatt_Write_Without_Response	0x02	0x23	0xFD23
36	Aci_Gatt_Signed_Write_Without_Resp	0x02	0x24	0xFD24
37	Aci_Gatt_Confirm_Indication	0x02	0x25	0xFD25

Item	Command	CGID	CID	OpCode
38	Aci_Gatt_Write_Resp	0x02	0x26	0xFD26
39	Aci_Gatt_Allow_Read	0x02	0x27	0xFD27
40	Aci_Gatt_Set_Security_Permission	0x02	0x28	0xFD28
41	Aci_Gatt_Set_Desc_Value	0x02	0x29	0xFD29
42	Aci_Gatt_Read_Handle_Value	0x02	0x2A	0xFD2A
43	Aci_Gatt_Read_Handle_Value_Offset	0x02	0x2B	0xFD2B
44	Aci_Gatt_Update_Char_Value_Ext	0x02	2C	0xFD2C

4.6.1 Aci_Gatt_Init

Table 128. Aci_Gatt_Init

Command name	Parameters	Return
Aci_Gatt_Init (0xFD01)		Status

Description:

Initialize the GATT server on the slave device. Initialize all the pools and active nodes. Also it adds GATT service with service changed characteristic. Until this command is issued the GATT channel does not process any commands even if the connection is opened. This command has to be given before using any of the GAP features.

Table 129. Aci_Gatt_Init return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error

4.6.2 Aci_Gatt_Add_Serv

Table 130. Aci_Gatt_Add_Serv

Command name	Parameters	Return
Aci_Gatt_Add_Serv (0xFD02)	Service_UUID_Type Service_UUID Service_Type Max_Attribute_Records	Status Service_handle

Description:

Add a service to GATT Server. When a service is created in the server, the host needs to reserve the handle ranges for this service using Max_Attribute_Records parameter. This parameter specifies the maximum number of attribute records that can be added to this service (including the service attribute, include attribute, characteristic attribute, characteristic value attribute and characteristic descriptor attribute). Handle of the created service is returned in command complete event.

Table 131. **Aci_Gatt_Add_Serv command parameters**

Parameter	Size	Description
Service_UUID_Type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
Service_UUID	2 bytes or 16 bytes	16-bit or 128-bit UUID based on the UUID type field
Service_Type	1 byte	0x01: Primary service 0x02: Secondary service
Max_Attribute_ Records	1 byte	Maximum number of attribute records that can be added to this service

Table 132. **Aci_Gatt_Add_Service return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of memory 0x61: Invalid parameter 0x62: Out of handle 0x64: Insufficient resources
Service_handle	2 bytes	Handle of the service When this service is added to the service, a handle is allocated by the server to this service. Also server allocates a range of handles for this service from Service_Handle to (Service_Handle + Max_Attribute_Records)

Event(s) generated:

When the Aci_Gatt_Add_Service command is completed, controller generates a command complete event with status and handle for the service as parameters.

4.6.3 Aci_Gatt_Include_Service

Table 133. **Aci_Gatt_Include_Service**

Command name	Parameters	Return
Aci_Gatt_Include_Service (0xFD03)	Service_Handle Include_Start_Handle Include_End_Handle Included_Uuid_Type Included_Uuid	Status Included_handle

Description:

Include a service given by Service_Include_Handle to another service given by Service_Handle. Attribute server creates an INCLUDE definition attribute and return the handle of this in command complete event as Included_Handle.

Table 134. **Aci_Gatt_Include_Service command parameters**

Parameter	Size	Description
Service_handle	2 bytes	Handle of the service to which another service has to be included
Include_Start_Handle	2 bytes	Start handle of the service which has to be included in service
Include_End_Handle	2 bytes	End handle of the service which has to be included in service
Include_UUID_Type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
Include_UUID	2 bytes or 16 bytes	16-bit or 128-bit UUID of the service

Table 135. **Aci_Gatt_Include_Service return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of memory 0x60: Invalid handle 0x61: Invalid parameter 0x62: Out of handle 0x63: Invalid operation 0x64: Insufficient resources
Included_handle	2 bytes	Handle of the included declaration

Event(s) generated:

When the Aci_Gatt_Add_Service command is completed, controller generates a command complete event with status and handle for the service as parameters.

4.6.4 Aci_Gatt_Add_Characteristic

Table 136. **Aci_Gatt_Add_Char**

Command name	Parameters	Return
Aci_Gatt_Add_Characteristic (0xFD04)	Service_Handle Char_UUID_Type Char_UUID Char_Value_Length Char_Properties Security_Permissions Gatt_Evt_Mask Encryption_Key_Size isVariable	Char_handle

Description:

Add a characteristic to a service.

Table 137. Aci_Gatt_Add_Characteristic command parameters

Parameter	Size	Description
Service_Handle	2 bytes	Handle of the service to which the characteristic has to be added
Char_UUID_Type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
Char_UUID	2 bytes or 16 bytes	16-bit or 128-bit UUID
Char_Value_Length	2 bytes	Maximum length of the characteristic value. Note that 1 byte for the BlueNRG-MS FW stack version < 7.2
Char_Properties	1 byte	Bitwise OR values of characteristic properties (defined in Volume 3, Section 3.3.3.1 of Bluetooth Specification 4.1)
Security_Permissions	1 byte	0x00: ATTR_PERMISSION_NONE 0x01: Need authentication to read 0x02: Need authorization to read 0x04: Link should be encrypted to read 0x08: Need authentication to write 0x10: Need authorization to write 0x20: Link should be encrypted for write
Gatt_Evt_Mask	1 byte	0x01: GATT_SERVER_ATTR_WRITE The application is notified when a client writes to this attribute 0x02: GATT_INTIMATE_AND_WAIT_FOR_APPL_AUTH The application is notified when a write request/write cmd/signed write cmd is received by the server for this attribute 0x04: GATT_INTIMATE_APPL_WHEN_READ_N_WAIT The application is notified when a read request of any type is got for this attribute
Encryption_Key_Size	1 byte	The minimum encryption key size requirement for this attribute. Valid range: 7 to 16
isVariable	1 byte	0x00: The attribute has a fixed length value field 0x01: The attribute has a variable length value field

Table 138. Aci_Gatt_Add_Characteristic return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of Memory 0x60: Invalid handle 0x61: Invalid parameter 0x62: Out of handle 0x64: Insufficient resources 0x66: Character Already Exists
Char_Handle	2 bytes	Handle of the Characteristic

Event(s) generated:

When the command is completed, a command complete event is generated by the controller which carries the status of the command and the handle of the characteristic as parameters.

4.6.5 Aci_Gatt_Add_Char_Desc

Table 139. Aci_Gatt_Add_Char_Desc

Command name	Parameters	Return
Aci_Gatt_Add_Char_Desc (0xFD05)	Service_Handle Char_Handle Char_Desc_UUID_Type Char_Desc_UUID Char_Desc_Value_Max_Length Char_Desc_Value_Length Char_Desc_Value Security_Permissions Access_Permissions Gatt_Evt_Mask Encryption_Key_Size isVariable	Status Char_desc_handle

Description:
Add a characteristic descriptor to a service.

Table 140. Aci_Gatt_Add_Char_Desc command parameters

Parameter	Size	Description
Service_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic to which description is to be added
Char_Desc_UUID type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
Char_Desc_UUID	2 bytes or 16 bytes	0x2900: Characteristic extended properties descriptor 0x2901: Characteristic user descriptor 0x2902: Client configuration descriptor 0x2903: Server configuration descriptor 0x2904: Characteristic presentation format 0x2905: Characteristic aggregated format 0xXXXX: 16/128 bit user specific descriptor
Char_Desc_Value_Max_Length	1 byte	The maximum length of the descriptor value
Char_Desc_Value_Length	1 byte	Current length of the characteristic descriptor value
Char_Desc_Value	0-N Bytes	Value of the characteristic description
Security_Permissions	1 byte	0x00: No security permission 0x01: Authentication required 0x02: Authorization required 0x04: Encryption required

Parameter	Size	Description
Access_Permissions	1 byte	0x00: No access 0x01: Readable 0x02: Writable 0x03: Read/write
Gatt_Event_Mask	1 byte	0x01: GATT_SERVER_ATTR_WRITE The application is notified when a client writes to this attribute 0x02: GATT_INTIMATE_AND_WAIT_FOR_APPL_AUTH The application is notified when a write request/write cmd/signed write cmd is received by the server for this attribute 0x04: GATT_INTIMATE_APPL_WHEN_READ_N_WAIT The application is notified when a read request of any type is got for this attribute
Encryption_Key_Size	1 byte	The minimum encryption key size requirement for this attribute. Valid range: 7 to 16
isVariable	1 byte	0x00: The attribute has a fixed length value field 0x01: The attribute has a variable length value field

Table 141. Aci_Gatt_Add_Char_Desc return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x1F: Out of memory 0x60: Invalid handle 0x61: Invalid parameter 0x62: Out of handle 0x63: Invalid operation 0x64: Insufficient resources
Char_Desc_Handle	2 bytes	Handle of the characteristic descriptor

Event(s) generated:

When this command is completed, a command complete event is generated by the controller which carries the status of the command and the handle of the characteristic descriptor.

4.6.6 Aci_Gatt_Update_Char_Value

Table 142. Aci_Gatt_Update_Char_Value

Command name	Parameters	Return
Aci_Gatt_Update_Char_Value (0xFD06)	Serv_Handle Char_Handle Val_Offset Char_Value_Length Char_Value	Status

Description:
Update a characteristic value in a service.

Table 143. Aci_Gatt_Update_Char_Value command parameters

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic
Val_Offset	1 byte	The offset from which the attribute value has to be updated. If this is set to 0, and the attribute value is of variable length, then the length of the attribute is set to the Char_Value_Length. If the Val_Offset is set to a value greater than 0, then the length of the attribute is set to the maximum length as specified for the attribute while adding the characteristic.
Char_Value_Length	1 byte	Length of the characteristic value in octets
Char_Value	0-N bytes	Characteristic value

Table 144. Aci_Gatt_Update_Char_Value return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:
When the command has completed, the controller will generate a command complete event.

4.6.7 Aci_Gatt_Del_Char

Table 145. Aci_Gatt_Del_Characteristic

Command name	Parameters	Return
Aci_Gatt_Del_Characteristic (0xFD07)	Serv_Handle Char_Handle	Status

Description:
Delete the characteristic specified from the service.

Table 146. Aci_Gatt_Del_Char command parameters

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic to description to be deleted

Table 147. **Aci_Gatt_Del_Char return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:

When the Aci_Gatt_Del_Characteristic command has completed, the controller generates a command complete event.

4.6.8 Aci_Gatt_Del_Service

Table 148. **Aci_Gatt_Del_Service**

Command name	Parameters	Return
Aci_Gatt_Del_Service (0xFD08)	Service_handle	Status

Description:

Delete the service specified from the GATT server database.

Table 149. **Aci_Gatt_Del_Service command parameters**

Parameter	Size	Description
Service_Handle	2 bytes	Handle of the service to be deleted

Table 150. **Aci_Gatt_Del_Service return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:

When the command has completed, the controller generates a command complete event.

4.6.9 Aci_Gatt_Del_Include_Service

Table 151. Aci_Gatt_Del_Include_Service

Command name	Parameters	Return
Aci_Gatt_Del_Include_Service	Serv_Handle Include_Handle	Status

Description:
Delete the include definition from the service.

Table 152. Aci_Gatt_Del_Include_Service command parameters

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which include definition belongs
Include_Handle	2 bytes	Handle of the Included definition to be deleted

Table 153. Aci_Gatt_Del_Include_Service return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

Event(s) generated:
When the command has completed, the controller generates a command complete event.

4.6.10 Aci_Gatt_Set_Event_Mask

Table 154. Aci_Gatt_Set_Event_Mask

Command name	Parameters	Return
Aci_Gatt_Set_Event_Mask (0xFD0A)	Event mask	Status

Description:
It allows masking events from the GATT. The default configuration is all the events masked.

Table 155. **Aci_Gatt_Set_Event_Mask command parameters**

Parameter	Size	Description
Event mask	4 bytes	0x00000001: Evt_Blue_Gatt_Attribute_modified 0x00000002: Evt_Blue_Gatt_Procedure_Timeout 0x00000004: Evt_Blue_Att_Exchange_MTU_Resp 0x00000008: Evt_Blue_Att_Find_Information_Resp 0x00000010: Evt_Blue_Att_Find_By_Type_Value_Resp 0x00000020: Evt_Blue_Att_Read_By_Type_Resp 0x00000040: Evt_Blue_Att_Read_Resp 0x00000080: Evt_Blue_Att_Read_Blob_Resp 0x00000100: Evt_Blue_Att_Read_Multiple_Resp 0x00000200: Evt_Blue_Att_Read_By_Group_Resp 0x00000800: Evt_Blue_Att_Prepare_Write_Resp 0x00001000: Evt_Blue_Att_Exec_Write_Resp 0x00002000: Evt_Blue_Gatt_Indication 0x00004000: Evt_Blue_Gatt_notification 0x00008000: Evt_Blue_Gatt_Error_Resp 0x00010000: Evt_Blue_Gatt_Procedure_Complete 0x00020000: Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp 0x00040000: Evt_Blue_Gatt_Tx_Pool_Available

Table 156. **Aci_Gatt_Set_Event_Mask return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid HCI command parameters

Event(s) generated:
A command complete event is generated on the completion of the command.

4.6.11 Aci_Gatt_Exchange_Configuration

Table 157. **Aci_Gatt_Exchange_Configuration**

Command name	Parameters	Return
Aci_Gatt_Exchange_Configuration (0xFD0B)	Connection_handle	Status

Description:
Perform an ATT MTU exchange.

Table 158. **Aci_Gatt_Exchange_Configuration command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given

Table 159. **Aci_Gatt_Exchange_Configuration return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the ATT MTU exchange procedure is completed, a Evt_Blue_Att_Exchange_MTU_Resp event is generated. Also procedure complete event is generated to indicate end of procedure.

4.6.12 Aci_Att_Find_Information_Req

Table 160. **Aci_Att_Find_Information_Req**

Command name	Parameters	Return
Aci_Att_Find_Information_Req (0xFD0C)	Connection_handle Start_handle End_handle	Status

Description:

Post the fnd information request.

Table 161. **Aci_Att_Find_Information_Req command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_handle	2 bytes	Starting handle of the range of attributes to be discovered on the server
End_handle	2 bytes	Ending handle of the range of attributes to be discovered on the server

Table 162. **Aci_Att_Find_Information_Req return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_ Att_Find_Information_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.6.13 Att_Find_By_Type_Value_Req

Table 163. Att_Find_By_Type_Value_Req

Command name	Parameters	Return
Att_Find_By_Type_Value_Req (0xFD0D)	Connection_handle Start_handle End_handle UUID AttrValLen AttrVal	Status

Description:

Post the find by type value request.

