1. Revision History

Date	Version	Description
2016-07-04	1.0	BLE command user guide release

2. Function of summary

2.1 Central function

2.1.1 NFBT_GATT_Scan

Function	**bt_status NFBT_GATT_Scan(nfUINT8 start)**	
Parameter	start	0x00: scanning stop 0x01: scanning start
Return	See bt_status	
Callback	**NFCBK_GATT_ScanResult**	
Description	It is used to start/stop scanning. Scanning is used to discover advertising devices nearby.	
Notice	Disconnect all connection before use.	

2.1.2 NFBT_GATT_SearchService

Function	**bt_status NFBT_GATT_SearchService(void)**
Parameter	n/a
Return	See bt_status
Callback	**NFCBK_GATT_SearchResult,** **NFCBK_GATT_SearchComplete[shampin, deprecate]**
Description	Search peripheral device which service supported. A peripheral may support several service that is a collection of data and associated behaviors to accomplish a particular function or feature of a device or portions of a device. If service search UUID result is 0, the search process is complete.
Notice	Connection established before use.

2.1.3 NFBT_GATT_GetCharacteristic

Function	**bt_status NFBT_GATT_GetCharacteristic(char *srvc_uuid)**	
Parameter	srvc_uuid	Pointer of which service UUID provided by Server.
Return	See bt_status	
Callback	**NFCBK_GATT_GetCharacteristicResult**	
Description	A service may include several Characteristic. A characteristic is a value used in a service along with properties and configuration information about how the value is accessed and information about how the value is displayed or represented. A characteristic definition contains a characteristic declaration, characteristic properties, and a value. It may also contain descriptors that describe the value or permit configuration of the server with respect to the characteristic value. The callback will return each Characteristic’s UUID and reference Handle ID for others operate control.	
Notice	Connection established before use.	

2.1.4 NFBT_GATT_GetDescriptor

Function	**bt_status NFBT_GATT_GetDescriptor(char *srvc_uuid, char *char_uuid)**	
Parameter	*srvc_uuid	Pointer of which service UUID provided by Server
	*char_uuid	Pointer of which characteristic UUID include in service UUID
Return	See bt_status	
Callback	**NFCBK_GATT_GetDescriptorResult**	
Description	A Characteristic may or not include Descriptor. A Descriptor describe the value or permit configuration of the server with respect to the characteristic value. The callback will return each Descriptor’s UUID and reference Handle ID for read/write control.	
Notice	Connection established before use.	

2.1.5 NFBT_GATT_NotificationRegister

Function	**bt_status NFBT_GATT_NotificationRegister(char * srvc_uuid, char * char_uuid, int registy)**	
Parameter	srvc_uuid	Pointer of which service UUID provided by Server
	char_uuid	Pointer of which characteristic UUID include in service UUID
	register	REGISTER(0x01) or UN-REGISTER(0x00)
Return	See bt_status	
Callback	**NFCBK_GATT_RegisterNotify**	
Description	It is used to notify a client of the value of a Characteristic Value from a server. Refer ence for BLE service specification which Notifications can be configured and using WriteDescriptor() to config the Client Characteristic Configuration descriptor corres pond with descriptor of service. If BLE connection disconnect, the Notification should be register again in next connection established.	
Notice	Connection established before use.	

2.1.6 NFBT_GATT_ReadCharacteristic

Function	**bt_status NFBT_GATT_ReadCharacteristic(char *srvc_uuid, char *char_uuid)**	
Parameter	srvc_uuid	Pointer of which service UUID provided by Server
	char_uuid	Pointer of which characteristic UUID include in service UUID
Return	See bt_status	
Callback	**NFCBK_GATT_ReadCharacteristicResult**	
Description	This is used to read a value of Characteristic from server. Reference for BLE service specification which Characteristic could be read.	
Notice	Connection established before use.	

2.1.7 NFBT_GATT_WriteCharacteristic

Function	**bt_status NFBT_GATT_WriteCharacteristic(char *srvc_uuid, char *char_uuid, char *descr_data, int data_len)**	
Parameter	srvc_uuid	Pointer of which service UUID provided by Server
	char_uuid	Pointer of which characteristic UUID include in service UUID
	descr_data	Pointer of which descriptor UUID include in characteristic UUID
	data_len	Characteristic data length, maximum is 273 bytes
Return	See bt_status	
Callback	**NFCBK_GATT_WriteCharacteristicResult**	
Description	This is used to write a value of Characteristic to server. Reference for BLE service specification which Characteristic could be write.	
Notice	Connection established before use.	

2.1.8 NFBT_GATT_ReadDescriptor

Function	**bt_status NFBT_GATT_ReadDescriptor(char *srvc_uuid, char *char_uuid, char *descr_uuid)**	
Parameter	srvc_uuid	Pointer of which service UUID provided by Server
	char_uuid	Pointer of which characteristic UUID include in service UUID
	descr_uuid	Pointer of which descriptor UUID include in characteristic UUID
Return	See bt_status	
Callback	**NFCBK_GATT_ReadDescriptorResult**	
Description	This is used to read a value of Descriptor from server. Reference for BLE service specification which Descriptor could be read.	
Notice	Connection established before use.	

2.1.9 NFBT_GATT_WriteDescriptor

Function	**bt_status NFBT_GATT_WriteDescriptor(char *srvc_uuid, char *char_uuid, char *descr_uuid, char *descr_data, int data_len)**	
Parameter	srvc_uuid	Pointer of which service UUID provided by Server.
	char_uuid	Pointer of which characteristic UUID include in service UUID.
	descr_uuid	Pointer of which descriptor UUID include in characteristic UUID.
	descr_data	Pointer of data that write to descriptor UUID.
	data_len	Descriptor data length, maximum is 253 bytes
Return	See bt_status	
Callback	**NFCBK_GATT_WriteDescriptorResult**	
Description	This is used to write a value of Descriptor to server. Reference for BLE service specification which Descriptor could be write.	
Notice	Connection established before use.	

2.2 Peripheral function

2.2.1 NFBT_GATT_Listen

Function	**bt_status NFBT_GATT_Listen(nfUINT8 start)**	
Parameter	start	0x00 : listening stop. 0x01 : listening start.
Return	See bt_status	
Callback	**n/a**	
Description	Start Listen the server device will advertising registered service to air. Client received advertising packet will be connect if it knows some information in packet. In listen state, server status will be change to GATT_CONN_LISTENING(0x02). If start fail, the status not change usually is GATT_CONN_DISCONNECT(0x00).	
Notice	Disconnect all connection before use.	

2.2.2 NFBT_GATT_ServiceAdd

Function	**bt_status NFBT_GATT_ServiceAdd(nfUINT16 *srvc_uuid)**	
Parameter	srvc_uuid	16-byte length. Pointer of which BLE service want to add. BLE service UUID reference to Bluetooth SIG website.
Return	See bt_status	
Callback	**NFCBK_GATT_ServiceAddedResult**	
Description	Add a new service into list before start advertising. Reference to BLE service specification to create service and also allow customer to create own's service.	
Notice	Disconnect all connection before use and service stop	

2.2.3 NFBT_GATT_CharacteristicAdd

Function	**bt_status NFBT_GATT_CharacteristicAdd(int srvc_hdl, char *char_uuid, nfUINT16 property, nfUINT16 permit)**	
Parameter	srvc_hdl	Handle of service that got from callback of service registered.
	char_uuid	Characteristic UUID which want to add to specific service
	property	Determines how the Characteristic Value can be used. possible bit field reference nfore_BlueGate_local.h
	permit	Allow to READ or WRITE. possible bit field reference nfore_BlueGate_local.h
Return	See bt_status	
Callback	**NFCBK_GATT_CharAddedResult**	
Description	Add a new Characteristic into service. Reference to BLE service specification to create Characteristic or customer own’s Characteristic. The Characteristic should create before Service created.	
Notice	Disconnect all connection before use and service stop	

2.2.4 NFBT_GATT_DescriptorAdd

Function	**bt_status NFBT_GATT_DescriptorAdd(int srvc_hdl, char *descr_uuid, nfUINT16 permit)**	
Parameter	srvc_hdl	Handle of service that got from callback of service registered.
	descr_uuid	Characteristic UUID which want to add to specific service
	permit	Allow to READ or WRITE. Possible bit field reference nfore_BlueGate_local.h
Return	See bt_status	
Callback	**NFCBK_GATT_DescriptorAddedResult**	
Description	Add a new Descriptor into Characteristic. This is optional to add. Reference to BLE service specification to create Descriptor or customer own’s Descriptor. The Descriptor should create before Characteristic created.	
Notice	Disconnect all connection before use and service stop	

2.2.5 NFBT_GATT_StartService

Function	**bt_status NFBT_GATT_StartService(int start, int srvc_hdl)**	
Parameter	start	start(0x01)/stop(0x00) peripheral service.
	srvc_hdl	service handle that get from callback of **NFCBK_GATT_ServiceAddedResult**
Return	See bt_status	
Callback	**NFCBK_GATT_ServiceResult**	
Description	Start/Stop a service that the service created and configuration finished.	
Notice	Disconnect all connection before use and service stop	

2.2.6 NFBT_GATT_DeleteService

Function	**bt_status NFBT_GATT_DeleteService(int srvc_hdl)**	
Parameter	srvc_hdl	Handle of service
Return	See bt_status	
Callback	**NFCBK_GATT_ServiceResult**	
Description	Delete a service. A service not to use any more or re-config Characteristic or Descriptor.	
Notice	Disconnect all connection before use and service stop	

2.2.7 NFBT_GATT_SendIndication

Function	**bt_status NFBT_GATT_SendIndication(int attr_hdl, char *data, int data_len, int notification)**	
Parameter	attr_hdl	Attribute(characteristic) handle
	data	Pointer of data that send data to remote device for indication.
	data_len	Length of data. Maximum is 253 bytes.
	notification	0x01 is notification, 0x00 is indication. The value must reference local characteristic property.
Return	See bt_status	
Callback	**NFCBK_GATT_IndicatorSentResult**	
Description	Client use WriteDescriptor to set Client Characteristic Configuration descriptor. If bit of Notification is set, the parameter of notification should use 0x01. If bit of Indicate is set, the parameter of notification should use 0x00. Reference of BLE service for which bit supported by service. It depends on Service’s Characteristic whether support Notification or Indicate.	
Notice	Connection established before use.	

