

# Lopy4

## LoRa et Sigfox

Rémi Jolin

# LoRa

- LoRa
  - Directement entre équipements (aujourd'hui)
    - LoRaWAN (demain, TTN), une couche protocolaire de plus
  - En Europe : bande des 868MHz
  - Fair use de la bande
  - Messages en « diffusion » : tous les récepteurs reçoivent tous les messages
    - pas forcément ceux que vous attendez...

# Sur LoPy4

- On fait appel à LoRa du package « network » et on crée une socket

```
import socket
from network import LoRa

# Open a Lora Socket, use tx_iq to avoid listening to our own messages
# Please pick the region that matches where you are using the device:
# Asia = LoRa.AS923
# Australia = LoRa.AU915
# Europe = LoRa.EU868
# United States = LoRa.US915
lora = LoRa(mode=LoRa.LORA, tx_iq=True, region=LoRa.EU868)

lora_sock = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
lora_sock.setblocking(False)
```

- Et ensuite on lit/écrit sur la socket

```
lora_sock.send(pkg)

data = lora_sock.recv(256)
```

# LoRa/Lopy4

- Demo time...

# Sigfox

- Réseau public
- Boite/Techno française
  - Labège (Toulouse)
  - Ludovic Le Moan (CEO)
- Couverture « quasi » mondiale (60 pays/régions couverts)
- Low consumption
- Low cost

