BLE Apple Notification Client

1.0

Features

- BLE ANCS Service GATT Client in GAP Peripheral role
- Low Power mode
- LED status and incoming call indication
- Workflow status and notification information reporting through UART
- Ability to accept or decline incoming calls by the push-button

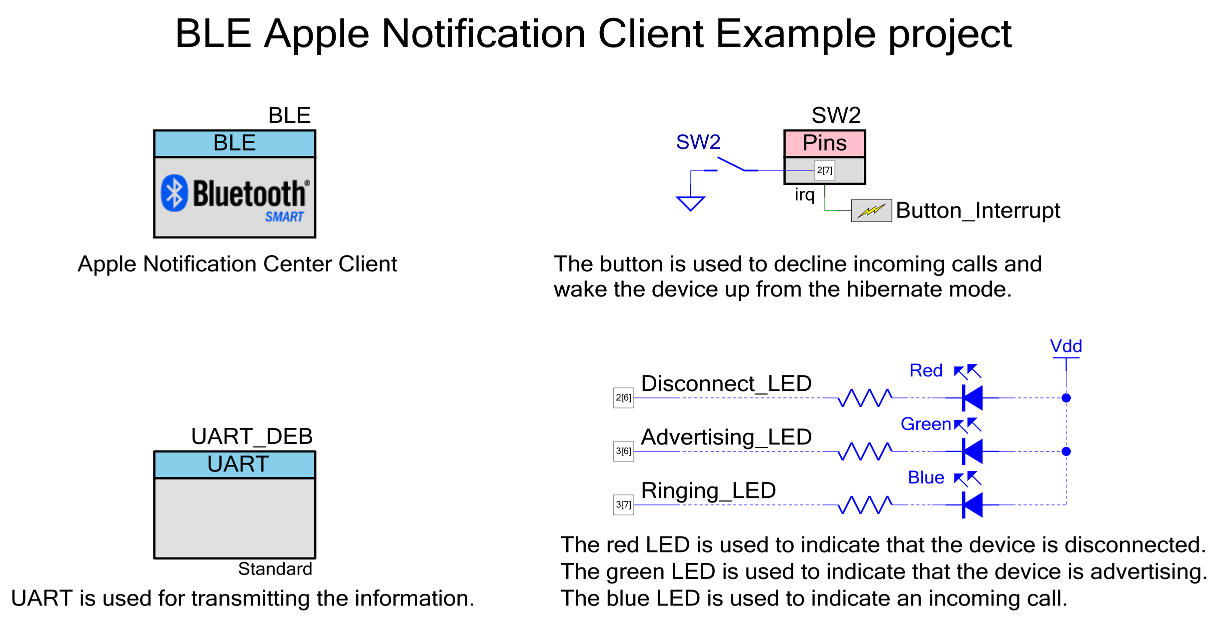
General Description

This example project demonstrates the BLE Apple Notification Client application workflow. The application uses the BLE Apple Notification Center Service in GATT Client mode to communicate with a BLE Apple Notification Center Server (iPhone, iPod, etc.).

Development Kit Configuration

Default CY8CKIT-042 BLE Pioneer Kit configuration.

Project Configuration



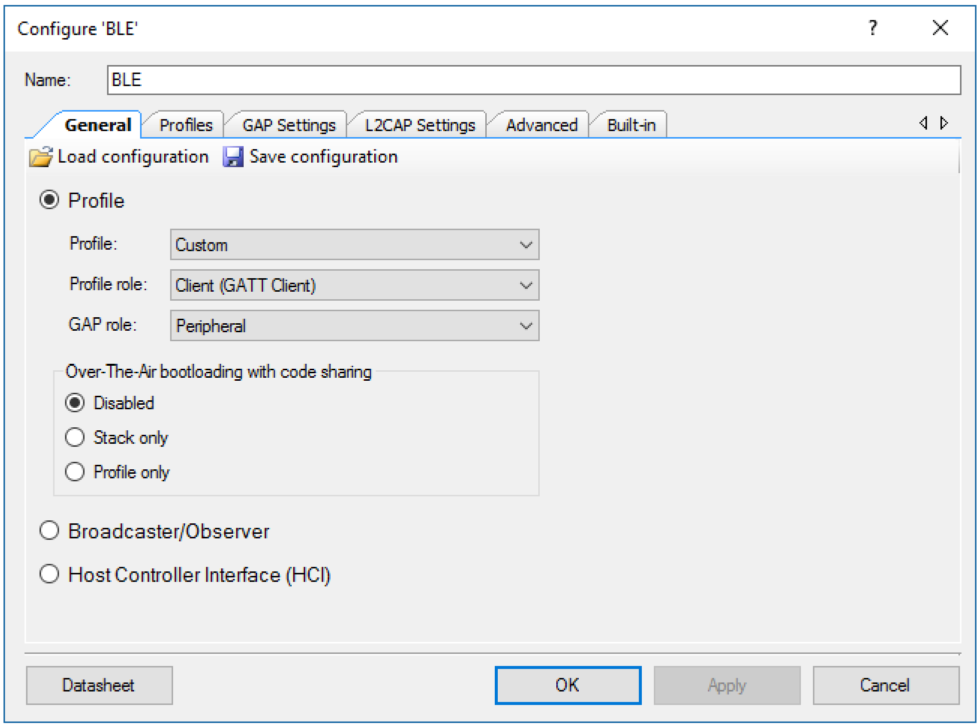
The button is used to decline incoming calls and wake the device up from the hibernate mode.