2.2.8 NFBT_GATT_SendResponse

Function	**bt_status NFBT_GATT_SendResponse(int trans_id, GATT_STATUS status, int attr_hdl, int attr_hdl_rsp, char *data, int data_len)**	
Parameter	trans_id	Include in last packet sent by remote device.
	status	See GATT_STATUS
	attr_hdl	Which handle of Descriptor or Characteristic
	attr_hdl_rsp	Which handle of Descriptor or Characteristic
	data	Response data. Defined by customer.
	data_len	Length of data. Maximum is 253 bytes.
Return	See bt_status	
Callback	**NFCBK_GATT_ResponseConfirmResult**	
Description	Client read/write Characteristic or Descriptor from/to service should send a response to Client.	
Notice	Connection established before use.	

2.2.9 NFBT_GATT_PeripheralListService

Function	**bt_status NFBT_GATT_PeripheralListService(void)**	
Parameter	n/a	
Return	See bt_status	
Callback	**NFCBK_GATT_PeripheralServiceListResult,** **NFCBK_GATT_PeripheralListComplete**	
Description	List all added Service to show UUID and Handle ID for control or delete service.	
Notice	Any status	

2.2.10 NFBT_GATT_PeripheralListAttribute

Function	**bt_status NFBT_GATT_PeripheralListAttribute(int srvc_hdl)**	
Parameter	srvc_hdl	service handle that include Characteristic/Descriptor list
Return	See bt_status	
Callback	**NFCBK_GATT_PeripheralAttrListResult,** **NFCBK_GATT_PeripheralListComplete**	
Description	List all added Characteristic or Descriptor in Service to show UUID and Handle ID for control.	
Notice	Any status	

2.2.11 NFBT_GATT_SetAdvertisingData

Function	**bt_status NFBT_GATT_SetAdvertisingData(int show_name, int show_txpower, char *srvc_uuid, int srvc_uuid_num, int speed)**	
Parameter	show_name	Adv. data include local device name. Data length is Header[2] + data[N]. N range is 1 to 26. （坑。0：不广播设备名；1：广播设备名）
	show_txpower	Adv. data include TX power level. Data length is Header[2] + data [1]
	*srvc_uuid	128-bit length per each service uuid. Data length is Header[2] + data[16*srvc_uuid_num].The Bluetooth base UUID is 0000xxxx00001000800000805f9b34fb. If srvc_uuid compare with base UUID only different part as xxxx. Only xxxx of UUID will be advertising. Typically, data length is 18 bytes,in this case, data length only 4 bytes.
	srvc_uuid_num	How many UUID list in advertising data.
	speed	Advertising speed. GATT_ADV_SLOW(1s), GATT_ADV_NORMAL(100ms), GATT_ADV_FAST(30ms). NOTE. Advertising data total length is 31 bytes. Because 3-byte is fixed for FLAG element, only remain 28 bytes can be use. Display data exceed limitation length will be ignore.
Return	See bt_status	
Callback	**NFCBK_GATT_SetAdvDataResult**	
Description	n/a	
Notice	Any status	

2.3 Peripheral and Central common function

2.3.1 NFBT_GATT_GetRssi

Function	**bt_status NFBT_GATT_GetRssi(void)**
Parameter	n/a
Return	See bt_status
Callback	**NFCBK_GATT_ReadRssi**
Description	Get connection's RF channel quality and signal power.
Notice	Connection established before use.

2.3.2 NFBT_GATT_ SetRole

Function	**bt_status NFBT_GATT_SetRole(GATT_ROLE role)**	
Parameter	role	See GATT_ROLE
Return	See bt_status	
Callback	**NFCBK_GATT_SetRoleResult**	
Description	Set the device is Role of Central or Peripheral.	
Notice	Disconnect all connection before use	

3. Function of callback

3.1 Central callback function

3.1.1 NFCBK_GATT_ScanResult

Function	**typedef void(*NFCBK_GATT_ScanResult)(BT_ADDR *addr, int rssi, nfUINT8 *adv_data)**	
Parameter	addr	Pointer of address of advertising packet device.
	rssi	Signal power of advertising device.
	adv_data	Pointer of advertising data packet.
OP function	**NFBT_GATT_Scan**	
Description	Scan advertising packet to get which device in nearly area. Advertising data packet i nclude several simple information of device. ex. name, kind of service, RSSI etc. All p acket size is not exceeds 31 bytes. See document of “Supplement to Bluetooth core Specification|CSSv6” from SIG website to get detail description.	

3.1.2 NFCBK_GATT_SearchResult

Function	**typedef void(*NFCBK_GATT_SearchResult) (char* srvc_uuid, nfUINT8 is_primary)**	
Parameter	srvc_uuid	Pointer of service UUID. To search what kind of ability supported by Server.
	Is_primary	There are two types of services: primary and secondary. A primary s ervice is a service that provides the primary functionality of a device. A secondary service is a service that provides auxiliary functionality of a device and is referenced from at least one primary service on th e device.
OP function	**NFBT_GATT_SearchService**	
Description	Callback of search all service on sever for identify which kind of device. A service is a collection of data and associated behaviors to accomplish a particular function or f eature of a device or portions of a device. A service may reference other primary or secondary services and/or a set of characteristics that make up the service.	

3.1.3 NFCBK_GATT_SearchComplete[shampin, deprecate]

Function	**typedef void(*NFCBK_GATT_SearchComplete) (GATT_STATUS status)**	
Parameter	state	See GATT_STATUS
OP function	**NFBT_GATT_SearchService**	
Description	Callback of all service listed, it will informed by system.	

3.1.4 NFCBK_GATT_GetCharacteristicResult

Function	**typedef void(*NFCBK_GATT_GetCharacteristicResult)(GATT_STATUS status, char *srvc_uuid, char *char_uuid, int char_prop)**	
Parameter	status	See GATT_STATUS
	srvc_uuid	Pointer of service UUID for operate.
	char_uuid	Pointer of characteristic UUID for operate.
	char_prop	Characteristic property is determined whether allow access by server.
OP function	**NFBT_GATT_GetCharacteristic**	
Description	Callback of get all characteristic list in service.	

3.1.5 NFCBK_GATT_GetDescriptorResult

Function	**typedef void(*NFCBK_GATT_GetDescriptorResult)(GATT_STATUS status, char *srvc_uuid, char *char_uuid, char *descr_uuid)**	
Parameter	status	See GATT_STATUS
	srvc_uuid	Pointer of service UUID for operate.
	char_uuid	Pointer of characteristic UUID for operate.
	descr_uuid	Pointer of descriptor UUID for operate.
OP function	**NFBT_GATT_GetDescriptor**	
Description	Callback of get specific Descriptor's value. It depends on Descriptor whether provided to access.	

3.1.6 NFCBK_GATT_ReadCharacteristicResult

Function	**typedef void(*NFCBK_GATT_ReadCharacteristicResult)(GATT_STATUS status, char *srvc_uuid, char *char_uuid, int data_type, int data_len, char *data)**	
Parameter	status	See GATT_STATUS
	srvc_uuid	Pointer of service UUID for operate.
	char_uuid	Pointer of characteristic UUID for operate.
	data_type	See Appendix B.
	data_len	Length of data.
	data	Pointer of data.
OP function	**NFBT_GATT_ReadCharacteristic**	
Description	Callback of get specific Characteristic's value. It depends on Characteristic whether provided access and depends on which kind of server to provide what kind of service.	

3.1.7 NFCBK_GATT_WriteCharacteristicResult

Function	**typedef void(*NFCBK_GATT_WriteCharacteristicResult)(GATT_STATUS status, nfUINT8 *srvc_uuid, nfUINT8 *char_uuid)**	
Parameter	status	See GATT_STATUS
	srvc_uuid	Pointer of service UUID for operate.
	char_uuid	Pointer of characteristic UUID for operate.
OP function	**NFBT_GATT_WriteCharacteristic**	
Description	Callback of Write a Characteristic's value to server. It depends on Characteristic whe ther provided access and depends on which kind of server to provide what kind of service.	

3.1.8 NFCBK_GATT_ReadDescriptorResult

Function	**typedef void(*NFCBK_GATT_ReadDescriptorResult)(GATT_STATUS status, nfUINT8 *srvc_uuid, nfUINT8 *char_uuid, nfUINT8 *descr_uuid, int data_type, int data_len, nfUINT8 *data)**	
Parameter	status	See GATT_STATUS
	srvc_uuid	Pointer service UUID for operate.
	char_uuid	Pointer characteristic UUID for operate.
	descr_uuid	Pointer descriptor UUID for operate.
	data_type	See Appendix B.
	data_len	Length of data.
	data	Pointer of data.
OP function	**NFBT_GATT_ReadDescriptor**	
Description	Callback of get specific Descriptor's value. It depends on Descriptor whether provided to access and depends on which kind of server to provide what kind of service.	

