SimpleLink™ *Bluetooth*® Low Energy CC2640 Wireless MCU

Simple Network Processor API Guide

For BLE-Stack™ Version: 2.1.0

July 2015

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. References

[1] Specification of the Bluetooth System, Core Version 4.0, June 30, 2010.
https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=282159

[2] HCI Vendor Specific Guide
C:\ti\simplelink\ble_cc26xx_2_01_00_xxxxx\TI_BLE_Vendor_Specific_HCI_Guide.pdf

[3] Software Developer's Guide
C:\ti\simplelink\ble_cc26xx_2_01_00_xxxxx\
SWRU393_CC2640_BLE_Software_Developer's_Guide.pdf

[4] Device Info Service
https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=238689

[5] Generic Access Service
https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.generic_access.xml

[6] Generic Attribute Service
https://developer.bluetooth.org/gatt/services/Pages/ServiceViewer.aspx?u=org.bluetooth.service.generic_attribute.xml

[7] Wiki Page
www.ti.com/ble-wiki

[8] TI BLE Stack
http://www.ti.com/blestack

2. Revision History

Date (YMD)	*Document version*	*Description of changes*
2015-08-25	*V1.0*	*Initial*

3. Document Specifics

Numerical Notation Conventions

Multiple-octets may be specified in hexadecimal notation in one of two ways:

Standard Hexadecimal Notation

In this notation, a single hexadecimal radix indicator "0x" precedes the entire value. The octet order as read from left to right is from most significant octet to least significant octet. For example, for the value 0x123456ABCDEF, '12' is the most significant octet and 'EF' is the least significant octet.

Colon Separated Hexadecimal Notation

In this notation, the hexadecimal radix indicator "0x" is *not* used and octets are colon separated. The octet order as read from left to right is from least significant octet to most significant octet. For example, for the value 12:34:56:AB:CD:EF, '12' is the least significant octet and 'EF' is the most significant octet.

Hyperlinks

All underlined Section headings can be clicked on to jump to the relevant section.

Definitions

AP	**Application Processor**
ATT	**Attribute Protocol**
BLE	**Bluetooth Low Energy**
BT	**Bluetooth**
CCCD	**Client Characteristic Configuration Descriptor**
GAP	**Generic Access Profile**

GATT	**Generic Attribute Profile**
HCI	**Host Controller Interface**
IDE	**Integrated Development Environment**
L2CAP	**Logical Link Control and Adaptation Protocol**
LE	**Low Energy**
LL	**Link Layer**
MTU	**Maximum Transmission Unit**
OTA	**Over The Air**
NPI	**Network Processor Interface**
NV	**Non-Volatile**
RFU	**Reserved for Future Use**
SDG	**SDG**
SNP	**Simple Network Processor**
TI	**Texas Instruments**

Table 1: Definitions

4. Purpose

The purpose of this document is to describe how to interface with the simple network processor (SNP). It will include a functional overview of the SNP, a description of the transport layer, and a detailed API. For more information on sample projects which interact with the SNP and how to develop a project with the SNP, see the TI BLE wiki page [7].

It is assumed that the reader has substantial knowledge in both BLE and the TI SDK. Therefore, it is recommended to have read the TI SDG [3] and be familiar with the BT specification [1] before using the SNP. Many topics covered here are further expounded upon by the SDG.

5. SNP Functional Overview

This section will provide an architectural overview of the SNP and describe what the AP is responsible for managing.

5.1 Architecture Overview

The SNP is a dual-device solution that requires an external MCU application processor (AP) to control the CC2650 simple network processor (SNP). The SNP was designed to simplify the complexity of the AP application by:

- Implementing as much BLE functionality as possible on the SNP, therefore simplifying what the AP needs to do.
- Using a simple interface between the AP and the SNP. For this reason, a proprietary SNP interface is used instead of the BT Spec-defined HCI interface that is used for the standard network processor project (HostTestApp included with the installer).

Almost the entire BLE stack is implemented on the SNP. This can be seen in Figure 1 below.

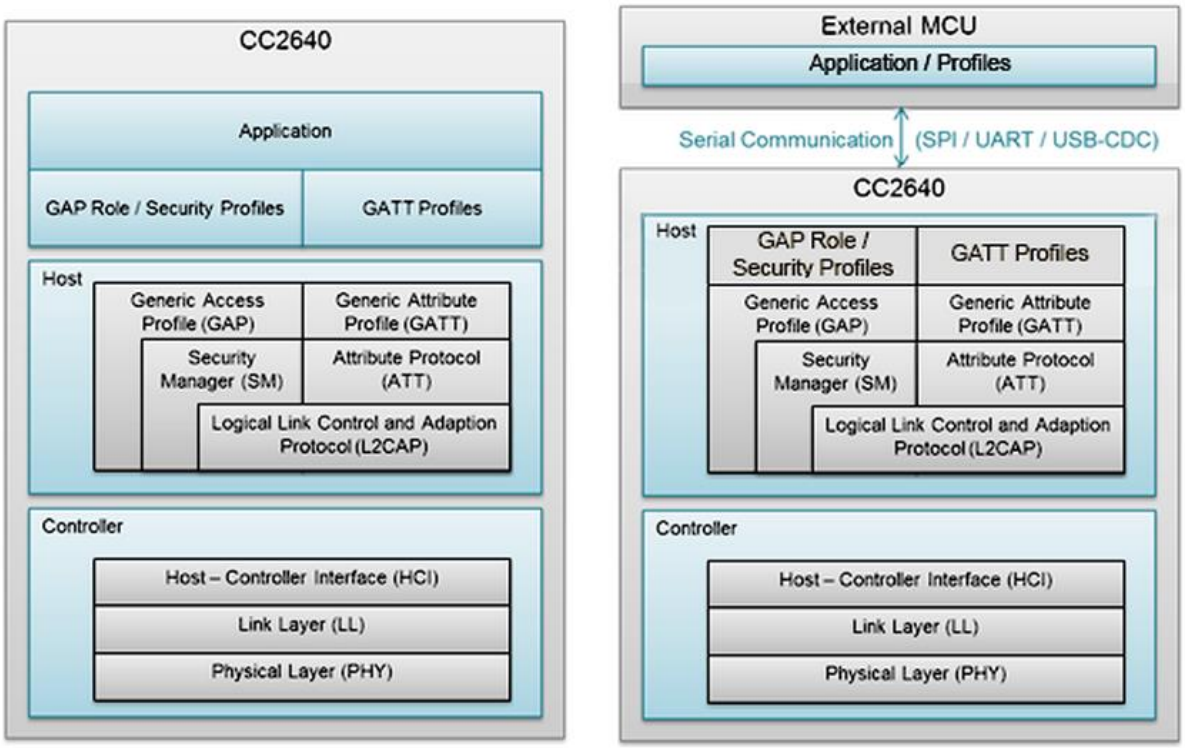

Figure 1: BLE Stack AP / NP separation

The figure on the left shows the standard embedded application and the figure on the right depicts the AP / SNP architecture. As indicated above and described throughout this document, the AP (external MCU) is only responsible for the initial configuration of the GAP and GATT services. It will also be notified of asynchronous events relating to these services such as connection establishments, attribute value updates, etc. Using the API provided here, the AP can then proceed accordingly after receiving these events. The general procedure is:

1. Initialize GATT (add services, characteristics, CCCD's, etc.). See Section 9.5.
2. Initialize GAP (advertisement data, connection parameters, etc.). See Section 9.3.
3. Advertise and optionally wait for a connection. See Section 9.4.
4. Respond to GATT requests and send notifications / indications as desired. See Section 9.6.

5.2 Default GATT Services

In order to comply with the BT spec and communicate with central devices such as mobile phones, there are two services which must be included in any BLE device:

- Generic Access Service [5]
- Device Information [4]

Therefore, these services are included by default in the SNP and are managed by the GATT server on the SNP. It is possible for the AP to modify and interact with these services via the commands and events described in the GATT Subgroup Commands and GATT Subgroup Events sections below.

The figure below depicts how these services are initialized by default in the SNP including what handle each attribute resides at. Any custom services and characteristics that the AP adds to the attribute table will always be added after these initial two services. Therefore, the handles shown below will always be the same.

Handle	Uuid	Uuid Description	Value	Properties
0x0001	0x2800	GATT Primary Service Declaration	00:18	
0x0002	0x2803	GATT Characteristic Declaration	02:03:00:00:2A	
0x0003	0x2A00	Device Name	SNP	Rd 0x02
0x0004	0x2803	GATT Characteristic Declaration	02:05:00:01:2A	
0x0005	0x2A01	Appearance	00:00	Rd 0x02
0x0006	0x2803	GATT Characteristic Declaration	02:07:00:04:2A	
0x0007	0x2A04	Peripheral Preferred Connection Parameters	50:00:A0:00:00:00:E8:03	Rd 0x02
0x0008	0x2800	GATT Primary Service Declaration	01:18	
0x0009	0x2800	GATT Primary Service Declaration	0A:18	
0x000A	0x2803	GATT Characteristic Declaration	02:0B:00:23:2A	
0x000B	0x2A23	System ID	00:00:00:00:00:00:00:00	Rd 0x02
0x000C	0x2803	GATT Characteristic Declaration	02:0D:00:24:2A	
0x000D	0x2A24	Model Number String	Model Number	Rd 0x02
0x000E	0x2803	GATT Characteristic Declaration	02:0F:00:25:2A	
0x000F	0x2A25	Serial Number String	Serial Number	Rd 0x02
0x0010	0x2803	GATT Characteristic Declaration	02:11:00:26:2A	
0x0011	0x2A26	Firmware Revision String	Firmware Revision	Rd 0x02
0x0012	0x2803	GATT Characteristic Declaration	02:13:00:27:2A	
0x0013	0x2A27	Hardware Revision String	Hardware Revision	Rd 0x02
0x0014	0x2803	GATT Characteristic Declaration	02:15:00:28:2A	
0x0015	0x2A28	Software Revision String	Software Revision	Rd 0x02
0x0016	0x2803	GATT Characteristic Declaration	02:17:00:29:2A	
0x0017	0x2A29	Manufacturer Name String	Manufacturer Name	Rd 0x02
0x0018	0x2803	GATT Characteristic Declaration	02:19:00:2A:2A	
0x0019	0x2A2A	IEEE 11073-20601 Regulatory Certification ...	FE:00:65:78:70:65:72:69:6...	Rd 0x02
0x001A	0x2803	GATT Characteristic Declaration	02:1B:00:50:2A	
0x001B	0x2A50	PnP ID	01:0D:00:00:00:10:01	Rd 0x02

Figure 2: Initial Attribute Table

Any other services that are added by the AP at run-time are managed by the AP. That is, the AP will manage the characteristic value: it will be notified when a read / write of the characteristic occurs and can respond (or not) respond with the value as it chooses.