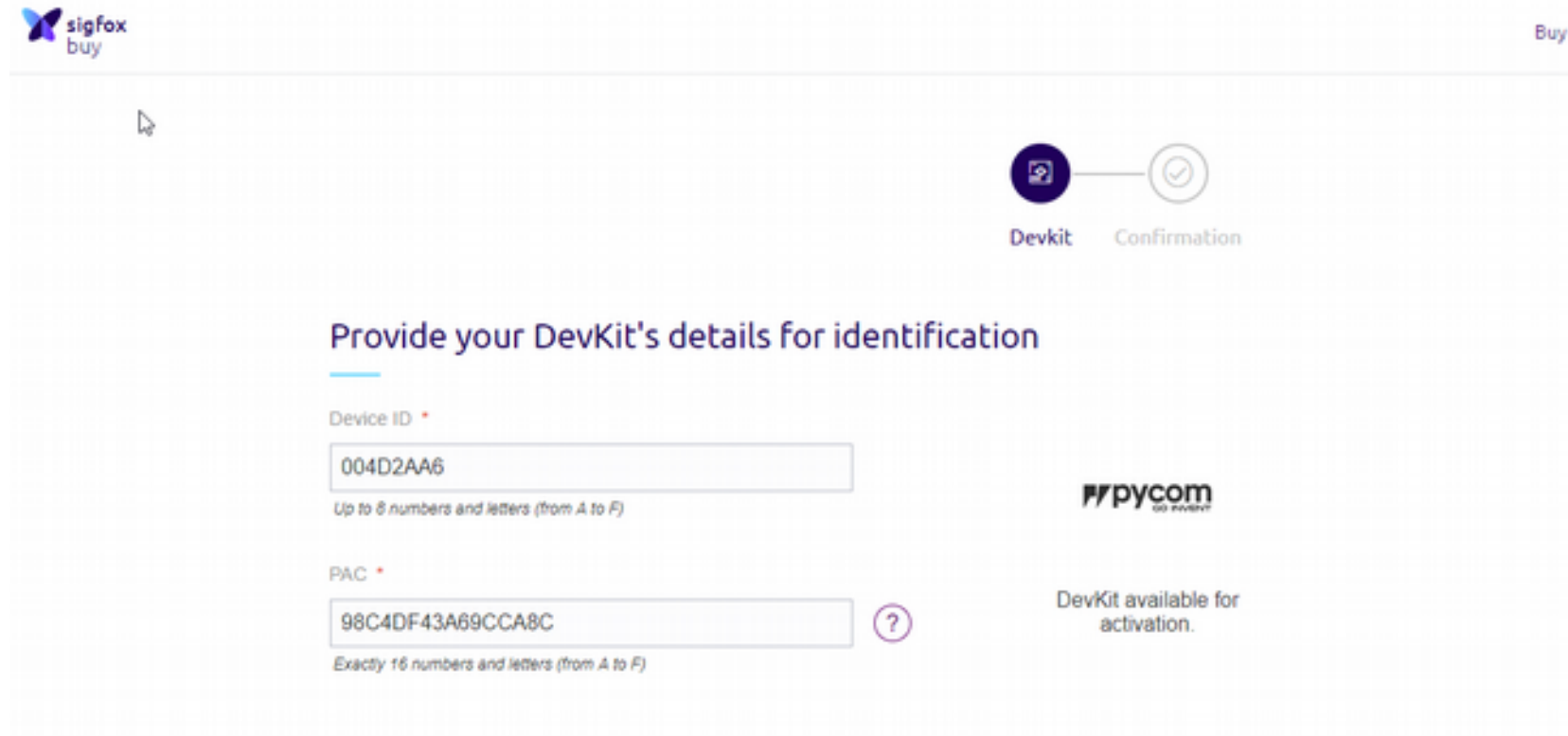


# Sigfox...

- Les devices envoient des messages à des stations qui les relaient à l'utilisateur final (via mail/webservice)
- 12 octets en émission (uplink), 8 en réception (downlink)
- 6 messages / heure en émission, 4 / JOUR en réception
  - Pas de contrôle « en dur »... Gentlemen agreement
- Messages chiffrés sur la partie radio

# Activation des devices LoPy4

- [build.sigfox.com/activate](https://build.sigfox.com/activate)



The screenshot shows the 'sigfox buy' header with a 'Buy' button. A progress bar indicates the 'Devkit' step is active, followed by 'Confirmation'. The main heading is 'Provide your DevKit's details for identification'. It contains two input fields: 'Device ID' with the value '004D2AA6' and a note 'Up to 8 numbers and letters (from A to F)'; and 'PAC' with the value '98C4DF43A69CCA8C' and a note 'Exactly 16 numbers and letters (from A to F)'. A Pycom logo is visible, along with a status message 'DevKit available for activation.' and a help icon.

sigfox buy Buy

Devkit Confirmation

Provide your DevKit's details for identification

Device ID \*

004D2AA6

Up to 8 numbers and letters (from A to F)

PAC \*

98C4DF43A69CCA8C

Exactly 16 numbers and letters (from A to F)

pycom

DevKit available for activation.

# Ajouter un callback

- backend.sigfox.com

The screenshot displays the Sigfox backend interface for managing device types. The top navigation bar includes tabs for 'DEVICE', 'DEVICE TYPE' (selected), 'USER', and 'GROUP'. The left sidebar contains a 'LIST' section with options: 'DEVICES BEING REGISTERED', 'GEOLOCATION PAYLOAD', and 'BULK OPERATIONS'. The main content area is titled 'Device type - List' and features a search bar with fields for 'Name' and 'Group', a 'Select a group' button, and a 'Display type' dropdown menu. Below the search bar are buttons for 'Include sub groups', 'RESET', and 'FILTER'. A table of device types is displayed, showing columns for 'Description', 'Display type', 'Group', 'Keep alive', and 'Name'. The table lists three device types: 'Auto created device type for EVK user : Remi Jolin', 'DevKit 1 (PYCOM)', and 'DevKit 2 (PYCOM)'. A context menu is visible over the table, showing options like 'Disengage sequence number', 'Restart', 'Edit', and 'Delete'.

Description	Display type	Group	Keep alive	Name
Auto created device type for EVK user : Remi Jolin	None	Remi Jolin EVK	N/A	Pycom TelecomValley kit
DevKit 1 (PYCOM)	None	Arsynet	N/A	PYCOM_DevKit_1
DevKit 2 (PYCOM)	None	Arsynet	N/A	PYCOM_DevKit_2



# Ajouter un callback...



Sigfox - Reinvent radio communication

LOCATION

ASSOCIATED DEVICES

DEVICES BEING REGISTERED

STATISTICS

EVENT CONFIGURATION

CALLBACKS

BULK OPERATIONS

DEVICE

DEVICE TYPE

USER

GROUP



## Device type PYCOM\_DevKit\_2 - Edition

### Device type information

Name PYCOM\_DevKit\_2

Description DevKit 2 (PYCOM)

Keep-alive (in minutes) 0

Subscription automatic renewal ☒ ?