Table 164. Aci_Att_Find_Attr_By_Typ_And_Value_Req command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_handle	2 bytes	Starting handle of the range of attributes to be discovered on the server
End_handle	2 bytes	Ending handle of the range of attributes to be discovered on the server
UUID	2 bytes	16-bit UUID
AttrValLen	1 byte	Length of the attribute value to follow. Note: Though the max attribute value that is allowed according to the spec is 512 octets, due to the limitation of the transport layer (command packet max length is 255 bytes) the value is limited to 255 bytes
AttrVal	0-N bytes	Value of the attribute to be matched

Table 165. Aci_Att_Find_Attr_By_Typ_And_Value_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_ Att_Find_By_Type_Value_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.6.14 Aci_Att_Read_Attr_By_Type_Req

Table 166. Aci_Att_Read_By_Type_Req

Command name	Parameters	Return
Aci_Att_Read_By_Type_Req (0xFD0E)	Connection_handle Start_handle End_handle UUID_type UUID	Status

Description:
Send a Read By Type Request.

Table 167. Aci_Att_Read_By_Type_Req command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_handle	2 bytes	Start handle of the range of values to be read on the server
End_handle	2 bytes	End handle of the range of values to be read on the server
UUID_type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 168. Aci_Att_Read_By_Typ_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:
A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Type_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.6.15 Aci_Att_Read_By_Group_Type_Req

Table 169. Aci_Att_Read_By_Group_Type_Req

Command name	Parameters	Return
Aci_Att_Read_By_Group_Type_Req (0xFD0F)	Connection_handle Start_handle End_handle UUID_type UUID	Status

Description:

It sends a Read By Group Type Request.

Table 170. Aci_Att_Read_By_Grp_Type_Req command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_handle	2 bytes	Start handle of the range of values to be read on the server
End_handle	2 bytes	End handle of the range of values to be read on the server
UUID_type	1 byte	0x01: 16-bit UUID 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 171. Aci_Att_Read_By_Grp_Type_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Group_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

The Read By Group Type Request is used to obtain the values of grouping attributes where the attribute type is known but the handle is not known. Grouping attributes are defined at GATT layer.

The grouping attribute types are: Primary Service, Secondary Service and Characteristic.

4.6.16 Aci_Att_Prepare_Write_Req

Table 172. Aci_Att_Prepare_Write_Req

Command name	Parameters	Return
Aci_Att_Prepare_Write_Req (0xFD10)	Connection_handle AttrHandle valOffset AttrValLen AttrVal	Status

Description:
It sends a Prepare Write Request.

Table 173. Aci_Att_Prepare_Write_Req command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute whose value has to be written
valOffset	2 bytes	The offset at which value has to be written
AttrValLen	1 byte	Length of the attribute value (maximum value is ATT_MTU - 5)
AttrVal	0-N bytes	Value of the attribute to be written

Table 174. Aci_Att_Prepare_Write_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:
A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Prepare_Write_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.6.17 Aci_Att_Execute_Write_Req

Table 175. Aci_Gatt_Execute_Write_Req

Command name	Parameters	Return
Aci_Att_Execute_Write_Req (0xFD11)	Connection_handle Execute	Status

Description:
It sends an Execute Write Request.

Table 176. Aci_Att_Execute_Write_Req command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Execute	1 byte	0x00 – Cancel all prepared writes 0x01 – Immediately write all pending prepared values

Table 177. Aci_Att_Execute_Write_Req return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If ATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:
A command status event is generated on the receipt of the command. The result of the procedure is given through the Evt_Blue_Att_Exec_Write_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.6.18 Aci_Gatt_Disc_All_Prim_Services

Table 178. Aci_Gatt_Disc_All_Prim_Services

Command name	Parameters	Return
Aci_Gatt_Disc_All_Prim_Services (0xFD12)	Connection_handle	Status

Description:
This command starts the GATT client procedure to discover all primary services on the server.

Table 179. **Aci_Gatt_Disc_All_Prim_Services command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given

Table 180. **Aci_Gatt_Disc_All_Prim_Services return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Group_Resp event. The end of the procedure is indicated by a Evt_Blue_Gatt_Procedure_Complete event.

4.6.19 Aci_Gatt_Disc_Prim_Service_By_UUID

Table 181. **Aci_Gatt_Disc_Prim_Service_By_Uuid**

Command name	Parameters	Return
Aci_Gatt_Disc_Prim_Serv_By_Uuid (0xFD13)	Connection_handle UUID_type UUID	Status

Description:

This command starts the procedure to discover the primary services of the specified UUID on the server.

Table 182. **Aci_Gatt_Disc_Prim_serv_By_UUID command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
UUID_type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 183. **Aci_Gatt_Disc_Prim_Service_By_Uuid return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Find_By_Type_Value_Resp event. The end of the procedure is indicated by a Evt_Blue_Procedure_Complete event.

4.6.20 Aci_Gatt_Find_Included_Services

Table 184. **Aci_Gatt_Find_Included_Services**

Command name	Parameters	Return
Aci_Gatt_Find_Included_Services (0xFD14)	Connection_handle Start_handle End_handle	Status

Description:

Start the procedure to find all included services.

Table 185. **Aci_Gatt_Find_Included_Services command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_handle	2 bytes	Start handle of the service
End_handle	2 bytes	End handle of the service

Table 186. **Aci_Gatt_Find_Included_Services return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. The responses of the procedure are given through the Evt_Blue_Att_Read_By_Type_Resp event. The end of the procedure is indicated by a Evt_Blue_Gatt_Procedure_Complete event.

4.6.21 Aci_Gatt_Disc_All_Charac_Of_Serv

Table 187. **Aci_Gatt_Disc_All_Charac_Of_Serv**

Command name	Parameters	Return
Aci_Gatt_Disc_All_Charac_Of_Serv (0xFD15)	Connection_handle Start_Attr_handle End_Attr_handle	Status

Description:

Start the procedure to discover all the characteristics of a given service.

Table 188. **Aci_Gatt_Disc_All_Charac_Of_Serv command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_Attr_handle	2 bytes	Start attribute handle of the service
End_Attr_handle	2 bytes	End attribute handle of the service

Table 189. **Aci_Gatt_Disc_All_Charac_Of_Serv return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_By_Type_Resp event.

4.6.22 Aci_Gatt_Disc_Charac_By_UUID

Table 190. **Aci_Gatt_Disc_Charac_By_UUID**

Command name	Parameters	Return
Aci_Gatt_Disc_Charac_By_UUID (0xFD16)	Connection_handle Start_Attr_handle End_Attr_handle UUID_type UUID	Status

Description:

Start the procedure to discover all the characteristics specified by the UUID.

Table 191. **Aci_Gatt_Disc_Charac_By_UUID command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_Attr_handle	2 bytes	Start attribute handle of the service
End_Attr_handle	2 bytes	End attribute handle of the service
UUID_type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 192. **Aci_Gatt_Disc_Charac_By_UUID return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp event.

4.6.23 Aci_Gatt_Disc_All_Charac_Descriptors

Table 193. **Aci_Gatt_Disc_All_Charac_Descriptors**

Command name	Parameters	Return
Aci_Gatt_Disc_All_Charac_Descriptors (0xFD17)	Connection_handle charValHandle EndHandle	Status

Description:

Start the procedure to discover all characteristic descriptors on the server.

Table 194. **Aci_Gatt_Disc_All_Charac_Descriptors command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
charValHandle	2 bytes	Handle of the characteristic value
EndHandle	2 bytes	End handle of the characteristic

Table 195. **Aci_Gatt_Disc_All_Charac_Descriptors return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Find_Info_Resp event.

4.6.24 Aci_Gatt_Read_Charac_Val

Table 196. **Aci_Gatt_Read_Charac_Val**

Command name	Parameters	Return
Aci_Gatt_Read_Charac_Val (0xFD18)	Connection_handle attrHandle	Status

Description:

Start the procedure to read the attribute value.

Table 197. **Aci_Gatt_Read_Charac_Val command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the characteristic to be read

Table 198. **Aci_Gatt_Read_Charac_Val return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packet is given through Evt_Blue_Att_Read_Resp event.

4.6.25 Aci_Gatt_Read_Charac_Using_UUID

Table 199. Aci_Gatt_Read_Charac_Using_UUID

Command name	Parameters	Return
Aci_Gatt_Read_Charac_Using_UUID (0xFD19)	Connection_handle Start_handle End_handle UUID_type UUID	Status

Description:

Start the procedure to read all the characteristics specified by the UUID.

Table 200. Aci_Gatt_Read_Charac_Using_UUID command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Start_handle	2 bytes	Starting handle of the range to be searched
End_handle	2 bytes	End handle of the range to be searched
UUID_type	1 byte	0x01: 16-bit UUID (default) 0x02: 128-bit UUID
UUID	2 bytes or 16 bytes	UUID

Table 201. Aci_Gatt_Read_Charac_Using_UUID return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp event.

4.6.26 Aci_Gatt_Read_Long_Charac_Val

Table 202. Aci_Gatt_Read_Long_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Read_Long_Charac_Val (0xFD1A)	Connection_handle attrHandle valOffset	Status

Description:
Start the procedure to read a long characteristic value.

Table 203. Aci_Gatt_Read_Long_Charac_Val command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
attrHandle	1 byte	Handle of the characteristic to be read
valOffset	2 bytes	Offset from which the value needs to be read

Table 204. Aci_Gatt_Read_Long_Charac_Val command return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:
A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_Blob_Resp event.

4.6.27 Aci_Gatt_Read_Multiple_Charac_Values

Table 205. Aci_Gatt_Read_Multiple_Charac_Values

Command name	Parameters	Return
Aci_Gatt_Read_Multiple_Charac_Values (0xFD1B)	Connection_handle numHandles setOfHandles	Status

Description:

Start a procedure to read multiple characteristic values from a server.

This sub-procedure is used to read multiple characteristic values from a server when the client knows the characteristic value handles.

Table 206. Aci_Gatt_Read_Multiple_Charac_Values command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
numHandles	1 byte	The number of handles for which the value has to be read
setOfHandles	numHandle * 2 bytes	The handles for which the attribute value has to be read

Table 207. Aci_Gatt_Read_Multiple_Charac_Values return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_Multiple_Resp event.

4.6.28 Aci_Gatt_Write_Charac_Val

Table 208. Aci_Gatt_Write_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Write_Charac_Val (0xFD1C)	Connection_handle attrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value.

Table 209. Aci_Gatt_Write_Charac_Val command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given

Parameter	Size	Description
attrHandle	2 bytes	Handle of the characteristic to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 210. Aci_Gatt_Write_Charac_Val return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated.

4.6.29 Aci_Gatt_Write_Long_Charac_Val

Table 211. Aci_Gatt_Write_Long_Charac_Val

Command name	Parameters	Return
Aci_Gatt_Write_Long_Charac_Val (0xFD1D)	Connection_handle AttrHandle ValOffset ValLen AttrVal	Status

Description:

Start the procedure to write a long characteristic value.

Table 212. Aci_Gatt_Write_Long_Charac_Val command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValOffset	2 bytes	Offset at which the attribute has to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 213. **Aci_Gatt_Write_Long_Charac_Val return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Prepare_Write_Resp and Evt_Blue_Att_Exec_Write_Resp events.

4.6.30 Aci_Gatt_Write_Charac_Reliable

Table 214. **Aci_Gatt_Write_Charac_Reliable**

Command name	Parameters	Return
Aci_Gatt_Write_Charac_Reliable (0xFD1E)	Connection_handle AttrHandle ValOffset ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic reliably.

Table 215. **Aci_Gatt_Write_Charac_Reliable command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValOffset	2 bytes	Offset at which the attribute has to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 216. **Aci_Gatt_Write_Charac_Reliable return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Prepare_Write_Resp and Evt_Blue_Att_Exec_Write_Resp events.

4.6.31 Aci_Gatt_Write_Long_Charac_Desc

Table 217. **Aci_Gatt_Write_Long_Charac_Descriptor**

Command name	Parameters	Return
Aci_Gatt_Write_Long_Charac_Descriptor (0xFD1F)	Connection_handle attrHandle ValOffset ValLen AttrVal	Status

Description:

Start the procedure to write a long characteristic descriptor.

Table 218. **Aci_Gatt_Write_Long_Charac_Descriptor command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the attribute to be written
ValOffset	2 bytes	Offset at which the attribute has to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 219. **Aci_Gatt_Write_Long_Charac_Desc return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through and Evt_Blue_Att_Prepare_Write_Resp and Evt_Blue_Att_Exec_Write_Resp event.

4.6.32 Aci_Gatt_Read_Long_Charac_Desc

Table 220. **Aci_Gatt_Read_Long_Charac_Desc**

Command name	Parameters	Return
Aci_Gatt_Read_Long_Charac_Descriptor (0xFD20)	Connection_handle attrHandle valOffset	Status

Description:

Start the procedure to read a long characteristic value.

Table 221. **Aci_Gatt_Read_Long_Charac_Descriptor command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
attrHandle	2 bytes	Handle of the characteristic descriptor
ValOffset	2 bytes	Offset from which the value needs to be read

Table 222. **Aci_Gatt_Read_Long_Charac_Descriptor return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packets are given through Evt_Blue_Att_Read_Blob_Resp event.

4.6.33 Aci_Gatt_Write_Charac_Descriptor

Table 223. **Aci_Gatt_Write_Charac_Descriptor**

Command name	Parameters	Return
Aci_Gatt_Write_Charac_Descriptor (0xFD21)	Connection_handle AttrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value.

Table 224. **Aci_Gatt_Write_Charac_Descriptor command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 225. **Aci_Gatt_Write_Charac_Descriptor return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated.

4.6.34 Aci_Gatt_Read_Charac_Desc

Table 226. **Aci_Gatt_Read_Charac_Descriptor**

Command name	Parameters	Return
Aci_Gatt_Read_Charac_Descriptor (0xFD22)	Connection_handle AttrHandle	Status

Description:

Start the procedure to read the descriptor specified.

Table 227. **Aci_Gatt_Read_Charac_Descr command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the descriptor to be read

Table 228. **Aci_Gatt_Read_Charac_Descriptor return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - If the exchange has already taken place - If GATT is expecting response for previous request - Already a request is in the queue to be sent - Channel not open - Already one GATT procedure is started

Event(s) generated:

A command status event is generated on the receipt of the command. When the procedure is completed, a Evt_Blue_Gatt_Procedure_Cmplt event is generated. Before procedure completion the response packet is given through Evt_Blue_Att_Read_Resp event.

4.6.35 Aci_Gatt_Write_Without_Response

Table 229. Aci_Gatt_Write_Without_Response

Command name	Parameters	Return
Aci_Gatt_Write_Without_Response (0xFD23)	Connection_handle AttrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value without waiting for any response from the server.

Table 230. Aci_Gatt_Write_Without_Response command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 231. Aci_Gatt_Write_Without_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed • Channel not open • previous Aci_Gatt_Write_Without_Response is still ongoing (only on BlueNRG-MS FW stack 7.1, 7.1.a) 0x64: Insufficient resources (from the BlueNRG-MS FW stack version 7.1.b or later)

Event(s) generated:

A command complete event is generated when this command is processed.