3.1.9 NFCBK_GATT_WriteDescriptorResult

Function	**typedef void(*NFCBK_GATT_WriteDescriptorResult)(GATT_STATUS status, nfUINT8 *srvc_uuid, nfUINT8 *char_uuid, nfUINT8 *descr_uuid)**	
Parameter	status	See GATT_STATUS
	*srvc_uuid	Which service UUID for operate.
	*char_uuid	Which characteristic UUID for operate.
	*descr_uuid	Which descriptor UUID for operate.
OP function	**NFBT_GATT_WriteDescriptor**	
Description	Callback of write a value to specific Descriptor. It depends on Descriptor whether provided access and depends on which kind of server provided service.	

3.1.10 NFCBK_GATT_RegisterNotify

Function	**typedef void(*NFCBK_GATT_RegisterNotify)(GATT_STATUS status, int registered)**	
Parameter	status	See GATT_STATUS
	registered	Characteristic register or un-register status.
OP function	**NFBT_GATT_NotificationRegister**	
Description	Callback of specific Characteristic to register or un-register	

3.1.11 NFCBK_GATT_Notify

Function	**typedef void(*NFCBK_GATT_Notify)(BT_ADDR *addr, nfUINT8 *srvc_uuid, nfUINT8 *char_uuid, int data_len, nfUINT8 *data)**	
Parameter	addr	Address of connected device.
	srvc_uuid	Pointer of service UUID for operate.
	char_uuid	Pointer of characteristic UUID for operate
	data_len	Length of data.
	data	Pointer of data.
OP function	**n/a**	
Description	Callback of registered Characteristic value retrieve from server.	

3.2 Peripheral callback function

3.2.1 NFCBK_GATT_ServiceAddedResult

Function	**typedef void (*NFCBK_GATT_ServiceAddedResult)(GATT_STATUS status, nfUINT8 *srvc_uuid, int srvc_handle)**	
Parameter	status	See GATT_STATUS
	srcv_uuid	Pointer of service UUID for operate.
	srvc_handle	Handle of service UUID.
OP function	**NFBT_GATT_ServiceAdd**	
Description	Callback of service added.	

3.2.2 NFCBK_GATT_CharAddedResult

Function	**typedef void (*NFCBK_GATT_CharAddedResult)(GATT_STATUS status, nfUINT8 *char_uuid, int srvc_handle, int char_handle)**	
Parameter	status	See GATT_STATUS
	char_uuid	Pointer of characteristic UUID for operate.
	srvc_handle	Handle of service. Characteristic added to which service handle.
	char_handle	Handle of characteristic.
OP function	**NFBT_GATT_CharacteristicAdd**	
Description	Callback of Characteristic added.	

3.2.3 NFCBK_GATT_DescriptorAddedResult

Function	**typedef void (*NFCBK_GATT_DescriptorAddedResult)(GATT_STATUS status, nfUINT8 *descr_uuid, int srvc_handle, int descr_handle)**	
Parameter	status	See GATT_STATUS.
	descr_uuid	Pointer of descriptor UUID for operate.
	srvc_handle	Handle of service. Descriptor added to which service handle
	descr_handle	Handle of descriptor.
OP function	**NFBT_GATT_DescriptorAdd**	
Description	Callback of Descriptor added.	

3.2.4 NFCBK_GATT_ServiceStartResult

Function	**typedef void (*NFCBK_GATT_ServiceStartResult)(GATT_STATUS status, int srvc_handle)**	
Parameter	status	See GATT_STATUS
	srvc_handle	Handle of service start.
OP function	**NFBT_GATT_StartService**	
Description	Callback of start a new service status	

3.2.5 NFCBK_GATT_ServiceStopResult

Function	**typedef void (*NFCBK_GATT_ServiceStopResult)(GATT_STATUS status, int srvc_handle)**	
Parameter	status	See GATT_STATUS
	srvc_handle	Handle of service stop
OP function	**NFBT_GATT_StartService**	
Description	Callback of stop a started service status.	

3.2.6 NFCBK_GATT_ServiceDeleteResult

Function	**typedef void (*NFCBK_GATT_ServiceDeleteResult)(GATT_STATUS status, int srvc_handle)**	
Parameter	status	See GATT_STATUS.
	srvc_handle	Which service handle for operate.
OP function	**NFBT_GATT_DeleteService**	
Description	Callback of a service to delete status.	

3.2.7 NFCBK_GATT_RequestReadEvent

Function	**typedef void (*NFCBK_GATT_RequestReadEvent)(BT_ADDR *addr, int trans_id, int attr_handle, int offset, int is_long)**	
Parameter	addr	Pointer of Client address to identify which one request to read.
	trans_id	Transaction ID. Only use for response.
	attr_handle	Which attribute handle for process read data from buffer.
	offset	Read data from offset buffer.
	is_long	TBD
OP function	**n/a**	
Description	Callback of client request to read which characteristic/descriptor value.	

3.2.8 NFCBK_GATT_RequestWriteEvent

Function	**typedef void (*NFCBK_GATT_RequestWriteEvent)(BT_ADDR *addr, int trans_id, int attr_handle, int offset, int is_prep, nfUINT8 *data, int length, int need_rsp)**	
Parameter	addr	Pointer of Client address to identify which one request to write.
	trans_id	Transaction ID. Only use for response if necessary.
	attr_handle	Which attribute handle process write data to buffer.
	offset	Write data to offset of buffer.
	is_prep	Prepare to write.
	data	Pointer of data.
	length	Length of data.
	need_rsp	If response requested by client, use NFBT_GATT_SendResponse to send response
OP function	**n/a**	
Description	Client device request to write a value in which characteristic/descriptor.	

3.2.9 NFCBK_GATT_ResponseConfirmResult

Function	**typedef void (*NFCBK_GATT_ResponseConfirmResult)(GATT_STATUS status)**	
Parameter	status	See GATT_STATUS
OP function	**NFBT_GATT_SendResponse**	
Description	Callback of client got a response from server.	

3.2.10 NFCBK_GATT_IndicatorSentResult

Function	**typedef void (*NFCBK_GATT_IndicatorSentResult)(GATT_STATUS status)**	
Parameter	status	See GATT_STATUS.
OP function	**NFBT_GATT_SendIndication**	
Description	Callback of client got a “notification” and sent a response back to server. This callback only occur when NFBT_GATT_SendIndication with parameter of notification set to 1.	

3.2.11 NFCBK_GATT_CongestionResult

Function	**typedef void (*NFCBK_GATT_CongestionResult)(int congested)**	
Parameter	congested	0x00: no congested. 0x01: system congested.
OP function	**n/a**	
Description	Callback of when use NFBT_GATT_SendIndication with parameter of notification set to 0. If data transmission too fast, congestion will be informed by system. Upper layer UI should suspend and waiting for system congestion release then resume for avoid data drop by system.	

3.2.12 NFCBK_GATT_PeripheralServiceListResult

Function	**typedef void (*NFCBK_GATT_PeripheralServiceListResult)(int handle, char *uuid)**	
Parameter	handle	Handle ID correspond with uuid created by system.
	uuid	Pointer of which UUID added in service list.
OP function	**NFBT_GATT_PeripheralListService**	
Description	Callback of list all added service.	

3.2.13 NFCBK_GATT_PeripheralAttributeListResult

Function	**typedef void (*NFCBK_GATT_PeripheralAttributeListResult)(int handle, nfUINT8 *uuid, int property, int type)**	
Parameter	handle	Handle ID correspond with uuid created by system.
	uuid	Pointer of UUID added in attribute list.
	property	Attribute property definition by CharacteristicAdd or DescriptorAdd.
	type	The attribute type is Characteristic or Descriptor.
OP function	**NFBT_GATT_PeripheralListAttribute**	
Description	Callback of list all added Characteristic and Descriptor.	

3.3 Peripheral and Central common callback function

3.3.1 NFCBK_GATT_SetRole

Function	**typedef void (*NFCBK_GATT_SetRole)(GATT_STATUS status, GATT_ROLE role)**	
Parameter	status	See GATT_STATUS.
	role	See GATT_ROLE.
OP function	**NFBT_GATT_SetRole**	
Description	Callback of set role to Central or Peripheral whether success.	

