BLE 4.0 (Bluetooth Low Energy)

BLE 4.0 (Bluetooth Low Energy)

- GAP
- GATT
  - Services
  - Decsriptors
  - Characteristics

BlueZ is the official Linux Bluetooth stack.
It provides support for core Bluetooth layers and protocol

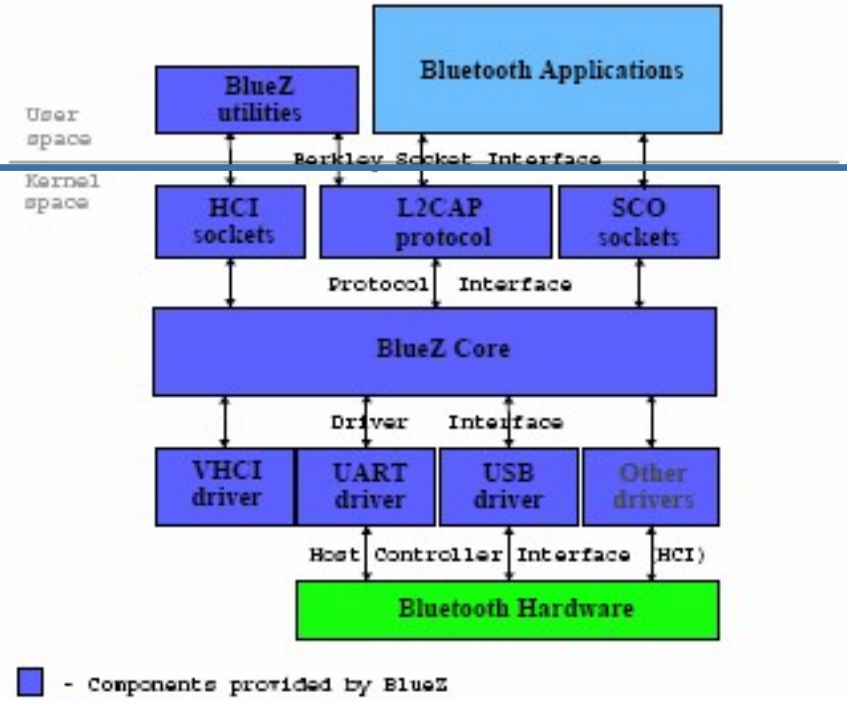
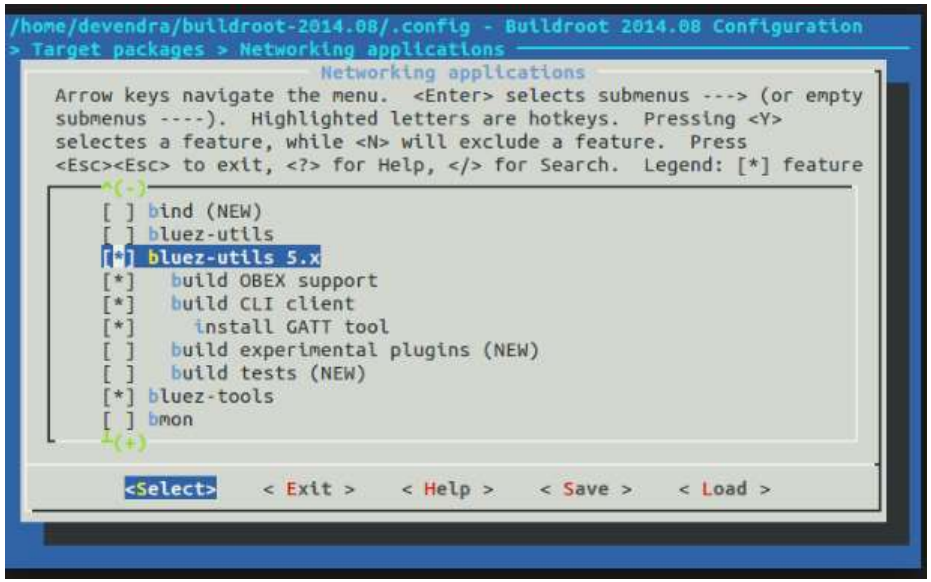


Figure 1: BlueZ Overview Diagram

Drivers and HW



bluepy

- Python interface to Bluetooth LE on Linux
- (proveds python bindings for BlueZ)
  - BlueZ provides support for the core Bluetooth layers and protocols
- This is a project to provide an API to allow access to Bluetooth Low Energy devices from Python. At present it runs on Linux only;

```
from bluepy.btle import Scanner

scanner = Scanner()
devices = scanner.scan(10.0)

for dev in devices:
    print "Device %s (%s), RSSI=%d dB" % (dev.addr, dev.addrType, dev.rssi)
    for (adtype, desc, value) in dev.getScanData():
        print "  %s = %s" % (desc, value)
```

Discover BLE devices

(by running blesca.py)

```
from bluepy.btle import Scanner

scanner = Scanner()
devices = scanner.scan(10.0)

for dev in devices:
    print "Device %s (%s), RSSI=%d dB" % (dev.addr, dev.addrType, dev.rssi)
    for (adtype, desc, value) in dev.getScanData():
        print "  %s = %s" % (desc, value)
```

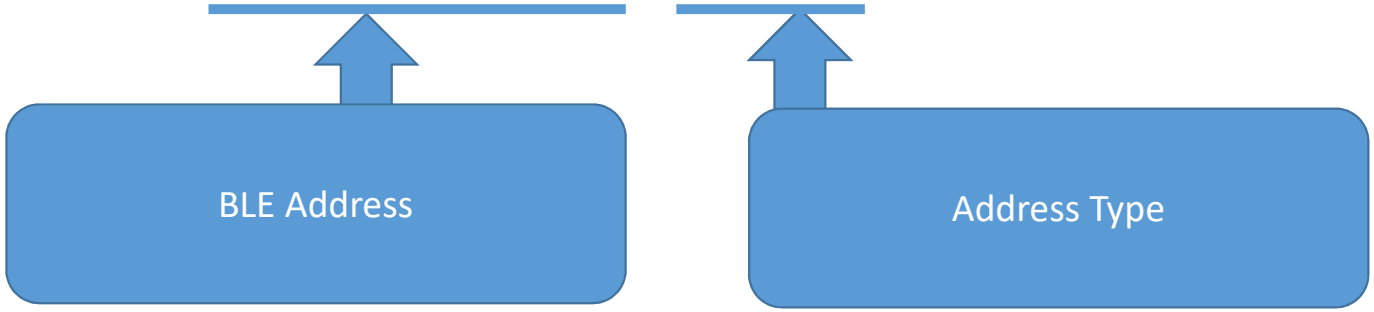
```
Device f9:ee:30:21:f6:6d (random), RSSI=-31 dB
  Flags = 06
  Complete 16b Services = 00a0
  Complete Local Name = Button
```