4.6.36 Aci_Gatt_Signed_Write_Without_Resp

Table 232. Aci_Gatt_Signed_Write_Without_Response

Command name	Parameters	Return
Aci_Gatt_Signed_Write_Without_Response (0xFD24)	Connection_handle AttrHandle ValLen AttrVal	Status

Description:

Start the procedure to write a characteristic value with an authentication signature without waiting for any response from the server. It cannot be used when the link is encrypted.

Table 233. Aci_Gatt_Signed_Write_Without_Resp command parameters

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
AttrHandle	2 bytes	Handle of the attribute to be written
ValLen	1 byte	Length of the value to be written
AttrVal	0-N bytes	Value to be written

Table 234. Aci_Gatt_Signed_Write_Without_Response return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x42: Invalid parameters 0x46: Not allowed - Channel not open previous Aci_Gatt_Write_Without_Response is still ongoing (only on BlueNRG-MS FW stack 7.1, 7.1.a) 0x64: Insufficient resources (from BlueNRG-MS FW stack version 7.1.b or later)

Event(s) generated:

A command complete event is generated when this command is processed.

4.6.37 Aci_Gatt_Confirm_Indication

Table 235. Aci_Gatt_Confirm_Indication

Command name	Parameters	Return
Aci_Gatt_Confirm_Indication (0xFD25)	Connection_handle	Status

Description:

Allow application to confirm indication. This command has to be sent when the application receives the event Evt_Blue_Gatt_Indication.

Table 236. **Aci_Gatt_Confirm_Indication command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given

Table 237. **Aci_Gatt_Confirm_Indication return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed - If the exchange has already taken place - Channel not open

Event(s) generated:

A command complete event is generated when this command is processed.

4.6.38 Aci_Gatt_Write_Resp

Table 238. **Aci_Gatt_Write_Resp**

Command name	Parameters	Return
Aci_Gatt_Write_Resp (0xFD26)	Connection_handle Attribute_handle Write_status Error_Code Att_Val_Len Att_Val	Status

Description:

It allows or rejects a write request from a client.This command has to be sent by the application when it receives the Evt_Blue_Gatt_Write_Permit_req. If the write can be allowed, then the status and error code has to be set to 0. If the write cannot be allowed, then the status has to be set to 1 and the error code has to be set to the error code that has to be passed to the client.

Table 239. **Aci_Gatt_Write_Resp command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given
Attribute_handle	2 bytes	Handle of the attribute that was passed in the event Evt_Blue_Gatt_Write_Permit_req
Write_status	1 byte	0x00: The value can be written to the attribute specified by handle 0x01: The value should not be written to the attribute specified by the handle
Error_Code	1 byte	The error code that has to be passed to the client in case the write has to be rejected
Att_Val_Len	1 byte	Length of the value to be written as passed in the event Evt_Blue_Gatt_Write_Permit_req
Att_Val	0-N bytes	Value as passed in the event Evt_Blue_Gatt_Write_Permit_req.

Table 240. **Aci_Gatt_Write_Resp return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed The command is not allowed in the current state of stack

Event(s) generated:

A command complete event is generated when this command is processed.

4.6.39 Aci_Gatt_Allow_Read

Table 241. **Aci_Gatt_Allow_Read**

Command name	Parameters	Return
Aci_Gatt_Allow_Read (0xFD27)	Connection_handle	Status

Description:

The application has to send this command when it receives the Evt_Blue_Gatt_Read_Permit_Req or Evt_Blue_Gatt_Read_Multi_Permit_Req. This command indicates to the stack that the response can be sent to the client. So if the application wishes to update any of the attributes before they are read by the client, it has to update the characteristic values using the Aci_Gatt_Update_Charac_Value and then give this command. The application should perform the required operations within 30 seconds. Otherwise the GATT procedure is timeout.

Table 242. **Aci_Gatt_Allow_Read command parameters**

Parameter	Size	Description
Connection_handle	2 bytes	Connection handle for which the command is given

Table 243. **Aci_Gatt_Allow_Read return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x46: Not allowed The command is not expected in the current state of the stack

Event(s) generated:

A command complete event is generated when this command is processed.

4.6.40 Aci_Gatt_Set_Security_Permission

Table 244. Aci_Gatt_Set_Security_Permission

Command name	Parameters	Return
Aci_Gatt_Set_Security_Permission (0xFD28)	Service_handle Attr_handle Security_permission	Status

Description:

This command sets the security permission for the attribute handle specified. Currently the setting of security permission is allowed only for client configuration descriptor.

Table 245. Aci_Gatt_Set_Security_Permission command parameters

Parameter	Size	Description
Service_handle	2 bytes	Handle of the service which contains the attribute whose security permission has to be modified.
Attr_handle	2 bytes	Handle of the attribute whose security permission has to be modified.
Security_permission	1 byte	0x00: ATTR_PERMISSION_NONE 0x01: ATTR_PERMISSION_AUTHEN_READ 0x02: ATTR_PERMISSION_AUTHOR_READ 0x04: ATTR_PERMISSION_ENCRY_READ 0x08: ATTR_PERMISSION_AUTHEN_WRITE 0x10: ATTR_PERMISSION_AUTHOR_WRITE 0x20: ATTR_PERMISSION_ENCRY_WRITE

Table 246. Aci_Gatt_Set_Security_Permission return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x12: Invalid parameters 0x41: Failed Could not find the handle specified or the handle is not of a client configuration descriptor

Event(s) generated:

A command complete event is generated when this command is processed.

4.6.41 Aci_Gatt_Set_Desc_Value

Table 247. **Aci_Gatt_Set_Desc_Value**

Command name	Parameters	Return
Aci_Gatt_Set_Descriptor_Value (0xFD29)	Service_Handle Char_Handle Desc_Handle val_offset desc_val_len desc_val	Status

Description:
This command sets the value of the descriptor specified by Desc_handle.

Table 248. **Aci_Gatt_Set_Desc_Value command parameters**

Parameter	Size	Description
Service_handle	2 bytes	Handle of the service which contains the descriptor.
Char handle	2 bytes	Handle of the characteristic which contains the descriptor.
Desc handle	2 bytes	Handle of the descriptor whose value has to be set.
val_offset	2 bytes	Offset from which the descriptor value has to be updated.
desc_val_len	2 bytes	Length of the descriptor value
desc_val	0-N bytes	Descriptor value

Table 249. **Aci_Gatt_Set_Desc_Value return parameters**

Parameter	Size	Description
Status	1 byte	0x00: Success 0x47: Error 0x60: Invalid handle 0x61: Invalid parameter

Event(s) generated:
A command complete event is generated when this command is processed.

4.6.42 Aci_Gatt_Read_Handle_Value

Table 250. **Aci_Gatt_Read_Handle_Value**

Command name	Parameters	Return
Aci_Gatt_Read_Handle_Value (0xFD2A)	Attribute_handle	Status Length Value

Description:

Reads the value of the attribute handle specified from the local GATT database.

Table 251. Aci_Gatt_Read_Handle_Value command parameters

Parameter	Size	Description
Attribute handle	2 bytes	Handle of the attribute to read

Table 252. Aci_Gatt_Read_Handle_Value return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x41: BLE_STATUS_FAILED 0x12: ERR_INVALID_HCI_CMD_PARAMS
Length	2 bytes	Length of the value
Value	0-N bytes	Value read. The length is as specified in the Length field.

Event(s) generated:

A command complete event is returned when this command is received. The event will contain the length and value of the attribute handle specified.

4.6.43 Aci_Gatt_Read_Handle_Value_Offset

Table 253. Aci_Gatt_Read_Handle_Value_Offset

Command name	Parameters	Return
Aci_Gatt_Read_Handle_Value_Offset (0xFD2B)	Attribute_Handle Offset	Status Attribute_value_Length Attribute_value

Description:

The command returns the value of the attribute handle from the specified offset. If the length to be returned is greater than 128, then only 128 bytes are returned. The application should send this command with incremented offsets until it gets an error with the offset it specified or the number of byes of attribute value returned is less than 128.

Table 254. Aci_Gatt_Read_Handle_Value_Offset command parameters

Parameter	Size	Description
Attribute_handle	2 bytes	The Attribute Handle to be read
Offset	1 byte	The offset specifies the position from which the value needs to be read.

Table 255. **Aci_Gatt_Read_Handle_Value_Offset return parameters**

Parameter	Size	Description
Status	1 byte	0x00 : Success 0x63 : Status Invalid Operation 0x12 : HCI_Invalid Parameters 0x41 : BLE_Status_Failed
Attribute_value_Length	2 bytes	Length of the attribute value to follow
Attribute_value	Data length bytes	Read attribute value

Event(s) generated:

On the completion of the command, a command complete event is generated. The status indicates success or failure. If the status is success, then the event data contains the data length of the attribute value followed by the attribute value.

4.6.44 Aci_Update_Char_Value_Ext

Table 256. **Aci_Gatt_Update_Char_Value_Ext**

Command name	Parameters	Return
Aci_Gatt_Update_Char_Value_Ext (0xFD2C)	Serv_Handle Char_Handle Update_Type Val_Total_Length Val_Update_Offset Val_Update_Length Char_Update_Value	Status

Description:

Update the Attribute Value of a Characteristic belonging to a specified service.

This command is an extension of the Aci_Gatt_Update_Char_Value command (OpCode=0xFD06) and support update of long attribute value (up to 512 bytes).

A specific parameter is also provided to choose if Indication and/or Notification can be sent if enabled in the related Client Characteristic Configuration Descriptor.

Table 257. **Aci_Gatt_Read_Update_Value_Ext command parameters**

Parameter	Size	Description
Serv_Handle	2 bytes	Handle of the service to which characteristic belongs
Char_Handle	2 bytes	Handle of the characteristic.
Update_Type	1 byte	Controls notification/indication generated by the attribute value update: 0x00: neither notification nor indication can be sent (the update is not shared with peer Clients) 0x01: only notification can be sent 0x02: only indication can be sent 0x03: both notification and indication can be sent

Parameter	Size	Description
Val_Total_Length	2 bytes	Total length of the Attribute value after the update. In case of a variable size characteristic, this field specifies the new length of the characteristic value after the update; in case of fixed length characteristic this field is ignored.
Val_Update_Offset	2 bytes	The offset from which the Attribute value has to be updated.
Val_Update_Length	1 bytes	Length of the following Char_Update_Value buffer
Char_Update_Value	Val_Update_Length bytes	Updated value of the characteristic

Table 258. Aci_Gatt_Update_Char_Value_Ext return parameters

Parameter	Size	Description
Status	1 byte	0x00 : Success 0x47: Error, GATT is not initialized 0x60: Invalid handle 0x61: Invalid parameter 0x64: Insufficient resources

4.7 GATT VS events

Table 259. GATT VS events

Item	Event	EGID	EID	ECODE
1	Evt_Blue_Gatt_Attribute_modified	0x03	0x01	0x0C01
2	Evt_Blue_Gatt_Procedure_Timeout	0x03	0x02	0x0C02
3	Evt_Blue_Att_Exchange_MTU_Resp	0x03	0x03	0x0C03
4	Evt_Blue_Att_Find_Information_Resp	0x03	0x04	0x0C04
5	Evt_Blue_Att_Find_By_Type_Value_Resp	0x03	0x05	0x0C05
6	Evt_Blue_Att_Read_By_Type_Resp	0x03	0x06	0x0C06
7	Evt_Blue_Att_Read_Resp	0x03	0x07	0x0C07
8	Evt_Blue_Att_Read_Blob_Resp	0x03	0x08	0x0C08
9	Evt_Blue_Att_Read_Multiple_Resp	0x03	0x09	0x0C09
10	Evt_Blue_Att_Read_By_Group_Type_Resp	0x03	0x0A	0x0C0A
11	Evt_Blue_Att_Prepare_Write_Resp	0x03	0x0C	0x0C0C
12	Evt_Blue_Att_Exec_Write_Resp	0x03	0x0D	0x0C0D
13	Evt_Blue_Gatt_Indication	0x03	0x0E	0x0C0E
14	Evt_Blue_Gatt_notification	0x03	0x0F	0X0C0F
15	Evt_Blue_Gatt_Procedure_Complete	0x03	0x10	0X0C10
16	Evt_Blue_Gatt_Error_Response	0x03	0x11	0X0C11
17	Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp	0x03	0x12	0X0C12
18	Evt_Blue_Gatt_Write_Permit_req	0x03	0x13	0X0C13
19	Evt_Blue_Gatt_Read_Permit_Req	0x03	0x14	0X0C14
20	Evt_Blue_Gatt_Read_Multi_Permit_Req	0x03	0x15	0X0C15
21	Evt_Blue_Gatt_Tx_Pool_Available	0x03	0x16	0X0C16

Item	Event	EGID	EID	ECODE
22	Evt_Blue_Gatt_Server_Confirmation	0x03	0x16	0x0C17

4.7.1 Evt_Blue_Gatt_Attribute_modified

This event is generated to the application by the GATT server when a client modifies any attribute on the server, as consequence of one of the following GATT procedures:

- write without response
- signed write without response
- write characteristic value
- write long characteristic value
- reliable write

Table 260. Evt_Blue_Gatt_Attribute_modified

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_handle	2 bytes	The connection handle which modified the attribute
Attribute_handle	2 bytes	Handle of the attribute that was modified
data_length	1 byte	The length of Attribute_value field
Offset	2 bytes	Bits 0-30: offset of the reported value inside the attribute. Bit 31: if the entire value of the attribute does not fit inside a single Evt_Blue_Gatt_Attribute_modified event, this bit is set to 1 to notify that other Evt_Blue_Gatt_Attribute_modified events follow to report the remaining value.
Attribute_value	0-N bytes	The new attribute value, starting from the given offset

4.7.2 Evt_Blue_Gatt_Procedure_Timeout

This event is generated by the client/server to the application on a GATT timeout (30 seconds).

Table 261. Evt_Blue_Gatt_Procedure_Timeout

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_handle	2 bytes	The connection handle on which the GATT procedure has timed out
data_len	1 byte	Following data (always 0)

4.7.3 Evt_Blue_Gatt_Procedure_Complete

This event is generated when a GATT client procedure completes either with error or successfully.

Table 262. Evt_Blue_Gatt_Procedure_Complete

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_handle	2 bytes	The connection handle which the GATT procedure has completed
data_len	1 byte	It is always 1 byte
error_code	0-N bytes	Indicates whether the procedure completed with error (BLE_STATUS_FAILED) or was successful (BLE_STATUS_SUCCESS)

4.7.4 Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp

This event can be generated during a "Discover Characteristics By UUID" procedure or a "Read using Characteristic UUID" procedure.

The attribute value is a service declaration as defined in Bluetooth Core v4.1spec (vol.3, Part G, ch. 3.3.1), when a "Discover Characteristics By UUID" has been started. It is the value of the Characteristic if a* "Read using Characteristic UUID" has been performed.

