API Skiply

What is the “Skiply Push API” (SPA) ?

Principle

- Each LPWAN operator sends data with different formats and/or protocols (REST, MQTT, Pub/Sub…): the SPA delivers a standard json payload to your server. It always remains the same, independently from the Network Operator (Sigfox or LoRaWAN). Therefore, it acts as a high level roaming system.
- The SPA decodes the data whatever the Running mode / frame is (pulse, count, code, battery level), taking into consideration the version of your device’s firmware.
- The SPA does the annoying job for you: calculation of the increments, checking the data consistency, and eliminating “bad frames”.
- We POST data to the URL of your choice (you can include API key as parameter or custom header). See the Skiply API documentation to know how to manage this parameters.

Frame details

The buttons are numbered as follow:

- Button1 = top left
- Button 2 = top right
- Button 3 = middle
- Button 4 = bottom left
- Button 5 = bottom right

> Push API: normal mode

```
{
  "device": "SA000272",
  "groupId": 5,
  "time": "2018-06-21T20:14:31.394Z", // ISO 8601 date
  "sq_num": 98,
  "frameType": "02",
  "data": {
    "index": [1,0,16,1,2],
    "increment": [0,0,0,0,2],
    "button_1": 0,
    "button_2": 0,
    "button_3": 0,
    "button_4": 0,
    "button_5": 2,
    "ack": 0
  },
  "negativeValue": 0,
  "rawPayload":"02000100A2000000000003"
}
```

Where:

- device = serial number of the product
- groupId = group of the device. Can be usefull to facilitate to consolidate device data, for instance in the context of live visualisation
- time = timestamp from the network
- sq_num = frame sequence number, to identify if a frame has been missed
- frameType = "02" or "03" (with ack)
- ack = 1 if the frame has been send due to a magnetic detection instead of normal operation (badge)
- index = state of the internal counters : [button1, button2, ... , button5]
- increment = variation of the index between to frames received by our servers : [button1, button2, ... , button5]
  code = type of frame
- negativeValue = return 1 if a negative increment has been detected (reset). Increment is [0,0,0,0,0] in this case to avoid bad counts
- rawPayload = payload receive before decoding

> Push API: code mode

```
{
  "device":"SA000272",
  "groupId": 5,
  "time":"2018-06-21T20:14:31.394Z", // ISO 8601 Date
  "dc_delay": 0,
  "sq_num": 98,
  "frameType": "11",
  "data": {
    "code": "021214",
    "increment": [2,2,0,1,0],
    "button_1": 2,
    "button_2": 2,
    "button_3": 0,
    "button_4": 1,
    "button_5": 2,
    "ack": 0
  },
  "previous": {
    "code": "012334",
    "increment": [1,1,2,1,0],
    "button_1": 1,
    "button_2": 1,
    "button_3": 2,
    "button_4": 1,
    "button_5": 0,
    "ack":1,
    "previous_time": 21
  },
  "rawPayload":"11000100A2000000000003"
};
```

Where:

- frameType = "11" or "13" or "31" for code mode
- dc_delay: potential delay from the Duty Cycle protection in minutes
- data: code entered by the user
    - code: value of the code (1 to 5, max. 6 digits)
    - increment (button 1 to 5, number of presses per button)
    - ack = 1 if the code has been validated by magnetic detection (badge)
- previous: code entered previously by the user
    - previous_time: time since last code what entered

> Push API: keep alive (sent every 24h)

```
{
  "device": "SA000272",
  "groupId": 5,
  "time": "2018-06-21T20:14:31.394Z", // ISO 8601 Date
  "sq_num": 99,
  "frameType": "01",
  "data": {
    "battery_level": 3.35
  },
  "rawPayload":"010EC640EC64"
}
```

Where:

- device = serial number of the product
- time = timestamp from the network
- sq_num = frame sequence number, to identify if a frame has been missed
- frameType = "01"
- ack = always 0 for this frame type
- battery_level = in volts
- code = type of frame

Retry

The Push API will send data to the endpoint defined in the group configuration, and wait for a 200 code in return. If the system doesn't receive a 200 response, a new try will be done after 60 seconds. If the retry fails, an error is logged with the missed data and error code returned.

Unique solution ID: #1049
Author: Jérôme Chambard
Last update: 2019-06-03 11:19