UART is used for transmitting the information.

The red LED is used to indicate that the device is disconnected.
The green LED is used to indicate that the device is advertising.
The blue LED is used to indicate an incoming call.

Figure 1. Top design schematic

The BLE component is configured as Apple Notification Center Client in GAP Peripheral role.



Configure 'BLE'

Name: BLE

(●) Profile

Profile: Custom

Profile role: Client (GATT Client)

GAP role: Peripheral

Over-The-Air bootloading with code sharing

(●) Disabled

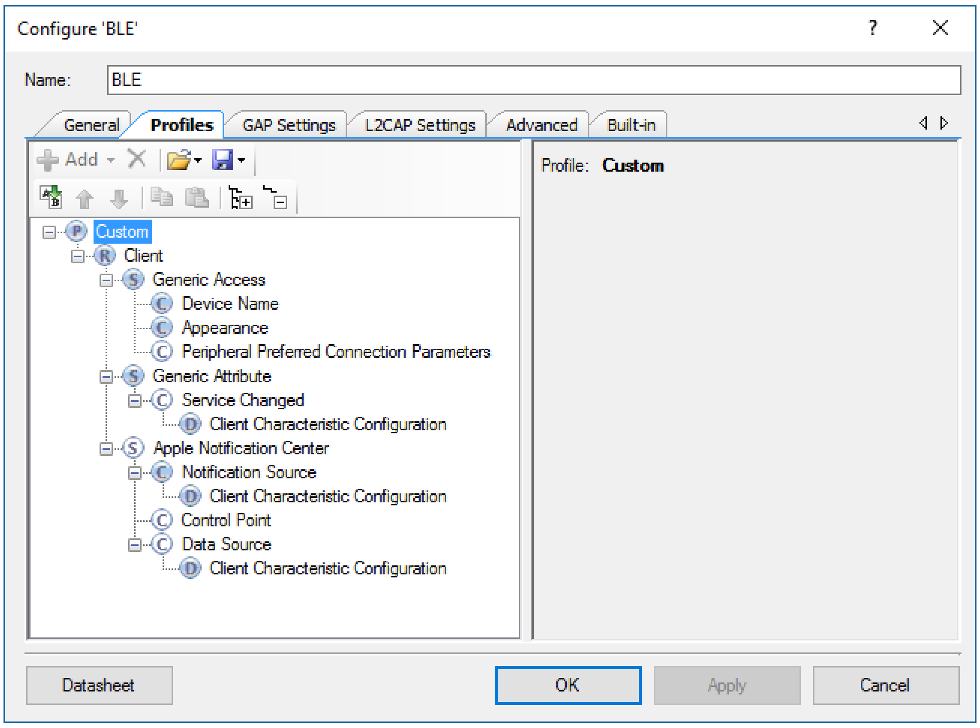
( ) Stack only

( ) Profile only

( ) Broadcaster/Observer

( ) Host Controller Interface (HCI)