3.3.2 NFCBK_GATT_ConnectStatus

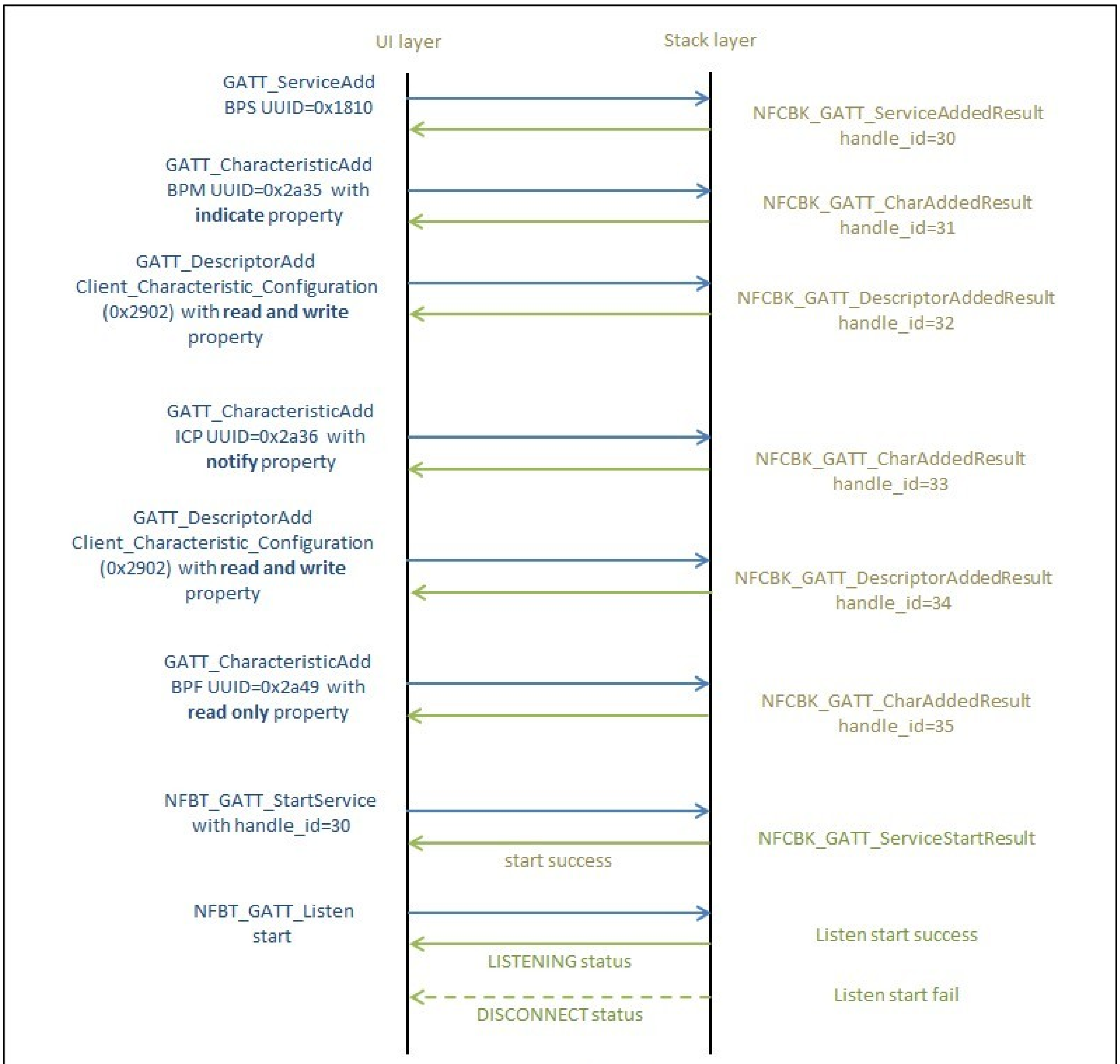
Function	**typedef void (*NFCBK_GATT_ConnectStatus)(GATT_CONN_STATUS status, BT_ADDR *addr, GATT_ROLE role)**	
Parameter	status	See GATT_CONN_STATUS
	addr	Pointer of Client address which one connected.
	role	See GATT_ROLE
OP function	**NFBT_ForceConnectProfile or Client device start establish connection**	
Description	Callback of indicate BLE connection status.	

4. Operate Flow Chart example

4.1 Peripheral

4.1.1 Create a service

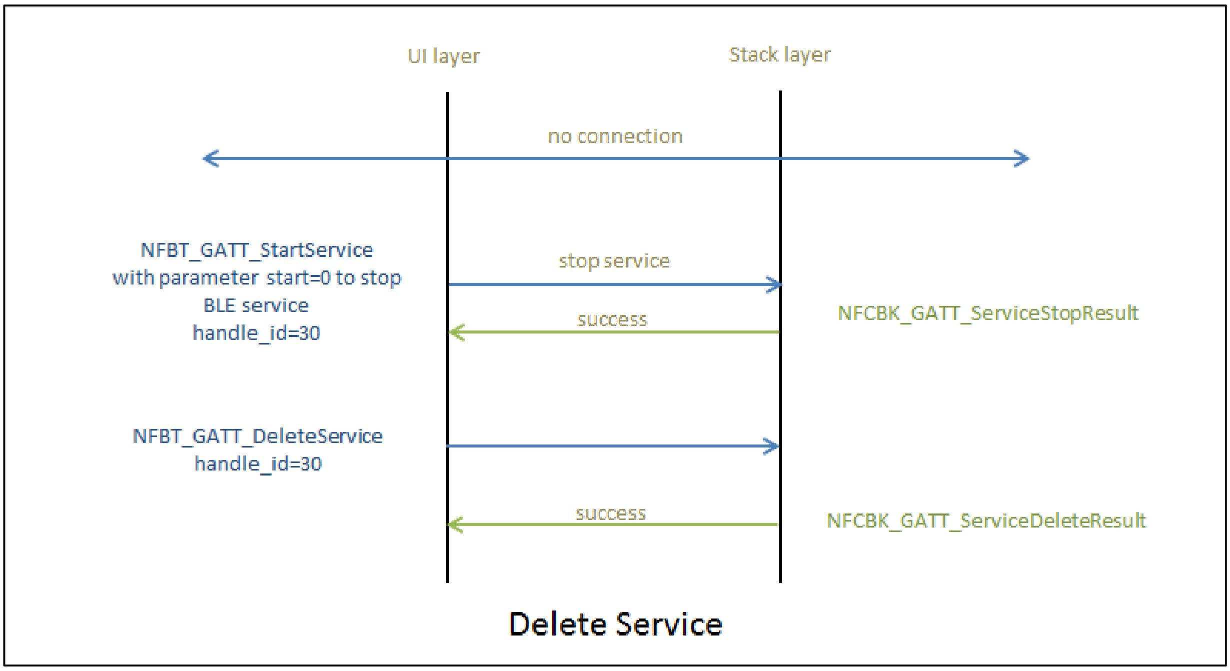
Make service example as Blood Pressure Service (BPS), the handle_id made by system stack return to UI layer for other operate, read, write and delete etc. handle_id value generated depending on different of system. Make sure BLE connection not connected and no service started before create new service.



Create BPS flowchart

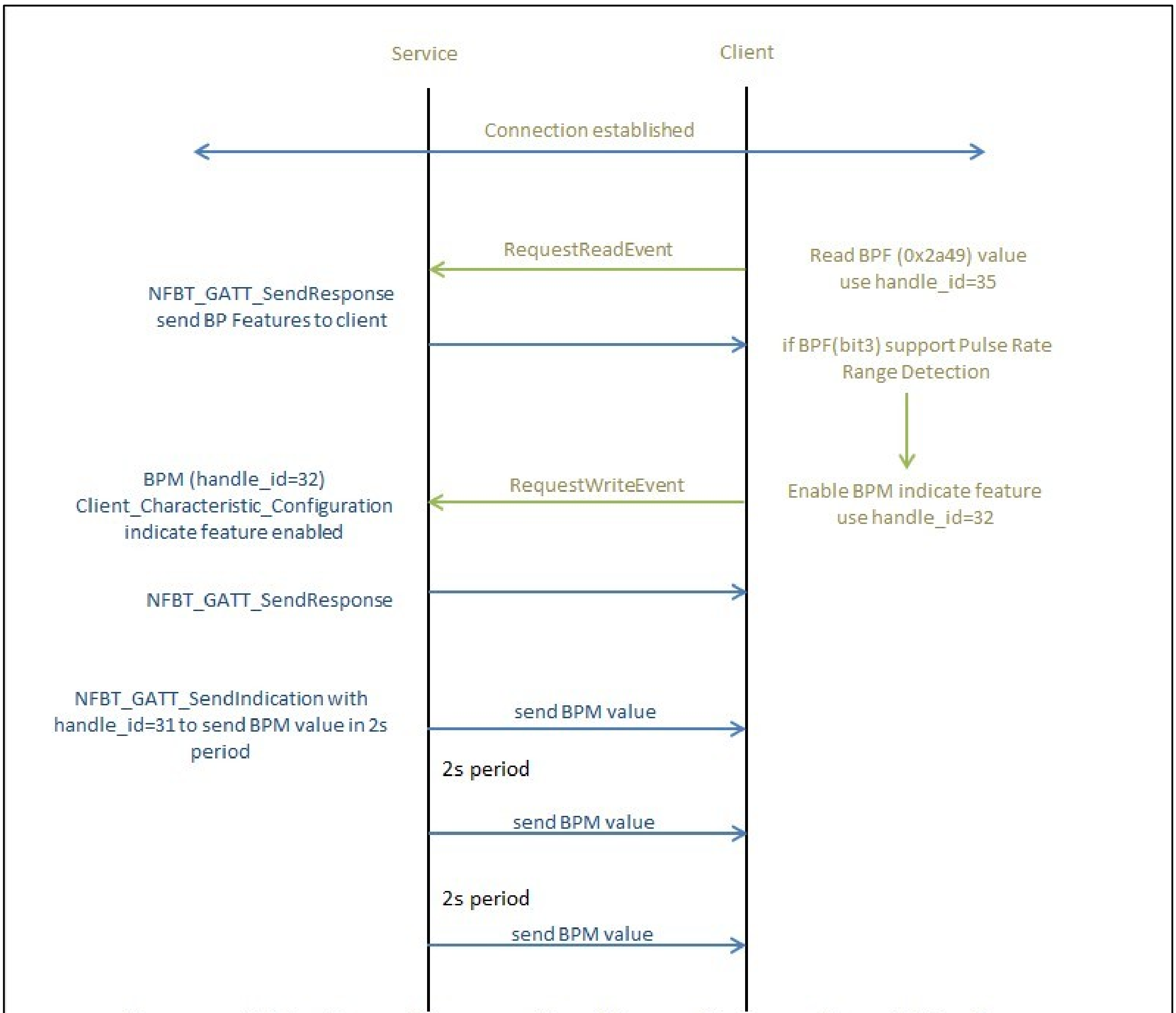
4.1.2 Delete a service

The service cannot visible in advertising data if service is not use any more, delete it from system for re lease resource. The handle_id got from service registered for stop and delete service. Make sure the s ervice is stop before to do delete.