Discover BLE devices

(by running blesca.py)

Expected output:

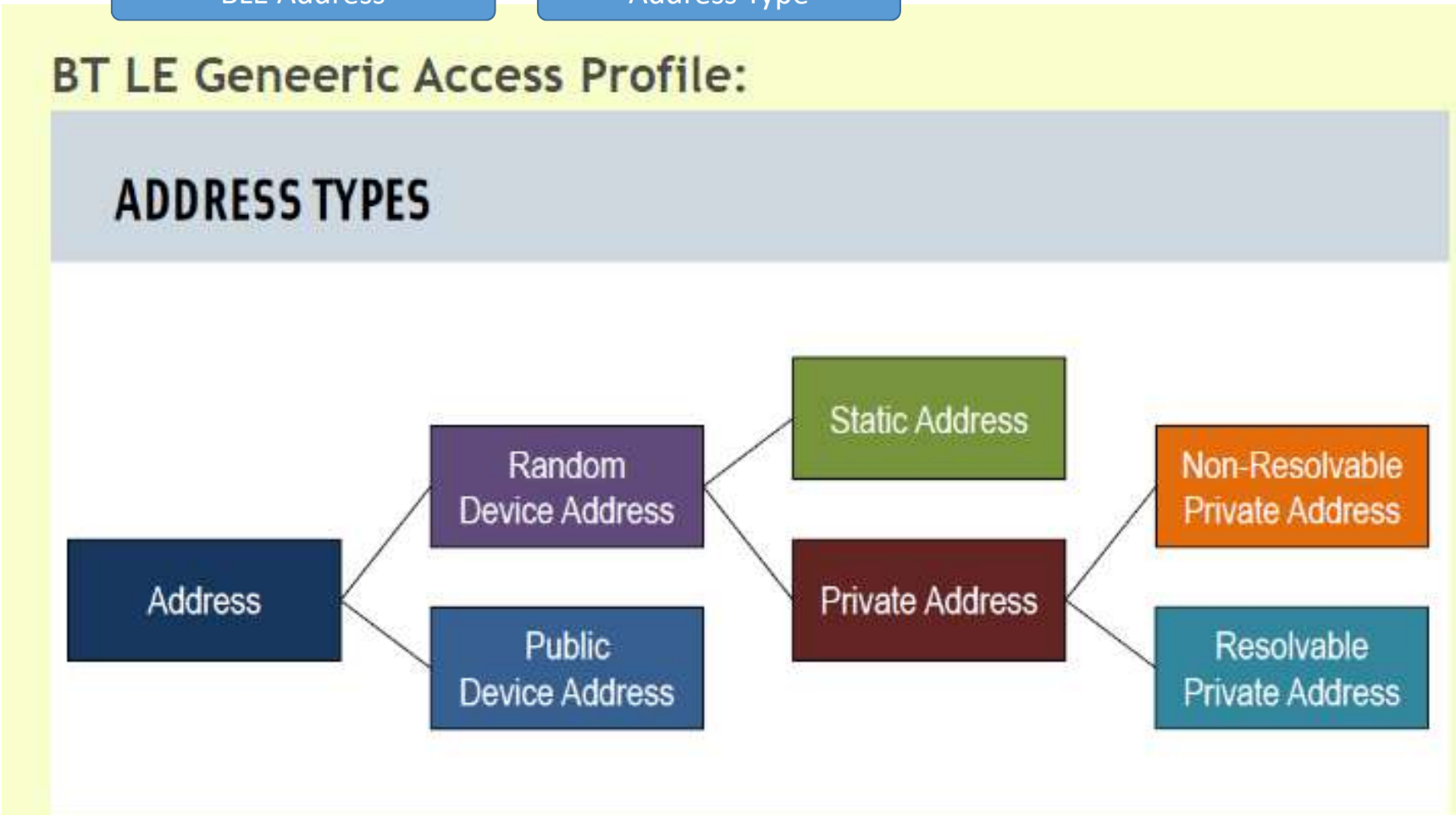
Device f9:ee:30:21:f6:6d (random), RSSI=-31 dB