Figure 2. General settings



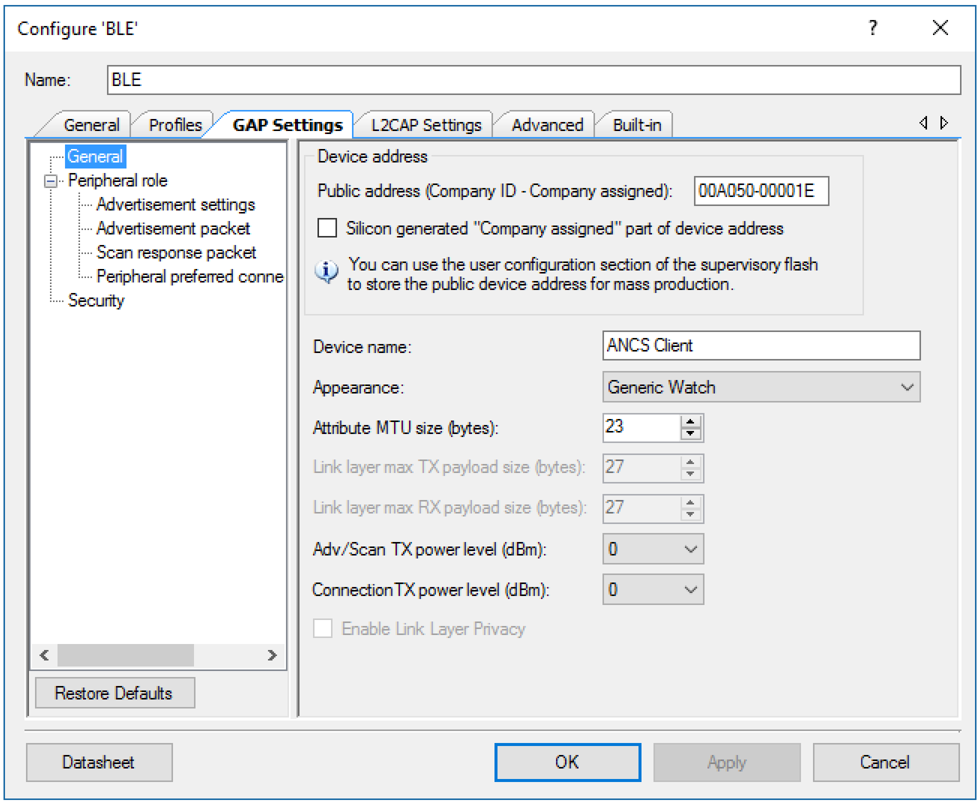
Configure 'BLE'

Name: BLE

```
Custom
  Client
    Generic Access
      Device Name
      Appearance
      Peripheral Preferred Connection Parameters
    Generic Attribute
      Service Changed
        Client Characteristic Configuration
    Apple Notification Center
      Notification Source
        Client Characteristic Configuration
      Control Point
      Data Source
        Client Characteristic Configuration
```

Profile: **Custom**

Figure 3. GATT Settings


Configure 'BLE'

Name: BLE

Tabs: General | Profiles | **GAP Settings** | L2CAP Settings | Advanced | Built-in

Tree: **General** / Peripheral role / Advertisement settings / Advertisement packet / Scan response packet / Peripheral preferred conne / Security

Device address

Public address (Company ID - Company assigned): 00A050-00001E

☐ Silicon generated "Company assigned" part of device address

You can use the user configuration section of the supervisory flash to store the public device address for mass production.