Contracts

If we fail to call one of your callbacks, an email will be sent to the address below so that you can take action to fix the problem.

Alert email

### Downlink data

Downlink mode CALLBACK ▼

Expression must either include hexadecimal encoded bytes (ex: **deadbeefcafebabe**) or the following variables: - {time} 4 bytes - {tapid} 4 bytes - {rssi} 2 bytes - {roaming} 1 byte


Downlink data in hexa  ?

### Payload display

Select below the most suitable parsing mode for the display of your payloads in the backend (mostly appropriate for debugging and development)

Payload parsing Regular (raw payload) ▼

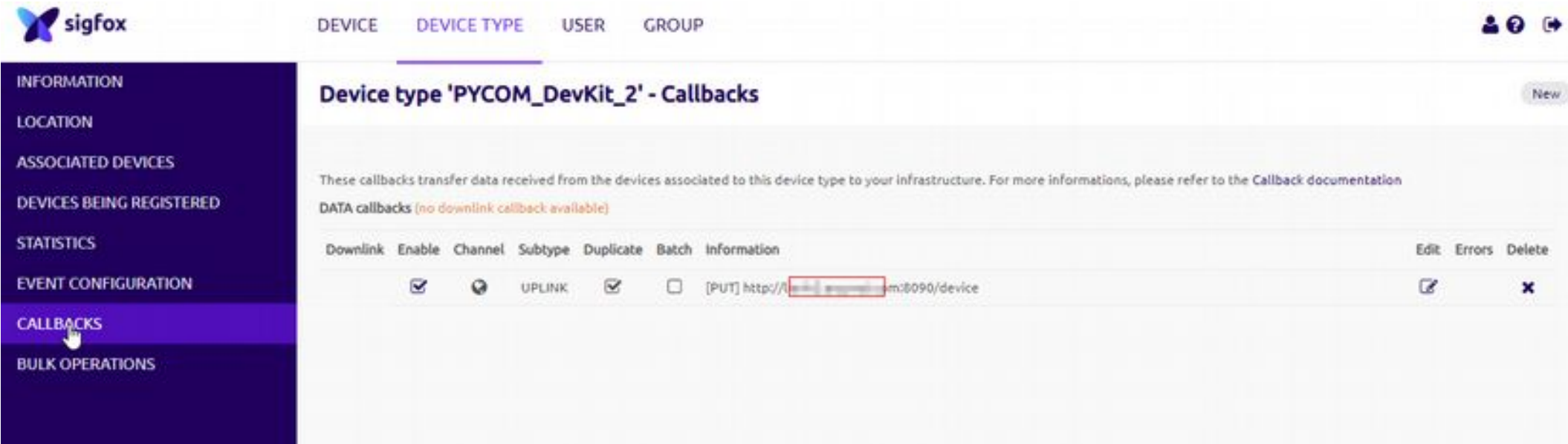
# Device / informations



The screenshot displays the Sigfox web interface. At the top left is the Sigfox logo. A navigation bar at the top contains links for DEVICE, DEVICE TYPE (which is highlighted with a purple underline), USER, and GROUP. On the left side, there is a dark purple sidebar with a list of menu items: INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING REGISTERED, STATISTICS, EVENT CONFIGURATION, CALLBACKS, and BULK OPERATIONS. The main content area on the right is titled 'Device type 'PYCOM\_DevKit\_2' - Information'. It lists the following details for the device:

- Id: 5cbc5fb60499f50883751bfd
- Name: PYCOM\_DevKit\_2
- Description: DevKit 2 (PYCOM)
- Keep alive: N/A
- Subscription automatic renewal: ☒ ⓘ
- Group: Arsynet
- Payload display: None
- Contracts:
  - 1. arsynet\_58fb\_9b85 (no token left - geoloc: yes, end date: 2021-04-20)
- Alert Email:
- Creation date: 2019-04-21 12:19:02
- Created by: buy.sigfox.com
- Last edition date: 2019-04-21 12:19:02
- Last edited by: buy.sigfox.com