Table 263. Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_Handle	2 bytes	Connection handle where the procedure started.
Data_len	1 byte	Length of following data
Attribute_Handle	2 bytes	Handle of the discovered attribute
Attribute_Value	(Data_len - 2) bytes	Attribute value

4.7.5 Evt_Blue_Gatt_Write_Permit_req

This event is given to the application when a write request, write command or signed write command is received by the server from the client. This event will be given to the application only if the event bit for this event generation is set when the characteristic was added.

When this event is received, the application has to check whether the value being requested for write can be allowed to be written and respond with the command Aci_Gatt_Write_Resp. The details of the parameters of the command can be found. Based on the response from the application, the attribute value is modified by the stack. If the write is rejected by the application, then the value of the attribute will not be modified. In case of a write REQ, an error response is sent to the client, with the error code as specified by the application. In case of write/ signed write commands, no response is sent to the client but the attribute is not modified.

Table 264. Evt_Blue_Gatt_Write_Permit_req

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_handle	2 bytes	Handle of the connection on which there was the request to write the attribute
Attribute_handle	2 bytes	The handle of the attribute for which the write request has been made by the client
data_len	1 byte	The length of the data to follow
data_buffer	0-N bytes	The data that the client has requested to write

4.7.6 Evt_Blue_Gatt_Read_Permit_Req

This event is given to the application when a read request or read blob request is received by the server from the client. This event IS given to the application only if the event bit for this event generation is set when the characteristic was added.

On receiving this event, the application can update the value of the handle if it desires and when done, it has to send the Aci_Gatt_Allow_Read command to indicate to the stack that it can send the response to the client.

Table 265. Evt_Blue_Gatt_Read_Permit_Req

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_handle	2 bytes	Handle of the connection on which there was the request to read the attribute
Attribute_handle	2 bytes	The handle of the attribute that has been requested by the client to be read
Datalen	1 byte	Length of the data to follow
Offset	0-N bytes	Contains the offset from which the read has been requested

4.7.7 Evt_Blue_Gatt_Read_Multi_Permit_Req

This event is given to the application when a read multiple request or read by type request is received by the server from the client. This event will be given to the application only if the event bit for this event generation is set when the characteristic was added.

On receiving this event, the application can update the values of the handles if it desires and when done, it has to send the Aci_Gatt_Allow_Read command to indicate to the stack that it can send the response to the client.

Table 266. Evt_Blue_Gatt_Read_Multi_Permit_Req

Parameter	Size	Description
Event code	2 bytes	The event code of the vendor specific event
Connection_handle	2 bytes	Handle of the connection which requested to read the attribute
data_len	1 byte	The length of the data to follow
data_buffer	0-N bytes	The handles of the attributes that have been requested by the client for a read

4.7.8 Evt_Blue_Gatt_Tx_Pool_Available

Each time BlueNRG-MS FW stack raises the error code BLE_STATUS_INSUFFICIENT_RESOURCES (0x64), the Evt_Blue_Gatt_Tx_Pool_Available event is generated as soon as there are at least two buffers available for notifications or write commands.

Table 267. Evt_Blue_Gatt_Tx_Pool_Available

Parameter	Size	Description
Event code	2 bytes	The event code of this vendor specific event (0x0C16)
Connection_handle	2 bytes	Connection handle on which the GATT procedure is running
Available_Buffers	2 bytes	Indicates the number of elements available in the attrTxPool List.

Note: This event is supported starting from the BlueNRG-MS FW stack version 7.1.b

4.7.9 Evt_Blue_Att_Exchange_MTU_Resp

This event is generated in response to an Exchange MTU request. See Section 4.6.11 Aci_Gatt_Exchange_Configuration.

Table 268. **Evt_Blue_Att_Exchange_MTU_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data (always 1).
server_rx_mtu	2 bytes	Attribute server receive MTU size

4.7.10 Evt_Blue_Att_Find_Information_Resp

This event is generated in response to a Find Information Request. See Section 4.6.12 Aci_Att_Find_Information_Req and Find Information Response in Bluetooth Core v4.0 spec.

Table 269. **Evt_Blue_Att_Find_Information_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
format	1 byte	The format of the handle_uuid_pair. 16-bit UUIDs 128-bit UUIDs

handle_uuid_pair	0-N bytes	A sequence of handle-uuid pairs.
		if format=1, each pair is:
		[2 octets for handle, 2 octets for UUIDs]
		if format=2, each pair is:
		[2 octets for handle, 16 octets for UUIDs]

4.7.11 Evt_Blue_Att_Find_By_Type_Value_Resp

This event is generated in response to a Find By Type Value Request.

Table 270. **Evt_Blue_Att_Find_By_Type_Value_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event-data_length	1 byte	Length of following data
handles_info_list	0-N bytes	Handles Information List as defined in Bluetooth Core v4.1 spec. A sequence of handle pairs: [2 octets for Found Attribute Handle, 2 octets for Group End Handle]

4.7.12 Evt_Blue_Att_Read_By_Type_Resp

This event is generated in response to a Read By Type Request. See Section 4.6.20 Aci_Gatt_Find_Included_Services and Section 4.6.21 Aci_Gatt_Disc_All_Charac_Of_Serv.

Table 271. **Evt_Blue_Att_Read_By_Type_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
handle_value_pair_length	1 byte	The size of each attribute handle-value pair
handle_value_pair	0-N bytes	Attribute Data List as defined in Bluetooth Core v4.1 spec. A sequence of handle-value pairs: [2 octets for Attribute Handle, (handle_value_pair_length - 2 octets) for Attribute Value]

4.7.13 Evt_Blue_Att_Read_Resp

This event is generated in response to a Read Request. See Section 4.6.24 Aci_Gatt_Read_Charac_Val.

Table 272. **Evt_Blue_Att_Read_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
attribute_value	0-N bytes	The value of the attribute

4.7.14 Evt_Blue_Att_Read_Blob_Resp

This event is generated in response to a Read Request. See Section 4.6.24 Aci_Gatt_Read_Charac_Val.

Table 273. **Evt_Blue_Att_Read_Blob_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
part_attribute_value	0-N bytes	Part of the attribute value

4.7.15 Evt_Blue_Att_Read_Multiple_Resp

This event is generated in response to a Read Multiple Request.

Table 274. **Evt_Blue_Att_Read_Multiple_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
set_of_values	0-N bytes	A set of two or more values.

4.7.16 Evt_Blue_Att_Read_By_Group_Type_Resp

This event is generated in response to a Read By Group Type Request. See Section 4.6.18 Aci_Gatt_Disc_All_Prim_Services.

Table 275. **Evt_Blue_Att_Read_By_Group_Type_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
attribute_data_length	1 byte	The size of each Attribute Data.
attribute_data_list	0-N bytes	A list of Attribute Data where the attribute data is composed by: 2 octets for Attribute Handle 2 octets for End Group Handle (attribute_data_length - 4) octets for Attribute Value

4.7.17 Evt_Blue_Att_Prepare_Write_Resp

This event is generated in response to a Prepare Write Request.

Table 276. **Evt_Blue_Att_Prepare_Write_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Length of following data
attribute_handle	2 bytes	The handle of the attribute to be written
offset	1 byte	The offset of the first octet to be written
part_attr_value	0-N bytes	The value of the attribute to be written

4.7.18 Evt_Blue_Att_Exec_Write_Resp

This event is generated in response to an Execute Write Request.

Table 277. **Evt_Blue_Att_Exec_Write_Resp**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the response
event_data_length	1 byte	Always 0

4.7.19 Evt_Blue_Gatt_Indication

This event is generated when an indication is received from the server.

Table 278. **Evt_Blue_Gatt_Indication**

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the event
event_data_length	1 byte	Length of following data
attribute_handle	2 bytes	The handle of the attribute
attr_value	0-N bytes	The current value of the attribute

4.7.20 Evt_Blue_Gatt_notification

This event is generated when a notification is received from the server.

Table 279. Evt_Blue_Gatt_notification

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	The connection handle related to the event
event_data_length	1 byte	Length of following data
attribute_handle	2 bytes	The handle of the attribute
attr_value	0-N bytes	The current value of the attribute

4.7.21 Evt_Blue_Gatt_Error_Resp

This event is generated when an Error Response is received from the server. The error response can be given by the server at the end of one of the GATT discovery procedures. This does not mean that the procedure ended with an error, but this error event is part of the procedure itself.

Table 280. Evt_Blue_Gatt_Error_Resp

Parameter	Size	Description
event_code	2 bytes	The event code of the vendor specific event
connection_handle	2 bytes	Connection handle on which the GATT procedure is running
data_len	1 byte	The length of the data to follow
req_opcode	1 byte	The request that generated this error response
attr_handle	2 bytes	The attribute handle that generated this error response
error_code	1 byte	The reason why the request has generated an error response

4.7.22 Evt_Gatt_Server_Confirmation

This event is generated when an Error Response is received from the server. The error response can be given by the server at the end of one of the GATT discovery procedures. This does not mean that the procedure ended with an error, but this error event is part of the procedure itself.

Table 281. Evt_Gatt_Server_Confirmation

Parameter	Size	Description
connection_handle	2 bytes	Handle of the connection on which there was the indication

4.8 HCI vendor specific commands

Table 282. HCI VS commands

Item	Event	CGID	CID	OpCode
1	Aci_Hal_Write_Config_Data	0x00	0x0C	0xFC0C
2	Aci_Hal_Read_Config_Data	0x00	0x0D	0xFC0D
3	Aci_Hal_Set_Tx_Power_Level	0x00	0x0F	0xFC0F
4	Aci_Hal_Device_Standby	0x00	0x13	0xFC13
5	Aci_Hal_LE_Tx_Test_Packet_Number	0x00	0x14	0xFC14
6	Aci_Hal_Tone_Start	0x00	0x15	0xFC15

Item	Event	CGID	CID	OpCode
7	Aci_Hal_Tone_Stop	0x00	0x16	0xFC16
8	Aci_Hal_Get_Link_Status	0x00	0x17	0xFC17
9	Aci_Hal_Get_Fw_Build_Number	0x00	0x00	0xFC00
10	Aci_Hal_Get_Anchor_Period	0x00	0xF8	0xFC8
11	Aci_Updater_Start	0x00	0x20	0xFC20
12	Aci_Updater_Reboot	0x00	0x21	0xFC21
13	Aci_Get_Updater_Version	0x00	0x22	0xFC22
14	Aci_Get_Updater_Buffer_Size	0x00	0x23	0xFC23
15	Aci_Erase_Blue_Flag	0x00	0x24	0xFC24
16	Aci_Reset_Blue_Flag	0x00	0x25	0xFC25
17	Aci_Updater_Erase_Sector	0x00	0x26	0xFC26
18	Aci_Updater_Program_Data_Block	0x00	0x27	0xFC27
19	Aci_Updater_Read_Data_Block	0x00	0x28	0xFC28
20	Aci_Updater_Calc_CRC	0x00	0x29	0xFC29
21	Aci_Updater_HW_Version	0x00	0x2A	0xFC2A

1. For the updater commands, i.e. OpCode with 0xFC2x, please refer to the separate document BlueNRG-MS Updater Specification.

4.8.1 Aci_Hal_Write_Config_Data

Table 283. Aci_Hal_Write_Config_Data

Command name	Parameters	Return
Aci_Hal_Write_Config_Data (0xFC0C)	Offset Length Value	Status

Description:

This command writes a value to a low level configure data structure. It is useful to set up directly some low level parameters for the system in the runtime.

Table 284. Aci_Hal_Write_Config_Data command parameters

Parameters	Size	Description
Offset	1 byte	Offset in the data structure. The starting member in the data structure will have an offset 0
Length	1 byte	Length of data to be written
Value	0-N bytes	Data to be written

Table 285. **Aci_Hal_Write_Config_Data members**

Data members	Size	Offset	Description
Public address	6 bytes	0x00	Bluetooth public address
DIV	2 bytes	0x06	DIV used to derive CSRK
ER	16 bytes	0x08	Encryption root key used to derive LTK and CSRK
IR	16 bytes	0x18	Identity root key used to derive LTK and CSRK
LLWithoutHost	1 byte	0x2C	Switch on/off Link Layer only mode
Role	1 byte	0x2D	Select the BlueNRG-MS roles and mode configuration. 1. Slave and master Only one connection 6 kB of RAM retention 2. Slave and master Only one connection 12 kB of RAM retention 3. Master and slave Up to 8 connections 12 kB of RAM retention 4. Master and slave Simultaneous advertising and scanning Up to 4 connections This mode is available starting from the BlueNRG-MS FW stack version 7.1.b

Table 286. **Aci_Hal_Write_Config_Data return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: ERR_INVALID_HCI_CMD_PARAMS

Event(s) generated:

The controller will generate a command complete event.

4.8.2 Aci_Hal_Read_Config_Data

Table 287. **Aci_Hal_Read_Config_Data**

Command name	Parameters	Return
Aci_Hal_Read_Config_Data (0xFC0D)	Offset	Status value

Description:

This command requests the value in the low level configure data structure. For more information see the command Aci_Hal_Write_Config_Data. The number of bytes of returned value changes for different offset.

Table 288. **Aci_Hal_Read_Config_Data command parameters**

Parameter	Size	Description
Offset	1 byte	Offset in the data structure. The starting member in the data structure will have an offset 0

Table 289. **Aci_Hal_Read_Config_Data return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: ERR_INVALID_HCI_CMD_PARAMS
Value	0-N bytes	Value read at the offset specified

Event(s) generated:

The controller will generate a command complete event.

4.8.3 Aci_Hal_Set_Tx_Power_Level

Table 290. **Aci_Hal_Set_Tx_Power_Level**

Command name	Parameters	Return
Aci_Hal_Set_Tx_Power_Level (0xFC0F)	EN_HIGH_POWER PA_LEVEL	Status

Description:

This command sets the TX power level of the BlueNRG-MS. By controlling the EN_HIGH_POWER and the PA_LEVEL, the combination of the 2 determines the output power level (dBm). See the table below.

When the system starts up or reboots, the default TX power level will be used, which is the maximum value of 8 dBm. Once this command is given, the output power will be changed instantly, regardless if there is Bluetooth communication going on or not. For example, for debugging purpose, the BlueNRG-MS can be set to advertise all the time. And use this command to observe the signal strength changing.

The system will keep the last received TX power level from the command, i.e. the 2nd command overwrites the previous TX power level. The new TX power level remains until another Set TX Power command, or the system reboots

Table 291. **Aci_Hal_Set_Tx_Power_Level command parameters**

Parameter	Size	Description
EN_HIGH_POWER	1 byte	Can be only 0 or 1. Set high power bit on or off
PA_LEVEL	1 byte	Can be from 0 to 7. Set the PA level value

Table 292. **Tx_power_level command parameters combination**

EN_HIGH_POWER	PA_LEVEL	Approx. TX power level (dBm)
0	0	-18
0	1	-15
0	2	-11

EN_HIGH_POWER	PA_LEVEL	Approx. TX power level (dBm)
0	3	-8
0	4	-5
0	5	-2
0	6	2
0	7	5
1	0	-15
1	1	-12
1	2	-8
1	3	-5
1	4	-2
1	5	1
1	6	5
1	7	8 (default)

Table 293. Aci_Hal_Set_Tx_Power_Level return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x12: ERR_INVALID_HCI_CMD_PARAMS

Event(s) generated:

The controller will generate a command complete event.