5.3 Limitations

The SNP will be configured by default to act only as a peripheral and/or broadcaster device. This can not be modified. This implies that the device can only advertise and accept/reject connection. It can not discover devices or initiate a connection. Also, the SNP will only be configured as a GATT server; it can not be a GATT client. The standard use case is for the SNP (peripheral GATT client) to connect to a central GATT server such as a smartphone.

Furthermore, while additional features will be added for future releases, the following limitations currently exist:

- There is no NV storage.
- No direct advertisements.
- The maximum ATT_MTU_SIZE is 251 bytes
- ATT read multiple requests are not supported
- Whitelist filtering is not supported.
- The SNP does not have the Generic Attribute service defined in the embedded project. If this is needed, it should be added and managed manually by the AP.

- L2CAP is not enabled. Therefore, connection-oriented-channel data transfer is not possible.
- There is no GAP Bond manager. Therefore, there is no means of securing the connection through pairing, encryption, or bonding.
- Only the 4.0 controller options are used: the 4.1 options are disabled.
- User description attributes are read-only. They can not be updated remotely by the GATT client.
- Advertisements are always performed on all 3 advertisement channels.
- The advertiser's address type is always static (no private or public address).
- Advertising filter policies have not been implemented.
- Since only one simultaneous connection is supported, all advertising during a connection will be non-connectable.
- In order to change the type of an ongoing advertisement, it must be stopped first.
- Services and characteristics are not stored in flash after they have been added. Therefore, they will need to be re-initialized upon reset.
- Authenticated notifications and indications are not supported.

6. Compiling the SNP

The SNP project is available at: $INSTALL$\Projects\ble\SimpleNP\CC26xx where $INSTALL$ is the install location of the TI BLE stack [8]. This directory has both an IAR and a CCS project. The SNP can be configured to support one of two serial interfaces: SPI and UART. The interface can be selected by choosing the relevant project configuration in the IDE. See the SDG [3] for more information on how to use the IDE's to select a configuration and compile a project.

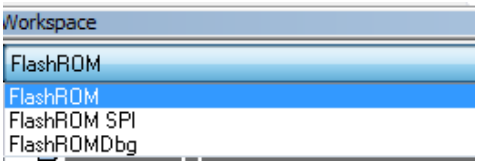

Figure 3: IAR Project Configurations

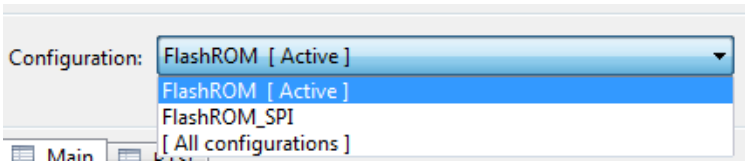

Figure 4: CCS Project Configurations.

Furthermore, power management can be enabled / disabled by including / excluding the POWER_SAVING preprocessor definition.

7. Transport Layer Specification

This section will describe the transport layer of the SNP. The SNP can be driven by UART or SPI. Note that the AP must be the protocol master device for the SNP architecture.

7.1 Packet Format

The following TL packet format is used.

SOF	Length	Cmd0	Cmd1	Data Payload	FCS
1 Byte (0xFE)	2 bytes	1 byte: bits 5-7: type, bits 0-4: subsystem	1 Byte	0-4096 bytes	1 Byte

SOF: start of Frame (1 byte). This byte must be 0xFE to indicate a start of frame.

Length: length of the data payload field (2 bytes). This length does not consider any other fields besides the data payload field.

Cmd0: the opcode is split in 2 fields:

- *Type* (bits 5-7): type of command / event

Type	Description
0x1	Synchronous Request
0x2	Asynchronous Request
0x3	Synchronous Response

Note that this type is independent of the TL messaging type described in Section 7.3.

- *Subsystem* (bits 0-4): the subsystem for which the command belongs. The subsystem used for SNP is the BLE SNP subsystem (0x15).

Cmd1: opcode of the command / event (1 byte).

Payload: the parameters of the command / event. See Section 9.

FCS: frame check sequence, calculated by an exclusive OR on all fields except SOF.

The following table shows an example of the SNP Add Service (0x81) command to send to SNP, with service type of primary service (0x01) and UUID 0xFFF0.

The SNP frame will be the following:

SOF	Length	Cmd0	Cmd1	Payload	FCS
0xFE	0x03 0x00	0x35	0x81	0x01 0xF0 0xFF	0xB9

Note that all field needs to use the little endian format (except for variable length data payload field where they are read in the order they arrive on the bus.)

7.2 Physical Interface

The purpose of this section is to describe the expected behavior of host MCUs implementing the AP regarding UART or SPI.

The SNP can be configured to support one of two serial interfaces as described in Section 6.

When configured to use SPI, the SNP project will by default use the second SPI module (SSI1). For each serial interface, power management can be enabled and regardless of interface type, MRDY and SRDY pins remain consistent. There are three different configurations:

- UART with Power Management
- UART without Power Management
- SPI with Power Management

Note that the SPI without Power Management configuration is not considered since the handshaking will be the same as the SPI with Power Management configuration.

The following signals are used to manage the power domains of the SNP: These are further explained in Section 7.3.

- “master ready” (MRDY)
- “slave ready” (SRDY).

These two signals, along with the respective SPI or UART serial bus signals, comprise the serial interface of the SNP.

7.2.1 UART without Power Management

To use this configuration, the UART project configuration should be selected and POWER_SAVING should not be defined. If power management is not enabled, only the UART TX and UART RX pins are needed. This is an always on configuration that does not allow the CC2640 to enter into low power mode.

The following table lists the default port setting for the SNP UART.

UART Parameters	Default Value
Port	CC2650_UART0
Baud Rate	115200
Data Length	8
Parity	None
Stop Bits	1
Flow Control	None

These are the default UART Pin Assignments:

PIN	Smart RF 06 Pins
UART0 Tx	P408.14
UART0 Rx	P408.12

7.2.2 UART with Power Management

To use this configuration, the UART project configuration should be selected and POWER_SAVING should be defined. This configuration allows low power modes through the inclusion of the MRDY and SRDY signals.

The following table lists the default port settings for the SNP UART.

UART Parameters	Default Value
Port	CC2650_UART0
Baud Rate	115200
Data Length	8
Parity	None
Stop Bits	1
Flow Control	None

These are the default UART Pin Assignments:

PIN	Smart RF 06 Pins
UART0 Tx	P408.14
UART0 Rx	P408.12
MRDY	P403.12
SRDY	P403.16

7.2.3 SPI

In order to allow full duplex communication, the SPI configuration requires the use of MRDY and SRDY regardless of whether or not power management is enabled.

The following table lists the default port settings for the SNP SPI

SPI Parameters	Default Value
Port	CC2650_SPI1
Baud Rate	800000
Data Length	8

These are the default SPI Pin Assignments:

PIN	Smart RF 06 SNP
SPI1 MISO	P404.18
SPI1 MOSI	P404.12

SPI CLK	P405.2
MRDY	P403.12
SRDY	P403.16

7.3 Handshaking

When enabled, the MRDY and SRDY handshaking signals comprise a handshake that determines how TL messages are sent to / from the SNP. MRDY is an input pin on the SNP and when set by the AP will wake the device enabling it to receive data over the serial bus. SRDY is an output pin on the SNP, and, when set by the SNP, signals to the AP that the AP may begin data transfer. This sequence of setting MRDY and SRDY pins comprises a “handshake” process to guarantee both devices are awake and ready to send and/or receive data.

Note that MRDY and SRDY are active low so “assertion” refers to a logic value of 0. Before any data can be sent, a “handshake” must occur and this handshake can be initiated by the SNP or the AP. There are two types of messages:

- Asynchronous Request – transfer initiated by AP, and
- Asynchronous Indication – transfer initiated by SNP.

When not using power management (and thus not using SRDY / MRDY), the two events that define an Asynchronous Request or Asynchronous Indication is the beginning of a data transfer on the UART Master TX pin or the UART Slave TX pin respectively.

7.3.1 Asynchronous Request (AREQ)

An asynchronous request is a message that is sent from AP to SNP. The AP must initiate the handshake by asserting MRDY, once SRDY has been asserted, then data can be transferred AP to SNP. MRDY is de-asserted by the AP once all data has been transferred and then SNP de-asserts SRDY notifying the AP that it may be in a low power mode. The order of events is shown below:

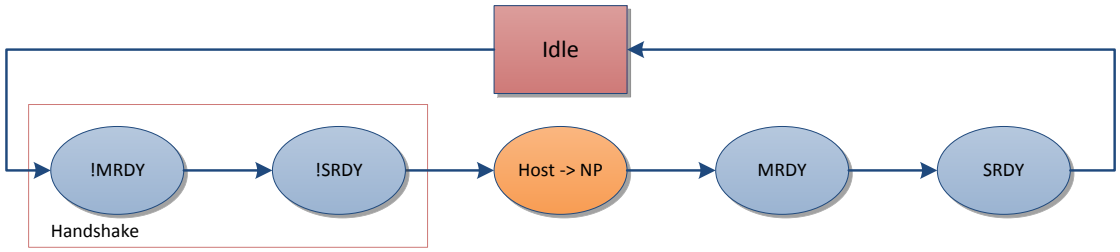

Figure 5: **AREQ Order of Events**

An AREQ message is presented in **Figure 6**. In this figure there is one AREQ message which ends once SRDY is de-asserted and a subsequent AREQ is initiated when MRDY is asserted. The following timings provided are based on testing. These do not represent absolute values but instead are values that were tested and shown to be functional.