Device name: ANCS Client

Appearance: Generic Watch

Attribute MTU size (bytes): 23

Link layer max TX payload size (bytes): 27

Link layer max RX payload size (bytes): 27

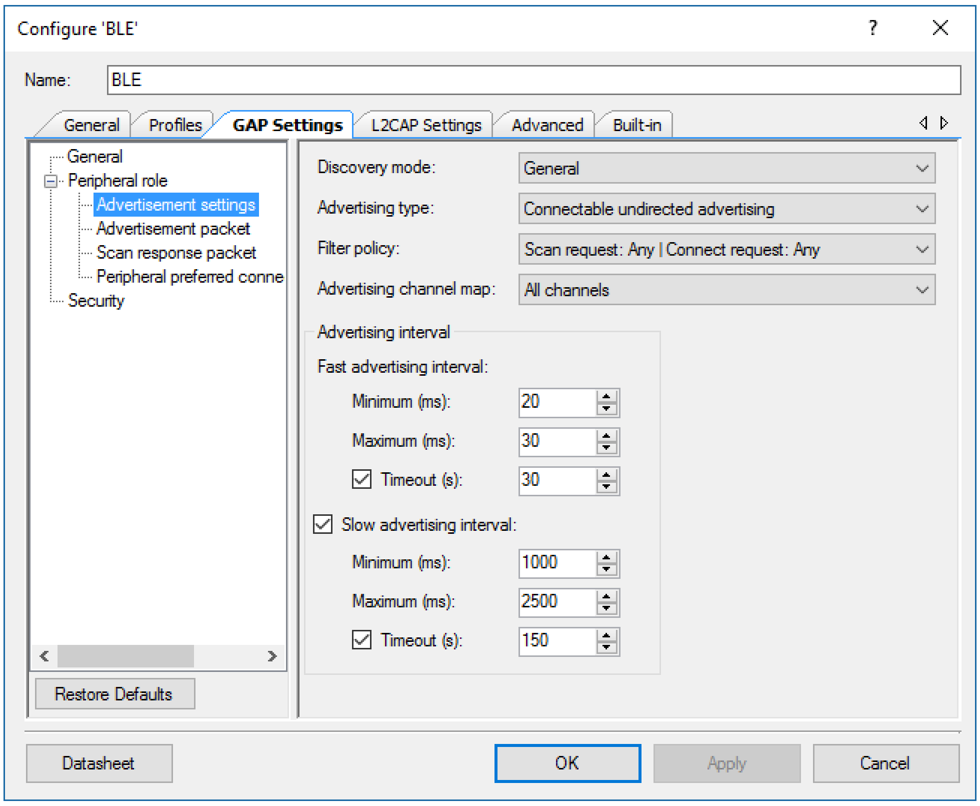
Adv/Scan TX power level (dBm): 0

Connection TX power level (dBm): 0

☐ Enable Link Layer Privacy

Restore Defaults | Datasheet | OK | Apply | Cancel

Figure 4. GAP Settings


Configure 'BLE'

Name: BLE

Tabs: General | Profiles | **GAP Settings** | L2CAP Settings | Advanced | Built-in

Tree: General / Peripheral role / **Advertisement settings** / Advertisement packet / Scan response packet / Peripheral preferred conne / Security

Discovery mode: General

Advertising type: Connectable undirected advertising

Filter policy: Scan request: Any | Connect request: Any

Advertising channel map: All channels

Advertising interval

Fast advertising interval:

Minimum (ms): 20

Maximum (ms): 30

☑ Timeout (s): 30

☑ Slow advertising interval:

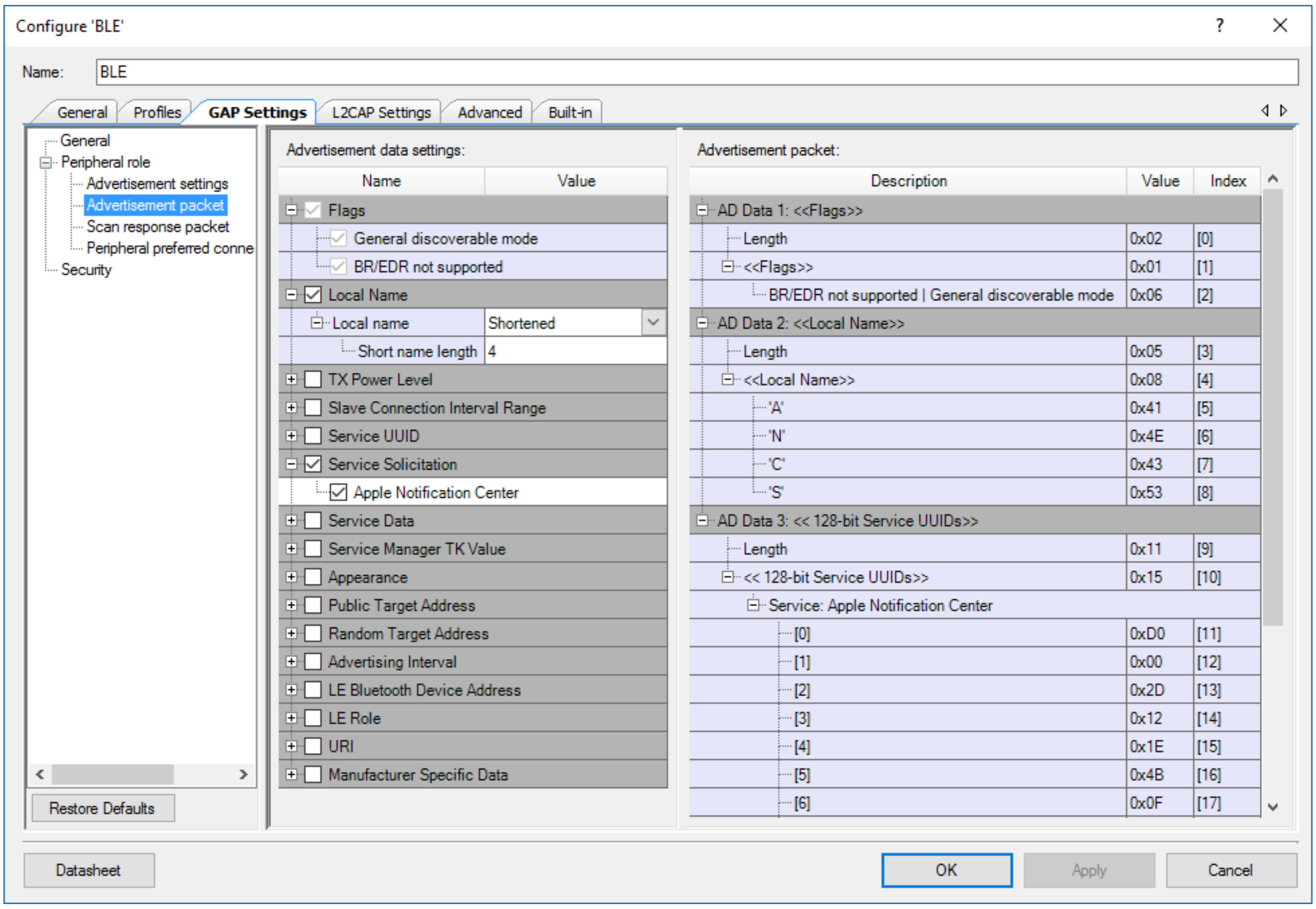
Minimum (ms): 1000

Maximum (ms): 2500

☑ Timeout (s): 150

Restore Defaults | Datasheet | OK | Apply | Cancel

Figure 5. GAP Settings -> Advertisement settings


Configure 'BLE'

Name: BLE

- General
- Peripheral role
  - Advertisement settings
  - Advertisement packet
  - Scan response packet
  - Peripheral preferred conne
- Security

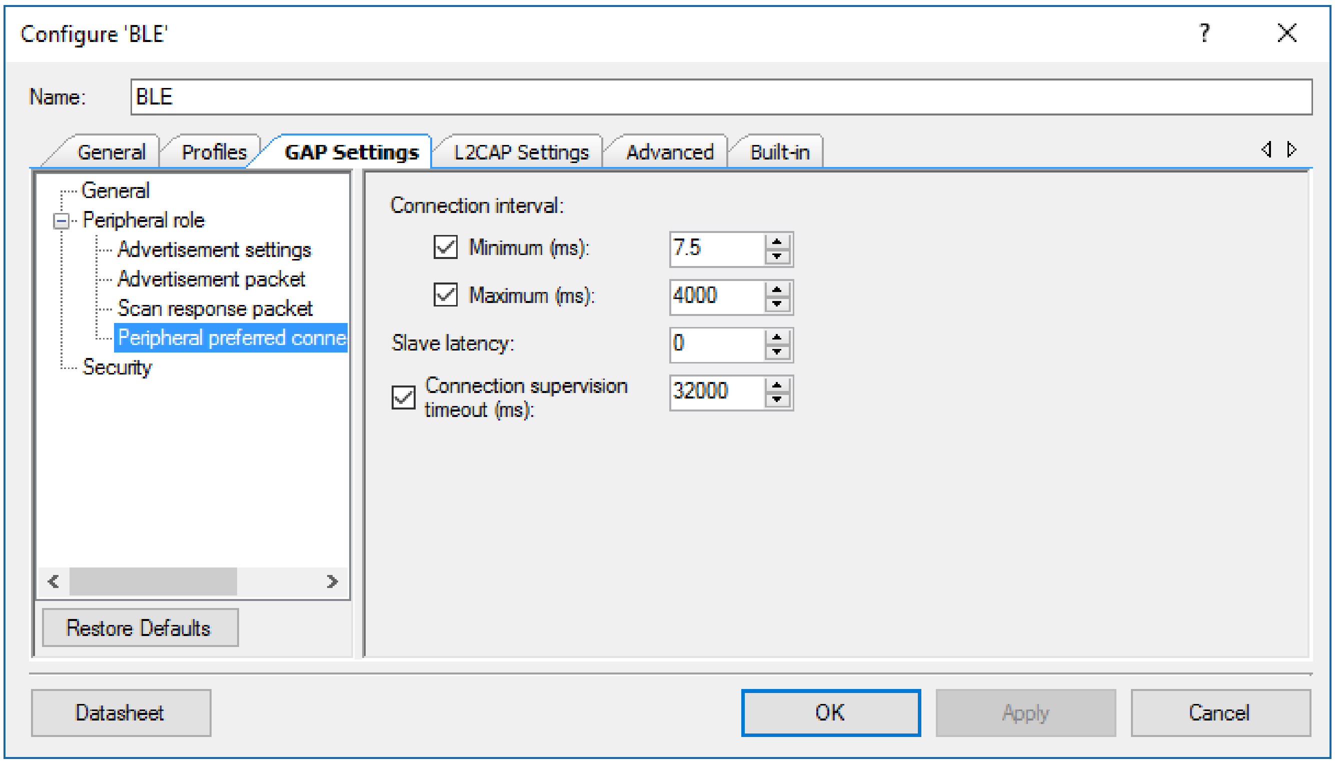
Advertisement data settings:

Name	Value
✓ Flags	
✓ General discoverable mode	
✓ BR/EDR not supported	
✓ Local Name	
Local name	Shortened
Short name length	4
TX Power Level	
Slave Connection Interval Range	
Service UUID	
✓ Service Solicitation	
✓ Apple Notification Center	
Service Data	
Service Manager TK Value	
Appearance	
Public Target Address	
Random Target Address	
Advertising Interval	
LE Bluetooth Device Address	
LE Role	
URI	
Manufacturer Specific Data	

