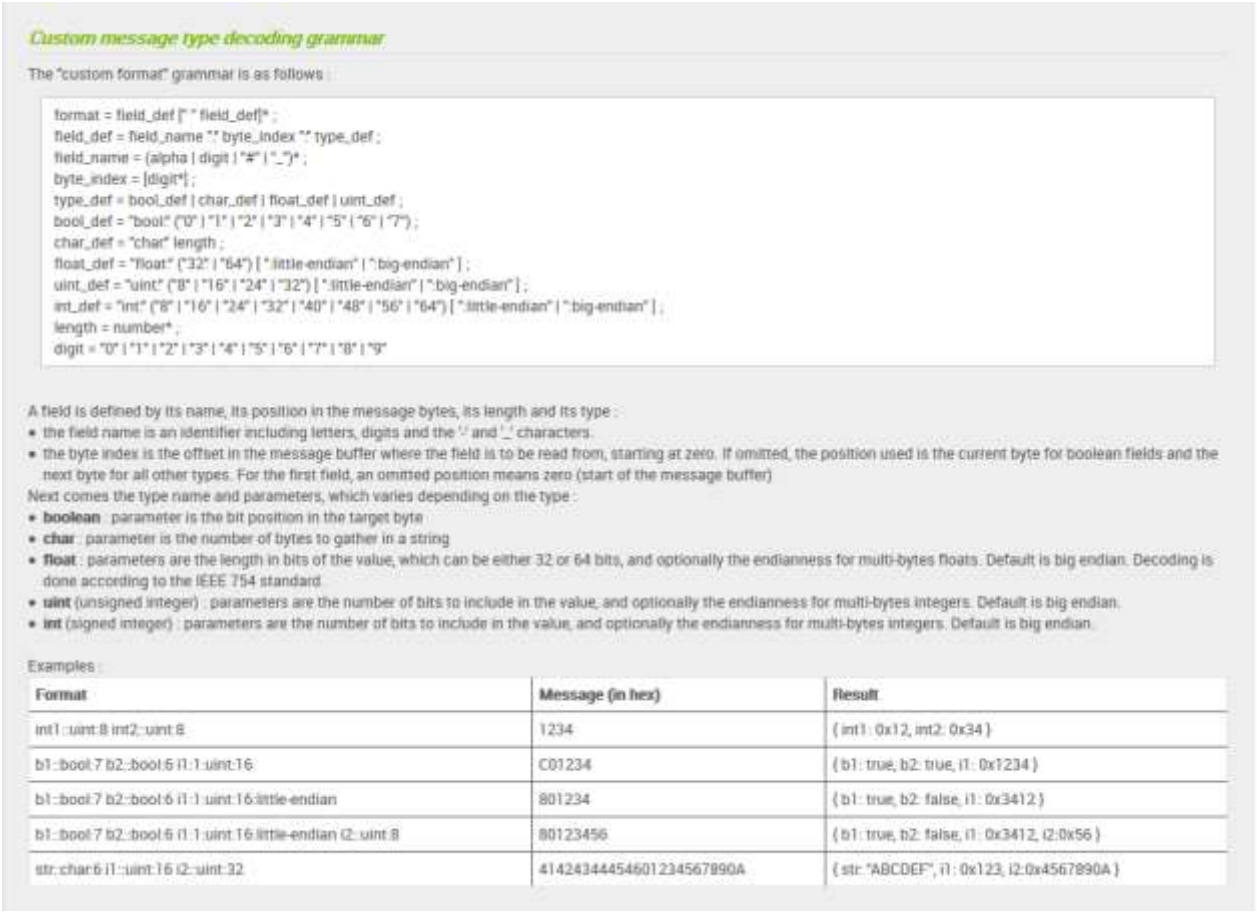
Connecting with a Sigfox Devicetype

Make sure you are logged in both the Sigfox dashboard and the Waylay dashboard.

1. Get the API login and API password from the Sigfox backend by going to ‘GROUP’ then select your preferred group and go to the tab Api Access. If needed create an API access entry and give it all the roles. Then copy the credentials to the Dashboard Sigfox authentication form.
2. To get a device type id, go to the ‘DEVICE TYPE’ page and choose the device type that you prefer (click link in the column Name). Now you see information about this devicetype. Use the field Id in the field ‘DeviceType ID’ in the Dashboard Sigfox authentication form.
3. Specify how the message for this devicetype is formatted. The format is the same as Sigfox uses.