4.1.3 Request Write Event and Send Indication

Request-write event informed by client that request write or config some values in service device. For example when client need to enable indication/notificationon service device. Client use RequestWriteEvent command inform to service turn on indication/notification function and feedback when service some values changed.

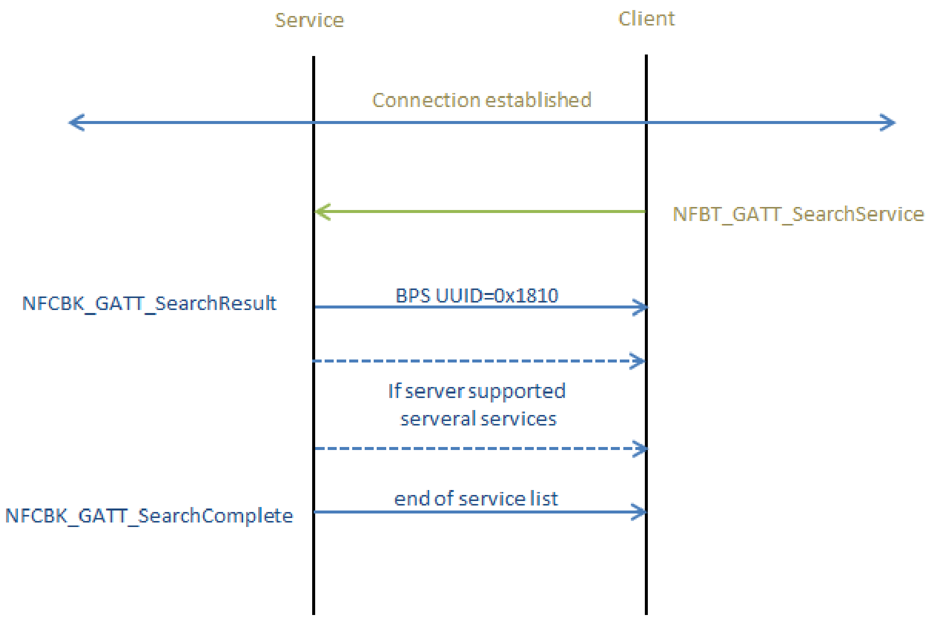


RequestWriteEvent/RequestReadEvent/RequestExecWriteEvent
and SendInication flowchart

4.2 Central

4.2.1 Service Search

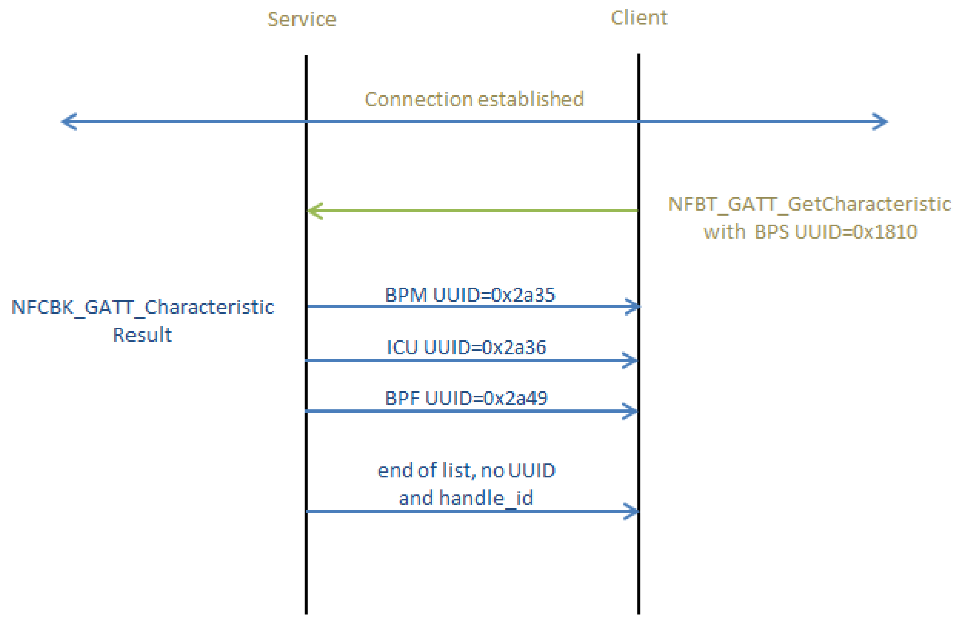
Search service to list all server provided service for operate. A device may have several services for transfer different data of part of service to watch. For example, when user in sport, a personal health device that there are the Blood Pressure Service detect user’s blood pressure and Heart Rate Service detect heart rate same time then display on phone or watch.



Search service from server flowchart

4.2.2 Get Characteristic

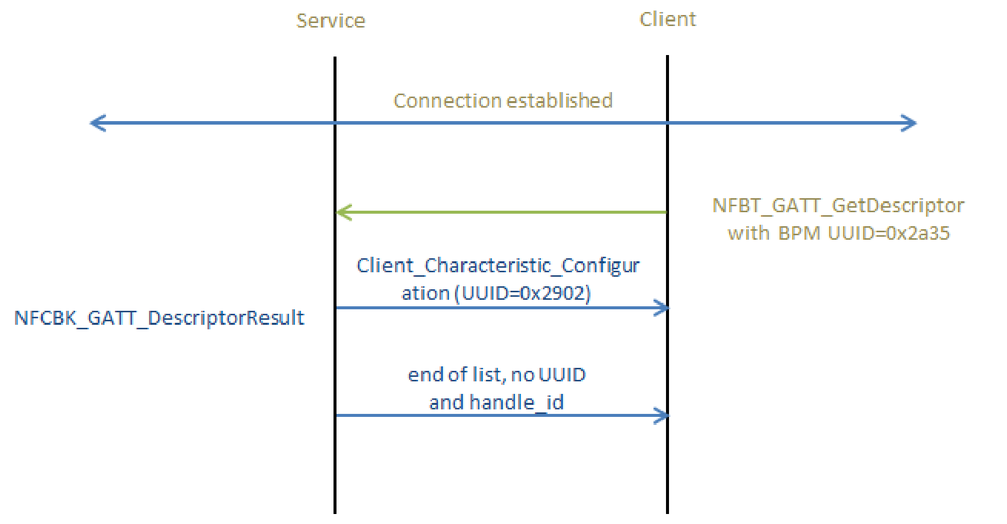
Which service characteristic provided by server, use NFBT_GATT_GetCharacteristic to discover service characteristics on a server. Once the characteristics are discovered additional information about the c haracteristics can be discovered or accessed using other procedures.



Get characteristic list from server flowchart

4.2.3 Get Characteristic Descriptors

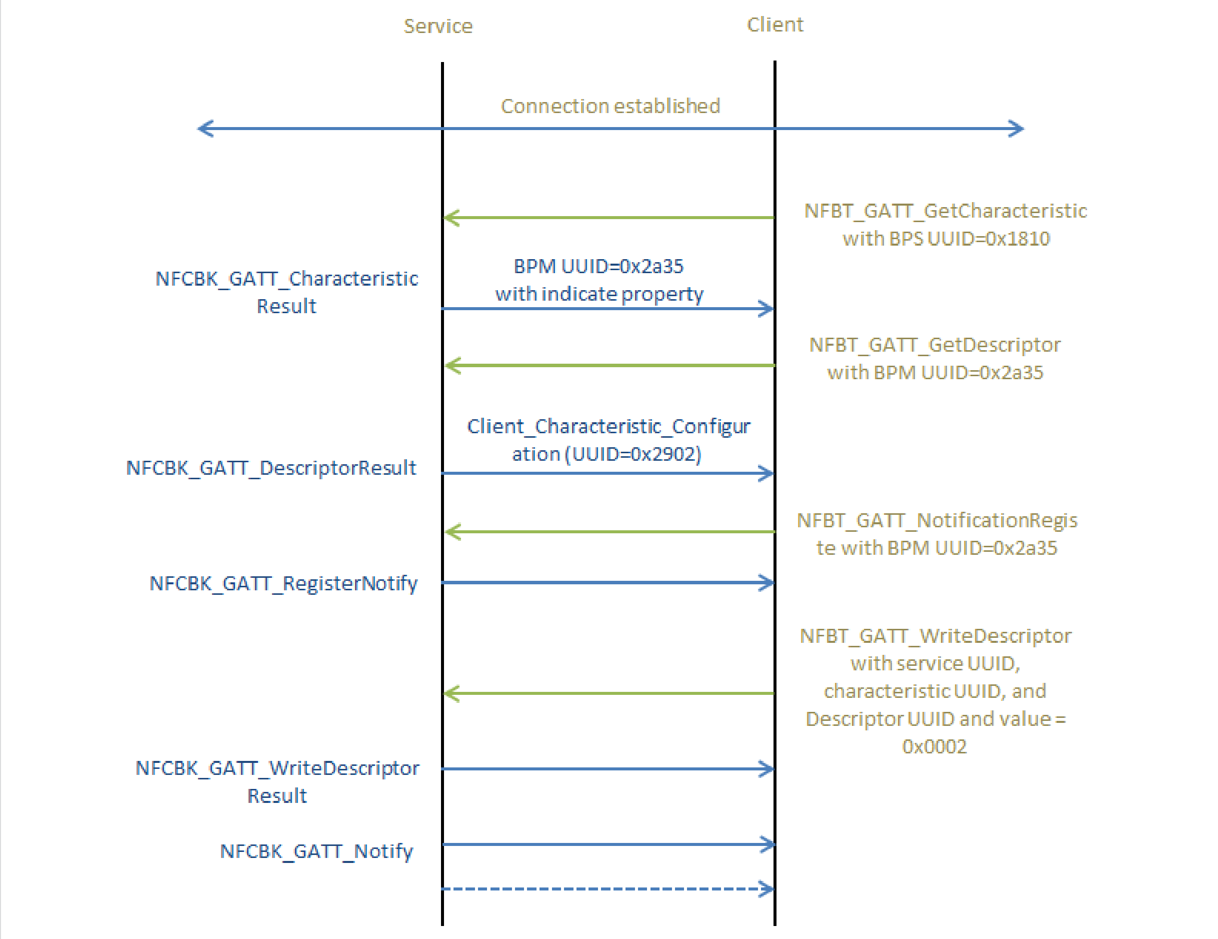
Which characteristic descriptors provided by server, use NFBT_GATT_GetDescriptor to discover characteris tic descriptors on a server. Once the characteristic descriptors are discovered additional information about the characteristic descriptors can be discovered or accessed using other procedures.