4.8.4 Aci_Hal_Device_Standby

Table 294. Aci_Hal_Device_Standby

Command name	Parameters	Return
Aci_Hal_Device_Standby(0xFC13)		Status

Description:

Normally the BlueNRG-MS will automatically enter sleep mode to save power. This Aci_Hal_Device_Standby command further put the device into the Standby mode instead of the sleep mode. The difference is that, in sleep mode, the device can still wake up itself with the internal timer. But in standby mode, this timer is also disabled. So the only possibility to wake up the device is by the external signals, e.g. a HCI command sent via SPI bus. Based on the measurement, the current consumption under sleep mode is ~2 uA. And this value is ~1.5 uA in standby mode.

The command is only accepted when there is no other Bluetooth activity. Otherwise an error code “command disallowed” will return.

Table 295. **Aci_Hal_Device_Standby return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS 0x0C: ERR_COMMAND_DISALLOWED This command is not allowed when Bluetooth activity is ongoing

Event(s) generated:

The controller generates a command complete event.

4.8.5 Aci_Hal_LE_Tx_Test_Packet_Number

Table 296. **Aci_Hal_LE_Tx_Test_Packet_Number**

Command name	Parameters	Return
Aci_Hal_LE_Tx_Test_Packet_Number (0xFC14)		Status Packet counter

Description:

During the Direct Test mode, in the TX tests, the number of packets sent in the test is not returned when executing the Direct Test End command. This command implements this feature.

If the Direct TX test is started, a 32-bit counter will be used to count how many packets have been transmitted. After the Direct Test End, this command can be used to check how many packets were sent during the Direct TX test.

The counter starts from 0 and counts upwards. As would be the case if 32-bits are all used, the counter wraps back and starts from 0 again. The counter is not cleared until the next Direct TX test starts.

Table 297. **Aci_Hal_LE_Tx_Test_Packet_Number return parameters**

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS
Packet counter	4 bytes	Number of packets sent during the last Direct TX test

Event(s) generated:

The controller generates a command complete event.

4.8.6 Aci_Hal_Tone_Start

Table 298. **Aci_Hal_Tone_Start**

Command name	Parameters	Return
Aci_Hal_Tone_Start (0xFC15)	Channel ID	Status

Description:

This command starts a carrier frequency, i.e. a tone, on a specific channel. The frequency sine wave at the specific channel may be used for debugging purpose only. The channel ID is a parameter from 0x00 to 0x27 for the 40 BLE channels, e.g. 0x00 for 2.402 GHz, 0x01

for 2.404 GHz etc. This command should not be used when normal Bluetooth activities are ongoing.
The tone should be stopped by Aci_Hal_Tone_Stop command.

Table 299. Aci_Hal_Tone_Start command parameters

Parameter	Size	Description
Channel ID	1 byte	BLE Channel ID, from 0x00 to 0x27 meaning (2.402 + 2*0xXX) GHz

Table 300. Aci_Hal_Tone_Start return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS

Event(s) generated:
The controller generates a command complete event.

4.8.7 Aci_Hal_Tone_Stop

Table 301. Aci_Hal_Tone_Stop

Command name	Parameters	Return
Aci_Hal_Tone_Stop (0xFC16)		Status

Description:
This command is used to stop the previously started Aci_Hal_Tone_Start command.

Table 302. Aci_Hal_Tone_Stop return parameters

Parameter	Size	Description
Status	1 byte	0x00: BLE_STATUS_SUCCESS

Event(s) generated:
The controller generates a command complete event.

4.8.8 Aci_Hal_Get_Link_Status

Table 303. Aci_Hal_Get_Link_Status

Command name	Parameters	Return
Aci_Hal_Get_Link_Status (0xFC17)		Status Link_Status Connection_Handle

Description:

This command is intended to return the Link Layer Status and Connection Handles.

Table 304. Aci_Hal_Get_Link_Status return parameters

Parameter	Size	Description
Status	1 byte	0x00: success

Description:
This command is intended to return the Link Layer Status and Connection Handles.

Table 305. Aci_Hal_Get_Link_Status return parameters

Parameter	Size	Description
Link_Status	8 bytes	Link status for each clients, where the byte 0 is the link status for client 0, byte 1 is the link status for client 1 and so on. 0: Idle 1: Advertising 2: Connected in Slave role 3: Scanning 4: Reserved 5: Connected in Master role 6: TX test 7: RX test
Connection_Handle	16 bytes	Connection handle for each clients, where the bytes [0,1] indicate the connection handle for client 0, bytes [2,3] indicate the connection handle for client 1 and so on

4.8.9 Aci_Hal_Get_Fw_Build_Number

Table 306. Aci_Hal_Get_Fw_Build_Number

Command name	Parameters	Return
Aci_Hal_Get_Fw_Build_Number (0xFC00)		Status Revision_Number

Description:
This command is intended to retrieve the firmware revision number.

Table 307. Aci_Hal_Get_Fw_Build_Number return parameters

Parameter	Size	Description
Status	1 byte	0x00: SUCCESS
Revision_Number	2 bytes	The firmware revision number

4.8.10 Aci_Hal_Get_Anchor_Period

Table 308. Aci_Hal_Get_Anchor_Period

Command name	Parameters	Return
Aci_Hal_Get_Anchor_Period (0xFCF8)		Status Anchor_Interval

Description:
This command is intended to retrieve information about the current Anchor Interval and allocable timing slots.

Table 309. Aci_Hal_Get_Anchor_Period return parameters

Parameter	Size	Description
Status	1 byte	0x00: Success 0x43: Busy. Returned in case a connection update is pending
Anchor_Interval	4 bytes	Current anchor interval in multiples of 0.625 ms
Max slot	4 bytes	Maximum available size (in multiples of 0.625 ms) that can be allocated to a new connection slot

Event(s) generated:
The controller will generate a command complete event.

4.9 HCI VS events

4.9.1 Evt_Blue_Initialized_event

When the BlueNRG-MS firmware is started normally, it gives a Evt_Blue_Initialized event to the user to indicate the system has started (with Reason Code 0x01). The Evt_Blue_Initialized event is an ACI event with the same format as the other events. In the table below all the fields are described.

Table 310. Evt_Blue_Initialized event

Parameter	Size	Description
Event code	1 byte	0x0001 - The event code for Evt_Blue_Initialized event
Reason code	1 byte	0x00 – Reserved 0x01 – Firmware started properly 0x02 – Updater mode entered because of Aci_Updater_Start command 0x03 - Updater mode entered because of a bad BLUE flag 0x04 - Updater mode entered with IRQ pin 0x05 - Reset caused by watchdog 0x06 - Reset due to lockup 0x07 - Brownout reset 0x08 - Reset caused by a crash (NMI or Hard Fault) 0x09 - Reset caused by an ECC error

4.9.2 Evt_Blue_Lost_Events

This event is generated when an indication is received from the server.

Table 311. **Evt_Blue_Lost_Events**

Parameter	Size	Description
event_code	2 bytes	0x0002: the event code for Evt_Blue_Lost_Events event
Lost_Events_Map	8 bytes	The lost events bitmap

Each bit of the lost events map corresponds to a specific event:

Table 312. **Lost_Events_Map (8 bytes)**

Lost_Event_Map bit	Description
0	EVT_DISCONN_COMPLETE
1	EVT_ENCRYPT_CHANGE
2	EVT_READ_REMOTE_VERSION_COMPLETE
3	EVT_CMD_COMPLETE
4	EVT_CMD_STATUS
5	EVT_HARDWARE_ERROR
6	EVT_NUM_COMP_PKTS
7	EVT_ENCRYPTION_KEY_REFRESH
8	EVT_BLUE_INITIALIZED
9	EVT_BLUE_GAP_SET_LIMITED_DISCOVERABLE
10	EVT_BLUE_GAP_PAIRING_CMPLT
11	EVT_BLUE_GAP_PASS_KEY_REQUEST
12	EVT_BLUE_GAP_AUTHORIZATION_REQUEST
13	EVT_BLUE_GAP_SECURITY_REQ_INITIATED
14	EVT_BLUE_GAP_BOND_LOST
15	EVT_BLUE_GAP_PROCEDURE_COMPLETE
16	EVT_BLUE_GAP_ADDR_NOT_RESOLVED
17	EVT_BLUE_L2CAP_CONN_UPDATE_RESP
18	EVT_BLUE_L2CAP_PROCEDURE_TIMEOUT
19	EVT_BLUE_L2CAP_CONN_UPDATE_REQ
20	EVT_BLUE_GATT_ATTRIBUTE_MODIFIED
21	EVT_BLUE_GATT_PROCEDURE_TIMEOUT
22	EVT_BLUE_EXCHANGE_MTU_RESP
23	EVT_BLUE_ATT_FIND_INFORMATION_RESP
24	EVT_BLUE_ATT_FIND_BY_TYPE_VAL_RESP
25	EVT_BLUE_ATT_READ_BY_TYPE_RESP
26	EVT_BLUE_ATT_READ_RESP
27	EVT_BLUE_ATT_READ_BLOB_RESP
28	EVT_BLUE_ATT_READ_MULTIPLE_RESP
29	EVT_BLUE_ATT_READ_BY_GROUP_RESP
30	EVT_BLUE_ATT_WRITE_RESP

Lost_Event_Map bit	Description
31	EVT_BLUE_ATT_PREPARE_WRITE_RESP
32	EVT_BLUE_ATT_EXEC_WRITE_RESP
33	EVT_BLUE_GATT_INDICATION
34	EVT_BLUE_GATT_NOTIFICATION
35	EVT_BLUE_GATT_PROCEDURE_COMPLETE
36	EVT_BLUE_GATT_ERROR_RESP
37	EVT_BLUE_GATT_DISC_READ_CHARAC_BY_UUID_RESP
38	EVT_BLUE_GATT_WRITE_PERMIT_REQ
39	EVT_BLUE_GATT_READ_PERMIT_REQ
40	EVT_BLUE_GATT_READ_MULTI_PERMIT_REQ
41	EVT_BLUE_GATT_TX_POOL_AVAILABLE
42	EVT_BLUE_GATT_SERVER_RX_CONFIRMATION
43	EVT_BLUE_GATT_PREPARE_WRITE_PERMIT_REQ
44	EVT_LL_CONNECTION_COMPLETE
45	EVT_LL_ADVERTISING_REPORT
46	EVT_LL_CONNECTION_UPDATE_COMPLETE
47	EVT_LL_READ_REMOTE_USED_FEATURES
48	EVT_LL_LTK_REQUEST

4.9.3 Fault data event

The fault data event is automatically sent after the Evt_Blue_Initialized event in case of NMI or Hard fault.

Table 313. Fault data event

Parameter	Size	Description
Event_code	2 bytes	0x0003: the event code for Evt_Fault_Data event
Fault reason	1 byte	6: NMI fault 7: Hard fault
SP	4 bytes	MCU SP register
R0	4 bytes	MCU R0 register
R1	4 bytes	MCU R1 register
R2	4 bytes	MCU R2 register
R3	4 bytes	MCU R3 register
R12	4 bytes	MCU R12 register
LR	4 bytes	MCU LR register
PC	4 bytes	MCU PC register
xPSR	4 bytes	MCU xPSR register
Fault data length	1 byte	N: length of the additional fault data section
Fault data	N bytes	Additional crash dump data

4.9.4 Hardware error event codes

The following error codes may be reported by the HCI Hardware_Error_Event:

- 0: SPI framing error

- 1: Radio state error. This error code is generated if the radio FSM has not reached the Active 2 state when the start command has been issued. This may be due to a slow crystal start-up.
- 2: Timer overrun error

5 SPI interface

The BlueNRG-MS device provides an SPI interface, which can be used to connect an external device, where the application runs, which sends ACI commands to control BlueNRG-MS and receives events from it. The ACI commands and events are transmitted through the SPI bus.

This chapter describes the BlueNRG-MS hardware SPI interface. Also it describes the communication protocol over the SPI bus between the BlueNRG-MS and the external device.

5.1 Hardware SPI interface

The BlueNRG-MS SPI interface is Motorola protocol compliant, which has 5 wires: CLK (clock), nCS (chip select), MOSI (master output slave input), MISO (master input slave output), and DataRdy_IRQ (transmit interrupt request).

The BlueNRG-MS always behaves as the slave device on the SPI bus. The external device must be the SPI master.

The table below shows the 5 SPI pins on the BlueNRG-MS device, including the pin number of the silicon chip and the pin direction from the BlueNRG-MS's side.

Table 314. SPI pins

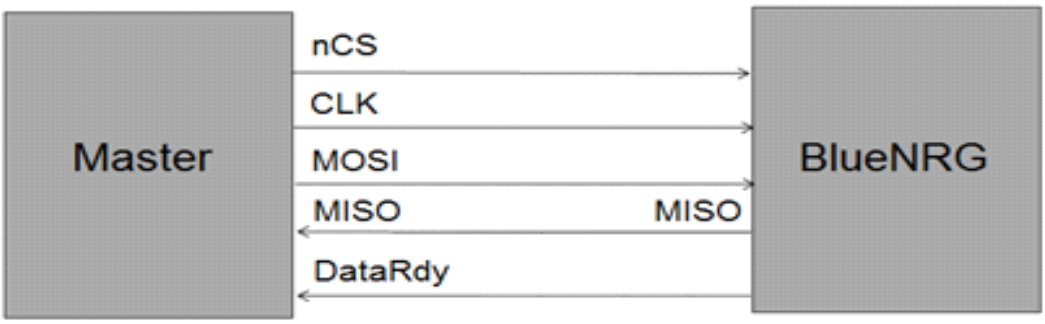
Pin name	Pin n°	Direction
SPI_CLK	2	Input
SPI_NCS	31	Input
SPI_MOSI	1	Input
SPI_MISO	32	Output
SPI_IRQ	3	Output

Here is the summary of the SPI parameters:

1. The CS line is active low.
2. The clock signal can go up to 8 MHz, or can go as low as 100 kHz.
3. The clock is inactive low, i.e. the clock polarity CPOL = 0.
4. The data is valid on clock leading edge, i.e. the clock phase CPHA = 0.
5. The data transfer is always 8-bit byte based.
6. For each byte, the Most Significant Bit is sent first.

Figure 9. SPI wire connection shows how to connect the BlueNRG-MS SPI bus to an external SPI master.

Figure 9. SPI wire connection



The SPI master, i.e. an application processor, always controls when to start an SPI transaction. This can happen at any time. It is started by putting low the nCS line to activate the BlueNRG-MS device. Then the master should provide the SPI clock. Together with the clock, the master should output the data on the SPI bus through the MOSI line, either dummy data or valid data. At the same time, the slave returns data from the MISO line, taking it form the internal shift register.