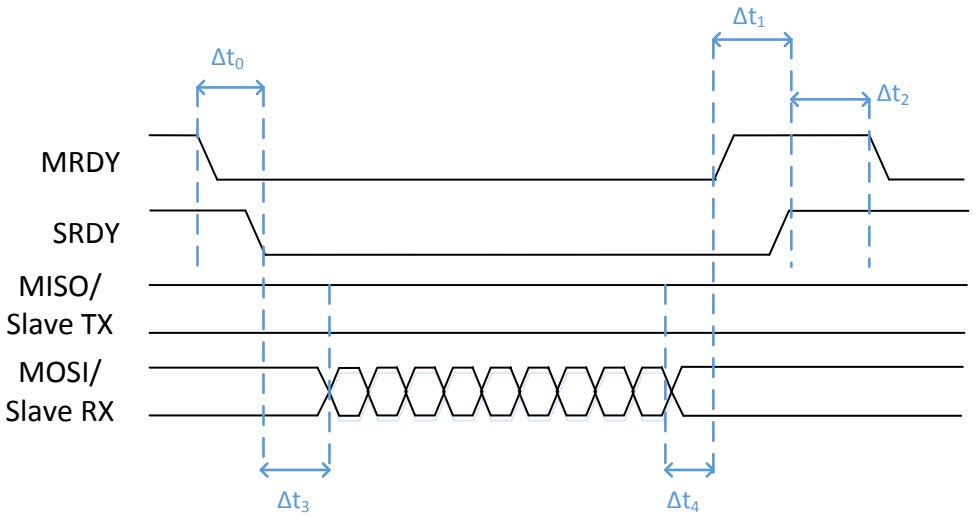

Figure 6 AREQ Timing Diagram

Interval	Minimum (µs)	Average (µs)	Maximum (µs)
Δt_0	48	90	-
Δt_1	41	46	-
Δt_2	-	510	-
Δt_3	4.1	4.5	-
Δt_4	-	101	-

Table 2 AREQ Timings

7.3.2 Asynchronous Response/Indication (AIND)

An asynchronous indication is a message that is sent from SNP to AP. The only difference between this message and AREQ is that SRDY is asserted first and the direction of the data transfer is inverted. The order of events can be seen below:

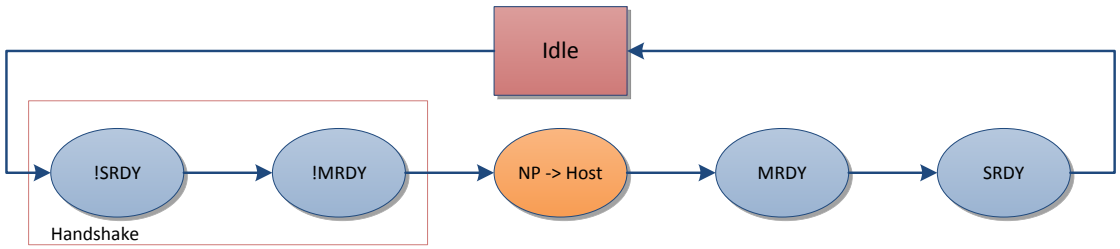

Figure 7: **AIND Order of Events**

An AIND message is presented in **Figure 6**. In this figure there is one AIND message which ends once SRDY is de-asserted and a subsequent AIND is initiated when SRDY is asserted. The following timings provided are based on testing. These do not represent absolute values but instead are values that were tested and shown to be functional.

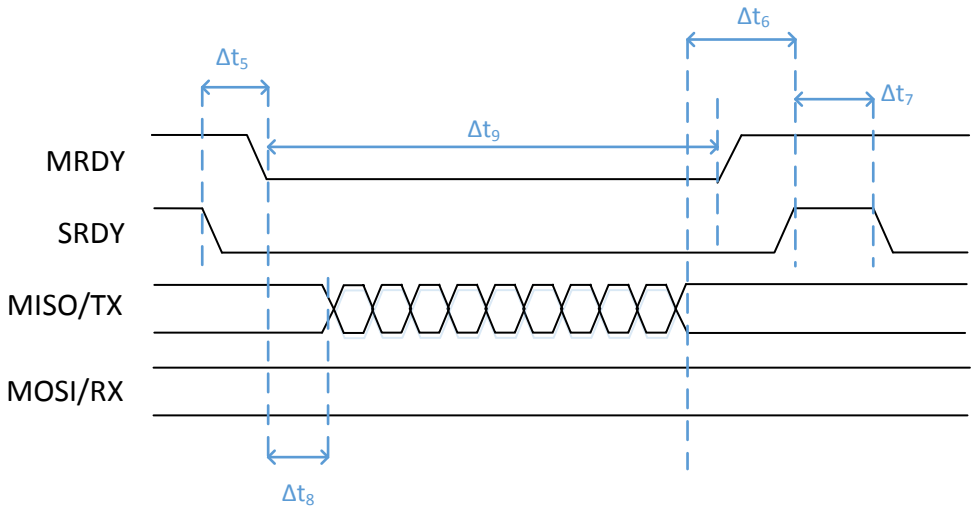

Figure 8 AIND Timing Diagram

Interval	Minimum (µs)	Average (µs)	Maximum (µs)
Δt_5	45	-	-
Δt_6	30	46	
Δt_7	270	450	-
Δt_8	.5	-	-
Δt_9	50	-	-

Table 3 AINDTimings

The timing diagram in **Figure 8** shows that MRDY is de-asserted only after the transfer on the serial bus has completed. This behavior however, is dependent on which underlying serial bus is being used.

For SPI, the SNP has no concept of when the SPI transfer has finished because the AP is in control of SCLK and determines how many bytes are clocked. It is the AP's job then to clock the correct amount of bytes and then de-assert MRDY which signals to the SNP that the transfer has indeed completed. The correct amount of bytes is determined by a length field that is present with the SPI frame that is being transferred.

For UART, the SNP is in control of when the data is transferred and has knowledge of when it has completed a transfer to the AP. Thus, the AP can de-assert MRDY prior to the actual end of the data transfer because SNP is not waiting to be signaled. This reduces the amount of processing required to receive each message since the packet is not parsed during the actual transmission. The AP will know that it has received a full message once SRDY is de-asserted.

7.3.3 Bidirectional Messaging

With any protocol that allows asynchronous messaging, where a signal transition denotes the start of a message as with MRDY and SRDY, there are inherent collisions between messages. Instead of requiring intricate collision handling, the SNP framework allows for bidirectional messaging to occur. This means that data can be sent from the AP to SNP and from SNP to the AP in the same message window regardless of the handshake order.

While reducing collision handling, bidirectional messaging adds some complexity to what operations must be performed or initiated by each device. For every AIND the SNP initiates, it must prepare to read as well as write when MRDY is asserted. For every AREQ, the AP must prepare to read as well as write once SRDY is asserted. Each device will also need to handle any FIFOs that could potentially be

overrun during a message and check at the end of every message to see what, if anything, has been received.

The scenario where bidirectional messaging will occur is when a device's input pin (SRDY wrt AP) has been asserted, and prior to asserting its output pin (MRDY wrt AP) the device gets a pending message to write. Below is the modified ordering of events where there exists now a partial order on the MRDY and SRDY assertions.

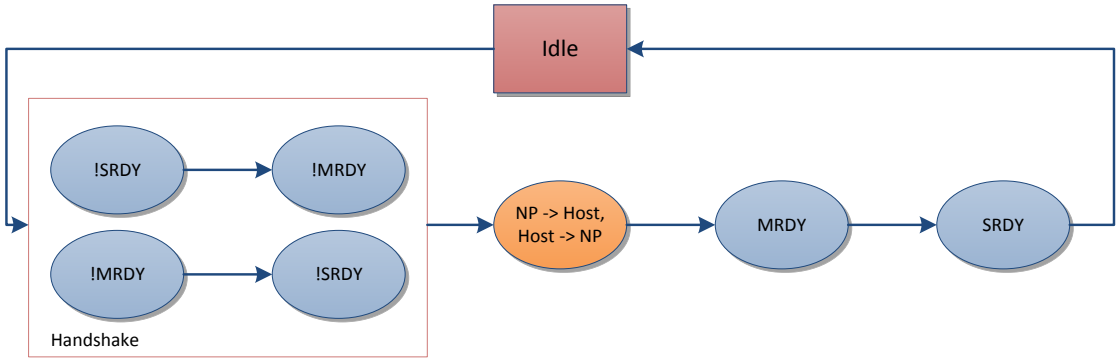

Figure 9 Bidirectional Messaging

As seen in **Figure 9**, it is still the responsibility of the AP to de-assert MRDY first. This again presents a serial bus dependent solution.

For SPI, MRDY can only be de-asserted after *both* enough bytes have been clocked out to transfer the message from AP to SNP and the full message from SNP to AP has been received. Again, the length of the message from SNP can be found in the length field of the frame that is being sent.

For UART, MRDY can be de-asserted after the AP to SNP transfer has completed. The SNP will de-assert SRDY after MRDY has been de-asserted and the SNP to AP transfer has been completed.

When SRDY / MRDY are not used, the only allowed serial interface is UART. UART inherently is capable of handling bidirectional transfers because there are two dedicated and independent channels for sending data between AP and SNP. In this situation, AIND and AREQ can be sent completely independently without the need for handshaking so the above techniques for handling handshaking collisions do not apply. Instead, AIND and AREQ messages should be sent and handled independently.

7.3.4 Fundamental Rules of MRDY and SRDY

Along with following the above scheme, there are two basic rules to the operation of the serial bus:

1. Each device must always initiate a read prior to asserting its respective output pin (MRDY wrt AP) regardless of the state of the its respective input pin (SRDY wrt AP).

2. Each device can only begin to write (or clock data in the case of SPI) once both MRDY and SRDY are asserted.

8. SNP Interface

This section will list all of the commands that can be sent to and the events that can be received from the SNP. See the Table of Contents for links to each command / event.

