bleak$_{sigspec}$

Release 0.0.4

Oct 12, 2020

Contents

Bleak SIG Bluetooth Characteristic Specification Formatter

This package enables characteristic metadata parsing and automatic formatting (bytes unpacking) into the proper characteristic values.

- Free software: MIT license
- Documentation: https://bleak_sigspec.readthedocs.io.

CHAPTER 1

Features

- xml SIG Bluetooth GATT Characteristics parser
- Automatic formatting based on xml metadata (from bytes to dictionary data type)

CHAPTER 2

Installation

Install `bleak_sigspec` by running:

```
$ pip install bleak_sigspec
```

Or get latest development version:

```
$ pip install https://github.com/Carglglz/bleak_sigspec.git
```

CHAPTER 3

Usage

Example: `service_explorer.py` in bleak examples:

```
from bleak_sigspec.utils import get_char_value
[...]
37
   bytes_value = bytes(await client.read_gatt_char(char.uuid))
   formatted_value = get_char_value(bytes_value, char)
[...]
43
   log.info(
     "Characteristic Name: {0}, Bytes Value: {1}, Formatted
     Value: {2}".format(char.description, bytes_value, formatted_value))
```

Output

```
$ python3 service_explorer.py
[...]
Characteristic Name: Temperature, Bytes Value: b'Z\x16', Formatted Value: {
↪'Temperature': {'Quantity': 'thermodynamic temperature',
'Unit': 'degree celsius',
'Symbol': '°C',
'Value': 57.22}}
```

Example: See characteristic metadata

```
>>> from bleak_sigspec.utils import get_xml_char
>>> temp = get_xml_char('Temperature')
>>> temp
Characteristic Metadata:

- NAME: Temperature
- UUID: 2A6E
- ABSTRACT: None
- SUMMARY: None
```

(continues on next page)

(continued from previous page)

```
- FIELDS:
    - Temperature:
        - InformativeText: Unit is in degrees Celsius with a resolution of 0.01␣
↪degrees Celsius
        - Requirement: Mandatory
        - Format: sint16
        - Ctype: h
        - Unit_id: org.bluetooth.unit.thermodynamic_temperature.degree_celsius
        - Quantity: thermodynamic temperature
        - Unit: degree celsius
        - Symbol: °C
        - DecimalExponent: -2
- TYPE: org.bluetooth.characteristic.temperature
- INFO TEXT: Unit is in degrees Celsius with a resolution of 0.01 degrees Celsius
- DESCRIPTION: None
- NOTE: None


>>>
```

CHAPTER 4

Contribute

- Issue Tracker.
- Source Code.

CHAPTER 5

Contents

5.1 How it works

`get_char_value` function uses the characteristic's name, parsing the respective xml file to get all the value fields. From there it reads the Flags field (if exists) and unpack the bytes accordingly. After that it performs any post-proccesing operation required (e.g. DecimalExponent, Multiplier etc). Finally it packs the result in a dictionary.

To see more about bytes packing/unpacking in python see: struct.

5.2 Limitations

`get_char_value` may fail, or return unexpected results due to the following reasons:

- The characteristic does not exist in `bleak` or the xml file in `bleak_sigspec` is missing
- There is a bug in the xml file.
- The format of the characteristic is not supported (see: struct.)
- There is a bug in `get_char_value`
- The data is not formatted following the xml file description.

5.3 Future Work

5.3.1 Vendor specific or custom characteristics

To format a characteristic value of a Vendor specific or custom characteristic (not defined in GATT specifications) there is a *Characteristic Presentation Format* Descriptor which defines the format of the Characteristic Value. So if this Descriptor is present it is possible to read it and get the characteristic value format, exponent, unit, name space and description.

To add custom characteristics it is possible to write a xml file describing the characteristic and provide a 128 UUID. Then this could be registered in `bleak/bleak_sigspec`.

Added in bleak_sigspec 0.0.4:

- Nordic UART TX, Nordic UART RX characteristics.

