Objenious
Codec definition

Version 1.1.1

Objenious
by Bouygues Telecom

1 Introduction

This document describes the configuration of codecs on the Objenious platform.

Codecs decode binary payloads to JSON objects. 3 types of codecs are currently implemented:

- Generic codec,
- NKE Batch, for NKE devices using batch mode,
- Senlab, for Sensing Lab devices using batch mode.

2 Generic codec

The codec definition has two sections:

- "defaults": this sections defines a list of global settings,
- "attributes": this section lists all possible attributes and their type,
- "format": this section defines the way the attributes are laid out.

Example codec:

```
{
    "defaults": {
       "endian": "little"
    },
    "attributes": {
      "id": {
          "type": "int",
          "hidden":true,
          "length": 8
      },
      "battery_level": {
          "type": "uint",
          "unit": 255,
          "length": 8
      },
      "internal": {
          "type": "int",
          "hidden":true,
          "length": -16
      },
      "temperature": {
          "type": "int",
          "length": 16,
          "divide": 16
      }
    },
    "format": [
      {
          "attributes": ["id"]
      },
      {
        "if": "id == 1",
```

```
        "then": [
            {
                "attributes": ["battery_level", "internal", "temperature"]
            }
        ]
      }
    ]
}
```

2.1 Defaults

The defaults section lists the default settings to be used for all attributes:

- "endian": order of bytes - "big" (default) or "little"
- "order": order of bits - "msb" (default) or "lsb",
- "negative": representation of negative numbers – "2_complement (default) or "sign_magnitude".

Those defaults can be overridden in specific attributes.

2.2 Attributes

An attribute has the following properties:

- "type": an attribute can have the following types: "int" (integer, big endian, signed using magnitude representation - 1 bit for the sign + N bits for the absolute value), "uint" (unsigned integer, big endian), "float" (IEEE 754), "bool" (1 bit boolean values, true = 1/false = 0) and "char" (ASCII 7 bits string),
- "length": number of bits,
- "variable": the attribute has a variable length, the first "length" contains the number of bytes of the attribute,
- "multiply": the decoded value will be multiplied by the value to get the final value,
- "divide": the decoded value will be divided by the value to get the final value,
- "hidden": if set to "true", the attribute will not be part of the decoded object,
- "endian": (see before)
- "order": (see before),
- "negative": (see before),
- "attributes": the attribute includes a list of other.

Example 1: A 16 bits integer, 2-complement negative numbers. A "250" value will be decoded as "15.625" (250/16).

```
"temperature": {
   "type": "int",
   "length": 16,
   "divide": 16
}
```

Example 2: A 32 bits integer including other attributes – endianness will be applied on the container attribute before parsing attribute1 and attribute2.

```
 "container": {
   "type": "uint",
   "hidden": true,
   "length": 32,
   "attributes": ["attribute1", "attribute2"]
},
"attribute1": {
```

```
        "type": "uint",
        "length": 4
    },
    "attribute2": {
        "type": "uint",
        "length": 3
    }
```

Example 3: A string of variable length: first byte contains the number of chars of the following string.

```
    "string": {
        "type": "char",
        "length": 8,
        "variable": true
    }
```

2.3 Format

Format is defined as a list of parts. A part can either be:

- "attributes": a list of attributes,
- "if/then": a list of parts, based on a condition.
    - Conditions use the following operators: "==" (equals to), "!=" (different from), ">", ">=" (greater than), "<", "<=" (less than), "&&" (logical and) and "||" (logical or). Spaces have to be present before/after the operator.
    - Condition can test:
        - previous values (hidden or not),
        - protocol values (e.g. port for LoRa devices).

Example 1: temperature_present is defined as a boolean

```
{
    "if": "temperature_present",
    "then": [
        {
            "attributes": ["Temperature"]
        }
    ]
}
```

Example 1: temperature_present is defined as a boolean

```
{
    "if": "temperature_present",
    "then": [
        {
            "attributes": ["Temperature"]
        }
    ]
}
```

Example 2:

```
{
  "if": "command_id == 1 && cluster_id == 1026 && attribute_id == 1",
    "then": [
```

```
            {
                "attributes": ["TemperatureMin"]
            }
        ]
    }
```

3 NKE Batch

Using batch mode, NKE devices can report multiple data points, either from a single attribute or from multiple attributes, at multiple times.

The codec needs to be configured in a similar way as the *br_uncompress* tool provided by NKE.

The configuration has 2 properties:

- "tag_size": the size of tags (e.g. 1)
- "measures": the list of attributes. Each attribute has the following properties:
  - "attribute": the name of the attribute,
  - "type": the type of the values (e.g. 7),
  - "divide": (see before).

Example:

```
{
    "tag_size": 1,
    "measures": [
        {
            "attribute": "temperature",
            "type": 7,
            "divide": 100
        }
    ]
}
```

4 Senlab

Using batch mode, SensingLab devices can report multiple data points. This codec only decodes complex payloads reporting multiple data points. Single payloads are decoded using a generic decoder configuration.

The configuration has 2 properties:

- "device_type", e.g. SenlabM, SenlabT, SenlabH...
- "device_version", e.g. 1.1

Example:

```
{
    "device_type": "SenlabT",
    "device_version": "1.1"
}
```

5 Changes

5.1 Version 1.1.0

- Add "defaults" section, remove "transform"
- Use "2 complement" instead of "sign-magnitude" as default
- Rename "unit" to "divide" and add "multiply"

5.2 Version 1.1.1

- Rename the incorrect attribute name "endianness" to the correct "endian"