8.1 Device Subgroup Commands

Command	Type	Opcode
SNP Mask Event	Sync Req	0x02
SNP Get Revision	Sync Req	0x03
SNP Encapsulated HCI	Async	0x04

SNP Get Status	Sync Req	0x06
SNP Test	Sync Req	0x10

Table 4: Device Subgroup Commands

8.2 Device Subgroup Events

Event	Type	Opcode
SNP Power Up	Async	0x01
SNP Mask Event Response	Sync Rsp	0x02
SNP Get Revision Response	Sync Rsp	0x03
SNP HCI Command Response	Async	0x04
SNP Event Indication	Async	0x05
SNP Get Status Response	Sync Rsp	0x06
SNP Invalid Synchronous Command Indication	Sync Rsp	0x07
SNP Test Response	Sync Rsp	0x10

Table 5: Device Subgroup Events

8.3 GAP Subgroup Commands

Command	Type	Opcode
SNP Start Advertisement	Sync Req	0x42
SNP Set Advertisement Data	Async	0x43
SNP Stop Advertisement	Sync Req	0x44
SNP Update Connection Parameters	Sync Req	0x45
SNP Terminate Connection	Async	0x46
SNP Set GAP Parameter	Sync Req	0x48
SNP Get Gap Parameter	Sync Req	0x49

Table 6: GAP Subgroup Commands

8.4 GAP Subgroup Events

Event	Type	Opcode
SNP Set Advertisement Data Response	Async	0x43
SNP Update Connection Parameter Response	Sync Rsp	0x45
SNP Set Parameter Response	Sync Rsp	0x48
SNP Get Parameter Response	Sync Rsp	0x49

Table 7: GAP Subgroup Events

8.5 GATT Subgroup Commands

Command	Type	Opcode
SNP Add Service	Sync Req	0x81

SNP Add Characteristic Value Declaration	Sync Req	0x82
SNP Add Characteristic Descriptor Declaration	Sync Req	0x83
SNP Register Service	Sync Req	0x84
SNP Get Attribute Value	Sync Req	0x85
SNP Set Attribute Value	Sync Req	0x86
SNP Characteristic Read Confirmation	Async	0x87
SNP Characteristic Write Confirmation	Async	0x88
SNP Send Notification Indication	Async	0x89
SNP CCCD Updated Confirmation	Async	0x8B
SNP Set GATT Parameter	Sync Req	0x8C
SNP Get GATT Parameter	Sync Req	0x8D

Table 8: GATT Subgroup Commands

8.6 GATT Subgroup Events

Event	Type	Opcode
SNP Add Service Response	Sync Rsp	0x81
SNP Add Characteristic Value Declaration Response	Sync Rsp	0x82
SNP Add Characteristic Descriptor Declaration Response	Sync Rsp	0x83
SNP Register Service Response	Sync Rsp	0x84
SNP Get Attribute Value Response	Sync Rsp	0x85
SNP Set Attribute Value Response	Sync Rsp	0x86
SNP Characteristic Read Indication	Async	0x87
SNP Characteristic Write Indication	Async	0x88
SNP Send Notification Indication Response	Async	0x89
SNP CCCD Updated Indication	Async	0x8B
SNP Set GATT Parameter Response	Sync Rsp	0x8C
SNP Get GATT Parameter Response	Sync Rsp	0x8D

Table 9: GATT Subgroup Events

8.7 SNP Error Codes

Error Code	Error
0x83	SNP Failure
0x84	SNP Invalid Parameters
0x85	Command Already in Progress
0x86	Command Rejected
0x87	Out of Resources
0x88	Unknown Attribute
0x89	Unknown Service
0x8A	Already Advertising

0x8B	Not Advertising
0x8C	HCI Response Collision
0x8D	HCI Unknown Command
0x8E	GATT Collision
0x8F	Notification / Indication not Enabled by Client
0x90	Notification / Indication not Allowed
0x91	Notification / Indication does not have a CCCD Attribute
0x92	Not Connected

Table 10: SNP Error Codes

9. SNP API

9.1 Device Subgroup Commands

9.1.1 SNP Mask Event (0x02)

The SNP Mask Event command enables the AP to mask some events returned from the SNP if they are not needed by the AP. By default, all events are enabled and will be returned to the AP. Masking events may be useful in order to limit the possible wake up conditions of the AP, thus reducing power consumption.

All of the events that can masked by this command are triggered asynchronously due to an action from the AP, an action of the remote peer, or a timer expiration.

See the SNP Event Indication (0x05) event for a description of the possible events.

Command Parameters

Event Mask: (2 bytes)

Value	Parameter Description
0x0001	SNP connection establishment event
0x0002	SNP connection termination event
0x0004	SNP connection parameters updated event
0x0008	SNP advertising started event
0x0010	SNP advertising ended event
0x0020	SNP ATT MTU event
0x8000	SNP error event

Table 11: SNP Event Values

Event(s) Generated

In response to this command, the SNP will return the SNP Mask Event Response (0x02) event.

9.1.2 SNP Get Revision (0x03)

The SNP Get Revision command is used to get the current revision of the SNP API as well as the full stack revision number as defined in the HCI vendor guide.

Command Parameters

This command does not take any parameters.

Event(s) Generated

In response to this command, the SNP will return the SNP Get Revision Response (0x03) event.

9.1.3 SNP Encapsulated HCI Command (0x04)

The SNP Encapsulated HCI Command is used to encapsulate and send an HCI command to the SNP. Only the HCI commands listed in the parameter section below are supported. The functionality of the HCI extension commands can be found in the HCI Vendor Specific Guide [2]. Note that some of these commands are defined in the Bluetooth Spec [1].

Command Parameters

HCI Command OpCode: (2 bytes)

Value	Parameter Description
0xFC01	HCI_EXT_SetTxPowerCmd
0xFC08	HCI_EXT_ModemTestRxCmd
0xFC09	HCI_EXT_ModemHopTestTxCmd
0xFC0A	HCI_EXT_ModemTestRxCmd
0xFC0B	HCI_EXT_EndModemTestCmd
0xFC0C	HCI_EXT_SetBDADDRCmd
0xFC0D	HCI_EXT_SetSCACmd
0xFC0E	HCI_EXT_EnablePTMCmd
0xFC11	HCI_EXT_SetMaxDtmTxPowerCmd
0xFC1D	HCI_EXT_ResetSystemCmd
0x1009	HCI_EXT_ReadBDADDRCmd
0x1405	HCI_ReadRssiCmd
0x201D	LE Receiver Test
0x201E	LE Transmitter Test
0x201F	LE Test End
0xFC14	HCI_EXT_PacketErrorRateCmd
0xFC05	HCI_EXT_DecryptCmd
0x2017	LE Encrypt
0xFC1A	HCI_EXT_SetSlaveLatencyOverrideCmd
0xFC07	HCI_EXT_SetFastTxResponseTimeCmd
0xFC02	HCI_EXT_OnePacketPerEventCmd
0xFC20	HCI_EXT_GetConnInfoCmd

Table 12: Supported HCI Commands

HCI Command Parameters: (n bytes)

Value	Parameter Description
0xXXXX….XXXX	The HCI Commands themselves may have parameters. These parameters are defined in the HCI Vendor Specific Guide and the same parameters are used here.

Event(s) Generated

In response to this command, the SNP will return an SNP HCI Command Response (0x04) event.

9.1.4 SNP Get Status (0x06)

The SNP Get Status command is used to get the current status of the SNP. See the SNP Get Status Response (0x06) event for a description of the possible statuses returned.

Command Parameters

This command does not take any parameters.

Event(s) Generated

In response to this command, the SNP will return the SNP Get Status Response (0x06) event.

9.1.5 SNP Test (0x10)

The SNP Test command is used to profile the heap usage of the SNP. This command is for debug use only: it can not be present in the final release.

Command Parameters

This command does not take any parameters.

Event(s) Generated

In response to this command, the SNP will return the SNP Test Response (0x10) event.

9.2 Device Subgroup Events

9.2.1 SNP Power Up (0x01)

The SNP Power Up event is sent from the SNP once the device has powered up and upon reset. Upon reception of this event, the AP should assume that the SNP has lost all previous configuration information.

When this event is received, the SNP is initialized and running.

Event Parameters

This event does not have any parameters.

9.2.2 SNP Mask Event Response (0x02)

The SNP will return the SNP Mask Event Response event to the AP after it has performed the SNP Mask Event (0x02) command. The event mask passed as a parameter in this event should be checked to verify the desired events were masked correctly.

Event Parameters

Masked Events (2 bytes):

Value	Parameter Description
0xXXXX	Events that have been masked. See SNP Error Codes

9.2.3 SNP Get Revision Response (0x03)

The SNP will return the SNP Get Revision Response event to the AP after it has performed the SNP Get Revision (0x03) command. The parameters defined below will describe the specific revision of the SNP.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

SNP Version (2 bytes)

Value	Parameter Description
0xXX	Version of the SNP (major, minor)

Stack Build Version (10 bytes)

Value	Parameter Description
0xXXXX….XXXX	Stack Revision. See the HCI Vendor Specific Guide for more info.

9.2.4 SNP HCI Command Response (0x04)

The SNP will return the SNP HCI Command Response event to the AP after it has performed the command specified by the preceding SNP Encapsulated HCI Command (0x04). The status parameter of this event should be checked to verify that the HCI command was performed successfully.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Opcode (2 bytes)

Value	Parameter Description
0xXXXX	HCI opcode that this event corresponds to

HCI Response (n bytes)

Value	Parameter Description
0xXXXX….XXXX	The relevant HCI event. This will match the event described in the HCI Vendor Specific Guide [2], including any parameters that are part of the event. The only difference is that the event opcode is not part of the event. Instead, the Opcode parameter described above can be used to see what command this event corresponds to.

9.2.5 SNP Event Indication (0x05)

The SNP Event Indication event is sent by the SNP to forward an asynchronous event to the AP. Note these events can be optionally masked by using the SNP Mask Event (0x02) command if they are not needed by the AP. Only one event will be indicated at a time per SNP Event Indication event. If several events occur simultaneously, they will each be encapsulated in their own event.

Note that each SNP Event Indication event will have, at minimum, the event type parameter. The proceeding parameters will depend on what the event type is. See the parameter section below. For example, the connection establishment event will only have the parameters described under “SNP Connection Establishment Parameters.”

General Event Parameters

Event Type: (2 bytes)

Value	**Parameter Description**
0x0001	Connection establishment event
0x0002	Connection termination event
0x0004	Connection parameters updated event
0x0008	Advertising started event
0x0010	Advertising ended event
0x0020	ATT MTU event
0x8000	Error event

SNP Connection Establishment Parameters

Connection Handle (2 byte)

Value	Parameter Description
0xXXXX	An index used to refer to a connection so that multiple connections can be differentiated between.

Connection Interval (2 bytes)

Value	Parameter Description
0xXXXX	Connection Interval used upon connection establishment

Slave Latency (2 bytes)

Value	Parameter Description
0xXXXX	Slave Latency used upon connection establishment

Supervision Timeout (2 bytes)

Value	Parameter Description
0xXXXX	Supervision timeout used upon connection establishment.