When the SPI master wants to send data to the BlueNRG-MS, i.e. an SPI write, it sets nCS to low, then sends the SPI clock to read an SPI header from the BlueNRG-MS. At the same time the first byte sent by the master must be a special byte indicating a write operation, while the following ones are dummy bytes used only to read the entire SPI header from the slave (5 bytes). The SPI header format will be described in the following section. If the SPI header indicates that the BlueNRG-MS has enough space in the buffer, the SPI master can then send real data over the SPI bus. The SPI master should not overwrite the BlueNRGMS SPI buffer, otherwise the communication will fail.

When the BlueNRG-MS wants to send data to the SPI master, i.e. an SPI read, it uses the IRQ pin, because the SPI slave cannot control the nCS line. The BlueNRG-MS will set the IRQ pin to high, which notifies the SPI master that there is data to be read. Then the SPI master should start an SPI transaction, reading the SPI header, from which it will know how many bytes it should read back from the BlueNRG-MS. The SPI master should not read bytes other than those indicated as the SPI header, otherwise the communication will fail.

The DataRdy (IRQ) pin of the BlueNRG-MS works as follows: (1) it is high when BlueNRGMS has data to send to the SPI master; (2) It is high impedance when BlueNRG-MS has no data to send to the SPI master. On the SPI master side, this pin must be configured as an input pin. The SPI master may poll this pin, or use it as an interrupt source, to detect when an SPI read is required. Also there should be a pull down resistor, either on the SPI master IRQ pin or externally connected. This pull down resistor makes sure the DataRdy value goes back to 0, so it does not confuse the SPI master to misread data.

Note also that the MISO also goes to high-Z when BlueNRG-MS does not send data. This allows multi-SPI slaves on the same bus. The MISO line requires a pull-down resistor for a correct operation when BlueNRG-MS is in sleep. A pull-down resistor also avoids unwanted current leakage on the MISO pin of the external microcontroller when the MISO pin on the BlueNRG-MS side is in High-Z.

5.2 SPI communication protocol

To communicate with the BlueNRG-MS, the data on the SPI bus must be formatted as described in this section.

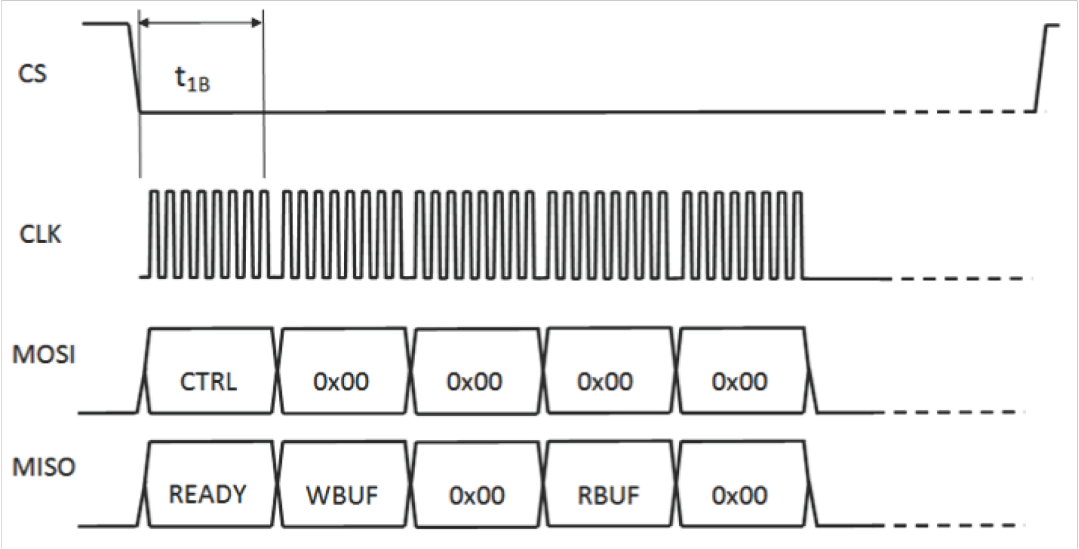
An SPI transaction is defined from the time instant when the master puts the nCS line to low, until the time instant when the nCS line is back to high.

Each SPI transaction should contain 1 data frame. Each data frame should contain at least 5 bytes of header, and may have 0-N bytes of data.

5.2.1 Header

The SPI frame header of the master on the MOSI line is composed of 1 control byte and 4 dummy bytes (0x00).

Figure 10. **SPI header format**



The control byte can have only two possible values:

- 0x0A for write operation
- 0x0B for read operation

The BlueNRG-MS returns the slave header on the MISO line at the same time. The 1st byte is the SPI READY indication. It is 0x02 to indicate that the slave SPI interface is ready. If it is any value other than 0x02, the master must ignore the following 4 bytes and abort the SPI transaction.

In order to correctly read the ready status of the device in any condition, the CTRL byte must be sent within 8 µs from the start of the frame (t1B) in the figure above.

If the BlueNRG-MS SPI is ready (ready byte is 0x02), the following 4 bytes give 2 buffer sizes for write and read:

- 2nd byte contains write buffer size (maximum value is 127)
- 4th byte contains read buffer size

The write buffer size means how many bytes the master can write to the BlueNRG-MS in a single SPI frame. The read buffer size means how many bytes in the BlueNRG-MS are waiting for the master to be read in the current SPI frame.

For example, if the slave header is [0x02, 0x7F, 0x0, 0x05, 0x00], the write buffer size is 0x7F (127 bytes) and the read buffer size is 0x05 (5 bytes). So the master is not allowed to write more than 127 bytes, and it should read back 5 bytes from the BlueNRG-MS as soon as possible.

The SPI master may poll the BlueNRG-MS by sending only SPI headers, just to check the size of the read and write buffers. For example, the SPI master can send an SPI frame with only the 5-byte header: the CTRL byte can be either write or read, followed by the four dummy bytes (e.g. 0x00). After the header, the SPI master can stop the frame without sending or reading data.

5.2.2 Write data to BlueNRG-MS

A write transaction is used to send a command to BlueNRG-MS. This is an asynchronous event generated by the external µC, regardless of the state of the device.

In order to write data to the BlueNRG-MS SPI, the SPI master must start an SPI frame with CTRL byte set to 0x0A.

Since BlueNRG-MS is performing a very advanced low power management that makes the device go to sleep as soon as no immediate tasks have to be performed (e.g. no radio activity), external µC does not know whether the device is actually sleeping or is awake when the SPI write transaction is performed. In order to handle these two cases, the nCS pin is not only used to select the SPI slave, but also as a wake up signal.

In summary when external µC would like to perform an SPI write transaction two cases can happen:

1. Device is awake
2. Device is asleep

If device is awake when nCS is driven low, the READY byte returned by the slave is 0x02.

The size of the available write buffer in the SPI header announces how many bytes can be written in the current SPI transaction. If READY byte is not 0x02 (device sleeping) or if write buffer size is 0 (SPI buffer not yet prepared or SPI not initialized), the SPI transaction must

be ended by releasing nCS line.

After releasing nCS line, BlueNRG-MS goes to sleep if no commands are given within 2 ms from the previous falling edge of the nCS signal. External µC has to poll BlueNRG-MS SPI to know when data can be written. It is mandatory to release and assert again nCS before

reading again the SPI header.

If READY byte is 0x02 and size of the write buffer is greater than 0, the SPI master can write data in the same SPI frame, up to the amount specified in the SPI header.

In case an ACI command packet is bigger than 127 bytes (the maximum number of bytes of an SPI frame, without header), the master has to send it in more SPI write transactions: the master firstly sends 127 bytes in a single SPI frame, then it must wait for the write buffer to be available again, and then it can write the remaining bytes in next SPI frame.

The write buffer size value refers to the payload. It does not include the 5-byte header.

When the master sends data bytes on the MOSI line, the BlueNRG-MS returns the same amount of bytes from its own shift register. The master should ignore these bytes during write transaction.

It could take around 500 us before BlueNRG-MS is able to receive data on the SPI after it has been woken up from sleep mode. This time depends on the HS crystal startup time and it is lower if 12 kB of RAM retention is used instead of 6 kB.

5.2.3 Read data from BlueNRG-MS

The BlueNRG-MS uses the IRQ pin to notify the SPI master when it has data to be read.

When the SPI master detects a high level on the IRQ pin, it should read data from BlueNRG-MS. The protocol is similar to SPI write. The SPI master starts a transaction to check the SPI header. The master puts 0x0B in the CTRL byte, and checks again if the

slave is ready (0x02), and the read buffer size is non-zero. Then the master must send dummy bytes (e.g. 0x00) on the MOSI line to read data from BlueNRG-MS on the MISO line. The reading process can happen in one or in multiple SPI transactions. The SPI master should read from BlueNRG-MS as long as the IRQ pin is high. BlueNRG-MS does not go to sleep until all data have been read.

Revision history

Table 315. **Document revision history**

Date	Revision	Changes
04-Mar-2015	1	Initial release.
21-Jul-2015	2	Updated: Figure 3, Table 38, Table 39, Table 123, Table 155, Table 256, Table 292. Added: Section 4.6.8.
30-Sep-2015	3	Updated GATT events section with full set of supported GATT events Updated some ACI parameters names and fields. Minor text changes throughout the document.
2-Dec-2015	4	Correct some ACIs and events names. Minor text changes throughout the document.
25-Aug-2016	5	Updated ACI commands and events
21-Dec-2016	6	Updated Aci_Gap_Pass_Key_Response and Aci_Gatt_Add_Char commands. Modified Section 5.2: SPI communication protocol.
24-Jan-2017	7	Updated section Section 5.2.1: Header.
18-Jul-2019	8	Updated Section 4.8.3 Aci_Hal_Set_Tx_Power_Level.

Contents

1 **Overview** 2
1.1 BlueNRG-MS ACI architecture 3
2 **ACI command data format** 4
2.1 Overview 4
2.2 HCI command packet 4
2.3 HCI event packet 6
2.4 HCI ACL data packet 6
3 **Standard HCI commands** 7
3.1 Supported HCI commands 7
3.2 Supported HCI events 8
3.3 Correct use of HCI commands 8
4 **Vendor specific commands** 9
4.1 VS command and VS event format 9
4.2 L2CAP VS commands and events 10
4.2.1 Aci_L2CAP_Connection_Parameter_Update_Request 10
4.2.2 Aci_L2CAP_Connection_Parameter_Update_Response 11
4.3 L2CAP VS events 12
4.3.1 Evt_Blue_L2CAP_Connection_Update_Resp 13
4.3.2 Evt_Blue_L2CAP_Procedure_Timeout 13
4.3.3 Evt_Blue_L2CAP_Connection_Update_Request 13
4.4 GAP VS commands and events 14
4.4.1 Overview 14
4.4.2 GAP VS commands 14
4.4.3 Aci_Gap_Set_Non_Discoverable 15
4.4.4 Aci_Gap_Set_Limited_Discoverable 15
4.4.5 Aci_Gap_Set_Discoverable 17
4.4.6 Aci_Gap_Set_Direct_Connectable 19
4.4.7 Aci_Gap_Set_IO_Capability 20
4.4.8 Aci_Gap_Set_Auth_Requirement 21
4.4.9 Aci_Gap_Set_Author_Requirement 22

4.4.10 Aci_Gap_Pass_Key_Response . . . 23
4.4.11 Aci_Gap_Authorization_Response . . . 24
4.4.12 Aci_Gap_Init . . . 24
4.4.13 Aci_Gap_Set_Non_Connectable . . . 25
4.4.14 Aci_Gap_Set_Undirected_Connectable . . . 26
4.4.15 Aci_Gap_Slave_Security_Request . . . 27
4.4.16 Aci_Gap_Update_Adv_Data . . . 27
4.4.17 Aci_Gap_Delete_AD_Type . . . 28
4.4.18 Aci_Gap_Get_Security_Level . . . 29
4.4.19 Aci_Gap_Set_Event_Mask . . . 30
4.4.20 Aci_Gap_Configure_WhiteList . . . 31
4.4.21 Aci_Gap_Terminate . . . 31
4.4.22 Aci_Gap_Clear_Security_Database . . . 32
4.4.23 Aci_Gap_Allow_Rebond . . . 32
4.4.24 Aci_Gap_Start_Limited_Discovery_Proc . . . 33
4.4.25 Aci_Gap_Start_General_Discovery_Proc . . . 34
4.4.26 Aci_Gap_Start_Name_Discovery_Proc . . . 34
4.4.27 Aci_Gap_Start_Auto_Conn_Establishment . . . 36
4.4.28 Aci_Gap_Start_General_Conn_Establishment . . . 38
4.4.29 Aci_Gap_Start_Selective_Conn_Establishment . . . 39
4.4.30 Aci_Gap_Create_Connection . . . 41
4.4.31 Aci_Gap_Terminate_Gap_Procedure . . . 43
4.4.32 Aci_Gap_Start_Connection_Update . . . 43
4.4.33 Aci_Gap_Send_Pairing_Request . . . 45
4.4.34 Aci_Gap_Resolve_Private_Address . . . 45
4.4.35 Aci_Gap_Get_Bonded_Devices . . . 46
4.4.36 Aci_Gap_Set_Broadcast_Mode . . . 47
4.4.37 Aci_Gap_Start_Observation_Proc . . . 48
4.4.38 Aci_Gap_Is_Device_Bonded . . . 49
4.5 GAP VS events . . . 50
4.5.1 Evt_Blue_Gap_Limited_Discoverable . . . 50
4.5.2 Evt_Blue_Gap_Pairing_Complete . . . 50

4.5.3 Evt_Blue_Pass_Key_Request . . . 50
4.5.4 Evt_Blue_Authorization_Request . . . 50
4.5.5 Evt_Blue_Slave_Security_Initiated . . . 51
4.5.6 Evt_Blue_Gap_Bond_Lost . . . 51
4.5.7 Evt_Blue_Gap_Procedure_Complete . . . 51
4.5.8 Evt_Blue_Gap_Addr_Not_Resolved . . . 51
4.6 GATT VS commands and events . . . 51
4.6.1 Aci_Gatt_Init . . . 53
4.6.2 Aci_Gatt_Add_Service . . . 53
4.6.3 Aci_Gatt_Include_Service . . . 54
4.6.4 Aci_Gatt_Add_Characteristic . . . 55
4.6.5 Aci_Gatt_Add_Char_Desc . . . 57
4.6.6 Aci_Gatt_Update_Char_Value . . . 58
4.6.7 Aci_Gatt_Del_Characteristic . . . 59
4.6.8 Aci_Gatt_Del_Service . . . 60
4.6.9 Aci_Gatt_Del_Include_Service . . . 60
4.6.10 Aci_Gatt_Set_Event_Mask . . . 61
4.6.11 Aci_Gatt_Exchange_Configuration . . . 62
4.6.12 Aci_Att_Find_Information_Req . . . 63
4.6.13 Att_Find_By_Type_Value_Req . . . 64
4.6.14 Aci_Att_Read_Attr_By_Type_Req . . . 65
4.6.15 Aci_Att_Read_By_Group_Type_Req . . . 65
4.6.16 Aci_Att_Prepare_Write_Req . . . 66
4.6.17 Aci_Att_Execute_Write_Req . . . 67
4.6.18 Aci_Gatt_Disc_All_Prim_Services . . . 68
4.6.19 Aci_Gatt_Disc_Prim_serv_By_Uuid . . . 69
4.6.20 Aci_Gatt_Find_Included_Services . . . 70
4.6.21 Aci_Gatt_Disc_All_Charac_Of_Serv . . . 71
4.6.22 Aci_Gatt_Disc_Charac_By_UUID . . . 72
4.6.23 Aci_Gatt_Disc_All_Charac_Descriptors . . . 73
4.6.24 Aci_Gatt_Read_Charac_Val . . . 74
4.6.25 Aci_Gatt_Read_Charac_Using_UUID . . . 75

