

ibeacon Parameter Setting for iOS

Battery Level Inquiry: To get present battery percentage

Connection first. After connection, Beacon will search for its services. One of these service is UUID= 00002a19-0000-1000-8000-00805f9b34fb, its characteristic BluetoothGattCharacteristic can be automatically read, and this value is battery percentage. The value is 1byte, and change byte[0]&0xFF into int.

Broadcasting Interval Setting: From 0 to 10000

Obtain Default Value: After connection the device will start pairing automatically, when it is paired, it will search for its services. One of these services is UUID=0000f356-0000-1000-8000-00805f9b34fb, its characteristic BluetoothGattCharacteristic can be automatically read, and this value is broadcasting interval.

The value is 2bytes. Analytical methods: To convert into a hexadecimal string first, example as follows:

```
public static String bytesToHexString(byte[] src) {
    StringBuilder stringBuilder = new StringBuilder("");
    if (src == null || src.length <= 0) {
        return null;
    }
    for (int i = 0; i < src.length; i++) {
        int v = src[i] & 0xFF;
        String hv = Integer.toHexString(v);
        if (hv.length() < 2) {
            stringBuilder.append(0);
        }
        stringBuilder.append(hv);
        stringBuilder.append(" ");
    }
    return stringBuilder.toString();
}
```

Then, plus the two bytes(val is Hexadecimal string after conversion):

```
String s[] = val.split(" ");
String vs = s[0]+s[1];
```

Then, convert vs into decimal digits:

```
public static int hexStringToAlgorism(String hex) {
    hex = hex.toUpperCase();
    int max = hex.length();
    int result = 0;
    for (int i = max; i > 0; i--) {
```

```

        char c = hex.charAt(i - 1);
        int algorism = 0;
        if (c >= '0' && c <= '9') {
            algorism = c - '0';
        } else {
            algorism = c - 55;
        }
        result += Math.pow(16, max - i) * algorism;
    }
    return result;
}

```

The return result is broadcasting interval. If you want to modify this broadcasting interval, the app should be paired with the device.

Set a new broadcasting interval: To convert into a hexadecimal string first(from 0 to 10000):

```

public static String algorismToHEXString(int algorism) {
    String result = "";
    result = Integer.toHexString(algorism);
    if (result.length() % 2 == 1) {
        result = "0" + result;
    }
    result = result.toUpperCase();
    return result;
}

```

Then convert it into byte array as the value of BluetoothGattCharacteristic:

```

public static byte[] hexStrToStr(String hexStr) {
    String str = "0123456789ABCDEF";
    char[] hexs = hexStr.toCharArray();
    byte[] bytes = new byte[hexStr.length() / 2];
    int n;
    for (int i = 0; i < bytes.length; i++) {
        n = str.indexOf(hexs[2 * i]) * 16;
        n += str.indexOf(hexs[2 * i + 1]);
        bytes[i] = (byte) (n & 0xff);
    }
    return bytes;
}

```

If the new broadcasting interval value is less than 256, then the first content in the byte array shall be 0. The byte array is 2 bytes.

Send the correct data under BluetoothGattCharacteristic to the module, you can modify the broadcasting intervals.

After the data sent, it will return value of `UUID=0000f3ff-0000-1000-8000-00805f9b34fb` under BluetoothGattCharacteristic. It is 3 bytes. If the last byte is 2, that means

broadcasting interval is successfully set; if it is 1, means the set is failed.

Tx Transmit Power Parameter Settings: from 0 to 3

Obtain Default Value: Connection first. After connection, Beacon will search for its services. One of these service is UUID=0000f355-0000-1000-8000-00805f9b34fb, its characteristic BluetoothGattCharacteristic can be automatically read, and this value is Tx transmit power.

The value is 1byte, and change byte[0]&0xFF into int.

Set a new value: from 0 to 3, the array is 1byte. It can be modified after paired.

Convert the data into byte, and store it to array as value of BluetoothGattCharacteristic, send BluetoothGattCharacteristic to the module, you can modify Tx transmit power.

After the data sent, it will return value of UUID=0000f3ff-0000-1000-8000-00805f9b34fb under BluetoothGattCharacteristic. It is 3 bytes. If the last byte is 2, that means broadcasting interval is successfully set; if it is 1, means the set is failed.

Beacon UUID: 36 bytes (1234567890abcdef-Fill in the original data format)

UUID Example eg: E2C56DB5-DFFB-48D2-B060-D0F5A71096E0

Obtain default value: Connection first. After connection, Beacon will search for its services. One of these service is UUID= 0000f351-0000-1000-8000-00805f9b34fb, its characteristic BluetoothGattCharacteristic can be automatically read, and this value is Beacon UUID.