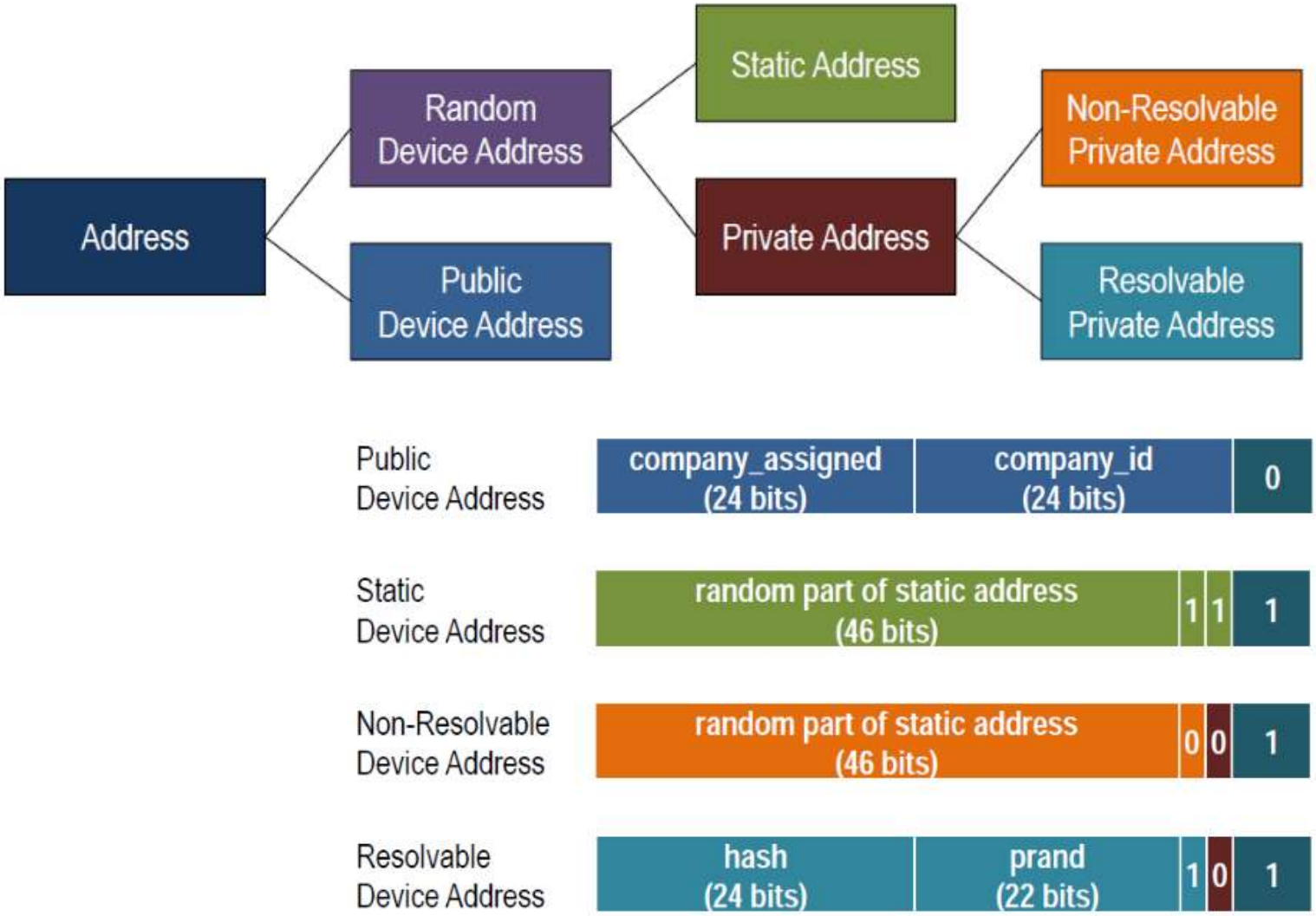


Device f9:ee:30:21:f6:6d (random), RSSI=-31 dB

BLE Address

Address Type




Address
Random Device Address
Public Device Address
Static Address
Private Address
Non-Resolvable Private Address
Resolvable Private Address
Public Device Address
company_assigned (24 bits)
company_id (24 bits)
0
Static Device Address
random part of static address (46 bits)
1
1
1
Non-Resolvable Device Address
random part of static address (46 bits)
0
0
1
Resolvable Device Address
hash (24 bits)
prand (22 bits)
1
0
1

Discover BLE devices

(by running blesca.py)

Expected output:

Device f9:ee:30:21:f6:6d (random)

The Generic Attributes (GATT)

GATT protocol is used to define the way that two Bluetooth Low Energy devices send and receive standard messages.

The Generic Attribute Profile (GATT) procedures define standard ways that services, characteristics and their descriptors can be discovered and then used to allow Bluetooth Low Energy devices to transfer data

GATT-Based Specifications

The Generic Attributes (GATT) define a hierarchical data structure that is exposed to connected Bluetooth with low energy devices.

- GATT Characteristics
- GATT Declarations
- GATT Descriptors
- GATT Services

GATT Characteristics

Characteristics

- are defined attribute types that contain a single logical value

SpecificationName	SpecificationType	AssignedNumb
Aerobic Heart Rate Upper Limit	org.bluetooth.characteristic.aerobic_heart_rate_upper_limit	0x2A84
Aerobic Threshold	org.bluetooth.characteristic.aerobic_threshold	0x2A7F
Age	org.bluetooth.characteristic.age	0x2A80
Aggregate	org.bluetooth.characteristic.aggregate	0x2A5A
Alert Category ID	org.bluetooth.characteristic.alert_category_id	0x2A43
Alert Category ID Bit Mask	org.bluetooth.characteristic.alert_category_id_bit_mask	0x2A42

GATT Descriptors

Descriptors are defined attributes that describe a characteristic value.

SpecificationName	SpecificationType	AssignedNumbe
Number of Digitals	org.bluetooth.descriptor.number_of_digitals	0x2909
Report Reference	org.bluetooth.descriptor.report_reference	0x2908
Server Characteristic Configuration	org.bluetooth.descriptor.gatt.server_characteristic_configuration	0x2903
Time Trigger Setting	org.bluetooth.descriptor.time_trigger_setting	0x290E
Valid Range	org.bluetooth.descriptor.valid_range	0x2906
Value Trigger Setting	org.bluetooth.descriptor.value_trigger_setting	0x290A

GATT Descriptors

Descriptors are defined attributes that describe a characteristic value.

SpecificationName	SpecificationType	AssignedNumbe
Number of Digitals	org.bluetooth.descriptor.number_of_digitals	0x2909

The Characteristic Number of Digitals descriptor is used for defining the number of digitals in a characteristic.

Value Fields

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
No of Digitals	Mandatory	uint8	N/A	N/A	None

GATT Services

Services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device

SpecificationName	SpecificationType	AssignedNumber
Alert Notification Service	org.bluetooth.service.alert_notification	0x1811
Automation IO	org.bluetooth.service.automation_io	0x1815
Battery Service	org.bluetooth.service.battery_service	0x180F
Blood Pressure	org.bluetooth.service.blood_pressure	0x1810
Body Composition	org.bluetooth.service.body_composition	0x181B
Bond Management	org.bluetooth.service.bond_management	0x181E
Continuous Glucose Monitoring	org.bluetooth.service.continuous_glucose_monitoring	0x181F
Current Time Service	org.bluetooth.service.current_time	0x1805
Cycling Power	org.bluetooth.service.cycling_power	0x1818

Display BLE Services runing getServices.py

(while the nrf51 is running the mbed BLE_Button Example)

```
import sys
from bluepy.btle import UUID, Peripheral

p = Peripheral(sys.argv[1],"random")

services=p.getServices()

#displays all services
for service in services:
   print service
```

Display BLE Services runing getServices.py

(while the nrf51 is running the mbed BLE_Button Example)

```
1   import sys
2   from bluepy.btle import UUID, Peripheral
3
4   p = Peripheral(sys.argv[1],"random")
5
6   services=p.getServices()
7
8   #displays all services
9   for service in services:
10     print service
```

```
Service <uuid=Generic Access handleStart=1 handleEnd=7>
Service <uuid=Generic Attribute handleStart=8 handleEnd=11>
Service <uuid=a000 handleStart=12 handleEnd=65535>
```

Display BLE Services runing getServices.py

(while the nrf51 is running the mbed BLE_Button Example)

SpecificationName	SpecificationType	AssignedNumber
Generic Access	org.bluetooth.service.generic_access	0x1800
Generic Attribute	org.bluetooth.service.generic_attribute	0x1801

Service <uuid=Generic Access handleStart=1 handleEnd=7>

Service <uuid=Generic Attribute handleStart=8 handleEnd=11>

Service <uuid=a000 handleStart=12 handleEnd=65535>

Uuid =a000 is not defined by the bluetooth org
(this is a cusom service / ouer Button Service)

GATT Services are **collections of characteristics** and relationships to other services that encapsulate the behavior of part of a device

Battery Service	org.bluetooth.service.battery_service	0x180F

Overview

Name:

Battery Level

Description:

The Battery Level characteristic is read using the GATT Read Characteristic Value sub-procedure and returns the current battery level as a percentage from 0% to 100%; 0% represents a battery that is fully discharged, 100% represents a battery that is fully charged.