Get characteristic descriptors list from server flowchart

4.2.4 Characteristic Notification

This is used to notify a client of the value of a Characteristic Value from a server. Notifications can be config ured using the Client Characteristic Configuration (0x2902) descriptor. Indication (0x0002) or Notification (0 x0001) configuration in Client Characteristic Configuration depend on which characteristic of service.


Service
Client
Connection established
NFBT_GATT_GetCharacteristic with BPS UUID=0x1810
NFCBK_GATT_Characteristic Result
BPM UUID=0x2a35 with indicate property
NFBT_GATT_GetDescriptor with BPM UUID=0x2a35
Client_Characteristic_Configur ation (UUID=0x2902)
NFCBK_GATT_DescriptorResult
NFBT_GATT_NotificationRegis te with BPM UUID=0x2a35
NFCBK_GATT_RegisterNotify
NFBT_GATT_WriteDescriptor with service UUID, characteristic UUID, and Descriptor UUID and value = 0x0002
NFCBK_GATT_WriteDescriptor Result
NFCBK_GATT_Notify

Notification characteristic from server flowchart

5. BLE data transmission throughput

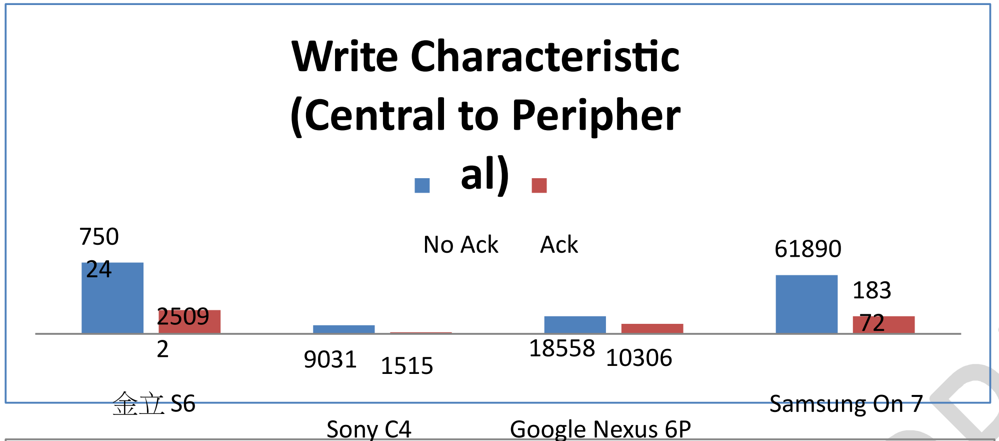
5.1 Test device

Test Platform(Central)	Sony C4	金立 S6	Samsung Galaxy On7	NFS7900(AP7)	Google nexus 6p	iPhone 5s
Software version	5.0	5.1	5.1.1	5.1 (Bluedroid)	6	iOS 9.1
BT Version	4.0	4.0	4.1	4.1	4.2	4.0

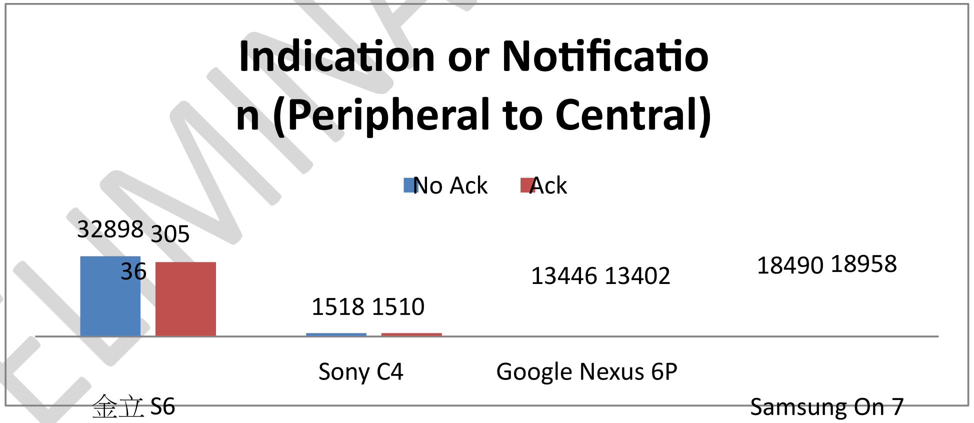
Test Platform (Peripheral)	NFS7900(AP8)
Android version	5.1 (Bluedroid)
BT Version	4.0

Test distance between 2 devices of ANT is 5cm

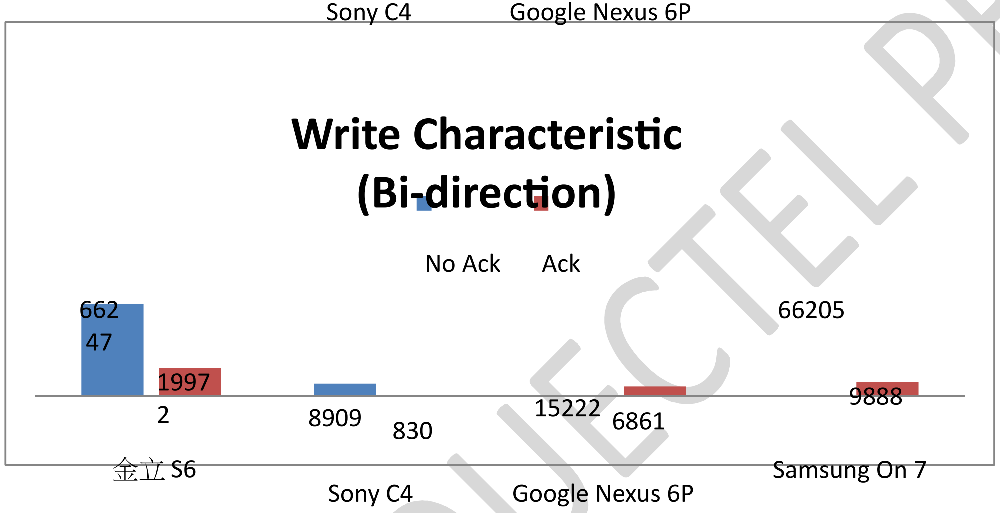
5.2 NF2208 data throughput(Kbps)



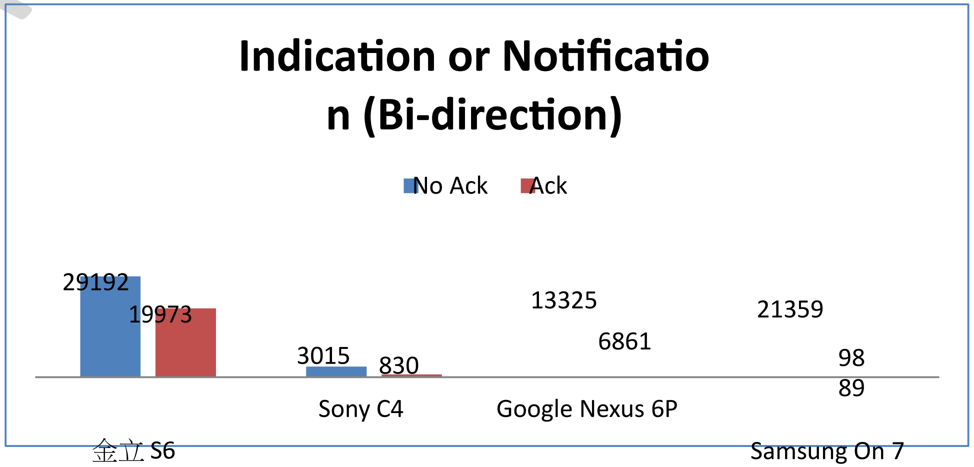
Write Characteristic (Central to Peripher al)