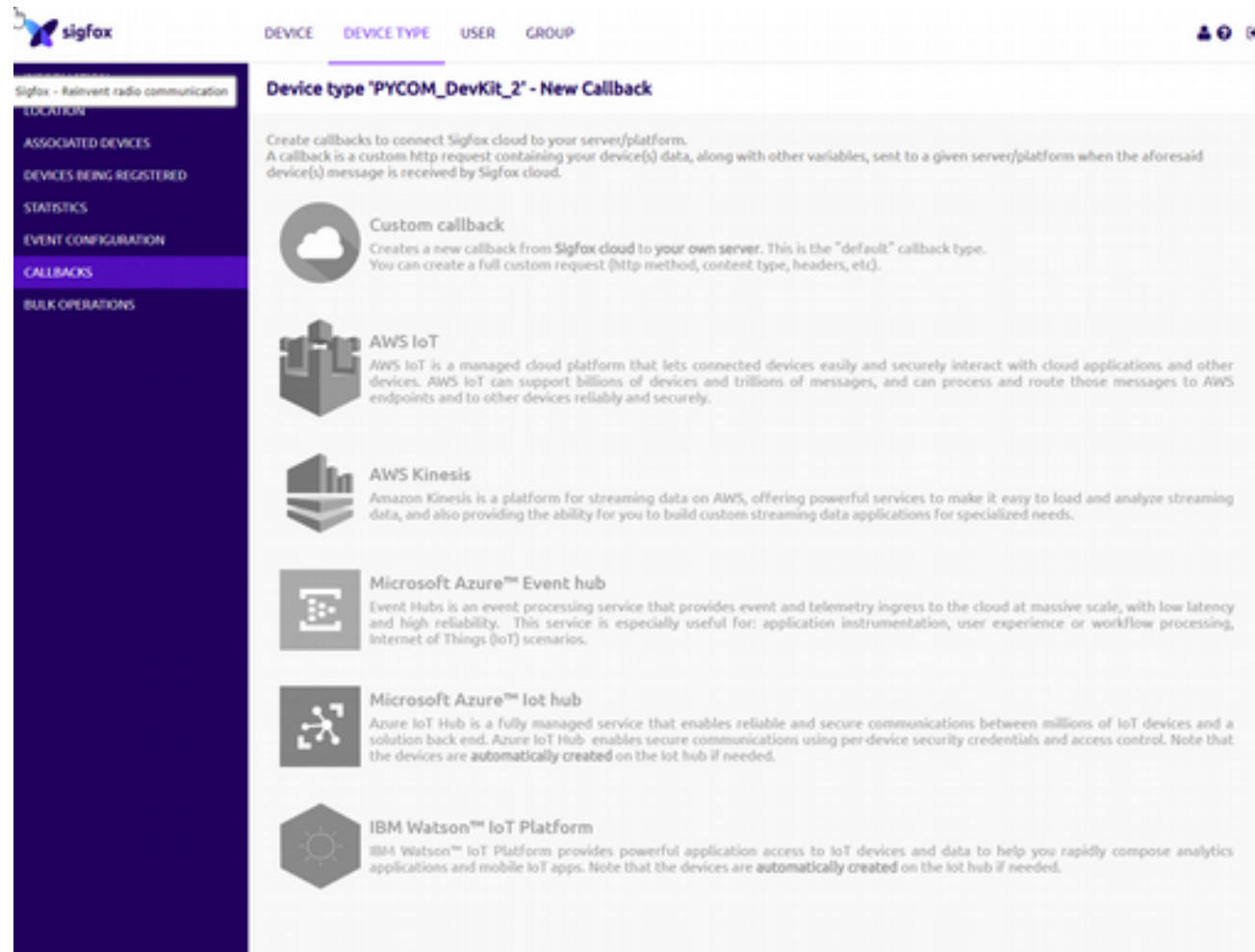
# Ajouter un callback...