Address Type (1 byte): address type of initiator

Value	Parameter Description
0x00	Public Address.
0x01	Static Address
0x02	Private Nonresolvable Address
0x03	Private Resolvable Address

Initiator Address (6 bytes)

Value	Parameter Description
0xXXXXXXXXXXXX	Address of the device which initiated the connection.

SNP Connection Termination Parameters

Connection Handle (2 byte)

Value	Parameter Description
0xXXXX	Handle of the connection that was terminated.

Reason (1 byte): reason that the connection was terminated

Value	Parameter Description
0x08	Supervision timeout
0x13	Peer Requested
0x16	Host Requested
0x22	Control Packet Timeout
0x28	Control Packet Instance Passed
0x3B	LSTO Violation
0x3D	MIC Failure

SNP Connection Parameters Updated Parameters

Connection Handle (2 byte)

Value	Parameter Description
0xXXXX	Handle of connection which was updated.

Connection Interval (2 bytes)

Value	Parameter Description
0xXXXX	Updated connection interval

Slave Latency (2 bytes)

Value	Parameter Description
0xXXXX	Updated slave latency

Supervision Timeout (2 bytes)

Value	Parameter Description
0xXXXX	Updated supervision timeout.

SNP Advertising Started Parameters

Status (1 byte)

Value	Parameter Description
0x00	Advertising Successfully Started
0xXX	See SNP Error Codes

SNP Advertising Ended Parameters

Status (1 byte)

Value	Parameter Description
0x00	Advertising Stopped Successfully
0xXX	See SNP Error Codes

SNP ATT MTU Parameters

Connection Handle (2 byte)

Value	Parameter Description
0xXXXX	Handle of the connection where the MTU size was updated.

MTU Size (2 bytes)

Value	Parameter Description
0xXXXX	New ATT MTU size negotiated between GATT client and server

SNP Error Event Parameters

Error (1 byte): error that occurred

Value	Parameter Description
0xXX	See SNP Error Codes

Additional Notes

It is important to keep track of the ATT_MTU_SIZE in order to manage fragmentation of GATT packets. The default ATT_MTU size is set to 23 bytes. This implies that, by default, there is no fragmentation at the HCI layer. A GATT client can request an ATT_MTU_EXCHANGE method to change the maximum possible ATT MTU size. The SNP is configured to manage ATT MTU size up to 251 bytes. If this update occurs, the corresponding event will be send by the SNP. If this event is never received, the AP should assume that ATT MTU size is 23.

9.2.6 SNP Get Status Response (0x06)

The SNP Get Status Response event is sent from the SNP in response to the SNP Get Status (0x06) command. It returns the following sets of statuses:

Event Parameters

GAPRole Status (1 byte)

Value	Parameter Description
0x00	Waiting to be started
0x01	Started but not advertising
0x02	Advertising using connectable advertising
0x03	Advertising using non-connectable advertising
0x04	In waiting period before advertising again
0x05	Just timed out from a connection and is in waiting period before advertising again
0x06	Connected
0x07	Connected and Advertising
0x08	Error Occurred

Advertising Status (1 byte)

Value	Parameter Description
0x00	Advertising Disabled
0x01	Advertising Enabled

ATT Status (1 byte)

Value	Parameter Description
0x00	No ATT operation ongoing
0x01	Ongoing ATT operation

ATT method (1 byte): current ATT operation in progress

Value	Parameter Description
0x01	ATT Error Response
0x02	ATT Exchange MTU Request
0x03	ATT Exchange MTU Response
0x04	ATT Find Information Request
0x05	ATT Find Information Response
0x06	ATT Find By Type Value Request
0x07	ATT Find By Type Value Response
0x08	ATT Read By Type Request
0x09	ATT Read By Type Response
0x0A	ATT Read Request
0x0B	ATT Read Response
0x0C	ATT Read Blob Request
0x0D	ATT Read Blob Response
0x0E	ATT Read Multiple Request
0x0F	ATT Read Multiple Response
0x10	ATT Read By Group Type Request
0x11	ATT Read By Group Type Response
0x12	ATT Write Request

0x13	ATT Write Response
0x16	ATT Prepare Write Request
0x17	ATT Prepare Write Response
0x18	ATT Execute Write Request
0x19	ATT Execute Write Response
0x1B	ATT Handle Value Notification
0x1D	ATT Handle Value Indication
0x1E	ATT Handle Value Confirmation
0x52	ATT Write Command
0xD2	ATT Signed Write Command

9.2.7 SNP Invalid Synchronous Command Indication (0x07)

The SNP Invalid Synchronous Command Indication event will be sent from the SNP in order to allow the AP to recover if an unknown synchronous packet is sent from the AP. This is necessary because an unknown command sent as a synchronous packet will prevent any other commands from being sent.

After this event is received, another asynchronous error event will be sent as SNP Event Indication (0x05) event (of SNP Error Event type) with the opcode of the offending command as a parameter.

Event Parameters

This event has no parameters.

9.2.8 SNP Test Response (0x10)

The SNP Test Response event is sent from the SNP in response to the SNP Test (0x10) command. It can only be used during debugging and is meant for profiling the heap.

Event Parameters

Mem Allocated (2 bytes)

Value	Parameter Description
0xXXXX	Heap memory that is allocated when this event was sent (in bytes).

Mem Max (2 bytes)

Value	Parameter Description
0xXXXX	High water mark of heap: maximum heap used (in bytes).

Mem Size (2 bytes)

Value	Parameter Description
0xXXXX	Total size of the heap (in bytes).

9.3 GAP Subgroup Commands

9.3.1 SNP Start Advertisement (0x42)

The SNP Start Advertisement command is sent to the SNP to start advertising on all 3 channels. Note that the SNP Set Advertisement Data (0x43) command should be called before this command in order to set the advertising data.

Command Parameters

Advertisement Type: (1 byte)

Value	Parameter Description

0x00	Connectable Undirected Advertisements
0x02	Scannable Undirected Advertisement
0x03	Non-Connectable Undirected Advertisement

Timeout (2 bytes)

Value	Parameter Description
0x0000	Advertise infinitely. See additional notes below.
0xXXXX	How long to advertise for (in ms)

Interval (2 bytes)

Value	Parameter Description
0xXXXX	Advertising Interval (n * 0.625 ms)

Filter Policy (1 byte)

Value	Parameter Description
0xXX	RFU

Initiator Address Type (1 byte)

Value	Parameter Description
0xXX	RFU

Initiator Address (6 bytes)

Value	Parameter Description
0xXXXXXXXXXXXX	RFU

Behavior (1 byte)

Value	Parameter Description
0x00	Advertising is disabled during connection and will not start after.
0x01	Advertising will continue with non-connectable advertising when connection is established. See additional notes below.
0x02	Advertising will restart with connectable advertising when a connection is terminated.

Additional Notes

If a timeout value equal to 0 is used, the SNP will advertise infinitely if it is in general advertisement mode or for 180 seconds if it is in limited discovery mode. See the SNP Get GAP Parameter (0x49) command for setting the advertising mode.

If an interval value equal to 0 is used, the default value of 100 ms will be used.

Since the SNP only supports one connection, advertisement in a connection can only be non-connectable advertisement.

If the behavior parameter is set to 0x01, advertising will continue with non-connectable advertising when a connection is established. The advertising interval in this case is set by the TGAP_CONN_ADV_INT_MIN and TGAP_CONN_ADV_INT_MAX parameters. By default, those parameters are set to 1280ms. They can be changed by using SNP Get GAP Parameter (0x49) command.

Event(s) generated

An SNP Event Indication (0x05) event with type Advertising Started will be sent from the SNP.

9.3.2 SNP Set Advertisement Data (0x43)

The SNP Set Advertisement Data command is sent to the SNP to update the raw data of either the scan response or the advertisement information.

There are 2 buffers for the advertisement data:

- A buffer for the non-connected state (device is not in a connection)
- A buffer for the connected state (device is in a connection)

When not in a connection, if advertisement is requested, the advertisement data stored in the non-connected state buffer will be advertised.

When in a connection, if advertisement is requested, the advertisement data stored in the connected state buffer will be advertised. If the connected state buffer has not been set, then the advertising data of the non-connected mode will be used. This way, if the user does not care about differentiating advertising data in connected mode and non- connected modes, the connected mode data buffer does not have to be set.

Command Parameters

Advertisement Type (1 byte): which buffer to update.

Value	Parameter Description
0x00	Scan Response Data
0x01	Non-connectable Advertisement Data
0x02	Connectable Advertisement Data

Data (1-31 bytes)

Value	Parameter Description
0xXXXX…XXXXXXXXX	Advertisement / Scan Response Data.

Additional Notes

The maximum advertisement / scan response size is 31 Bytes.

If this command is not called, the following default advertisement data will be used:

- 0x02 (length of this field)
- 0x01 (advertising type flags)
- 0x06 (general discovery, no BREDR)
- 0x53 (‘S’)
- 0x4E (‘N’)
- 0x50 (‘P’)

Event(s) generated

The SNP Set Advertisement Data Response (0x43) event will be returned from the SNP.

9.3.3 SNP Stop Advertisement (0x44)

The SNP Stop Advertisement command is sent to the SNP to stop advertising.

Command Parameters

This command does not have any parameters.

Event(s) generated

An SNP Event Indication (0x05) event with type Advertising Ended will be sent from the SNP.

9.3.4 SNP Update Connection Parameters (0x45)

The SNP Update Connection Parameters command is sent to the SNP to update the connection parameters while in a connection.

Command Parameters

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Connection Handle to update parameters

Minimum Interval (2 bytes)

Value	Parameter Description
0xXXXX	Minimum desired connection interval

Maximum Interval (2 bytes)

Value	Parameter Description
0xXXXX	Maximum desired connection interval

Slave Latency (2 bytes)

Value	Parameter Description
0xXXXX	Desired slave latency

Supervision Timeout (2 bytes)

Value	Parameter Description
0xXXXX	Desired supervision timeout

Event(s) generated

The SNP Set Parameter Update Response (0x48) event will be returned from the SNP.

9.3.5 SNP Terminate Connection (0x46)

The SNP Terminate Connection command is sent to the SNP to terminate an ongoing connection.

Command Parameters

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Connection Handle to terminate connection.
0xFFFE	Terminate ongoing connection request.
0xFFFF	Terminate all connections.

Option (1 byte): type of disconnection wanted

Value	Parameter Description
0x00	Default: gracefully disconnect by sending a termination over-the-air.
0x01	Terminate immediately: abruptly end the connection by not sending any more RF packets. The other side of the connection will experience a time out.