4.6.26 Aci_Gatt_Read_Long_Charac_Val . . . 75
4.6.27 Aci_Gatt_Read_Multiple_Charac_Values . . . 76
4.6.28 Aci_Gatt_Write_Charac_Val . . . 77
4.6.29 Aci_Gatt_Write_Long_Charac_Val . . . 78
4.6.30 Aci_Gatt_Write_Charac_Reliable . . . 79
4.6.31 Aci_Gatt_Write_Long_Charac_Descriptor . . . 80
4.6.32 Aci_Gatt_Read_Long_Charac_Descriptor . . . 81
4.6.33 Aci_Gatt_Write_Charac_Descriptor . . . 82
4.6.34 Aci_Gatt_Read_Charac_Descriptor . . . 83
4.6.35 Aci_Gatt_Write_Without_Response . . . 84
4.6.36 Aci_Gatt_Signed_Write_Without_Response . . . 84
4.6.37 Aci_Gatt_Confirm_Indication . . . 85
4.6.38 Aci_Gatt_Write_Resp . . . 86
4.6.39 Aci_Gatt_Allow_Read . . . 87
4.6.40 Aci_Gatt_Set_Security_Permission . . . 87
4.6.41 Aci_Gatt_Set_Descriptor_Value . . . 88
4.6.42 Aci_Gatt_Read_Handle_Value . . . 89
4.6.43 Aci_Gatt_Read_Handle_Value_Offset . . . 90
4.6.44 Aci_Update_Char_Value_Ext . . . 91
4.7 GATT VS events . . . 92
4.7.1 Evt_Blue_Gatt_Attribute_modified . . . 93
4.7.2 Evt_Blue_Gatt_Procedure_Timeout . . . 93
4.7.3 Evt_Blue_Gatt_Procedure_Complete . . . 93
4.7.4 Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp . . . 93
4.7.5 Evt_Blue_Gatt_Write_Permit_req . . . 94
4.7.6 Evt_Blue_Gatt_Read_Permit_Req . . . 95
4.7.7 Evt_Blue_Gatt_Read_Multi_Permit_Req . . . 95
4.7.8 Evt_Blue_Gatt_Tx_Pool_Available . . . 95
4.7.9 Evt_Blue_Att_Exchange_MTU_Resp . . . 95
4.7.10 Evt_Blue_Att_Find_Information_Resp . . . 96
4.7.11 Evt_Blue_Att_Find_By_Type_Value_Resp . . . 96
4.7.12 4.6.12 Evt_Blue_Att_Read_By_Type_Resp . . . 96

4.7.13 Evt_Blue_Att_Read_Resp . . . 97
4.7.14 Evt_Blue_Att_Read_Blob_Resp . . . 97
4.7.15 Evt_Blue_Att_Read_Multiple_Resp . . . 97
4.7.16 Evt_Blue_Att_Read_By_Group_Type_Resp . . . 97
4.7.17 Evt_Blue_Att_Prepare_Write_Resp . . . 98
4.7.18 Evt_Blue_Att_Exec_Write_Resp . . . 98
4.7.19 Evt_Blue_Gatt_Indication . . . 98
4.7.20 Evt_Blue_Gatt_notification . . . 98
4.7.21 Evt_Blue_Gatt_Error_Resp . . . 99
4.7.22 Evt_Gatt_Server_Confirmation . . . 99
4.8 HCI vendor specific commands . . . 99
4.8.1 Aci_Hal_Write_Config_Data . . . 100
4.8.2 Aci_Hal_Read_Config_Data . . . 101
4.8.3 Aci_Hal_Set_Tx_Power_Level . . . 102
4.8.4 Aci_Hal_Device_Standby . . . 103
4.8.5 Aci_Hal_LE_Tx_Test_Packet_Number . . . 104
4.8.6 Aci_Hal_Tone_Start . . . 104
4.8.7 Aci_Hal_Tone_Stop . . . 105
4.8.8 Aci_Hal_Get_Link_Status . . . 105
4.8.9 Aci_Hal_Get_Fw_Build_Number . . . 106
4.8.10 Aci_Hal_Get_Anchor_Period . . . 106
4.9 HCI VS events . . . 107
4.9.1 Evt_Blue_Initialized event . . . 107
4.9.2 Evt_Blue_Lost_Events . . . 107
4.9.3 Fault data event . . . 109
4.9.4 Hardware error event codes . . . 109
5 SPI interface . . . 111
5.1 Hardware SPI interface . . . 111
5.2 SPI communication protocol . . . 112
5.2.1 Header . . . 112
5.2.2 Write data to BlueNRG-MS . . . 113
5.2.3 Read data from BlueNRG-MS . . . 114

Revision history 115

List of tables

Table 1. OGF value . . . 5
Table 2. HCI commands . . . 7
Table 3. HCI events . . . 8
Table 4. CGID group . . . 9
Table 5. EGID group . . . 10
Table 6. L2CAP VS commands . . . 10
Table 7. Aci_L2CAP_Connection_Parameter_Update_Request . . . 10
Table 8. Aci_L2CAP_Connection_Parameter_Update_Requests command parameters . . . 11
Table 9. Aci_L2CAP_Connection_Parameter_Update_Requests return parameters . . . 11
Table 10. Aci_L2CAP_Connection_Parameter_Update_Response . . . 11
Table 11. Aci_L2CAP_Connection_Parameter_Update_Response command parameters . . . 12
Table 12. Aci_L2CAP_Connection_Parameter_Update_Response return parameters . . . 12
Table 13. L2CAP VS events . . . 12
Table 14. Evt_Blue_L2CAP_Connection_Update_Resp . . . 13
Table 15. Evt_Blue_L2CAP_Procedure_Timeout . . . 13
Table 16. Evt_Blue_L2CAP_Connection_Update_Request . . . 13
Table 17. GAP VS commands . . . 14
Table 18. Aci_gap_set_non_discoverable . . . 15
Table 19. Aci_gap_set_non_discoverable return parameters . . . 15
Table 20. Aci_Gap_Set_Limited_Discoverable . . . 16
Table 21. Aci_Gap_Set_limited_Discoverable command parameters . . . 16
Table 22. Aci_Gap_Set_Limited_Discoverable return parameters . . . 17
Table 23. Aci_Gap_Set_Discoverable . . . 18
Table 24. Aci_Gap_Set_Discoverable command parameters . . . 18
Table 25. Aci_Gap_Set_Discoverable return parameters . . . 19
Table 26. Aci_Gap_Set_Direct_Connectable . . . 19
Table 27. Aci_Gap_Set_Direct_Connectable command parameters . . . 20
Table 28. Aci_Gap_Set_Direct_Connectable return parameters . . . 20
Table 29. Aci_Gap_Set_IO_Capability . . . 20
Table 30. Aci_Gap_Set_IO_Capability command parameters . . . 21
Table 31. Aci_Gap_Set_IO_Capability return parameters . . . 21
Table 32. Aci_Gap_Set_Auth_Requirement . . . 21
Table 33. Aci_Gap_Set_Auth_Requirement command parameters . . . 22
Table 34. Aci_Gap_Set_Auth_Requirement return parameters . . . 22
Table 35. Aci_Gap_Set_Author_Requirement . . . 22
Table 36. Aci_Gap_Set_Author_Requirement command parameters . . . 22
Table 37. Aci_Gap_Set_Author_Requirement return parameters . . . 23
Table 38. Aci_Gap_Pass_Key_Response . . . 23
Table 39. Aci_Gap_Pass_Key_Response command parameters . . . 23
Table 40. Aci_Gap_Pass_Key_Response return parameters . . . 23
Table 41. Aci_Gap_Authorization_Response . . . 24
Table 42. Aci_Gap_Authorization_Response command parameters . . . 24
Table 43. Aci_Gap_Authorization_Response return parameters . . . 24
Table 44. Aci_Gap_Init . . . 24
Table 45. Aci_Gap_Init command parameters . . . 25
Table 46. Aci_Gap_Init return parameters . . . 25
Table 47. Aci_Gap_Set_Non_Connectable . . . 25
Table 48. Aci_Gap_Set_Non_Connectable command parameters . . . 25
Table 49. Aci_Gap_Set_Non_Connectable return parameters . . . 26
Table 50. Aci_Gap_Set_Undirected_Connectable . . . 26
Table 51. Aci_Gap_Set_Undirected_Connectable command parameters . . . 26
Table 52. Aci_Gap_Set_Undirected_Connectable return parameters . . . 27

Table 53. Aci_Gap_Slave_Security_Request . . . 27
Table 54. Aci_Gap_Slave_Security_Request command parameters . . . 27
Table 55. Aci_Gap_Slave_Security_Request return parameters . . . 27
Table 56. Aci_Gap_Update_Adv_Data . . . 28
Table 57. Aci_Gap_Update_Adv_Data command parameters . . . 28
Table 58. Aci_Gap_Update_Adv_Data return parameters . . . 28
Table 59. Aci_Gap_Delete_AD_Type . . . 28
Table 60. Aci_Gap_Delete_AD_Type command parameters . . . 29
Table 61. Aci_Gap_Delete_AD_Type return parameters . . . 29
Table 62. Aci_Gap_Get_Security_Level . . . 29
Table 63. Aci_Gap_Get_Security_Level return parameters . . . 30
Table 64. Aci_Gap_Set_Event_Mask . . . 30
Table 65. Aci_Gap_Set_Event_Mask command parameters . . . 30
Table 66. Aci_Gap_Set_Event_Mask return parameters . . . 30
Table 67. Aci_Gap_Configure_WhiteList . . . 31
Table 68. Aci_Gap_Configure_WhiteList return parameters . . . 31
Table 69. Aci_Gap_Terminate . . . 31
Table 70. Aci_Gap_Terminate command parameters . . . 31
Table 71. Aci_Gap_Terminate return parameters . . . 32
Table 72. Aci_Gap_Clear_Security_Database . . . 32
Table 73. Aci_Gap_Clear_Security_Database return parameters . . . 32
Table 74. Aci_Gap_Allow_Rebond . . . 32
Table 75. Aci_Gap_Allow_Rebond command parameters . . . 32
Table 76. Aci_Gap_Allow_Rebond return parameters . . . 33
Table 77. Aci_Gap_Start_Limited_Discovery_Proc . . . 33
Table 78. Aci_Gap_Start_Limited_Discovery_Proc command parameters . . . 33
Table 79. Aci_Gap_Start_Limited_Discovery_Proc return parameters . . . 33
Table 80. Aci_Gap_Start_General_Discovery_Proc . . . 34
Table 81. Aci_Gap_Start_General_Discovery_Proc command parameters . . . 34
Table 82. Aci_Gap_Start_General_Discovery_Proc return parameters . . . 34
Table 83. Aci_Gap_Start_Name_Discovery_Proc . . . 35
Table 84. Aci_Gap_Start_Name_Discovery_Proc command parameters . . . 35
Table 85. Aci_Gap_Start_Name_Discovery_Proc return parameters . . . 36
Table 86. Aci_Gap_Start_Auto_Conn_Establishment . . . 37
Table 87. Aci_Gap_Start_Auto_Conn_Establishment command parameters . . . 37
Table 88. Aci_Gap_Start_Auto_Conn_Establishment return parameters . . . 38
Table 89. Aci_Gap_Start_General_Conn_Establishment . . . 38
Table 90. Aci_Gap_Start_General_Conn_Establishment command parameters . . . 39
Table 91. Aci_Gap_Start_General_Conn_Establishment return parameters . . . 39
Table 92. Aci_Gap_Start_Selective_Conn_Establishment . . . 40
Table 93. Aci_Gap_Start_Selective_Conn_Establishment command parameters . . . 40
Table 94. Aci_Gap_Start_General_Conn_Establishment return parameters . . . 40
Table 95. Aci_Gap_Create_Connection . . . 41
Table 96. Aci_Gap_Create_Connection command parameters . . . 41
Table 97. Aci_Gap_Create_Connection return parameters . . . 42
Table 98. Aci_Gap_Terminate_Gap_Procedure . . . 43
Table 99. Aci_Gap_Terminate_Gap_Procedure command parameters . . . 43
Table 100. Aci_Gap_Terminate_Gap_Procedure return parameters . . . 43
Table 101. Aci_Gap_Start_Connection_Update . . . 43
Table 102. Aci_Gap_Start_Connection_Update command parameters . . . 44
Table 103. Aci_Gap_Start_Connection_Update return parameters . . . 44
Table 104. Aci_Gap_Send_Pairing_Request . . . 45
Table 105. Aci_Gap_Send_Pairing_Request command parameters . . . 45
Table 106. Aci_Gap_Send_Pairing_Request return parameters . . . 45