The screenshot shows the Sigfox web interface. On the left is a dark purple sidebar with a list of menu items: INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING REGISTERED, STATISTICS, EVENT CONFIGURATION, CALLBACKS (highlighted with a mouse cursor), and BULK OPERATIONS. The main content area has a top navigation bar with tabs: DEVICE, DEVICE TYPE (selected), USER, and GROUP. Below the tabs, the title 'Device type 'PYCOM\_DevKit\_2' - Callbacks' is displayed, followed by a 'New' button. A descriptive text states: 'These callbacks transfer data received from the devices associated to this device type to your infrastructure. For more informations, please refer to the [Callback documentation](#)'. Below this, it says 'DATA callbacks (no downlink callback available)'. A table lists the callbacks with columns: Downlink, Enable, Channel, Subtype, Duplicate, Batch, Information, Edit, Errors, and Delete. One callback is shown with a checked 'Downlink' box, a globe icon for 'Channel', 'UPLINK' for 'Subtype', a checked 'Duplicate' box, an unchecked 'Batch' box, and the 'Information' field containing '[PUT] http://[redacted]m:090/device'. The 'Edit' column has a pencil icon, and the 'Delete' column has an 'X' icon.

Downlink	Enable	Channel	Subtype	Duplicate	Batch	Information	Edit	Errors	Delete
<input checked="" type="checkbox"/>	<input type="checkbox"/>		UPLINK	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[PUT] http://[redacted]m:090/device			







# Ajouter un callback...



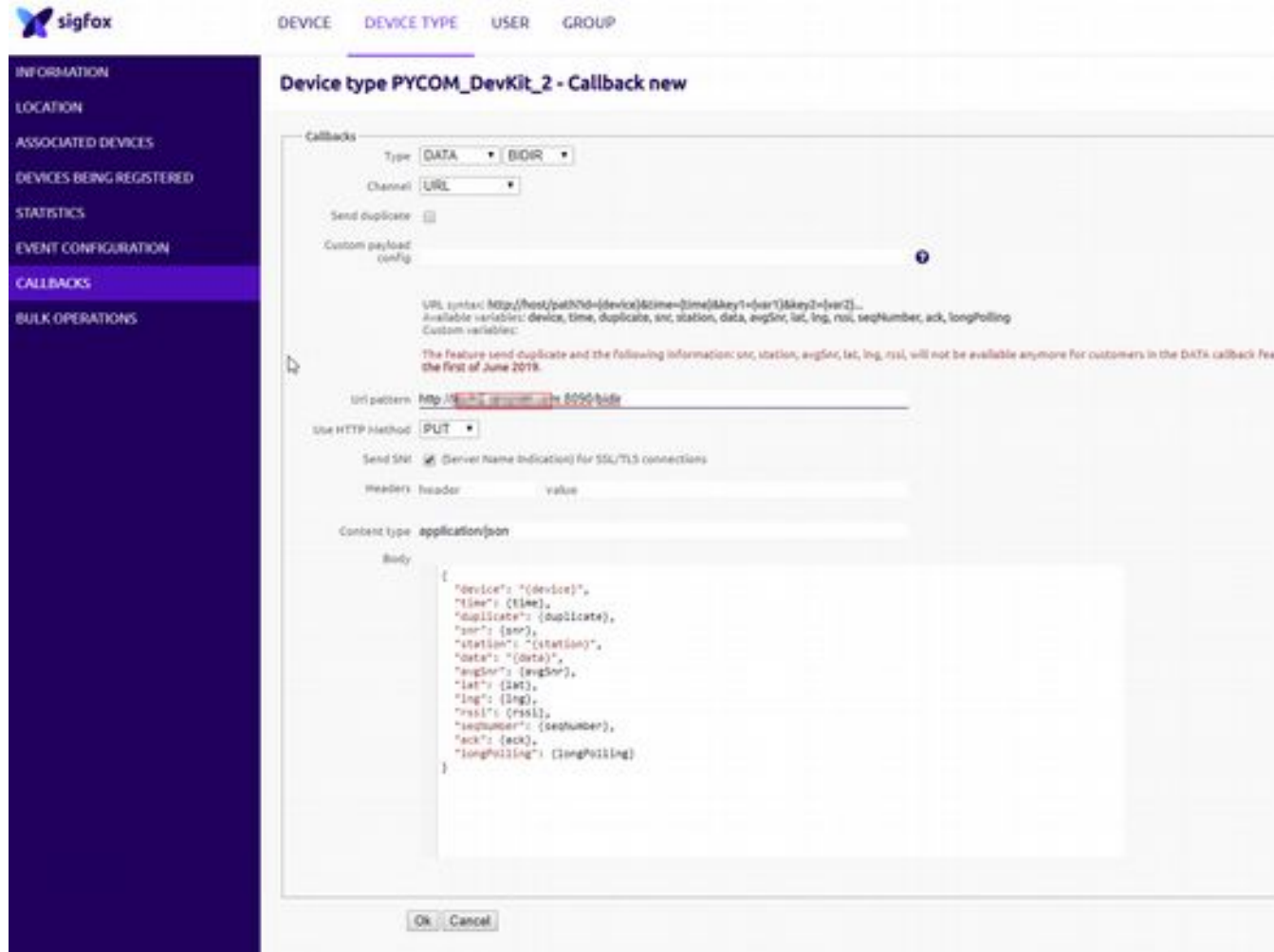
The screenshot shows the Sigfox web interface. On the left is a dark blue sidebar with navigation links: 'Sigfox - Reinvent radio communication', 'LOCATION', 'ASSOCIATED DEVICES', 'DEVICES BEING REGISTERED', 'STATISTICS', 'EVENT CONFIGURATION', 'CALLBACKS' (highlighted in purple), and 'BULK OPERATIONS'. The top navigation bar includes 'DEVICE', 'DEVICE TYPE', 'USER', and 'GROUP'. The main content area is titled 'Device type 'PYCOM\_DevKit\_2' - New Callback'. It contains a description of callbacks and a list of callback types with their respective icons and descriptions.