Type:

org.bluetooth.characteristic.battery_level

Requirement:

Mandatory

Properties

Property	Requirement
Read	Mandatory
Write	Excluded
WriteWithoutResponse	Excluded
SignedWrite	Excluded
Notify	Optional
Indicate	Excluded
WritableAuxiliaries	Excluded
Broadcast	Excluded
ExtendedProperties	

Descriptors

Overview

Name:

Characteristic Presentation Format

Type:

org.bluetooth.descriptor.gatt.characteristic_presentation_format

Requirement:

if_multiple_service_instances

Permissions

Permission	Requirement
Read	Mandatory
Write	Excluded

Overview

Name:

Client Characteristic Configuration

Type:

org.bluetooth.descriptor.gatt.client_characteristic_configuration

Requirement:

if_notify_or_indicate_supported

Permissions

Permission	Requirement
Read	Mandatory
Write	Mandatory

GATT Services are **collections of characteristics** and relationships to other services that encapsulate the behavior of part of a device

Battery Service	org.bluetooth.service.battery_service	0x180F

Overview

Name:

Battery Level

Description:

The Battery Level characteristic is read using the GATT Read Characteristic Value sub-procedure and returns the current battery level as a percentage from 0% to 100%; 0% represents a battery that is fully discharged, 100% represents a battery that is fully charged.

Type:

org.bluetooth.characteristic.battery_level

Requirement:

Mandatory

GATT Characteristics

Properties

Property	Requirement
Read	Mandatory
Write	Excluded
WriteWithoutResponse	Excluded
SignedWrite	Excluded
Notify	Optional
Indicate	Excluded
WritableAuxiliaries	Excluded
Broadcast	Excluded
ExtendedProperties	

Descriptors

Overview

Name:

Characteristic Presentation Format

Type:

org.bluetooth.descriptor.gatt.characteristic_presentation_format

Requirement:

if_multiple_service_instances

Permissions

Permission	Requirement
Read	Mandatory
Write	Excluded

Overview

Name:

Client Characteristic Configuration

Type:

org.bluetooth.descriptor.gatt.client_characteristic_configuration

Requirement:

if_notify_or_indicate_supported

Permissions

Permission	Requirement
Read	Mandatory
Write	Mandatory

GATT Characteristics

defined attribute types that contain a single logical value

battery level

Type: org.bluetooth.characteristic.battery_level
Assigned Number: 0x2A19

Abstract:

The current charge level of a battery. 100% represents fully charged while 0% represents fully discharged.

Value Fields

Names	Field Requirement	Format	Minimum Value	Maximum Value	Additional Information
Level **Unit:** org.bluetooth.unit.percentage	Mandatory	uint8	0	100	Enumerations Key: 101 - 255 Value: Reserved

GATT Services are **collections of characteristics** and relationships to other services that encapsulate the behavior of part of a device

Battery Service	org.bluetooth.service.battery_service	0x180F

Overview

Name:

Battery Level

Description:

The Battery Level characteristic is read using the GATT Read Characteristic Value sub-procedure and returns the current battery level as a percentage from 0% to 100%; 0% represents a battery that is fully discharged, 100% represents a battery that is fully charged.

Type:

org.bluetooth.characteristic.battery_level

Requirement:

Mandatory

Properties

Property	Requirement
Read	Mandatory
Write	Excluded
WriteWithoutResponse	Excluded
SignedWrite	Excluded
Notify	Optional
Indicate	Excluded
WritableAuxiliaries	Excluded
Broadcast	Excluded
ExtendedProperties	

Descriptors

Overview

Name:

Characteristic Presentation Format

Type:

org.bluetooth.descriptor.gatt.characteristic_presentation_format

Requirement:

if_multiple_service_instances

Permissions

Permission	Requirement
Read	Mandatory
Write	Excluded

Overview

Name:

Client Characteristic Configuration

Type:

org.bluetooth.descriptor.gatt.client_characteristic_configuration

Requirement:

if_notify_or_indicate_supported

Permissions

Permission	Requirement
Read	Mandatory
Write	Mandatory

GATT Services are **collections of characteristics** and relationships to other services that encapsulate the behavior of part of a device

Battery Service	org.bluetooth.service.battery_service	0x180F

Overview

Name:

Battery Level

Description:

The Battery Level characteristic is read using the GATT Read Characteristic Value sub-procedure and returns the current battery level as a percentage from 0% to 100%; 0% represents a battery that is fully discharged, 100% represents a battery that is fully charged.

Type:

org.bluetooth.characteristic.battery_level

Requirement:

Mandatory

Properties

Property	Requirement
Read	Mandatory
Write	Excluded
WriteWithoutResponse	Excluded
SignedWrite	Excluded
Notify	Optional
Indicate	Excluded
WritableAuxiliaries	Excluded
Broadcast	Excluded
ExtendedProperties	

Descriptors

Overview

Name:

Characteristic Presentation Format

Type:

org.bluetooth.descriptor.gatt.characteristic_presentation_for

Requirement:

if_multiple_service_instances

Permissions

Permission	Requirement
Read	Mandatory
Write	Excluded

Overview

Name:

Client Charact configuration

Type:

etooth.descriptor.gatt.client_char guration

Requirement:

cate_supported

Permissions

Permission	Requirement
Read	Mandatory
Write	Mandatory

GATT Declarations

GATT Services are **collections of characteristics** and relationships to other services that encapsulate the behavior of part of a device

Battery Service	org.bluetooth.service.battery_service	0x180F

Overview

Name:

Battery Level

Description:

The Battery Level characteristic is read using the GATT Read Characteristic Value sub-procedure and returns the current battery level as a percentage from 0% to 100%; 0% represents a battery that is fully discharged, 100% represents a battery that is fully charged.