Event(s) generated

An SNP Event Indication (0x05)event with type Connection Termination will be sent from the SNP.

9.3.6 SNP Set GAP Parameter (0x48)

The SNP Set GAP Parameter command is sent to the SNP to modify the value of a GAP parameter. The parameters are listed here. See section 12.17 of the HCI Vendor Specific guide [2] or the GAP API in the SDG [3] for more information on each parameter.

Command Parameters

Parameter ID (2 bytes): which parameter to update

Value	Parameter Description
0x0001	Time (ms) to remain advertising in General Discovery mode. Setting this to 0 turns off this timeout, thus advertising infinitely. Default is 0 (continue indefinitely)
0x0002	Time (sec) to remain advertising in Limited Discovery mode. Default is 180 seconds.
0x0003	Time (ms) to perform scanning for General Discovery.
0x0004	Time (ms) to perform scanning for Limited Discovery.
0x0005	Advertising timeout (ms) when performing Connection Establishment.
0x0006	Timeout (ms) for link layer to wait to receive connection parameter update response.
0x0007	Minimum advertising interval in limited discovery mode (n * 0.625 ms)
0x0008	Maximum advertising interval in limited discovery mode (n * 0.625 ms)
0x0009	Minimum advertising interval in general discovery mode (n * 0.625 ms)
0x000A	Maximum advertising interval in general discovery mode (n * 0.625 ms)
0x000B	Minimum advertising interval when in connectable mode (n * 0.625 ms)
0x000C	Maximum advertising interval when in connectable mode (n * 0.625 ms)
0x000D	Scan interval used during Link Layer Initiating state, when in Connectable mode (n * 0.625 mSec)
0x000E	Scan window used during Link Layer Initiating state, when in Connectable mode (n * 0.625 mSec)
0x000F	Scan interval used during Link Layer Initiating state, when in Connectable mode, high duty scan cycle scan parameters (n * 0.625 mSec)
0x0010	Scan window used during Link Layer Initiating state, when in Connectable mode, high duty scan cycle scan parameters (n * 0.625 mSec)
0x0011	Scan interval used during Link Layer Scanning state, when in General Discovery proc (n * 0.625 mSec).
0x0012	Scan window used during Link Layer Scanning state, when in General Discovery proc (n * 0.625 mSec)
0x0013	Scan interval used during Link Layer Scanning state, when in Limited Discovery proc (n * 0.625 mSec)
0x0014	Scan window used during Link Layer Scanning state, when in Limited Discovery proc (n * 0.625 mSec)
0x0015	Minimum Link Layer connection interval, when using Connection Establishment proc (n * 1.25 mSec)
0x0016	Maximum Link Layer connection interval, when using Connection Establishment proc (n * 1.25 mSec)
0x0017	Scan interval used during Link Layer Initiating state, when using Connection Establishment proc (n * 0.625 mSec)
0x0018	Scan window used during Link Layer Initiating state, when using Connection Establishment proc (n * 0.625 mSec)
0x0019	Link Layer connection supervision timeout, when using Connection Establishment proc (n * 10 mSec)
0x001A	Link Layer connection slave latency, when using Connection Establishment proc (in number of connection events)

0x001B	Local informational parameter about min len of connection needed, when using Connection Establishment proc (n * 0.625 mSec)
0x001C	Local informational parameter about max len of connection needed, when using Connection Establishment proc (n * 0.625 mSec).
0x001D	Minimum Time Interval between private (resolvable) address changes. In minutes (default 15 minutes)
0x001E	Central idle timer. In seconds (default 1 second)
0x001F	Minimum time upon connection establishment before the peripheral starts a connection update procedure. In seconds (default 5 seconds)
0x0020	Time (ms) to wait for security manager response before returning bleTimeout. Default is 30 seconds.
0x0021	SM Minimum Key Length supported. Default 7.
0x0022	SM Maximum Key Length supported. Default 16.
0x0023	TRUE to filter duplicate advertising reports. Default TRUE.
0x0024	Minimum RSSI required for scan responses to be reported to the app. Default -127.
0x0025	Whether or not to reject Connection Parameter Update Request received on Central device. Default FALSE.

Value (2 byte)

Value	Parameter Description
0xXXXX	New value of GAP Parameter

Event(s) generated

An SNP Set Parameter Update Response (0x48) event will be sent from the SNP.

9.3.7 SNP Get GAP Parameter (0x49)

The SNP Get GAP Parameter command is sent to the SNP to read the value of a GAP parameter. The parameters are listed in the SNP Set GAP Parameter (0x48) command. See section 12.17 of the HCI Vendor Specific guide or the GAP API in the SDG for more information on each parameter.

Command Parameters

Parameter ID (2 bytes)

Value	Parameter Description
0xXXXX	Which parameter to read. See SNP Set GAP Parameter (0x48)

9.4 GAP Subgroup Events

9.4.1 SNP Set Advertisement Data Response (0x43)

The SNP will return the SNP Set Advertisement Data Response event to the AP after it has performed the SNP Set Advertisement Data (0x43) command. The status of this event should be checked to verify that the advertising data was set correctly.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success

0xXX	See SNP Error Codes

9.4.2 SNP Update Connection Parameter Response (0x45)

The SNP will return the SNP Update Connection Parameter Response event to the AP after it has performed the SNP Update Connection Parameters (0x45) command. The status of this event should be checked to verify that the connection parameters were updated successfully.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Connection handle of parameter update

9.4.3 SNP Set Parameter Update Response (0x48)

The SNP will return the SNP Set Parameter Update Response event to the AP after it has performed the SNP Set GAP Parameter (0x48) command. The status of this event should be checked to verify that the parameter was set correctly.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

9.4.4 SNP Get Parameter Update Response (0x49)

The SNP will return the SNP Get Parameter Update Response event to the AP after it has performed the SNP Get GAP Parameter (0x49) command. The status of this event should be checked to verify that the parameter was read successfully and to see what the value of the parameter is.

Command Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Parameter ID (2 bytes)

Value	Parameter Description
0xXXXX	Which parameter to read. See SNP Set GAP Parameter (0x48)

Value (2 bytes)

Value	Parameter Description
0xXXXX	Value of the GAP Parameter

9.5 GATT Subgroup Commands

Note that, after they are added to the SNP's GATT server database, any AP-initialized services and characteristics are not stored in flash on the SNP. Therefore, if a reset of the SNP occurs, all services and characteristics will be lost and need to be added again from the AP.

It is necessary to have a strong background in GATT and ATT to understand how to use these commands. For more information, see the GATT section of the SDG **Error! Reference ource not found.** or Bluetooth spec.

9.5.1 SNP Add Service (0x81)

The SNP Add Service command is sent to the SNP to start the addition of a new service to the GATT server running on the SNP. This command should be followed by the SNP Add Characteristic Value Declaration (0x82) command and optionally the SNP Add Characteristic Descriptor Declaration (0x83) command to add characteristics to the service. Lastly, when all characteristics have been added, the SNP Register Service command should be sent to the SNP.

Note that only one service can be added at a time. That is, once this command has been sent, all characteristics must then be added and the service must to be registered before adding another service.

Command Parameters

Type (1 byte): type of the service to add

Value	Parameter Description
0x01	Primary Service
0x02	Secondary Service

UUID (16 bytes)

Value	Parameter Description
0xXXXX….XXXX	UUID of service to add.

Event(s) generated

The SNP Add Service Response (0x81) event will be returned from the SNP.

9.5.2 SNP Add Characteristic Value Declaration (0x82)

The SNP Add Characteristic Value Declaration command is sent to the SNP to add the two BT-spec defined mandatory attributes for a characteristic:

- characteristic value
- characteristic declaration.

This command should be sent after the SNP Add Service (0x81) command has been sent to begin service declaration.

Note that it is possible to add additional optional attributes to the characteristic using the SNP Add Characteristic Descriptor Declaration (0x83) command.

Command Parameters

Value Permissions (1 byte): type of the service to add

Value	Parameter Description
0x01	GATT Read Permission

0x02	GATT Write Permission

Value Properties (2 bytes)

Value	Parameter Description
0x0002	GATT Read Properties
0x0004	GATT Write No Response Properties
0x0008	GATT Write Properties
0x0010	GATT Notification Properties
0x0020	GATT Indication Properties

Management Option (1 byte)

Value	Parameter Description
0xXX	RFU

Value Max Length (2 bytes)

Value	Parameter Description
0xXXXX	Maximum length of the attribute value.

UUID (16 bytes)

Value	Parameter Description
0xXXXX….XXXX	UUID of characteristic to add.

Additional Notes

If the GATT Notify or GATT Indicate properties are enabled, a CCCD attribute for characteristic must be added with the SNP Add Characteristic Descriptor Declaration (0x83) command. Otherwise, it will not be possible to notify or indicate the characteristic value.

Note that the TI BLE Stack expects the properties to be aligned logically with the permissions. For example, a characteristic with read properties would most likely have read permissions. This is explained in more detail in the GATT section of the SDG **Error! Reference source not found.**.

Event(s) generated

The SNP Add Characteristic Value Declaration (0x82) event will be returned from the SNP.

9.5.3 SNP Add Characteristic Descriptor Declaration (0x83)

The SNP Add Characteristic Descriptor Declaration command is sent to the SNP to add one or more of the following attributes to a characteristic:

- User description string
- CCCD
- Presentation format
- Server Characteristic configuration (RFU)
- Aggregate format (RFU)

The first parameter of the command is a header indicating which attribute is being added. When a bit is set corresponding to the attribute type, the corresponding set of parameters must be present in the following parameters and will be added to the characteristic. These sets of parameters need to appear in the same order as the bits appear in the header (from least-significant bit to most-significant bit).

For example, if both a CCCD and User Description attribute are being added, the header will have the value 0x84. The CCCD parameter (1 byte) will then follow the header, and the user description parameter (n Bytes) will follow the CCCD parameter. Other parameters must be omitted.