Custom message type decoding grammar

The "custom format" grammar is as follows :

```
format = field_def [" " field_def]* ;
field_def = field_name ":" byte_index ":" type_def ;
field_name = (alpha | digit | "#" | "_")* ;
byte_index = [digit*] ;
type_def = bool_def | char_def | float_def | uint_def ;
bool_def = "bool:" ("0" | "1" | "2" | "3" | "4" | "5" | "6" | "7") ;
char_def = "char:" length ;
float_def = "float:" ("32" | "64") [ ":little-endian" | ":big-endian" ] ;
uint_def = "uint:" ("8" | "16" | "24" | "32") [ ":little-endian" | ":big-endian" ] ;
int_def = "int:" ("8" | "16" | "24" | "32" | "40" | "48" | "56" | "64") [ ":little-endian" | ":big-endian" ] ;
length = number* ;
digit = "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"
```

A field is defined by its name, its position in the message bytes, its length and its type :

- the field name is an identifier including letters, digits and the '-' and '_' characters.
- the byte index is the offset in the message buffer where the field is to be read from, starting at zero. If omitted, the position used is the current byte for boolean fields and the next byte for all other types. For the first field, an omitted position means zero (start of the message buffer)

Next comes the type name and parameters, which varies depending on the type :

- **boolean** : parameter is the bit position in the target byte
- **char** : parameter is the number of bytes to gather in a string
- **float** : parameters are the length in bits of the value, which can be either 32 or 64 bits, and optionally the endianness for multi-bytes floats. Default is big endian. Decoding is done according to the IEEE 754 standard
- **uint** (unsigned integer) : parameters are the number of bits to include in the value, and optionally the endianness for multi-bytes integers. Default is big endian.
- **int** (signed integer) : parameters are the number of bits to include in the value, and optionally the endianness for multi-bytes integers. Default is big endian.

Examples :

Format	Message (in hex)	Result
int1::uint:8 int2::uint:8	1234	{ int1: 0x12, int2: 0x34 }
b1::bool:7 b2::bool:6 i1:1:uint:16	C01234	{ b1: true, b2: true, i1: 0x1234 }
b1::bool:7 b2::bool:6 i1:1:uint:16:little-endian	801234	{ b1: true, b2: false, i1: 0x3412 }
b1::bool:7 b2::bool:6 i1:1:uint:16:little-endian i2::uint:8	80123456	{ b1: true, b2: false, i1: 0x3412, i2:0x56 }
str::char:6 i1::uint:16 i2::uint:32	4142434445460123456789 0A	{ str: "ABCDEF", i1: 0x123, i2:0x4567890A }

Figure 1 Sigfox message type decoding grammar

4. When all fields are configured, click the authenticate button. This will import old data and a webhook to the Waylay dashboard will be added to the selected device type. Should it not exist you can press the Refresh resources button under DEBUG.
5. Now when a Sigfox message is received it will be propagated to Waylay. The message send to the broker depends on the format string you have provided.

The parsing of the messages happens on the webhookside. When a message is received the webhook will use the *@waylay/sigfox-parser* package and parse the message according to the previously specified format string. The webhook wil receive a message in this format:

```
{
    "longitude": "4",
    "latitude": "51",
    "device": "151D8",
    "data": "144418",
    "timestamp": "1461321009",
    "payloadConfig": "lightAmbi::uint:16 temperature:2:int:8"
  }
```

And then parses this data to this.

```
{
    "lightAmbi": 5188,
    "temperature": 24
  }
```

And then you should combine these messages and send this to all the resources where the deviceid is equal to the device. Note that sometimes the parsed message already contains geodata and this should not be overwritten. Below is the code that implements the webhook (no requires are done).

```
module.exports =  function(req, res) {
 res.sendStatus(200);
 try {
   const body = req.body;
   const message = parseMessage(body.data, body.payloadConfig);
   message.timestamp = body.timestamp * 1000;
   if (!message.longitude && !message.latitude) { // Make sure the parsedMessage does not contain
geolocation
     message.longitude = body.longitude;
     message.latitude = body.latitude;
   }
   Resource.find({provider: 'sigfox', providerId: body.device})
     .then((devices) => {
       return devices.map((device) => waylay.data.postSeries(device.resource, message));
     });
 } catch (err) {
   log.error('ERROR', err);
 }
};
```