**Device type 'PYCOM\_DevKit\_2' - New Callback**

Create callbacks to connect Sigfox cloud to your server/platform.  
A callback is a custom http request containing your device(s) data, along with other variables, sent to a given server/platform when the aforesaid device(s) message is received by Sigfox cloud.

-  **Custom callback**  
Creates a new callback from Sigfox cloud to your own server. This is the "default" callback type. You can create a full custom request (http method, content type, headers, etc).
-  **AWS IoT**  
AWS IoT is a managed cloud platform that lets connected devices easily and securely interact with cloud applications and other devices. AWS IoT can support billions of devices and trillions of messages, and can process and route those messages to AWS endpoints and to other devices reliably and securely.
-  **AWS Kinesis**  
Amazon Kinesis is a platform for streaming data on AWS, offering powerful services to make it easy to load and analyze streaming data, and also providing the ability for you to build custom streaming data applications for specialized needs.
-  **Microsoft Azure™ Event hub**  
Event Hubs is an event processing service that provides event and telemetry ingress to the cloud at massive scale, with low latency and high reliability. This service is especially useful for: application instrumentation, user experience or workflow processing, Internet of Things (IoT) scenarios.
-  **Microsoft Azure™ IoT hub**  
Azure IoT Hub is a fully managed service that enables reliable and secure communications between millions of IoT devices and a solution back end. Azure IoT Hub enables secure communications using per-device security credentials and access control. Note that the devices are **automatically created** on the IoT hub if needed.
-  **IBM Watson™ IoT Platform**  
IBM Watson™ IoT Platform provides powerful application access to IoT devices and data to help you rapidly compose analytics applications and mobile IoT apps. Note that the devices are **automatically created** on the IoT hub if needed.