General Command Parameters

Header (1 byte): which attributes will be added to the service

Value	Parameter Description
0x01	Generic-Short UUID
0x02	Generic-Long UUID
0x04	CCCD
0x08	Presentation Format
0x80	User Description String

Generic-Short UUID Parameters

Short UUID (5 bytes)

Value	Parameter Description
0xXXXXXXXXXX	Short UUID

Generic-Long UUID Parameters

Long UUID (19 bytes)

Value	Parameter Description
0xXXXX…XXXX	Long UUID

CCCD Parameters

Permissions (1 byte): GATT permissions of CCCD attribute

Value	Parameter Description
0x01	GATT Read Permissions
0x02	GATT Write Permissions

Presentation Format Parameters

Format (1 byte)

Value	Parameter Description
0xXX	Format, as described in BLE Spec

Exponent (1 byte)

Value	Parameter Description
0xXX	Exponent, as described in BLE Spec

Unit (2 bytes)

Value	Parameter Description
0xXXXX	Unit, as described in BLE Spec

Desc (2 bytes)

Value	Parameter Description
0xXXXX	Namespace, as described in BLE Spec

User Description String Parameters

Permissions (1 byte): GATT permissions of the User Description Attribute

Value	Parameter Description
0x01	GATT Read Permissions
0x02	GATT Write Permissions

Max Length (2 bytes)

Value	Parameter Description

0xXXXX	Maximum Possible length of the user description string (from 1 to 512)

Initial Length (2 bytes)

Value	Parameter Description
0xXXXX	Initial length of the string. Must be less than the Max Length.

Description (1-512 bytes)

Value	Parameter Description
0xXXXX…XXXX	Initial user description string

Additional Notes

The presentation format is described in the BLE spec [1], Vol 3, Part F, chapter 3.3.

The attribute permission for the User Description attribute will be set to GATT Read Permissions.

Event(s) generated

The SNP Add Characteristic Descriptor Declaration Response (0x83) event will be returned from the SNP.

9.5.4 SNP Register Service (0x84)

The SNP Register Service command will be sent to the SNP to register the service and characteristics previously added to the GATT server, thus ending the creation of the service.

Command Parameters

This command does not have any parameters.

Event(s) generated

The SNP Register Service Response (0x84)SNP Add Characteristic Descriptor Declaration Response (0x83) event will be returned from the SNP.

9.5.5 SNP Get Attribute Value (0x85)

The SNP Get Attribute Value command is sent to the SNP to read the attribute value of a characteristic that is managed by the GATT server on the SNP. This includes characteristics in the Device Information Service [4] and the Generic Access Service [5]. More information about these characteristics can be found in the relevant service specification. This command can only be used to write the following attribute types:

- User Description
- Presentation Format

For reading the characteristic value of an SNP-managed attribute, the SNP Get GATT Parameter (0x8D) command can be used.

Command Parameters

Attribute Handle (2 bytes): handle of attribute value to be read

Value	Parameter Description
0xXXXX	See Figure 2: Initial Attribute Table.

Additional Notes

If the characteristic value is managed by the AP, this request will be rejected.

Generated Event(s)

The SNP Get Attribute Value Response (0x85) event will be returned from the SNP.

9.5.6 SNP Set Attribute Value (0x86)

The SNP Set Attribute Value command is sent to the SNP to write the attribute value of a characteristic that is managed by the GATT server on the SNP. This includes characteristics in the Device

Information Service [4] and the Generic Access Service [5]. More information about these characteristics can be found in the relevant service specification. This command can only be used to write the following attribute types:

- User Description
- Presentation Format

For writing the characteristic value of an SNP-managed attribute, the SNP Get GATT Parameter (0x8D) command can be used.

Note that the attribute value to set is indexed by the attribute handle.

Command Parameters

Attribute Handle (2 bytes): handle of attribute value to write

Value	Parameter Description
0xXXXX	See Figure 2: Initial Attribute Table.

Additional Notes

If the characteristic value is managed by the AP, this request will be rejected.

Generated Event(s)

The SNP Set Attribute Value (0x86) event will be returned from the SNP.

9.5.7 SNP Characteristic Read Confirmation (0x87)

If a GATT client requests a read of a characteristic that was added by the AP, the SNP GATT server will forward an SNP Characteristic Read Indication (0x87) event to the AP. The AP then has 30 seconds to respond with an SNP Characteristic Read Confirmation stating whether the read is allowed based on application-specific rules and, if so, what the characteristic value is. If no response is received in 30 seconds, an ATT timeout will occur and no more ATT communication can happen without reestablishing the connection.

Command Parameters

Status (1 byte): status of the read request

Value	Parameter Description
0x00	Success
0xXX	AP-define error code.

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Handle of the connection.

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Handle of the characteristic value attribute being read.

Offset (2 bytes)

Value	Parameter Description
0xXXXX	Offset of the characteristic to start reading from.

Data (1-ATT_MTU_SIZE bytes)

Value	Parameter Description
0xXXXX….XXXX	Characteristic value data.

Additional Notes

See the SNP Characteristic Read Indication (0x87) event for more information on the offset parameter and partial reads.

Generated Event(s)

This command will not cause any corresponding events to be sent.

9.5.8 SNP Characteristic Write Confirmation (0x88)

If a GATT client requests a write of a characteristic that was added by the AP, the SNP GATT server will forward an SNP Characteristic Write Indication (0x88) to the AP. The AP then has 30 seconds to decide if the write is allowed based on any application-specific rules, perform the write, and respond with a SNP Characteristic Write Confirmation stating whether the write occurred successfully. If no response is received in 30 seconds, an ATT timeout will occur and no more ATT communication can happen without reestablishing the connection.

Command Parameters

Status (1 byte): status of the write request

Value	Parameter Description
0x00	Success
0xXX	AP-defined error code.

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Handle of the connection.

Additional Notes

See the SNP Characteristic Write Indication (0x88) event for more information on the offset parameter and partial reads

Generated Event(s)

This command will not cause any corresponding events to be sent.

9.5.9 SNP Send Notification Indication (0x89)

The SNP Send Notification Indication command is sent to the SNP to send a notification or indication to the GATT Client. This is only possible if a CCCD for the characteristic has been created and notifications / indications have been enabled by writing to the CCCD. Note that the final transmission of notifications and indications is managed by the GATT server. Therefore, they can only be sent to GATT clients that are requesting them. That is, notifications / indications can not be forced unless the CCCD is configured correctly.

Command Parameters

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Handle of the connection.

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Handle of the characteristic value attribute to notify / indicate.

Authenticated (1 byte)

Value	Parameter Description
0xXXXX	RFU

Request Type (1 byte)

Value	Parameter Description
0x00	Notification

0x01	Indication

Data (1-ATT_MTU_SIZE bytes)

Value	Parameter Description
0xXXXX….XXXX	Data to be notified / indicated.

Additional Notes

The maximum size possible for a notification or indication value is ATT_MTU_SIZE. If the value sent within the command is larger than ATT_MTU_SIZE bytes, it will be truncated.

Generated Event(s)

Indications require that the remote GATT client sends a confirmation so therefore only one indication can be sent at a time. Once the indication confirmation is received form the GATT client, it will be forwarded to the SNP via the SNP Send Notification Indication Response (0x89) event with the status SUCCESS. If the confirmation is not received from the GATT client within 30s, the SNP Send Notification Indication Response (0x89) event will be sent back by the SNP with the status FAILURE. In this case, an ATT timeout has occurred and no more ATT communication can occur without reestablishing the connection.

Notifications do not require that the remote GATT clients send a confirmation. Therefore, the SNP Send Notification Indication Response (0x89) event will be returned from the SNP once the notification has been forwarded successfully to the BLE host.

9.5.10 SNP CCCD Updated Confirmation (0x8B)

Depending on the desired implementation, the AP may be responsible for authorizing writes to CCCD's. When a GATT client attempts to write to a CCCD, the SNP will send a SNP CCCD Updated Indication (0x8B) event to the AP. The AP must then reply with this command to indicate whether or not it will allow the CCCD to be updated. If the update it authorized, this command should be sent. If the update is not authorized, no command should be sent.

Command Parameters

Status (1 byte): status of the CCCD update request

Value	Parameter Description
0x00	Success
0xXX	AP-defined error code.

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Handle of the connection.

Additional Notes

If desired, the AP is responsible for caching CCCD values. That is, it should remember the CCCD attribute for a specific GATT client for future connections.

Event(s) Generated

This command will not cause any corresponding events to be sent.

9.5.11 SNP Set GATT Parameter (0x8C)

The SNP Set GATT Parameter command is sent to the SNP to write the characteristic value attribute of a characteristic that is managed by the GATT server on the SNP. This includes characteristics in the Device Information Service [4] and the Generic Access Service [5]. More information about these characteristics can be found in the relevant service specification. Note that this command can only be used to write the characteristic value attribute. The SNP Set Attribute Value (0x86) command should be used for writing to other attribute types in these two services.

Command Parameters

Service ID (1 byte)

Value	Parameter Description
0x01	Generic Access Service
0x02	Device Info Service

Parameter ID (2 bytes)

Value	Device Info Service Parameter	Generic Access Service Parameter
0x0000	System ID	Device Name
0x0001	Model Number String	Appearance
0x0002	Serial Number String	Peripheral Privacy Flag (RFU)
0x0003	Firmware Revision String	Reconnection Address (RFU)
0x0004	Hardware Revision String	Peripheral Preferred Connection Parameters
0x0005	Software Revision String	Peripheral Privacy Flag Properties (RFU)
0x0006	Manufacturer Name String	Device Name Permissions (RFU)
0x0007	IEEE 11073-20601 Regulatory Certification Data List	Appearance Permissions (RFU)
0x0008	PnP ID	Peripheral Privacy Flag Permissions (RFU)

Value (n bytes)

Value	Parameter Description
0xXXXX….XXXX	Value to update parameter with.

Event(s) generated

The SNP Set GATT Parameter (0x8C) event will be returned from the SNP.

9.5.12 SNP Get GATT Parameter (0x8D)

The SNP Get GATT Parameter command is sent to the SNP to read the characteristic value attribute of a characteristic that is managed by the GATT server on the SNP. This includes characteristics in the Device Information Service [4] and the Generic Access Service [5]. More information about these characteristics can be found in the relevant service specification. Note that this command can only be used to read the characteristic value attribute. The SNP Get GATT Parameter (0x8D) command should be used for reading other attribute types in these two services.