Type:

org.bluetooth.characteristic.battery_level

Requirement:

Mandatory

Properties

Property	Requirement
Read	Mandatory
Write	Excluded
WriteWithoutResponse	Excluded
SignedWrite	Excluded
Notify	Optional
Indicate	Excluded
WritableAuxiliaries	Excl
Broadcast	ded
ExtendedProperties	

Descriptors

Overview

Name:

Characteristic Presentation Format

Type:

org.bluetooth.descriptor.gatt.characteristic_presentation_format

Requirement:

if_multiple_service_instances

Permissions

Permission	Requirement
Read	Mandatory
Write	Excluded

Name:

Client Characteristic Configuration

Type:

org.bluetooth.descriptor.gatt.client_characteristic_configuration

Requirement:

if_notify_or_indicate_supported

Permissions

Permission	Requirement
Read	Mandatory
Write	Mandatory

GATT Descriptor

Display GATT Descriptors and Delcarations

(while the nrf51 is running the mbed [BLE_Button Example](#))

```
import sys
from bluepy.btle import UUID, Peripheral

p = Peripheral(sys.argv[1],"random")

descriptors=p.getDescriptors(1,0x00F) #Bug if no limt is specified the function wil hang
                        # (go in a endless loop and not return anything)
print("UUID                                    Handle UUID by name")
for descriptor in descriptors:
    print ( " "+ str(descriptor.uuid) + "  0x" + format(descriptor.handle,"02X") +"   "+ str(descriptor) )
```

```python
import sys
from bluepy.btle import UUID, Peripheral

p = Peripheral(sys.argv[1],"random")

descriptors=p.getDescriptors(1,0x00F) #Bug if no limt is specified the function wil hang
                        # (go in a endless loop and not return anything)
print("UUID                                    Handle UUID by name")
for descriptor in descriptors:
    print ( " "+ str(descriptor.uuid) + "  0x" + format(descriptor.handle,"02X") +"   "+ str(descriptor) )
```

```
UUID                                    Handle UUID by name
 00002800-0000-1000-8000-00805f9b34fb  0x01   Descriptor <2800>
 00002803-0000-1000-8000-00805f9b34fb  0x02   Descriptor <2803>
 00002a00-0000-1000-8000-00805f9b34fb  0x03   Descriptor <Device Name>
 00002803-0000-1000-8000-00805f9b34fb  0x04   Descriptor <2803>
 00002a01-0000-1000-8000-00805f9b34fb  0x05   Descriptor <Appearance>
 00002803-0000-1000-8000-00805f9b34fb  0x06   Descriptor <2803>
 00002a04-0000-1000-8000-00805f9b34fb  0x07   Descriptor <Peripheral Preferred Connection Parameters>
 00002800-0000-1000-8000-00805f9b34fb  0x08   Descriptor <2800>
 00002803-0000-1000-8000-00805f9b34fb  0x09   Descriptor <2803>
 00002a05-0000-1000-8000-00805f9b34fb  0x0A   Descriptor <Service Changed>
 00002902-0000-1000-8000-00805f9b34fb  0x0B   Descriptor <Client Characteristic Configuration>
 00002800-0000-1000-8000-00805f9b34fb  0x0C   Descriptor <2800>
 00002803-0000-1000-8000-00805f9b34fb  0x0D   Descriptor <2803>
 0000a001-0000-1000-8000-00805f9b34fb  0x0E   Descriptor <a001>
 00002902-0000-1000-8000-00805f9b34fb  0x0F   Descriptor <Client Characteristic Configuration>
```

Display BLE Services runing getServices.py

(while the nrf51 is running the mbed BLE_Button Example)

```
1   import sys
2   from bluepy.btle import UUID, Peripheral
3
4   p = Peripheral(sys.argv[1],"random")
5
6   services=p.getServices()
7
8   #displays all services
9   for service in services:
10     print service
```

```
Service <uuid=Generic Access handleStart=1 handleEnd=7>
Service <uuid=Generic Attribute handleStart=8 handleEnd=11>
Service <uuid=a000 handleStart=12 handleEnd=65535>
```

Service <uuid=Generic Access handleStart=1 handleEnd=7>

Service <uuid=Generic Attribute handleStart=8 handleEnd=11>

Service <uuid=a000 handleStart=12 handleEnd=65535>

UUID	Handle	UUID by name
00002800-0000-1000-8000-00805f9b34fb	0x01	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x02	Descriptor <2803>
00002a00-0000-1000-8000-00805f9b34fb	0x03	Descriptor <Device Name>
00002803-0000-1000-8000-00805f9b34fb	0x04	Descriptor <2803>
00002a01-0000-1000-8000-00805f9b34fb	0x05	Descriptor <Appearance>
00002803-0000-1000-8000-00805f9b34fb	0x06	Descriptor <2803>
00002a04-0000-1000-8000-00805f9b34fb	0x07	Descriptor <Peripheral Preferred Connection Parameters>
00002800-0000-1000-8000-00805f9b34fb	0x08	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x09	Descriptor <2803>
00002a05-0000-1000-8000-00805f9b34fb	0x0A	Descriptor <Service Changed>
00002902-0000-1000-8000-00805f9b34fb	0x0B	Descriptor <Client Characteristic Configuration>
00002800-0000-1000-8000-00805f9b34fb	0x0C	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x0D	Descriptor <2803>
0000a001-0000-1000-8000-00805f9b34fb	0x0E	Descriptor <a001>
00002902-0000-1000-8000-00805f9b34fb	0x0F	Descriptor <Client Characteristic Configuration>

Handle 1-7

Handle 8-11

Handle 12--

GAT declaration

- GATT Declarations are defined GATT profile read only attribute types
- The GATT Declarations is always sent before the service's Descriptors (in increasing handle order) within the definition and that introduces most of the metadata about the data that follows. All declarations have read-only permissions with no security required, because they cannot contain sensitive data. They are only structural attributes that allow the client to find out and discover the layout and nature of the attributes on the server.