Advertisement packet:

Description	Value	Index
AD Data 1: <<Flags>>		
Length	0x02	[0]
<<Flags>>	0x01	[1]
BR/EDR not supported | General discoverable mode	0x06	[2]
AD Data 2: <<Local Name>>		
Length	0x05	[3]
<<Local Name>>	0x08	[4]
'A'	0x41	[5]
'N'	0x4E	[6]
'C'	0x43	[7]
'S'	0x53	[8]
AD Data 3: << 128-bit Service UUIDs>>		
Length	0x11	[9]
<< 128-bit Service UUIDs>>	0x15	[10]
Service: Apple Notification Center		
[0]	0xD0	[11]
[1]	0x00	[12]
[2]	0x2D	[13]
[3]	0x12	[14]
[4]	0x1E	[15]
[5]	0x4B	[16]
[6]	0x0F	[17]

Figure 6. GAP Settings -> Advertisement packet



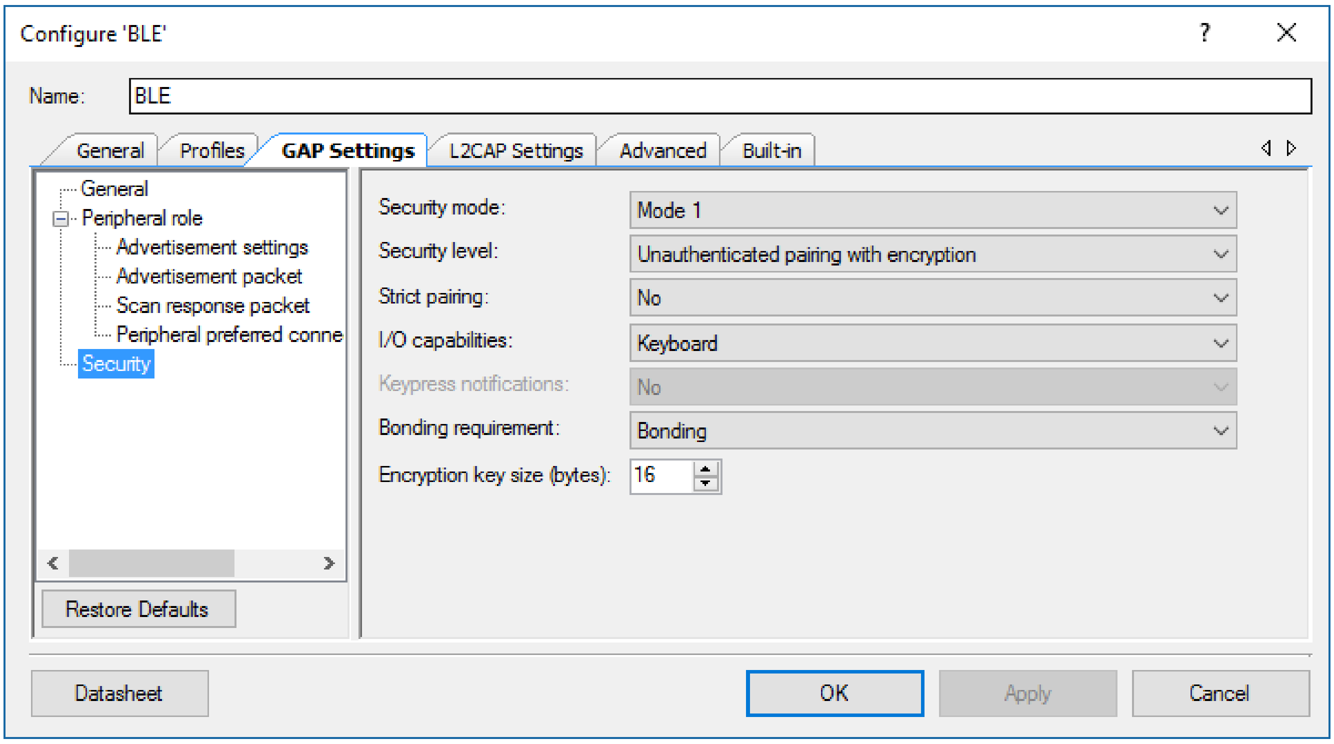
Connection interval:

- [✓] Minimum (ms): 7.5
- [✓] Maximum (ms): 4000

Slave latency: 0

[✓] Connection supervision timeout (ms): 32000

Figure 7. GAP Settings -> Peripheral preferred connection parameters



Security mode:	Mode 1
Security level:	Unauthenticated pairing with encryption
Strict pairing:	No
I/O capabilities:	Keyboard
Keypress notifications:	No
Bonding requirement:	Bonding
Encryption key size (bytes):	16

Figure 8. GAP Settings -> Security

Project Description

The project demonstrates the functionality of the BLE component configured as a BLE Apple Notification Center Service Client.