Indication or Notificatio n (Peripheral to Central)

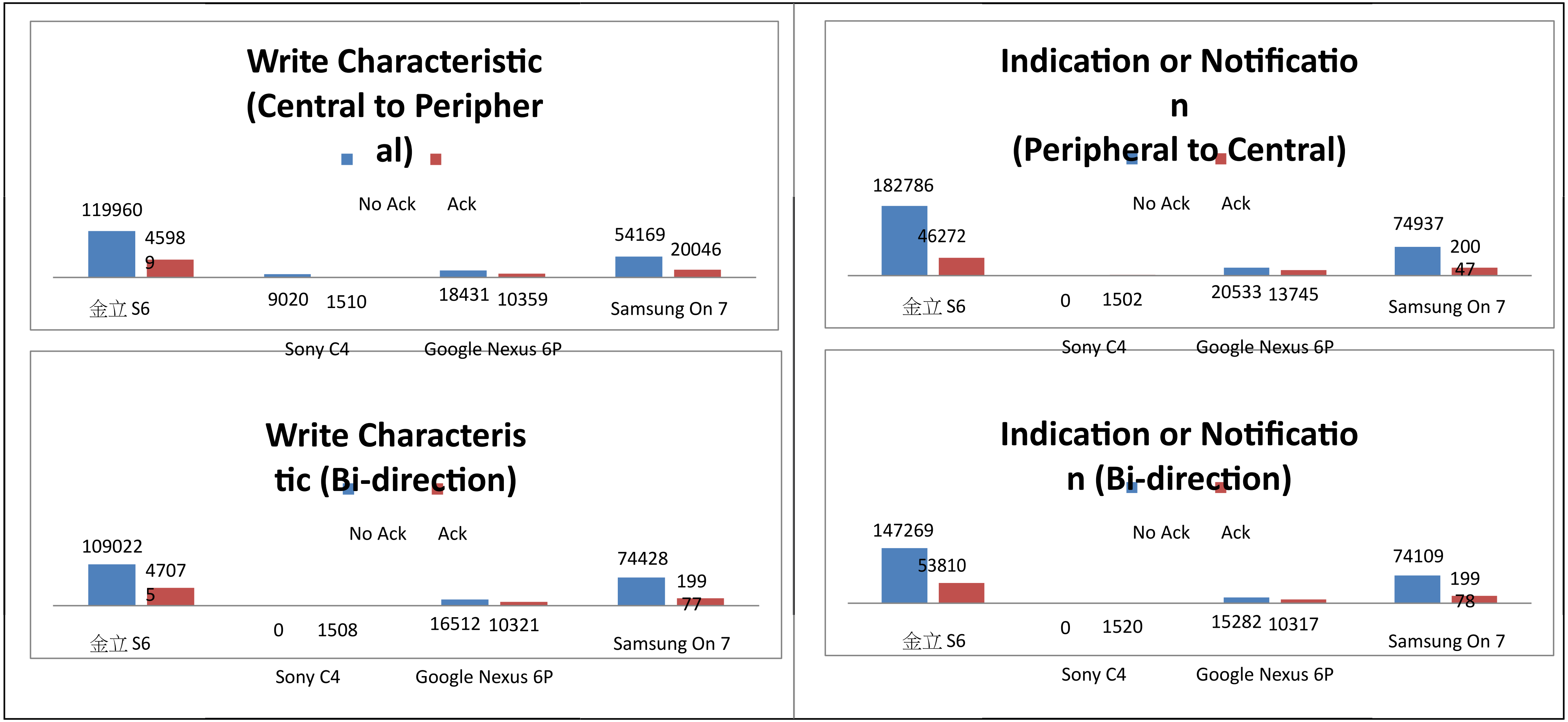


Write Characteristic (Bi-direction)

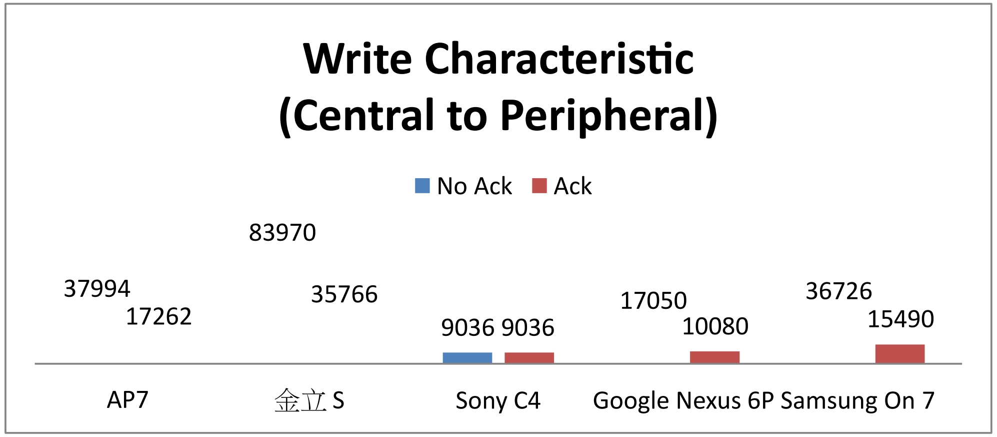


Indication or Notificatio n (Bi-direction)

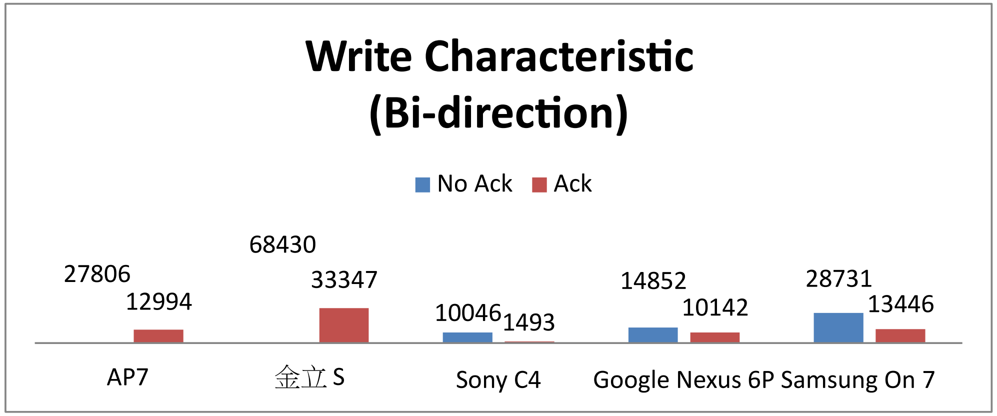
5.3 NF3303 data throughput(Kbps)



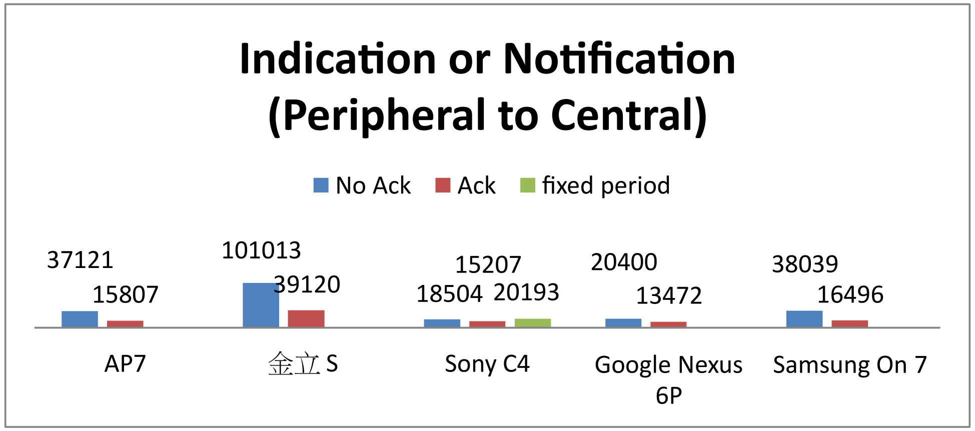
5.4 NF8350 data throughput(Kbps)



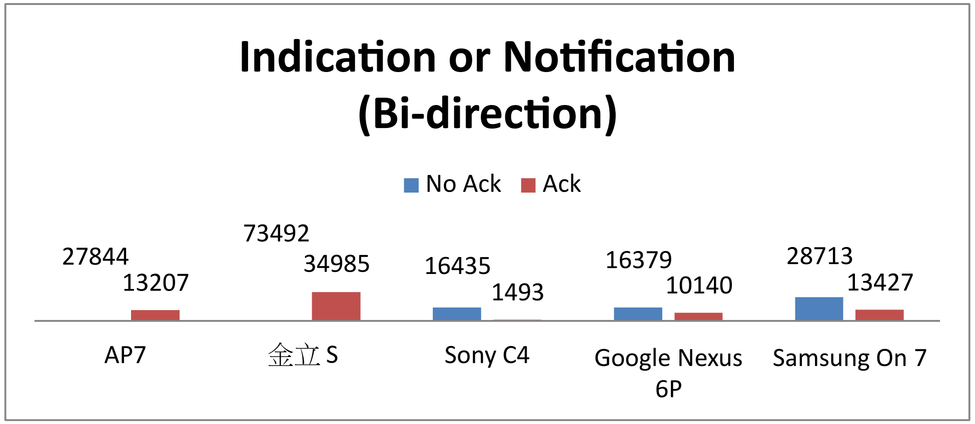
Write Characteristic (Central to Peripheral)



Write Characteristic (Bi-direction)