SpecificationName	SpecificationType	AssignedNumber	SpecificationLevel
GATT Characteristic Declaration	org.bluetooth.attribute.gatt.characteristic_declaration	0x2803	Adopted
GATT Include Declaration	org.bluetooth.attribute.gatt.include_declaration	0x2802	Adopted
GATT Primary Service Declaration	org.bluetooth.attribute.gatt.primary_service_declaration	0x2800	Adopted
GATT Secondary Service Declaration	org.bluetooth.attribute.gatt.secondary_service_declaration	0x2801	Adopted

The GATT Declarations is always sent before the service's Descriptors (in increasing handle order) within the definition and that introduces most of the metadata about the data that follows. All declarations have read-only permissions with no security required, because they cannot contain sensitive data. They are only structural attributes that allow the client to find out and discover the layout and nature of the attributes on the server.

SpecificationName	SpecificationType	AssignedNumber	SpecificationLevel
GATT Characteristic Declaration	org.bluetooth.attribute.gatt.characteristic_declaration	0x2803	Adopted
GATT Include Declaration	org.bluetooth.attribute.gatt.include_declaration	0x2802	Adopted
GATT Primary Service Declaration	org.bluetooth.attribute.gatt.primary_service_declaration	0x2800	Adopted
GATT Secondary Service Declaration	org.bluetooth.attribute.gatt.secondary_service_declaration	0x2801	Adopted

UUID	Handle	UUID by name
00002800-0000-1000-8000-00805f9b34fb	0x01	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x02	Descriptor <2803>
00002a00-0000-1000-8000-00805f9b34fb	0x03	Descriptor <Device Name>
00002803-0000-1000-8000-00805f9b34fb	0x04	Descriptor <2803>
00002a01-0000-1000-8000-00805f9b34fb	0x05	Descriptor <Appearance>
00002803-0000-1000-8000-00805f9b34fb	0x06	Descriptor <2803>
00002a04-0000-1000-8000-00805f9b34fb	0x07	Descriptor <Peripheral Preferred Connection Parameters>
00002800-0000-1000-8000-00805f9b34fb	0x08	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x09	Descriptor <2803>
00002a05-0000-1000-8000-00805f9b34fb	0x0A	Descriptor <Service Changed>
00002902-0000-1000-8000-00805f9b34fb	0x0B	Descriptor <Client Characteristic Configuration>
00002800-0000-1000-8000-00805f9b34fb	0x0C	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x0D	Descriptor <2803>
0000a001-0000-1000-8000-00805f9b34fb	0x0E	Descriptor <a001>
00002902-0000-1000-8000-00805f9b34fb	0x0F	Descriptor <Client Characteristic Configuration>

Declarations

Descriptor List widthout the declarations.

UUID	Handle	UUID by name
00002a00-0000-1000-8000-00805f9b34fb	0x03	Descriptor <Device Name>
00002a01-0000-1000-8000-00805f9b34fb	0x05	Descriptor <Appearance>
00002a04-0000-1000-8000-00805f9b34fb	0x07	Descriptor <Peripheral Preferred Connection Parameters>
00002a05-0000-1000-8000-00805f9b34fb	0x0A	Descriptor <Service Changed>
00002902-0000-1000-8000-00805f9b34fb	0x0B	Descriptor <Client Characteristic Configuration>
0000a001-0000-1000-8000-00805f9b34fb	0x0E	Descriptor <a001>
00002902-0000-1000-8000-00805f9b34fb	0x0F	Descriptor <Client Characteristic Configuration>

Reding a descriptor

UUID	Handle UUID by name	
00002a00-0000-1000-8000-00805f9b34fb	0x03	Descriptor <Device Name>

```
import sys
from bluepy.btle import UUID, Peripheral

dev_name_uuid = UUID(0x2A00)

p = Peripheral(sys.argv[1],"random")

ch = p.getCharacteristics(uuid=dev_name_uuid)[0]
    if (ch.supportsRead()):
            print ch.read()
```

```
import sys
from bluepy.btle import UUID, Peripheral

dev_name_uuid = UUID(0x2A00)

p = Peripheral(sys.argv[1],"random")

ch = p.getCharacteristics(uuid=dev_name_uuid)[0]
    if (ch.supportsRead()):
            print ch.read()
```

Expected output when the nRF51-DK is running the BLE_Button Example

NRF5x

The Button Service (UUID =0xA000)

Service <uuid=a000 handleStart=12 handleEnd=65535>

UUID	Handle	UUID by name
00002800-0000-1000-8000-00805f9b34fb	0x0C	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x0D	Descriptor <2803>
0000a001-0000-1000-8000-00805f9b34fb	0x0E	Descriptor <a001>
00002902-0000-1000-8000-00805f9b34fb	0x0F	Descriptor <Client Characteristic Configuration>

Handle 12--

```
from bluepy.btle import UUID, Peripheral

button_service_uuid = UUID(0xA000)
button_char_uuid    = UUID(0xA001)

p = Peripheral(sys.argv[1], "random")
ButtonService=p.getServiceByUUID(button_service_uuid)

ch = ButtonService.getCharacteristics(button_char_uuid)[0]
if (ch.supportsRead()):
    while 1:
        val = binascii.b2a_hex(ch.read())
        print ("0x" + val)
        time.sleep(1)
```

The Button Service (UUID =0xA000)

https://developer.mbed.org/compiler

main.cpp | ButtonService.h

```
class ButtonService {
public:
    const static uint16_t BUTTON_SERVICE_UUID              = 0xA000;
    const static uint16_t BUTTON_STATE_CHARACTERISTIC_UUID = 0xA001;
```

```
from bluepy.btle import UUID, Peripheral

button_service_uuid = UUID(0xA000)
button_char_uuid    = UUID(0xA001)

p = Peripheral(sys.argv[1], "random")
ButtonService=p.getServiceByUUID(button_service_uuid)

ch = ButtonService.getCharacteristics(button_char_uuid)[0]
if (ch.supportsRead()):
    while 1:
        val = binascii.b2a_hex(ch.read())
        print ("0x" + val)
        time.sleep(1)
```

Writing to a LED (writeLed2.py)

while the nrf51 is running the mbed BLE_LED Example)

```
import sys
import binascii
import struct
import time
from bluepy.btle import UUID, Peripheral

led_service_uuid = UUID(0xA000)
led_char_uuid = UUID(0xA001)

p = Peripheral(sys.argv[1], "random")
LedService=p.getServiceByUUID(led_service_uuid)

ch = LedService.getCharacteristics(led_char_uuid)[0]

while 1:
    ch.write(struct.pack('<B', 0x00));
    print ("Led2 on")
    time.sleep(1)
    ch.write(struct.pack('<B', 0x01));
    print ("Led2 off")
    time.sleep(1)
```

Client Characteristic Configuration ?