The return result is array of byte, convert it into string via bytesToHexString(), you can get the actual setting UUID.

Format after conversion example:E2 C5 6D B5 DF FB 48 D2 B0 60 D0 F5 A7 10 96 E0

Convert your interception characters into UUID sample format

Set A New UUID: 36 bytes (1234567890abcdef-) It can be set after paired. Interception characters shall be in string, example: E2 C5 6D B5 DF FB 48 D2 B0 60 D0 F5 A7 10 96 E0.

Covert the string into byte array via hexStrToStr(String str), and the result is the value of BluetoothGattCharacteristic. Send it to the module, the UUID can be modified.

Afterwards, it will return a value of BluetoothGattCharacteristic of UUID=0000f3ff-0000-1000-8000-00805f9b34fb, it is 3 bytes, the last byte is is 2, that means broadcasting interval is successfully set; if it is 1, means the set is failed.

Beacon Major Setting: from 0 to 65535

Obtain default value: Connection first. After connection, Beacon will search for its services.

One of these service is `UUID=0000f352-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is Beacon Major.

The value is 2bytes. Analytical methods: To convert into a hexadecimal string via `bytesToHexString()`, and then, plus the two bytes(val is Hexadecimal string after conversion):

```
String s[] = val.split(" ");
String vs = s[0]+s[1];
```

Then convert string vs into decimal digits via `hexStringToAlgorism(vs)`, the result is actual Major.

Set a new value: from 0 to 65535. It can be modified after paired.

First turn data into digital format, and convert it into a hexadecimal string `algorismToHEXString(value)`, and then convert it into byte array via `hexStrToStr(String value)`. The result is the value of `BluetoothGattCharacteristic`.

If the modifying value is less than 256, then the first content in the byte array shall be 0. The byte array is 2 bytes.

Send the correct data under `BluetoothGattCharacteristic` to the module, tha major can be modified.

Afterwards, it will return a value of `BluetoothGattCharacteristic` of `UUID=0000f3ff-0000-1000-8000-00805f9b34f`, it is 3 bytes, the last byte is is 2, that means broadcastng interval is successfully set; if it is 1, means the set is failed.

Beacon Minor Setting: from 0 to 65535

Obtain default value: Connection first. After connection, Beacon will search for its services. One of these service is `UUID= 0000f353-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is Beacon Minor.

The value is 2bytes. Analytical methods: To convert into a hexadecimal string via `bytesToHexString()`, and then, plus the two bytes(val is Hexadecimal string after conversion):

```
String s[] = val.split(" ");
String vs = s[0]+s[1];
```

Then convert string vs into decimal digits via `hexStringToAlgorism(vs)`, the result is actual Minor.

Set a new value: from 0 to 65535. It can be modified after paired.

First turn data into digital format, and convert it into a hexadecimal string `algorismToHEXString(value)`, and then convert it into byte array via `hexStrToStr(String value)`. The result is the value of `BluetoothGattCharacteristic`.

If the modifying value is less than 256, then the first content in the byte array shall be 0. The byte array is 2 bytes.

Send the correct data under `BluetoothGattCharacteristic` to the module, tha major can be modified.

Afterwards, it will return a value of `BluetoothGattCharacteristic` of `UUID=0000f3ff-0000-1000-8000-00805f9b34fb`, it is 3 bytes, the last byte is 2, that means broadcasting interval is successfully set; if it is 1, means the set is failed.

Tx power: from -127 to 0

Obtain default value: Connection first. After connection, Beacon will search for its services. One of these service is `UUID= 0000f354-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is Beacon Tx power.

The value is 1byte, and change `byte[0]` into `int`, value can be -127 to 0.

Set a new value: from -127 to 0. It can be modified after paired.

Convert the digital format into bytes as the value of `BluetoothGattCharacteristic`, send it to the module, the Tx power can be modified.

Afterwards, it will return a value of `BluetoothGattCharacteristic` of `UUID=0000f3ff-0000-1000-8000-00805f9b34fb`, it is 3 bytes, the last byte is 2, that means broadcasting interval is successfully set; if it is 1, means the set is failed.

Beacon Display Name: 1-20 characters(English alphabet, Digital)

Obtain default value: Connection first. After connection, Beacon will search for its services. One of these service is `UUID= 0000f35b-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is Beacon Display name.

The value got in byte array, convert it into string as following. The string is the actual Beacon Display name.

```
public String readcharacteristicbyte(byte[] by) {
    String value = "";
    if (by != null) {
        //convert into string
        for (int i = 0; i < by.length; i++) {
            byte b = by[i];
            char cha = (char)b;
            value = value + cha;
        }
    }
    return value;
}
```

Set a new value: 1-20 characters(English alphabet, Digital). It can be modified after paired.