Right after startup the device performs BLE component initialization. In this project two callback functions are required for the BLE operation. Callback function AppCallBack() is required to receive generic events from BLE Stack, and the service-specific callback function AncsCallBack() is required for Apple Notification Center service-specific events. The CYBLE_EVT_STACK_ON event indicates a successful initialization of BLE Stack. After this event is received, the component starts advertising with the packet structure as described above (see **Figure 6**). The BLE component stops advertising as soon as 180 seconds advertising period expires.

The Apple Notification Client device can be connected to any Apple gadget which supports BLE Apple Notification Center Service configured as GAP Central role and GATT Server. To connect to the Apple Notification Client device, go to *Settings->Bluetooth* and find the "ANCS" while a device is advertising (green LED is blinking).

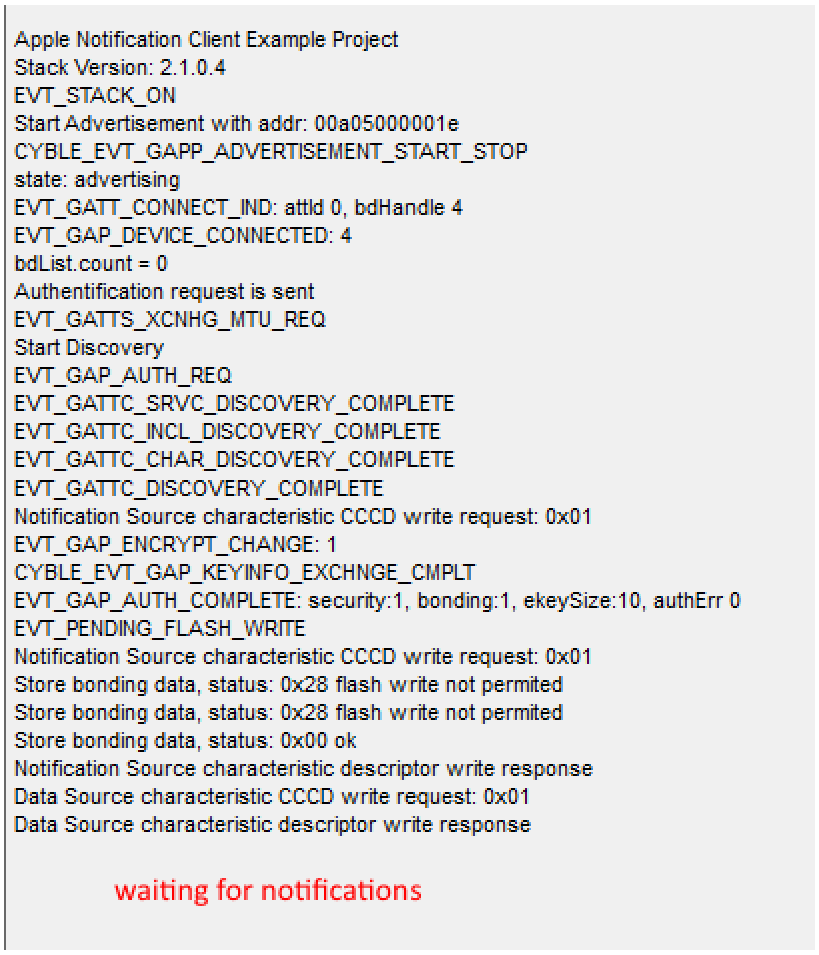
The red LED will turn on after fast and slow advertisement period elapsed to indicate that no Client is connected to the device and it felt asleep into hibernate mode. To wake up a device, use the SW2 button. When the Central device connects successfully, the Apple Notification Client discovers Server's GATT database (including Apple Notification Center Server's characteristics and descriptors) and enables the notifications.

The Apple Notification Client is able to show unread emails, incoming calls (also text messages, pending missed calls, etc.) from Viber application (and decline them) and regular incoming calls on iPhone (and accept or decline them). Pressing the SW2 button one time per second performs a "decline" action for incoming calls. Pressing the SW2 button two times per second performs an "accept" action for incoming calls. The WDT is used to make LEDs blinking.