UUID	Handle	UUID by name
00002a00-0000-1000-8000-00805f9b34fb	0x03	Descriptor <Device Name>
00002a01-0000-1000-8000-00805f9b34fb	0x05	Descriptor <Appearance>
00002a04-0000-1000-8000-00805f9b34fb	0x07	Descriptor <Peripheral Preferred Connection Parameters>
00002a05-0000-1000-8000-00805f9b34fb	0x0A	Descriptor <Service Changed>
00002902-0000-1000-8000-00805f9b34fb	0x0B	Descriptor <Client Characteristic Configuration>
0000a001-0000-1000-8000-00805f9b34fb	0x0E	Descriptor <a001>
00002902-0000-1000-8000-00805f9b34fb	0x0F	Descriptor <Client Characteristic Configuration>

UUID	Handle	UUID by name
00002a05-0000-1000-8000-00805f9b34fb	0x0A	Descriptor <Service Changed>
00002902-0000-1000-8000-00805f9b34fb	0x0B	Descriptor <Client Characteristic Configuration>

SpecificationName	SpecificationType	AssignedNumber
Client Characteristic Configuration	org.bluetooth.descriptor.gatt.client_characteristic_configuration	0x2902

The following Client Characteristic Configuration bits are defined:

Configuration	Value	Description
Notification	0x0001	The Characteristic Value shall be notified. This value can only be set if the characteristic's property has the notify bit set.
Indication	0x0002	The Characteristic Value shall be indicated. This value can only be set if the characteristic's property has the indicate bit set.
Reserved for Future Use	0xFFF4	Reserved for future use.

The default value for the *Client Characteristic Configuration* descriptor value shall be 0x0000.

GATT Services are **collections of characteristics** and relationships to other services that encapsulate the behavior of part of a device

Battery Service	org.bluetooth.service.battery_service	0x180F

Overview

Name:
Battery Level

Type:
org.bluetooth.characteristic.battery_level

Requirement:
Mandatory

Properties

Property	Requirement
WritableAuxiliaries	Excluded
Broadcast	Excluded
ExtendedProperties	

The following Client Characteristic Configuration bits are defined:

Configuration	Value	Description
Notification	0x0001	The Characteristic Value shall be notified.This value can only be set if the characteristic's property has the notify bit set.
Indication	0x0002	The Characteristic Value shall be indicated. This value can only be set if the characteristic's property has the indicate bit set.

Descriptors

Overview

Name:
Characteristic Presentation Format

Type:
org.bluetooth.descriptor.gatt.characteristic_presentation_format

Requirement:
if_multiple_service_instances

Permissions

Permission	Requirement
Read	Mandatory
Write	Excluded

Overview

Name:
Client Characteristic Configuration

Type:
org.bluetooth.descriptor.gatt.client_characteristic_configuration

Requirement:
if_notify_or_indicate_supported

Permissions

Permission	Requirement
Read	Mandatory
Write	Mandatory

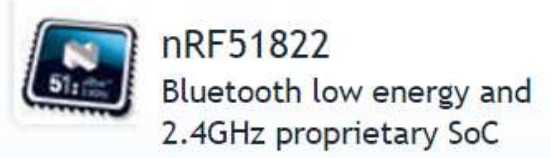
The Button Service

Service <uuid=a000 handleStart=12 handleEnd=65535>

UUID	Handle	UUID by name
00002800-0000-1000-8000-00805f9b34fb	0x0C	Descriptor <2800>
00002803-0000-1000-8000-00805f9b34fb	0x0D	Descriptor <2803>
0000a001-0000-1000-8000-00805f9b34fb	0x0E	Descriptor <a001>
00002902-0000-1000-8000-00805f9b34fb	0x0F	Descriptor <Client Characteristic Configuration>

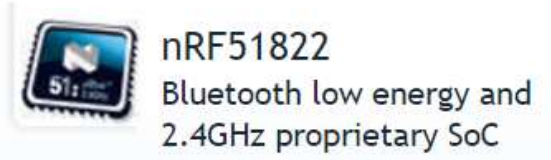
python readButton1Notify.py

```
Button1 Client Characteristic Configuration found at handle
0x0F
Notification is turned on for Button1
Waiting... Waited more than one sec for notification
Notification from Handle: 0x0E Value: 1
Notification from Handle: 0x0E Value: 0
```