Command Parameters

Service ID (1 byte)

Value	Parameter Description
0x01	Generic Access Service
0x02	Device Info Service

Parameter ID (2 bytes)

Value	Device Info Service Parameter	Generic Access Service Parameter
0x0000	System ID	Device Name
0x0001	Model Number String	Appearance
0x0002	Serial Number String	Peripheral Privacy Flag (RFU)

0x0003	Firmware Revision String	Reconnection Address (RFU)
0x0004	Hardware Revision String	Peripheral Preferred Connection Parameters
0x0005	Software Revision String	Peripheral Privacy Flag Properties (RFU)
0x0006	Manufacturer Name String	Device Name Permissions (RFU)
0x0007	IEEE 11073-20601 Regulatory Certification Data List	Appearance Permissions (RFU)
0x0008	PnP ID	Peripheral Privacy Flag Permissions (RFU)

Value (0-512 bytes)

Value	Parameter Description
0xXXXX….XXXX	Parameter value.

Event(s) generated

The SNP Get GAP Parameter (0x49) event will be returned from the SNP.

9.6 GATT Subgroup Events

9.6.1 SNP Add Service Response (0x81)

The SNP will return the SNP Add Service Response event to the AP after it has performed the SNP Add Service (0x81) command. The status of this event should be checked to verify that the service was added correctly.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

9.6.2 SNP Add Characteristic Value Declaration Response (0x82)

The SNP will return the SNP Add Characteristic Value Declaration Response event to the AP after it has performed the SNP Add Characteristic Value Declaration (0x82) command. The status of this event should be checked to verify that the characteristic was added successfully and, if needed, to store the handle the characteristic value was added at for future use.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Attribute handle of the characteristic value.

9.6.3 SNP Add Characteristic Descriptor Declaration Response (0x83)

The SNP will return the SNP Add Characteristic Descriptor Declaration Response event to the AP after it has performed the SNP Add Characteristic Descriptor Declaration (0x83) command. The status of this event should be checked to verify that the attribute was added successfully and, if needed, to store the handle the attribute was added at for future use.

The first parameter of the event is a header indicating which attributes were requested to be added. This is the same structure as the SNP Add Characteristic Descriptor Declaration command where multiple bits can be set and each bit corresponds to an attribute.
The status parameter should be checked to verify that the attributes were successfully added. There is only one status parameter per event. There are, however, as many attribute handle parameters as there were attributes added and the attribute handles will be returned in ascending order of the parameter bit values.
For example, if both the CCCD and user description attributes were to be added, the header will have the value 0x84. The one byte status will come next followed by two bytes for the attribute handle of the CCCD and then two bytes for the attribute handle of the user description.

Event Parameters

Header (1 byte): which attributes will be added to the service

Value	Parameter Description
0x01	Generic-Short UUID
0x02	Generic-Long UUID
0x04	CCCD
0x08	Presentation Format
0x80	User Description String

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Handles (2-12 bytes): handles of attributes added. There will be 1-6 of these: one for each attribute added.

Value	Parameter Description
0xXXXX	Handle of first attribute added.
…..	……
0xXXXX	Handle of last attribute added.

9.6.4 SNP Register Service Response (0x84)

The SNP will return the SNP Register Service Response event to the AP after it has performed the SNP Register Service (0x84) command. The status of this event should be checked to verify that the service was registered successfully and, if needed, to store the handles that service was registered at.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Service Start Handle (2 bytes)

Value	Parameter Description
0xXXXX	First attribute handle of the registered service.

Service End Handle (2 bytes)

Value	Parameter Description
0xXXXX	Last attribute handle of the registered service.

9.6.5 SNP Get Attribute Value Response (0x85)

The SNP will return the SNP Get Attribute Value Response event to the AP after it has performed the SNP Get Attribute Value (0x85) command. The status of this event should be checked to verify that the attribute value was successfully read.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Attribute handle of the characteristic value.

Data (n bytes)

Value	Parameter Description
0xXXXX….XXXX	Data stored in the characteristic value.

9.6.6 SNP Set Attribute Value Response (0x86)

The SNP will return the SNP Set Attribute Value Response event to the AP after it has performed the SNP Set Attribute Value (0x86) command. The status of this event should be checked to verify that the attribute value was successfully set.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Attribute handle of the characteristic value.

9.6.7 SNP Characteristic Read Indication (0x87)

The SNP Characteristic Read Indication event is sent from the SNP when the remote GATT client requests a read of a characteristic value that is managed by the AP. Upon receiving this event, the AP must send an SNP Characteristic Read Confirmation (0x87) within 30 seconds.

Event Parameters

Connection Handle (2bytes)

Value	Parameter Description

0xXXXX	Handle of the connection requesting the read.

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Attribute handle of the characteristic value being read.

Offset (2 bytes)

Value	Parameter Description
0xXXXX	Offset into characteristic value to start reading from

Max Size (2bytes)

Value	Parameter Description
0xXXXX	Maximum amount of bytes that can be read at once.

Additional Notes

A characteristic value can be up to 512 Bytes long. However, not more than ATT_MTU_SIZE bytes can be read at once by an ATT operation. If the characteristic size is bigger than ATT_MTU_SIZE, the remote GATT client will send several ATT_MTU_SIZE reads until it reads the entire characteristic. Therefore, each read request send by the GATT client is translated into a SNP Characteristic Read Indication with two parameters: an offset and the maximum data size. The offset represents the start of the data to read in the characteristic value and the maximum data size represents the maximum amount of data (in bytes) that can be sent in one SNP Characteristic Read Confirmation. The AP needs to reply with the SNP Characteristic Read Confirmation (0x87) and, in this command, indicate the offset the data was read from and the size of the data.

9.6.8 SNP Characteristic Write Indication (0x88)

The SNP Characteristic Write Indication event is sent from the SNP when the remote GATT client requests a write of a characteristic value that is managed by the AP. Upon receiving this event and if the Response Needed parameter is set to 0x01, the AP must send an SNP Characteristic Write Confirmation (0x88) within 30 seconds. If the Response Needed parameter is set to 0x00, no Characteristic Write Confirmation is needed. This will occur in the case of an ATT Write Command instead of an ATT Write Request.

Event Parameters

Connection Handle (2bytes)

Value	Parameter Description
0xXXXX	Handle of the connection requesting the write.

Attribute Handle (2 bytes)

Value	Parameter Description
0xXXXX	Attribute handle of the characteristic value being written to.

Response Needed (1 byte)

Value	Parameter Description
0x00	No SNP Characteristic Write Confirmation response needed.
0x01	A SNP Characteristic Write Confirmation response is needed.

Offset (2 bytes)

Value	Parameter Description
0xXXXX	Offset into characteristic value to start writing at.

Data (0-bytes)

Value	Parameter Description
0xXXXX….XXXX	Data to write to the characteristic value.

Additional Notes

A characteristic value can be up to 512 Bytes long. However, not more than ATT_MTU_SIZE bytes can be written at once by an ATT operation. If the characteristic size is bigger than ATT_MTU_SIZE, the remote GATT client will send several ATT_MTU_SIZE writes until it has written the entire characteristic. The Offset parameter represents the start of the characteristic value data to write. The data size can be deduced from the NPI frame length of the SNP Characteristic Write Indication.

Once a remote GATT client starts to write a characteristic value with offset 0, this means it will write the entire value (see the BT Spec [1] 4.9.4, Part G, Vol3). If the remote GATT Client starts to write with an offset different than 0, this means it is a partial write.

9.6.9 SNP Send Notification Indication Response (0x89)

The SNP Send Notification Indication Response event is sent from the SNP after the SNP Send Notification Indication (0x89) command has been performed. If an indication was sent, this event means that the GATT client received the indication and has responded with a confirmation. If a notification was sent, this event means that the notification was successfully queued up for transmission in the BLE stack. In either case, the status parameter should be checked to see that the indication / notification has completed successfully.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

Connection Handle (2 bytes)

Value	Parameter Description
0xXXXX	Connection handle of the characteristic value.

9.6.10 SNP CCCD Updated Indication (0x8B)

The SNP CCCD Updated Indication event is sent from the SNP when the remote GATT Client has requested to update the CCCD value of an attribute managed by the AP. Upon receiving this event, and if the Response Needed parameter is set to 0x01, the AP must send an SNP CCCD Updated Confirmation (0x8B) within 30 seconds. If the Response Needed parameter is set to 0x00, no CCCD Updated Confirmation is needed. This last case can occur if an ATT_WRITE_CMD operation has been used by the remote GATT client.

This indication / confirmation scheme allows the AP to add a layer of authorization to modifying CCCD's and allowing indications / notifications. That is, the AP is free to allow / deny CCCD updates based on the desired implementation.

Event Parameters

Connection Handle (2bytes)

Value	Parameter Description
0xXXXX	Handle of the connection requesting the CCCD update.

CCCD Handle (2 bytes)

Value	Parameter Description

0xXXXX	Handle of the characteristic value attribute to notify / indicate.

Response Needed (1 byte)

Value	Parameter Description
0x00	No SNP CCCD Updated Confirmation needed.
0x01	A SNP CCCD Updated Confirmation is needed.

Value (2 bytes)

Value	Parameter Description
0xXXXX	Value to write to the CCCD.

9.6.11 SNP Set GATT Parameter Response (0x8C)

The SNP will return the SNP Set GATT Parameter Response event to the AP after it has performed the SNP Set GATT Parameter (0x8C) command. The status of this event should be checked to verify that the parameter was successfully set.

Event Parameters

Status (1 byte): status of the request

Value	Parameter Description
0x00	Success
0xXX	See SNP Error Codes

9.6.12 SNP Get GATT Parameter Response (0x8D)

The SNP will return the SNP Get GATT Parameter Response event to the AP after it has performed the SNP Get GATT Parameter (0x8D) command. The status of this event should be checked to verify that the parameter was successfully read and to see what the value is.

Event Parameters

Service ID (1 byte)

Value	Parameter Description
0x01	Generic Access Service
0x02	Device Info Service

Parameter ID (2 bytes)

Value	Device Info Service Parameter	Generic Access Service Parameter
0x0000	System ID	Device Name
0x0001	Model Number String	Appearance
0x0002	Serial Number String	Peripheral Privacy Flag (RFU)
0x0003	Firmware Revision String	Reconnection Address (RFU)
0x0004	Hardware Revision String	Peripheral Preferred Connection Parameters
0x0005	Software Revision String	Peripheral Privacy Flag Properties (RFU)
0x0006	Manufacturer Name String	Device Name Permissions (RFU)
0x0007	IEEE 11073-20601 Regulatory Certification Data List	Appearance Permissions (RFU)
0x0008	PnP ID	Peripheral Privacy Flag Permissions

		(RFU)

Value (n bytes)

Value	Parameter Description
0xXXXX….XXXX	Value to update parameter with.