Convert the string into byte array via `getBytes()` as the value of `BluetoothGattCharacteristic`, send it to the module, the display name can be modified.

Afterwards, it will return a value of `BluetoothGattCharacteristic` of `UUID=0000f3ff-0000-1000-8000-00805f9b34fb`, it is 3 bytes, the last byte is 2

2, that means broadcasting interval is successfully set; if it is 1, means the set is failed.

Pairing Code: six figures of digitals (from 000000 to 999999)

Obtain default value: Connection first. After connection, Beacon will search for its services. One of these service is UUID= 0000f357-0000-1000-8000-00805f9b34fb, its characteristic BluetoothGattCharacteristic can be automatically read, and this value is pairing code.

The return value is byte array, convert it into **long** type digitals via bytesToint (by, 0) :

```
/**
 * convert a 4byte array into a 32 figures long
 * by: byte array to be converted
 * pos location in byte array when conversion Start, high order first
 * return digits after conversion
 */
public static long bytesToint(byte[] by, int pos) {
    int ab = 0;
    int bbb=0;
    int cb =0;
    int db=0;
    int index =pos;
    ab = (0x000000FF&((int)by[index]));
    bbb = (0x000000FF&((int)by[index+1]));
    cb = (0x000000FF&((int)by[index+2]));
    db = (0x000000FF&((int)by[index+3]));
    index = index +4;
    long s = ((long) (ab<<24|bbb<<16|cb<<8|db)) &0xFFFFFFFFL;
    return s;
}
```

If the long type digital after conversion is less than six figures, then add the number 0 (zero) in front to make up to six-digit. This six-digit is the actual pairing code.

Set a new value: six-digit (from 000000 to 999999)

Convert the data to digit format before modification, and then convert it into byte array via intTobytes(num).

```
/**
 * convert a 32 figures integer into a 4byte array, high order first
 * nm integer
 */
public static byte[] intTobytes(int nm) {
    byte bb[] = new byte[4];
    for (int i = 0; i < 4; i++) {
        int of = (bb.length-1-i)*8;
```

```

        bb[i] = (byte) ((nm>>>of) &0xff);
    }
    return bb;
}

```

The return value as the value of `BluetoothGattCharacteristic`, and send it to the module, the pairing code can be modified.

Afterwards, it will return a value of `BluetoothGattCharacteristic` of `UUID=0000f3ff-0000-1000-8000-00805f9b34fb`, it is 3 bytes, the last byte is 2, that means broadcasting interval is successfully set; if it is 1, means the set is failed.

system ID(Display system parameters): unprogrammable

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a23-0000-1000-8000-00805f9b34fb`, its `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte(by)`. The string after conversion is system ID, and it may be null.

Model Number(Display system parameters): unprogrammable, Value example: *Model Number*

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a24-0000-1000-8000-00805f9b34fb`, its `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte(by)`. The string after conversion is Model Number, and it may be null or *Model Number*.

Serial Number(Display system parameters): unprogrammable, Value example: Serial Number

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a25-0000-1000-8000-00805f9b34fb`, its `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte(by)`. The string after conversion is Serial Number, and it may be null or *Serial Number*.

FW rev(Display system parameters): unprogrammable, Value example: 1.3.1a

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a26-0000-1000-8000-00805f9b34fb`, its `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte(by)`. The string after conversion is FW rev, and it may be null.

HW rev(Display system parameters): unprogrammable, Value example: 1.0

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a27-0000-1000-8000-00805f9b34fb`, its `BluetoothGattCharacteristic` can be automatically read, and this value is byte array.

Convert it into string via `readcharacteristiczhbyte (by)`. The string after conversion is HW rev, and it may be null.

SW rev(Display system parameters): unprogrammable, Value example: Software Revision

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a28-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte (by)`. The string after conversion is SW rev, and it may be null.

Manufacturer Name(Display system parameters): unprogrammable, Value example: Linlinqi Studio

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a29-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte (by)`. The string after conversion is Manufacturer Name, and it may be null.

IEEE(Display system parameters): unprogrammable, Value example: ?experimental

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a2a-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte (by)`. The string after conversion is IEEE, and it may be null.

PnP ID(Display system parameters): unprogrammable

Connection first. After connection, Beacon will search for its services. One of these service is `UUID=00002a50-0000-1000-8000-00805f9b34fb`, its characteristic `BluetoothGattCharacteristic` can be automatically read, and this value is byte array. Convert it into string via `readcharacteristiczhbyte (by)`. The string after conversion is PnP ID, and it may be null.