Expected Results

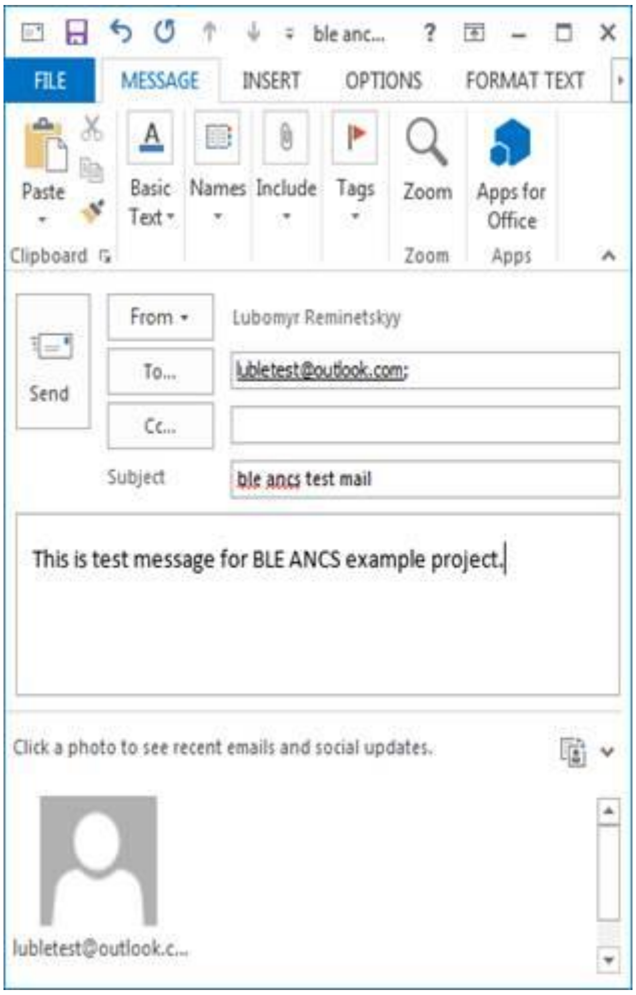
Example on how to operate the Apple Notification Client:

- Create an outlook account on your iPod (or iPhone) to operate with a regular "Mail" application.
- Configure the Notifications (*Settings->Notifications->Mail->Allow Notifications*) on iPod. (The project currently supports maximum 10 notifications. You can easily change this number by modifying CYBLE_ANCS_NS_CNT).
- Run the project (connect any terminal software to an appropriate COM port to observe the workflow).
- Connect to the device: go to the *Settings->Bluetooth*, find "ANCS", and tap on it, then tap on "Pair" in the dialog window.
- Now the device should discover the server (iPod) and wait the notifications:



```
Apple Notification Client Example Project
Stack Version: 2.1.0.4
EVT_STACK_ON
Start Advertisement with addr: 00a05000001e
CYBLE_EVT_GAPP_ADVERTISEMENT_START_STOP
state: advertising
EVT_GATT_CONNECT_IND: attId 0, bdHandle 4
EVT_GAP_DEVICE_CONNECTED: 4
bdList.count = 0
Authentification request is sent
EVT_GATTS_XCNHG_MTU_REQ
Start Discovery
EVT_GAP_AUTH_REQ
EVT_GATTC_SRVC_DISCOVERY_COMPLETE
EVT_GATTC_INCL_DISCOVERY_COMPLETE
EVT_GATTC_CHAR_DISCOVERY_COMPLETE
EVT_GATTC_DISCOVERY_COMPLETE
Notification Source characteristic CCCD write request: 0x01
EVT_GAP_ENCRYPT_CHANGE: 1
CYBLE_EVT_GAP_KEYINFO_EXCHNGE_CMPLT
EVT_GAP_AUTH_COMPLETE: security:1, bonding:1, ekeySize:10, authErr 0
EVT_PENDING_FLASH_WRITE
Notification Source characteristic CCCD write request: 0x01
Store bonding data, status: 0x28 flash write not permited
Store bonding data, status: 0x28 flash write not permited
Store bonding data, status: 0x00 ok
Notification Source characteristic descriptor write response
Data Source characteristic CCCD write request: 0x01
Data Source characteristic descriptor write response
```

- Send an email to the created outlook account:



From: Lubomyr Reminetskyy
To...: lubletest@outlook.com;
Subject: ble ancs test mail

This is test message for BLE ANCS example project.

- Observe the device is receiving emails, for example:

```
Data Source characteristic CCCD write request: 0x01
Data Source characteristic descriptor write response

From: Lubomyr Reminetskyy
Subject: FW: ble ancs test mail
Message: This is forwarded reply for test message for BLE ANCS example project.

From: Lubomyr Reminetskyy
Subject: RE: ble ancs test mail
Message: This is reply for test message for BLE ANCS example project.

From: Lubomyr Reminetskyy
Subject: ble ancs test mail
Message: This is test message for BLE ANCS example project.
```