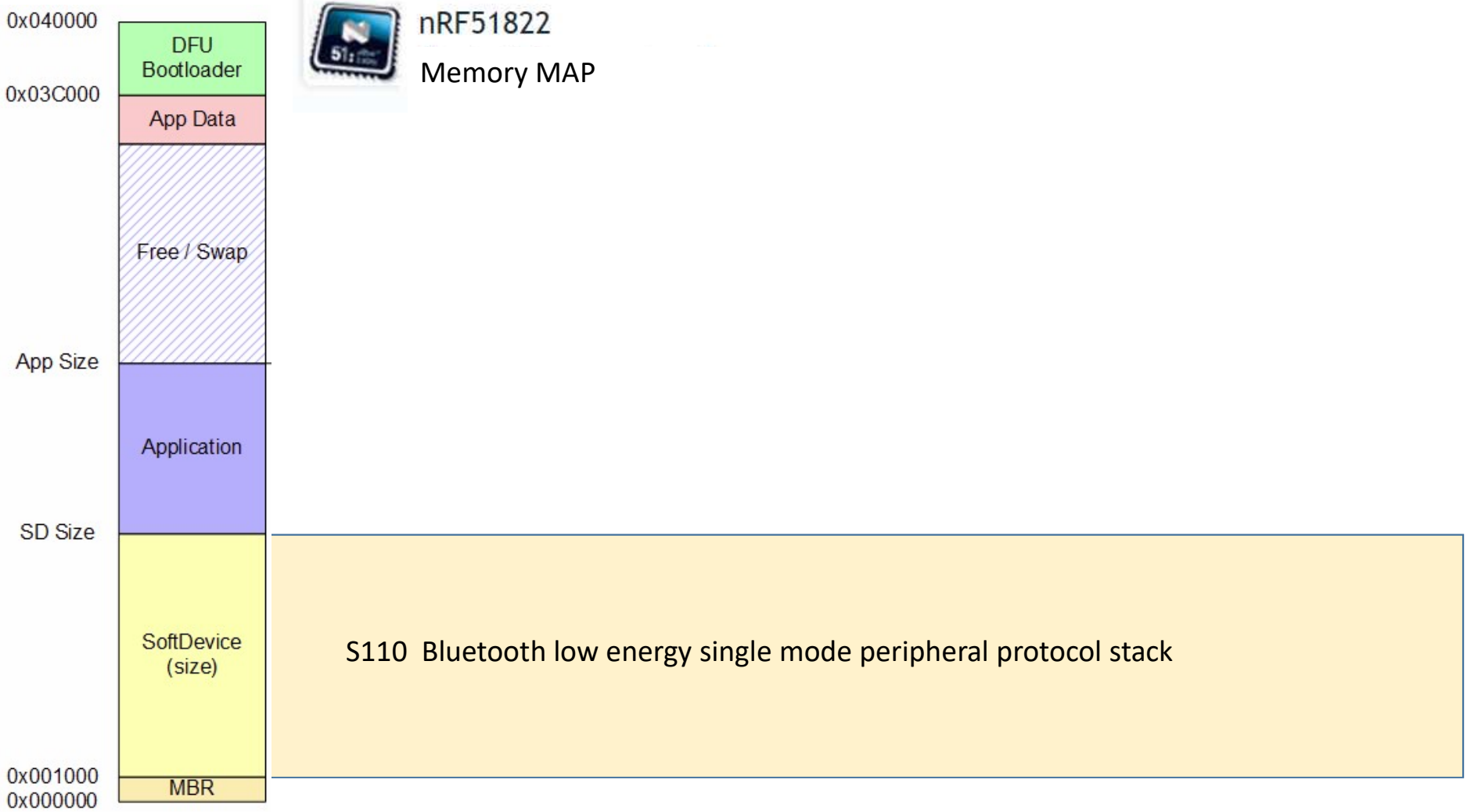
Soft Devices avilable NRF51822

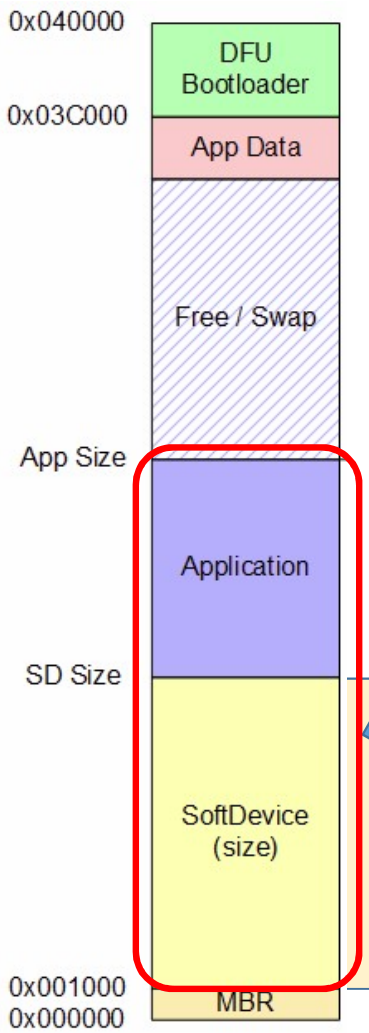
- S110
  - Bluetooth low energy single mode peripheral protocol stack
- S120
  - Bluetooth low energy single mode master protocol stack offering 8 simultaneous connections
- S130
  - Bluetooth low energy protocol stack, supporting multi-link central, peripheral, observer and broadcaster roles



Soft Devices avilable NRF51822

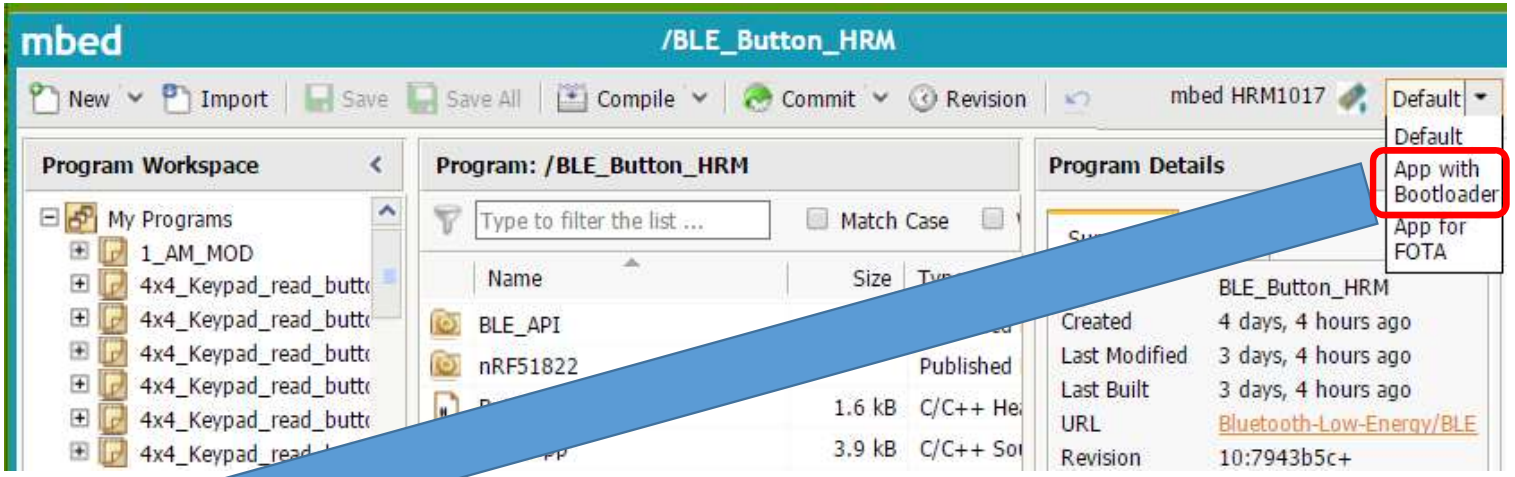
- S110
  - Bluetooth low energy single mode peripheral protocol stack
- S120
  - Bluetooth low energy single mode master protocol stack offering 8 simultaneous connections
- S130
  - Bluetooth low energy protocol stack, supporting multi-link central, peripheral, observer and broadcaster roles


nRF51822
Memory MAP
0x040000
0x03C000
DFU Bootloader
App Data
Free / Swap
App Size
Application
SD Size
SoftDevice (size)
0x001000
0x000000
MBR
S110 Bluetooth low energy single mode peripheral protocol stack



nRF51822

Memory MAP



S110 Bluetooth low energy single mode peripheral protocol stack