Table 107. Aci_Gap_Resolve_Private_Address . . . 45
Table 108. Aci_Gap_Resolve_Private_Address command parameters . . . 46
Table 109. Aci_Gap_Resolve_Private_Address return parameters . . . 46
Table 110. Aci_Gap_Get_Bonded_Devices . . . 46
Table 111. Aci_Gap_Get_Bonded_Devices return parameters . . . 46
Table 112. Aci_Gap_Set_Broadcast_Mode . . . 47
Table 113. Aci_Gap_Set_Broadcast_Mode command parameters . . . 47
Table 114. Aci_Gap_Set_Broadcast_Mode return parameters . . . 48
Table 115. Aci_Gap_Start_Observation_Proc . . . 48
Table 116. Aci_Gap_Start_Observation_Proc commands parameters . . . 48
Table 117. Aci_Gap_Start_Observation_Proc return parameters . . . 49
Table 118. Aci_Gap_Is_Device_Bonded . . . 49
Table 119. Aci_Gap_Is_Device_Bonded command parameters . . . 49
Table 120. Aci_Gap_Is_Device_Bonded return parameters . . . 49
Table 121. GAP VS events . . . 50
Table 122. Evt_Blue_Gap_Pairing_Complt . . . 50
Table 123. Evt_Blue_Pass_Key_Request . . . 50
Table 124. Evt_Blue_Authorization_Request . . . 51
Table 125. Evt_Blue_Gap_Procedure_Complete . . . 51
Table 126. Evt_Blue_Gap_Addr_Not_Resolved . . . 51
Table 127. GATT VS commands . . . 52
Table 128. Aci_Gatt_Init . . . 53
Table 129. Aci_Gatt_Init return parameters . . . 53
Table 130. Aci_Gatt_Add_Serv . . . 53
Table 131. Aci_Gatt_Add_Serv command parameters . . . 54
Table 132. Aci_Gatt_Add_Service return parameters . . . 54
Table 133. Aci_Gatt_Include_Service . . . 54
Table 134. Aci_Gatt_Include_Service command parameters . . . 55
Table 135. Aci_Gatt_Include_Service return parameters . . . 55
Table 136. Aci_Gatt_Add_Char . . . 55
Table 137. Aci_Gatt_Add_Characteristic command parameters . . . 56
Table 138. Aci_Gatt_Add_Characteristic return parameters . . . 56
Table 139. Aci_Gatt_Add_Char_Desc . . . 57
Table 140. Aci_Gatt_Add_Char_Desc command parameters . . . 57
Table 141. Aci_Gatt_Add_Char_Desc return parameters . . . 58
Table 142. Aci_Gatt_Update_Char_Value . . . 58
Table 143. Aci_Gatt_Update_Char_Value command parameters . . . 59
Table 144. Aci_Gatt_Update_Char_Value return parameters . . . 59
Table 145. Aci_Gatt_Del_Characteristic . . . 59
Table 146. Aci_Gatt_Del_Char command parameters . . . 59
Table 147. Aci_Gatt_Del_Char return parameters . . . 60
Table 148. Aci_Gatt_Del_Service . . . 60
Table 149. Aci_Gatt_Del_Service command parameters . . . 60
Table 150. Aci_Gatt_Del_Service return parameters . . . 60
Table 151. Aci_Gatt_Del_Include_Service . . . 61
Table 152. Aci_Gatt_Del_Include_Service command parameters . . . 61
Table 153. Aci_Gatt_Del_Include_Service return parameters . . . 61
Table 154. Aci_Gatt_Set_Event_Mask . . . 61
Table 155. Aci_Gatt_Set_Event_Mask command parameters . . . 62
Table 156. Aci_Gatt_Set_Event_Mask return parameters . . . 62
Table 157. Aci_Gatt_Exchange_Configuration . . . 62
Table 158. Aci_Gatt_Exchange_Configuration command parameters . . . 62
Table 159. Aci_Gatt_Exchange_Configuration return parameters . . . 63
Table 160. Aci_Att_Find_Information_Req . . . 63

Table 161. Aci_Att_Find_Information_Req command parameters . . . 63
Table 162. Aci_Att_Find_Information_Req return parameters . . . 63
Table 163. Att_Find_By_Type_Value_Req . . . 64
Table 164. Aci_Att_Find_Attr_By_Typ_And_Value_Req command parameters . . . 64
Table 165. Aci_Att_Find_Attr_By_Typ_And_Value_Req return parameters . . . 64
Table 166. Aci_Att_Read_By_Type_Req . . . 65
Table 167. Aci_Att_Read_By_Type_Req command parameters . . . 65
Table 168. Aci_Att_Read_By_Typ_Req return parameters . . . 65
Table 169. Aci_Att_Read_By_Group_Type_Req . . . 66
Table 170. Aci_Att_Read_By_Grp_Type_Req command parameters . . . 66
Table 171. Aci_Att_Read_By_Grp_Type_Req return parameters . . . 66
Table 172. Aci_Att_Prepare_Write_Req . . . 67
Table 173. Aci_Att_Prepare_Write_Req command parameters . . . 67
Table 174. Aci_Att_Prepare_Write_Req return parameters . . . 67
Table 175. Aci_Gatt_Execute_Write_Req . . . 68
Table 176. Aci_Att_Execute_Write_Req command parameters . . . 68
Table 177. Aci_Att_Execute_Write_Req return parameters . . . 68
Table 178. Aci_Gatt_Disc_All_Prim_Services . . . 68
Table 179. Aci_Gatt_Disc_All_Prim_Services command parameters . . . 69
Table 180. Aci_Gatt_Disc_All_Prim_Services return parameters . . . 69
Table 181. Aci_Gatt_Disc_Prim_Service_By_Uuid . . . 69
Table 182. Aci_Gatt_Disc_Prim_serv_By_UUID command parameters . . . 69
Table 183. Aci_Gatt_Disc_Prim_Service_By_Uuid return parameters . . . 70
Table 184. Aci_Gatt_Find_Included_Services . . . 70
Table 185. Aci_Gatt_Find_Included_Services command parameters . . . 70
Table 186. Aci_Gatt_Find_Included_Services return parameters . . . 71
Table 187. Aci_Gatt_Disc_All_Charac_Of_Serv . . . 71
Table 188. Aci_Gatt_Disc_All_Charac_Of_Serv command parameters . . . 71
Table 189. Aci_Gatt_Disc_All_Charac_Of_Serv return parameters . . . 72
Table 190. Aci_Gatt_Disc_Charac_By_UUID . . . 72
Table 191. Aci_Gatt_Disc_Charac_By_UUID command parameters . . . 72
Table 192. Aci_Gatt_Disc_Charac_By_UUID return parameters . . . 73
Table 193. Aci_Gatt_Disc_All_Charac_Descriptors . . . 73
Table 194. Aci_Gatt_Disc_All_Charac_Descriptors command parameters . . . 73
Table 195. Aci_Gatt_Disc_All_Charac_Descriptors return parameters . . . 74
Table 196. Aci_Gatt_Read_Charac_Val . . . 74
Table 197. Aci_Gatt_Read_Charac_Val command parameters . . . 74
Table 198. Aci_Gatt_Read_Charac_Val return parameters . . . 74
Table 199. Aci_Gatt_Read_Charac_Using_UUID . . . 75
Table 200. Aci_Gatt_Read_Charac_Using_UUID command parameters . . . 75
Table 201. Aci_Gatt_Read_Charac_Using_UUID return parameters . . . 75
Table 202. Aci_Gatt_Read_Long_Charac_Val . . . 76
Table 203. Aci_Gatt_Read_Long_Charac_Val command parameters . . . 76
Table 204. Aci_Gatt_Read_Long_Charac_Val command return parameters . . . 76
Table 205. Aci_Gatt_Read_Multiple_Charac_Values . . . 76
Table 206. Aci_Gatt_Read_Multiple_Charac_Values command parameters . . . 77
Table 207. Aci_Gatt_Read_Multiple_Charac_Values return parameters . . . 77
Table 208. Aci_Gatt_Write_Charac_Val . . . 77
Table 209. Aci_Gatt_Write_Charac_Val command parameters . . . 77
Table 210. Aci_Gatt_Write_Charac_Val return parameters . . . 78
Table 211. Aci_Gatt_Write_Long_Charac_Val . . . 78
Table 212. Aci_Gatt_Write_Long_Charac_Val command parameters . . . 78
Table 213. Aci_Gatt_Write_Long_Charac_Val return parameters . . . 79
Table 214. Aci_Gatt_Write_Charac_Reliable . . . 79

Table 215. Aci_Gatt_Write_Charac_Reliable command parameters . . . 79
Table 216. Aci_Gatt_Write_Charac_Reliable return parameters . . . 80
Table 217. Aci_Gatt_Write_Long_Charac_Descriptor . . . 80
Table 218. Aci_Gatt_Write_Long_Charac_Descriptor command parameters . . . 80
Table 219. Aci_Gatt_Write_Long_Charac_Desc return parameters . . . 81
Table 220. Aci_Gatt_Read_Long_Charac_Desc . . . 81
Table 221. Aci_Gatt_Read_Long_Charac_Descriptor command parameters . . . 81
Table 222. Aci_Gatt_Read_Long_Charac_Descriptor return parameters . . . 82
Table 223. Aci_Gatt_Write_Charac_Descriptor . . . 82
Table 224. Aci_Gatt_Write_Charac_Descriptor command parameters . . . 82
Table 225. Aci_Gatt_Write_Charac_Descriptor return parameters . . . 83
Table 226. Aci_Gatt_Read_Charac_Descriptor . . . 83
Table 227. Aci_Gatt_Read_Charac_Descr command parameters . . . 83
Table 228. Aci_Gatt_Read_Charac_Descriptor return parameters . . . 83
Table 229. Aci_Gatt_Write_Without_Response . . . 84
Table 230. Aci_Gatt_Write_Without_Response command parameters . . . 84
Table 231. Aci_Gatt_Write_Without_Response return parameters . . . 84
Table 232. Aci_Gatt_Signed_Write_Without_Response . . . 85
Table 233. Aci_Gatt_Signed_Write_Without_Resp command parameters . . . 85
Table 234. Aci_Gatt_Signed_Write_Without_Response return parameters . . . 85
Table 235. Aci_Gatt_Confirm_Indication . . . 85
Table 236. Aci_Gatt_Confirm_Indication command parameters . . . 86
Table 237. Aci_Gatt_Confirm_Indication return parameters . . . 86
Table 238. Aci_Gatt_Write_Resp . . . 86
Table 239. Aci_Gatt_Write_Resp command parameters . . . 86
Table 240. Aci_Gatt_Write_Resp return parameters . . . 87
Table 241. Aci_Gatt_Allow_Read . . . 87
Table 242. Aci_Gatt_Allow_Read command parameters . . . 87
Table 243. Aci_Gatt_Allow_Read return parameters . . . 87
Table 244. Aci_Gatt_Set_Security_Permission . . . 88
Table 245. Aci_Gatt_Set_Security_Permission command parameters . . . 88
Table 246. Aci_Gatt_Set_Security_Permission return parameters . . . 88
Table 247. Aci_Gatt_Set_Desc_Value . . . 89
Table 248. Aci_Gatt_Set_Desc_Value command parameters . . . 89
Table 249. Aci_Gatt_Set_Desc_Value return parameters . . . 89
Table 250. Aci_Gatt_Read_Handle_Value . . . 89
Table 251. Aci_Gatt_Read_Handle_Value command parameters . . . 90
Table 252. Aci_Gatt_Read_Handle_Value return parameters . . . 90
Table 253. Aci_Gatt_Read_Handle_Value_Offset . . . 90
Table 254. Aci_Gatt_Read_Handle_Value_Offset command parameters . . . 90
Table 255. Aci_Gatt_Read_Handle_Value_Offset return parameters . . . 91
Table 256. Aci_Gatt_Update_Char_Value_Ext . . . 91
Table 257. Aci_Gatt_Read_Update_Value_Ext command parameters . . . 91
Table 258. Aci_Gatt_Update_Char_Value_Ext return parameters . . . 92
Table 259. GATT VS events . . . 92
Table 260. Evt_Blue_Gatt_Attribute_modified . . . 93
Table 261. Evt_Blue_Gatt_Procedure_Timeout . . . 93
Table 262. Evt_Blue_Gatt_Procedure_Complete . . . 93
Table 263. Evt_Blue_Gatt_Disc_Read_Charac_By_UUID_Resp . . . 94
Table 264. Evt_Blue_Gatt_Write_Permit_req . . . 94
Table 265. Evt_Blue_Gatt_Read_Permit_Req . . . 95
Table 266. Evt_Blue_Gatt_Read_Multi_Permit_Req . . . 95
Table 267. Evt_Blue_Gatt_Tx_Pool_Available . . . 95
Table 268. Evt_Blue_Att_Exchange_MTU_Resp . . . 96

Table 269. Evt_Blue_Att_Find_Information_Resp . . . 96
Table 270. Evt_Blue_Att_Find_By_Type_Value_Resp . . . 96
Table 271. Evt_Blue_Att_Read_By_Type_Resp . . . 97
Table 272. Evt_Blue_Att_Read_Resp . . . 97
Table 273. Evt_Blue_Att_Read_Blob_Resp . . . 97
Table 274. Evt_Blue_Att_Read_Multiple_Resp . . . 97
Table 275. Evt_Blue_Att_Read_By_Group_Type_Resp . . . 98
Table 276. Evt_Blue_Att_Prepare_Write_Resp . . . 98
Table 277. Evt_Blue_Att_Exec_Write_Resp . . . 98
Table 278. Evt_Blue_Gatt_Indication . . . 98
Table 279. Evt_Blue_Gatt_notification . . . 99
Table 280. Evt_Blue_Gatt_Error_Resp . . . 99
Table 281. Evt_Gatt_Server_Confirmation . . . 99
Table 282. HCI VS commands . . . 99
Table 283. Aci_Hal_Write_Config_Data . . . 100
Table 284. Aci_Hal_Write_Config_Data command parameters . . . 100
Table 285. Aci_Hal_Write_Config_Data members . . . 101
Table 286. Aci_Hal_Write_Config_Data return parameters . . . 101
Table 287. Aci_Hal_Read_Config_Data . . . 101
Table 288. Aci_Hal_Read_Config_Data command parameters . . . 102
Table 289. Aci_Hal_Read_Config_Data return parameters . . . 102
Table 290. Aci_Hal_Set_Tx_Power_Level . . . 102
Table 291. Aci_Hal_Set_Tx_Power_Level command parameters . . . 102
Table 292. Tx_power_level command parameters combination . . . 102
Table 293. Aci_Hal_Set_Tx_Power_Level return parameters . . . 103
Table 294. Aci_Hal_Device_Standby . . . 103
Table 295. Aci_Hal_Device_Standby return parameters . . . 104
Table 296. Aci_Hal_LE_Tx_Test_Packet_Number . . . 104
Table 297. Aci_Hal_LE_Tx_Test_Packet_Number return parameters . . . 104
Table 298. Aci_Hal_Tone_Start . . . 104
Table 299. Aci_Hal_Tone_Start command parameters . . . 105
Table 300. Aci_Hal_Tone_Start return parameters . . . 105
Table 301. Aci_Hal_Tone_Stop . . . 105
Table 302. Aci_Hal_Tone_Stop return parameters . . . 105
Table 303. Aci_Hal_Get_Link_Status . . . 105
Table 304. Aci_Hal_Get_Link_Status return parameters . . . 106
Table 305. Aci_Hal_Get_Link_Status return parameters . . . 106
Table 306. Aci_Hal_Get_Fw_Build_Number . . . 106
Table 307. Aci_Hal_Get_Fw_Build_Number return parameters . . . 106
Table 308. Aci_Hal_Get_Anchor_Period . . . 107
Table 309. Aci_Hal_Get_Anchor_Period return parameters . . . 107
Table 310. Evt_Blue_Initialized event . . . 107
Table 311. Evt_Blue_Lost_Events . . . 108
Table 312. Lost_Events_Map (8 bytes) . . . 108
Table 313. Fault data event . . . 109
Table 314. SPI pins . . . 111
Table 315. Document revision history . . . 115

List of figures

Figure 1. Bluetooth LE system with separate host and controller . . . 2
Figure 2. Bluetooth LE system with combined host and controller . . . 3
Figure 3. BlueNRG-MS ACI architecture . . . 3
Figure 4. HCI command packet . . . 4
Figure 5. HCI event packet . . . 6
Figure 6. HCI ACL data packet . . . 6
Figure 7. OpCode format for VS commands . . . 9
Figure 8. Event parameter 0 format for VS events . . . 10
Figure 9. SPI wire connection . . . 111
Figure 10. SPI header format . . . 113