# Ajouter un callback...



The screenshot shows the Sigfox web interface for configuring a new callback for a device. The left sidebar contains navigation links: INFORMATION, LOCATION, ASSOCIATED DEVICES, DEVICES BEING REGISTERED, STATISTICS, EVENT CONFIGURATION, CALLBACKS (highlighted), and BULK OPERATIONS. The top navigation bar includes tabs for DEVICE, DEVICE TYPE, USER, and GROUP. The main heading is "Device type PYCOM\_DevKit\_2 - Callback new".

The "Callbacks" section contains the following fields and options:

- Type: DATA (selected), BIOR (selected)
- Channel: URL (selected)
- Send duplicate: ☐
- Custom payload config:
- URL syntax: `http://host/path?id={device}&time={time}&key1={var1}&key2={var2}...`  
Available variables: device, time, duplicate, src, station, data, avgSnr, lat, lng, rssi, seqNumber, ack, longPolling  
Custom variables:
- The feature send duplicate and the following information: src, station, avgSnr, lat, lng, rssi, will not be available anymore for customers in the DATA callback Fee the first of June 2019.
- Url pattern: `http://192.168.1.100:8080/body`
- Use HTTP Method: PUT (selected)
- Send SNR: ☒ (Server Name Indication) for SSL/TLS connections
- Headers: Header value
- Content type: application/json
- Body: 

```
{
  "device": "{device}",
  "time": {time},
  "duplicate": {duplicate},
  "src": {src},
  "station": "{station}",
  "data": "{data}",
  "avgSnr": {avgSnr},
  "lat": {lat},
  "lng": {lng},
  "rssi": {rssi},
  "seqNumber": {seqNumber},
  "ack": {ack},
  "longPolling": {longPolling}
}
```

At the bottom, there are "Ok" and "Cancel" buttons.

# Ajouter un callback...

DEVICE DEVICE TYPE USER GROUP



## Device type 'PYCOM\_DevKit\_2' - Callbacks

New

These callbacks transfer data received from the devices associated to this device type to your infrastructure. For more informations, please refer to the [Callback documentation](#)

DATA callbacks (no downlink callback available)

Downlink	Enable	Channel	Subtype	Duplicate	Batch	Information	Edit	Errors	Delete
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		UPLINK	<input checked="" type="checkbox"/>	<input type="checkbox"/>	[PUT] http://[redacted]om:8090/device			
	<input checked="" type="checkbox"/>		BIDIR	<input type="checkbox"/>	<input type="checkbox"/>	[PUT] http://[redacted]m:8090/bidir			

Select for downlink

# Coté « callback »

- Un serveur web qui va répondre à l'URL

```
@app.route('/device', methods=['POST', 'PUT'])
def read_device_info():
    result = dict(call='device',
                  now=utcnow(),
                  method=request.method,
                  headers=dict(h for h in request.headers),
                  ip=request.remote_addr,
                  json=request.json)

    print('result')
    pprint(result)

    return jsonify(dict(result='ok'))
```

# Callback...

- **bidir**

```
@app.route('/bidir', methods=['POST', 'PUT'])
def read_device_info_bidir():
    result = dict(call='bidir', now=utcnow(), method=request.method,
                  headers=dict(h for h in request.headers),
                  ip=request.remote_addr, json=request.json)
    print('result')
    pprint(result)

    device = request.json['device']

    r, v, b = random.randrange(256), random.randrange(256), random.randrange(256)
    resp = format_rvb.build(dict(r=r, v=v, b=b)).encode('hex')
    response = {device: (dict(downlinkData=resp) if request.json['ack'] else\
                           dict(noData=True))}
    print('reponse:', response)
    return jsonify(response)
```



# Envoyer un message avec LoPy

```
from network import Sigfox
import socket
import struct
import pycom

# init Sigfox for RCZ1 (Europe)
sigfox = Sigfox(mode=Sigfox.SIGFOX, rcz=Sigfox.RCZ1)

# create a Sigfox socket
s = socket.socket(socket.AF_SIGFOX, socket.SOCK_RAW)

# make the socket blocking
s.setblocking(True)

# configure it as both ways link
s.setsockopt(socket.SOL_SIGFOX, socket.SO_RX, True) # False if uplink only

# send message
s.send('azerty')

# receive message
s.settimeout(60)
result = s.recv(32)
```