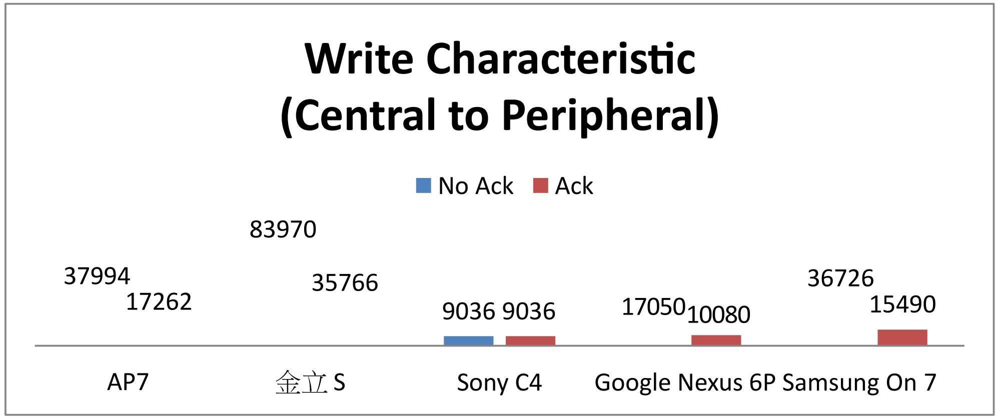


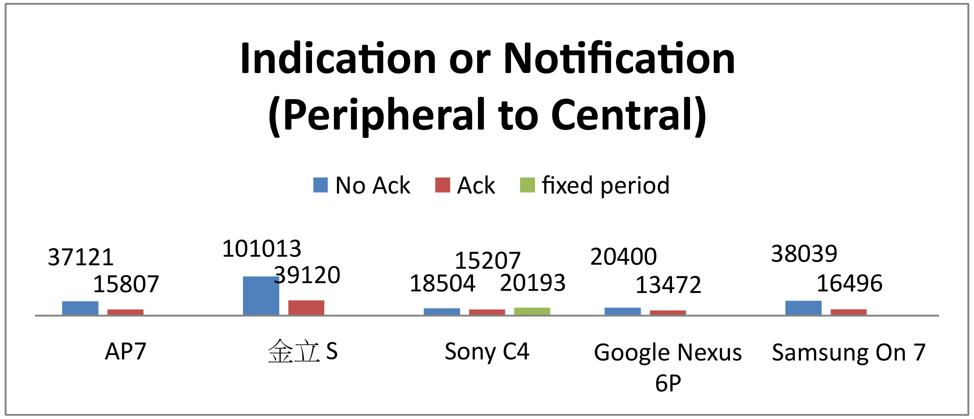
Indication or Notification (Peripheral to Central)

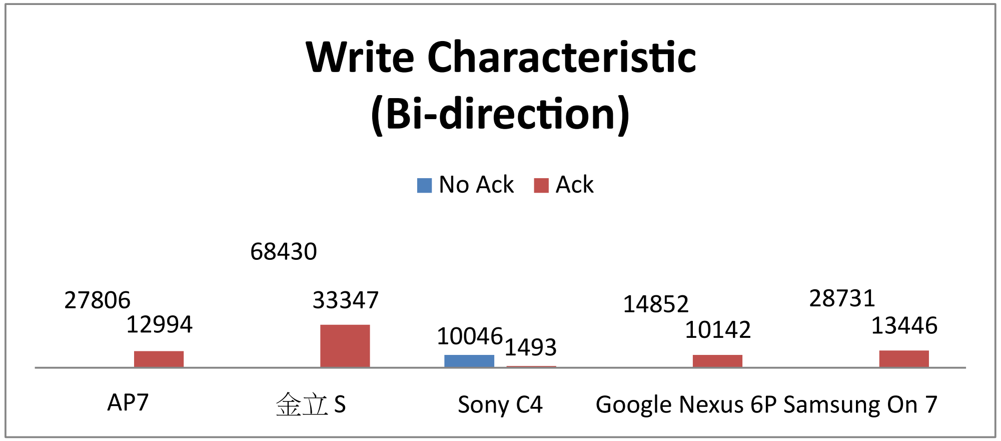


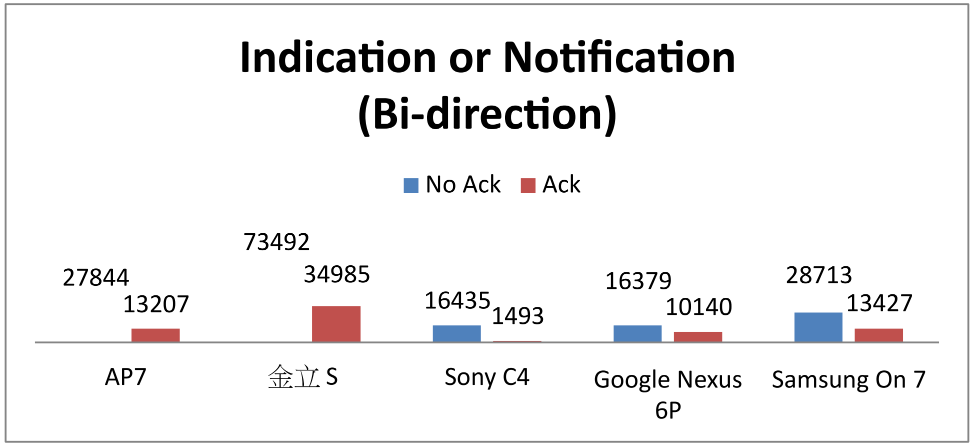
Indication or Notification (Bi-direction)

5.5 NFS7900 data throughput(Kbps)









6. Appendix

6.1 Name Definition

```
/*    Bluetooth status */
typedef enum
{
      NFBT_STATUS_SUCCESS = 0, NFBT_STATUS
      _FAIL, NFBT_STATUS_NOT_READY, NFBT_S
      TATUS_NOMEM, NFBT_STATUS_BUSY, NFB
      T_STATUS_DONE, NFBT_STATUS_UNSUPP
      ORTED, NFBT_STATUS_PARM_INVALID, NF
      BT_STATUS_UNHANDLED, NFBT_STATUS_A
      UTH_FAILURE, NFBT_STATUS_RMT_DEV_D
      OWN, NFBT_STATUS_AUTH_REJECTED, NF
      BT_STATUS_WRONG_STATUS
} bt_status;

/* connection status used for CLIENT/SERVER connection status */ typede
f enum
{
      GATT_CONN_DISCONNECT = 0,
      GATT_CONN_START_LISTENING = 1, /* ONLY PERIPHERAL(SERVER) TO USED */
      GATT_CONN_LISTENING = 2, /* ONLY PERIPHERAL(SERVER) TO USED */ GATT_
      CONN_CONNECTING = 3,
      GATT_CONN_CONNECTED = 4,

      GATT_CONN_SRV_REG_ING = 5, /* ONLY PERIPHERAL(SERVER) TO USED */ G
      ATT_CONN_SRV_UNREG_ING = 6 /* ONLY PERIPHERAL(SERVER) TO USED */
} GATT_CONN_STATUS;
```

```
typedef enum
{
        GATT_SERVICE_STOP = 0,
        GATT_SERVICE_START = 1
} GATT_SERVICE_STATUS;

/* role status */ typede
f enum
{
        GATT_ROLE_PERIPHERAL = 0,
        GATT_ROLE_CENTRAL = 1
} GATT_ROLE;
```

6.2 Characteristic Format Types

Format	Short Name	Description	Exponent Value
0x00	rfu	Reserved for future use	No
0x01	boolean	unsigned 1-bit; 0=false, 1=true	No
0x02	2bit	unsigned 2-bit integer	No
0x03	nibble	unsigned 4-bit integer	No
0x04	uint8	unsigned 8-bit integer	Yes
0x05	uint12	unsigned 12-bit integer	Yes
0x06	uint16	unsigned 16-bit integer	Yes
0x07	uint24	unsigned 24-bit integer	Yes
0x08	uint32	unsigned 32-bit integer	Yes
0x09	uint48	unsigned 48-bit integer	Yes
0x0A	uint64	unsigned 64-bit integer	Yes
0x0B	uint128	unsigned 128-bit integer	Yes
0x0C	sint8	signed 8-bit integer	Yes
0x0D	sint12	signed 12-bit integer	Yes
0x0E	sint16	signed 16-bit integer	Yes
0x0F	sint24	signed 24-bit integer	Yes
0x10	sint32	signed 32-bit integer	Yes
0x11	sint48	signed 48-bit integer	Yes
0x12	sint64	signed 64-bit integer	Yes
0x13	sint128	signed 128-bit integer	Yes
0x14	float32	IEEE-754 32-bit floating point	No
0x15	float64	IEEE-754 64-bit floating point	No
0x16	SFLOAT	IEEE-11073 16-bit SFLOAT	No
0x17	FLOAT	IEEE-11073 32-bit FLOAT	No
0x18	duint16	IEEE-20601 format	No
0x19	utf8s	UTF-8 string	No
0x1A	utf16s	UTF-16 string	No
0x1B	struct	Opaque structure	No

0x1C-0xFF	rfu	Reserved for Future Use	No

Additional Formatting Notes

- When encoding an IPv4 address, the uint32 Format type shall be used.
- When encoding an IPv6 address, the uint128 Format type shall be used.
- When encoding a Bluetooth BD_ADDR, the uint48 Format type shall be used.
- A duint16 is two uint16 values concatenated together.