5.4 API

5.4.1 bleak_sigspec.formatter

Utils to work with binary data or bytes

exception bleak_sigspec.formatter.**FormatIncompleteError**(**args*)
Bases: Exception

class bleak_sigspec.formatter.**SuperStruct**
Bases: object

Struct class Bluetooth SIG compliant

calcsize(*fmt*)
Return the size in bytes of a string format, same as `struct.calcsize`

pack(*fmt*, **args*)
Pack values (*args*) into bytes following the specified format (fmt)

unpack(*fmt*, *bb*)
Unpack values from bytes(bb) following the specified format (fmt)

bleak_sigspec.formatter.**decode_2_uint12**(*bb*)
Decode 3 bytes as two unsigned 12 bit integers

Specs: 2_uint12 len: 3 bytes

Format string: 'o'

bleak_sigspec.formatter.**decode_FLOAT_ieee11073**(*value*)
Defined in ISO/IEEE Std. 11073-20601TM-2008:

FLOAT-Type is defined as a 32-bit value with a 24-bit mantissa and an 8-bit exponent.

Special Values:

- +INFINITY : [exponent 0, mantissa +(2^23 –2) → 0x007FFFFE]
- NaN *(Not a Number)*: [exponent 0, mantissa +(2^23 –1) → 0x007FFFFF]
- NRes *(Not at this Resolution)*: [exponent 0, mantissa –(2^23) → 0x00800000]
- Reserved for future use : [exponent 0, mantissa –(2^23–1) → 0x00800001]
- – INFINITY : [exponent 0, mantissa –(2^23 –2) → 0x00800002]

bleak_sigspec.formatter.**decode_SFLOAT_ieee11073**(*value*)
Defined in ISO/IEEE Std. 11073-20601TM-2008:

SFLOAT-Type is defined as a 16-bit value with 12-bit mantissa and 4-bit exponent. The 16–bit value contains a 4-bit exponent to base 10, followed by a 12-bit mantissa. Each is in twos- complement form.

Special Values:

- +INFINITY : [exponent 0, mantissa +(2^11 –2) → 0x07FE]
- NaN *(Not a Number)*: [exponent 0, mantissa +(2^11 –1) → 0x07FF]
- NRes *(Not at this Resolution)*: [exponent 0, mantissa –(2^11) → 0x0800]
- Reserved for future use: [exponent 0, mantissa –(2^11 –1) → 0x0801]
- – INFINITY : [exponent 0, mantissa –(2^11 –2) → 0x0802]

bleak_sigspec.formatter.**decode_nibbles**(*bb*)
Decode 1 byte as two nibbles (ints)

Specs: **bb_len** : 1

returns: (int, int)

bleak_sigspec.formatter.**decode_sint24**(*bb*)
Decode 3 bytes as a signed 24 bit integer

Specs:

- **sint24 len**: 3 bytes
- **Format string**: 'K'

bleak_sigspec.formatter.**decode_uint128**(*bb*)
Decode 16 bytes as a unsigned 128 bit integer

Specs:

- **uint128 len**: 16 bytes
- **Format string**: 'z'

bleak_sigspec.formatter.**decode_uint24**(*bb*)
Decode 3 bytes as a unsigned 24 bit integer

Specs:

- **uint24 len**: 3 bytes
- **Format string**: 'k'

bleak_sigspec.formatter.**decode_uint40**(*bb*)
Decode 5 bytes as an unsigned 40 bit integer

Specs:

- **uint40 len**: 5 bytes
- **Format string**: 'j'

bleak_sigspec.formatter.**decode_uint48**(*bb*)
Decode 6 bytes as an unsigned 48 bit integer

Specs:

- **uint48 len**: 6 bytes
- **Format string**: 'J'

bleak_sigspec.formatter.**encode_2_uint12**(*val*, *val2*)
Format two values as two unsigned 12 bit integers

Specs: **2_uint12 len**: 3 bytes

Format string: 'o'

bleak_sigspec.formatter.**encode_FLOAT_ieee11073**(*value*, *precision=1*, *debug=False*)
Binary representation of float value as IEEE-11073:20601 32-bit FLOAT

FLOAT-Type is defined as a 32-bit value with a 24-bit mantissa and an 8-bit exponent.

- https://community.hiveeyes.org/t/implementing-ble-gatt-ess-characteristics-with-micropython/2413/3

bleak_sigspec.formatter.**encode_SFLOAT_ieee11073**(*value*, *precision=1*, *debug=False*)
Binary representation of float value as ISO/IEEE Std. 11073-20601TM-2008: 16-Bit FLOAT

The SFLOAT-Type is defined as a 16-bit value with 12-bit mantissa and 4-bit exponent

bleak_sigspec.formatter.**encode_nibbles**(*val*, *val2*)
Encode two values as two nibbles in a byte

Specs:

- **Nibble**: MSN LSN
- **Byte**: 0b0000 0000
- **Indexes**: 7654 3210
- **Values**: val2 val

Requirement:

- Only values (0-15) allowed

bleak_sigspec.formatter.**encode_sint24**(*val*)
Format a value as a signed 24 bit integer

Specs:

- **sint24 len**: 3 bytes
- **Format string**: 'K'

bleak_sigspec.formatter.**encode_uint128**(*val*)
Format a value as a unsigned 128 bit integer

Specs:

- **uint128 len**: 16 bytes
- **Format string**: 'z'

bleak_sigspec.formatter.**encode_uint24**(*val*)
Format a value as a unsigned 24 bit integer

Specs:

- **uint24 len**: 3 bytes
- **Format string**: 'k'

bleak_sigspec.formatter.**encode_uint40**(*val*)
Format a value as an unsigned 40 bit integer

Specs:

- **uint40 len**: 5 bytes
- **Format string**: 'j'

bleak_sigspec.formatter.**encode_uint48**(*val*)
Format a value as an unsigned 48 bit integer

Specs:

- **uint48 len**: 6 bytes
- **Format string**: 'J'

`bleak_sigspec.formatter.`**`twos_comp`**(*val*, *bits*)
returns the 2's complement of int value val with n bits

- https://stackoverflow.com/questions/1604464/twos-complement-in-python

`bleak_sigspec.formatter.`**`twos_comp_dec`**(*val*, *bits*)
returns the signed int value from the 2's complement val with n bits

- https://stackoverflow.com/questions/1604464/twos-complement-in-python

5.4.2 bleak_sigspec.utils

Utils to decode binary data from SIG Characteristics

`class` `bleak_sigspec.utils.`**`CHAR_XML`**(*xml_file*, *path='/home/docs/checkouts/readthedocs.org/user_builds/bleak-sigspec/envs/stable/lib/python3.7/site-packages/bleak_sigspec-0.0.4-py3.7.egg/bleak_sigspec/characteristics_xml'*)
Bases: `object`

Parse characteristic xml file

`bleak_sigspec.utils.`**`dict_char_value`**(*data*, *raw=False*)
Simplify the characteristic value in dict format

`bleak_sigspec.utils.`**`get_char_value`**(*value: bytes, characteristic: Union[bleak.backends.characteristic.BleakGATTCharacteristic, str, bleak_sigspec.utils.CHAR_XML], rtn_flags: bool = False, debug: bool = False*) → dict
Given a characteristic and its raw value in bytes, obtain the formatted value as a dict instance:

Args:

- **value (bytes)**:

  The result of `read_gatt_char()`

- **characteristic** *(BleakGATTCharacteristic, str, CHAR_XML)*:

  The characteristic from which get metadata.

- **rnt_flags**:

  return the bitflags too if present

- **debug**:

  print debug information about bytes unpacking

Returns:

- **dict**:

  dict instance with the formatted value and its metadata.

`bleak_sigspec.utils.`**`get_plain_format`**(*field*)
Iterates until the last level where Value is

`bleak_sigspec.utils.`**`get_ref_char_field`**(*_field*, *name_field*)
Get characteristics field references recursively

bleak_sigspec.utils.**get_xml_char**(*characteristic: Union[str, bleak.backends.characteristic.BleakGATTCharacteristic]*) → bleak_sigspec.utils.CHAR_XML

Get characteristic metadata from its xml file

Args:

- **characteristic** *(str, BleakGATTCharacteristic)*:

  The name of the characteristic or bleak characteristic class

Returns:

- **characteristic metatada class** *(CHAR_XML)*:

  The characteristic metadata parsed from its xml file

bleak_sigspec.utils.**map_char_value**(*data*, *keys=[]*, *string_fmt=False*, *one_line=True*, *sep=', '*)

Map characteristic value with the given keys, return dict or string format

bleak_sigspec.utils.**pformat_char_flags**(*data*, *sep='\n'*, *prnt=False*, *rtn=True*)

Print or return the characteristic flag in string format

bleak_sigspec.utils.**pformat_char_value**(*data*, *char=''*, *only_val=False*, *one_line=False*, *sep=', '*, *custom=None*, *symbols=True*, *prnt=True*, *rtn=False*)

Print or return the characteristic value in string format

bleak_sigspec.utils.**pformat_field_value**(*field_data*, *field=''*, *sep=', '*, *prnt=True*, *rtn=False*)

Print or return the field value in string format

bleak_sigspec.utils.**pformat_ref_char_value**(*char_value*)

Print or return the characteristic value in string format

CHAPTER 6

License

The project is licensed under the MIT license.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

b

Index

S

T

U