



Connect with Actility Servers and Create your IoT Application

REVISION 2.0

JULY 2018

INDEX

1.	INTRODUCTION	3
2.	Receive your devices payload from Actility.....	3
2.1	Receive the data from Actility	3
2.1.1	Using the AS Routing Profile.....	3
2.1.2	Using the DX-API Dataflow.....	6
3.	Display your device data.....	12
3.1.1	Freeboard and MQTT Example.....	12
4.	CONCLUSION.....	15

1. INTRODUCTION

ThingPark DX-API provides a dataflow management solution to process uplink and downlink data. It allows developers to design and deploy new dataflows for a specific set of devices, thus enabling data processing on top of the ThingPark Wireless solution for those devices.

In this guide you will find tutorials to retrieve and display devices data from Actility networks servers.

Keep in mind that these tutorials are intended for users that already play with Actility ThingPark Wireless, so you will need to have the next pre-requisites for going successfully through the tutorials:

- ThingPark Developer account in one of the following network servers: Dev1, POC or IoT.
- Gateway and device registered and activated in your account
- Know how to create and associate AS routing profiles to IoT devices in the ThingPark Wireless platform.

If you don't know how to make this and want to create your IoT application with Actility, you just need to create a new account in our [Developers Portal](#) and read our [Advanced Developer Guide](#). Here you have all the tools to get started with Actility!

2. RECEIVE YOUR DEVICES PAYLOAD FROM ACTILITY

2.1 Receive the data from Actility

Exists two different ways to get devices data from Actility ThingPark servers. The first one is using an AS routing profile, which gives users the opportunity to send data to any HTTP server, with the constrain that this data will be encoded by an algorithm made by the device maker. To solve this problem, Actility gives the opportunity to use the DX-API Dataflow as well, which is a software that allows subscribers to manage their ThingPark account and create new connectors called "Dataflows". These Dataflows send data in the same way of an AS Routing Profile but manage different protocols (Besides HTTP), could use keys and certificates to make a more secure connection and decode your device data to send ready to process information to your application server.

Here you will find two tutorials about how to use these two ways.

2.1.1 Using the AS Routing Profile

The AS (Application Server) routing profile defines how the sensor data is routed to an application server.

This is the easiest way to obtain data from Actility, only set a routing profile that point at the HTTP server of your preference, and you will start receiving coded data of the device in form of a POST request in JSON or XML format.

As an example, you could create a simple Generic HTTP Listener following the steps describer in the Actility Git-Hub:

<https://github.com/actility/generic-http-listener>

With this HTTP server, you could see how Activity send coded device data and create your own application. But if you don't want to create your own application right now, and you want to use this feature you can follow the next tutorial about how to send data through AS Routing Profiles using <http://dweet.io/> as an example. Dweet give users the opportunity to send messages to a HTTP server and display them in an easy way.

Activity and Dweet Tutorial

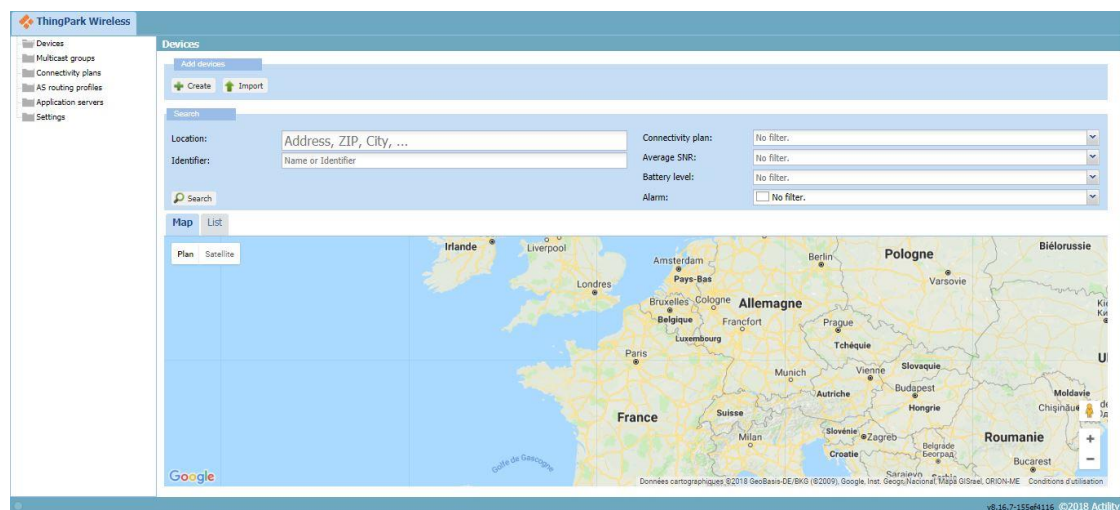
Dweet allows users to send data to a specific URL that follow the next pattern:

<https://dweet.io/dweet/for/my-thing-name?hello=world>

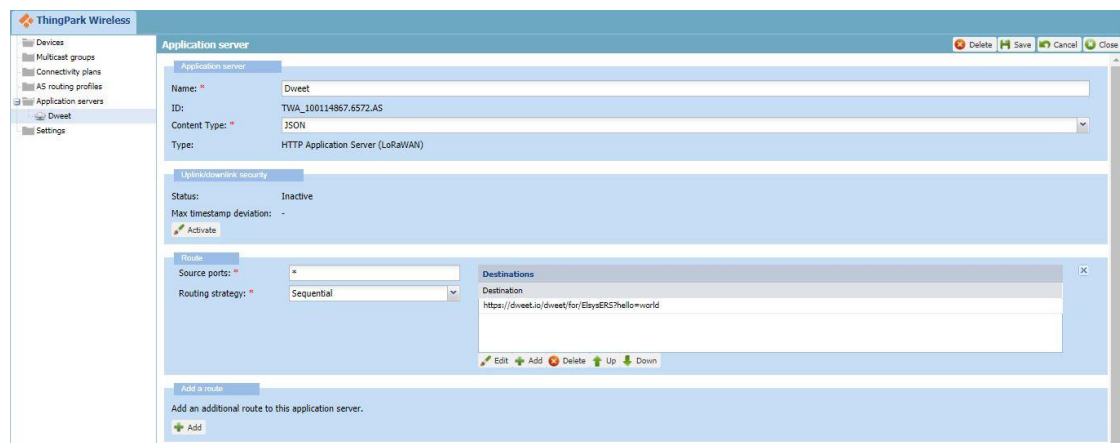
Where “my-thing-name” is a unique name of your choice. So, when you have your thing name decided, go to the ThingPark Wireless interface and create a routing profile that points to this new URL.

To achieve this, follow the next steps:

- Enter in your ThingPark Wireless user interface



- Create a new application server with the URL given above and your thing name (In this tutorial the application server will be called Dweet and will point to <https://dweet.io/dweet/for/ElsysERS?hello=world>)



- Create a new AS Routing Profile and associate the previous application server created (In this tutorial the AS Routing Profile will be called Dweet as well)

AS routing profile

Name: Dweet
 ID: TWA_100114867.8510
 Type: LoRaWAN
 Is default: ☒

Type	Destination
Local application server	Dweet

Last modification: 27/07/2018 à 17:24:04
 Updated by:

- Open your device network settings going to Devices -> List -> Edit (Pencil tool)

Device

Manufacturer: Generic
 Model: LoRaWAN 1.0.2 - class C - AS923 - Rx2_SF11
 Name: Elys ERS
 Motion indicator: Near static
 Activation mode: Over The Air Activation (OTAA)
 DevEUI: A81758FFFFE3273A
 DevAddr: FE03273A

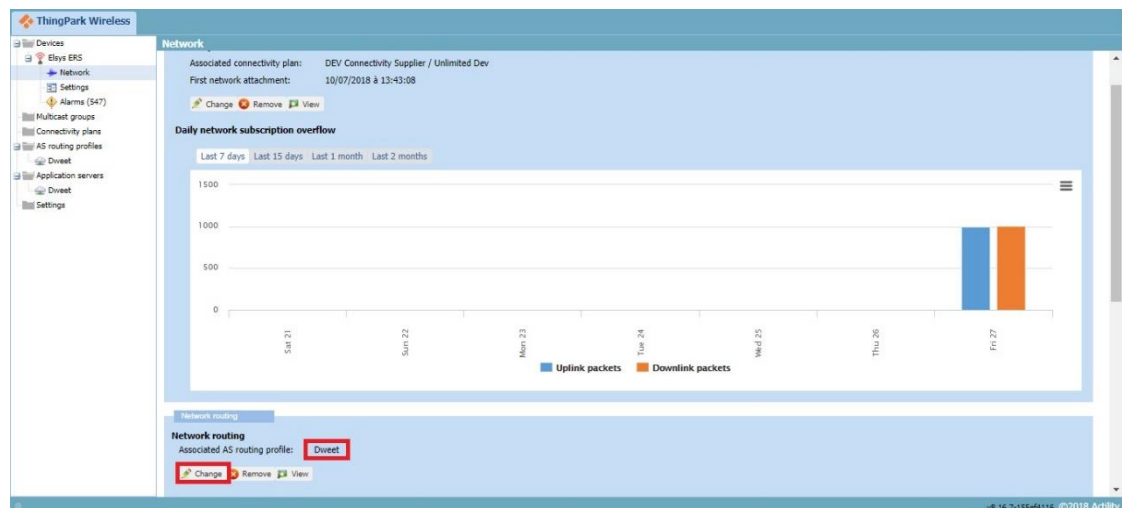
Administrative info:

Average packets: 15880.0/day
 Average SNR: 5.8 dB
 Average RSSI: -105.0 dBm
 Last instantaneous PER: 0.0%
 Last mean PER: 0.0%

Last spreading factor: SF7
 Last SNR: 7.3 dB
 Last RSSI: -108.0 dBm
 Last uplink frame: 27/07/2018 à 17:28:26
 Last downlink frame: 27/07/2018 à 17:28:27

Battery:
 Battery replaced: 10/07/2018
 Replace battery by: 18/01/2019

- Go to Network settings and associate your new AS Routing Profile



That's it your AS Routing Profile is ready! If you want to retrieve the data just follow the next link, replacing your “thing name” and you will see your device payload.

<https://dweet.io/get/latest/dweet/for/my-thing-name>

In the example of this tutorial the information is showed in the next way by Dweet:

```
← → ↺ ⌂ 🔒 Sécurisé https://dweet.io/get/latest/dweet/for/ElsysERS
{"this":"succeeded","by":"getting","the":"dweets","with":[{"thing":"ElsysERS","created":"2018-07-27T15:36:28.244Z","content":
{"hello":"world","LrnDevEui":"A81758FFFFE03273A","LrnPort":5,"LrnInfos":{"Twa_100114867.6572.AS-1-5028147","DevEui_uplink":{"Time":"2018-07-
27T17:36:26.789+02:00","DevEui":"A81758FFFFE03273A","FPort":5,"FPortUp":133358,"ACKBit":1,"ADBit":1,"Type":4,"Payload_hex":"0100ff02310401490500070d0e","mic_hex":"cf5a7110"
,"Lrcid":"00000127","LrrRSSI":-106,"LrrSNR":8.75,"SpFact":7,"SubBand":"G0","Channel":"LC4","DevLrrCnt":1,"LrrId":"004A265B","LrrLAT":48.875221,"LrrLON":2.334037,"Lrrs":{"Lrr":
{"LrrId":"004A265B","Chain":0,"LrrRSSI":-106,"LrrSNR":8.75,"LrrESP":-106.543648}}},"CustomerID":"100114867","CustomerData":{"loc":
{"lat":"48.876570543321755","lon":"2.3324060440063477"},"a1r":
{"pro":"LORA/Generic","ver":"1"}},{"ModelCfg":"0","InstantPER":0,"MeanPER":0,"DevAddr":"FE03273A","AckRequested":1,"rawMacCommands":""}}]}
```

2.1.2 Using the DX-API Dataflow

The purpose of this API is to provide the best experience for all developers who want to interface with ThingPark applications to receive and send messages from their IoT devices.

Link to theory

If you want to familiarize yourself with the functioning of the DX-API Dataflow, do not hesitate and go to the DX-API documentation. Here you could read how this tool works in the front and back end.

[DX-API Documentation](#)

On the other hand, if you know how the Dataflows works or you only want to create a quick proof-of-concept continue through the tutorial!

DX-API Tutorial

First go to the DX-API Dataflow application following the next link, introduce your credentials and you will have access to all DX-API applications and documentation:

[Get Started](#)

Select the “DX Dataflow API Swagger-UI” and follow the next steps to create your Dataflow.

Get started with the ThingPark DX API Platform!

You successfully generated a new token:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzY2Y2

This token is associated with scope [SUBSCRIBER:114867]
and expires on July 30th 2018, 3:26:22 pm.

Now you can use any of the following DX APIs:

DX Admin API

Token management
and other DX platform
administration features

[Doc](#) | [Swagger-UI](#)

DX Core API

Account management,
offer subscription, device and
base station provisioning

[Doc](#) | [Swagger-UI](#)

DX Maker API

Device pre-provisioning
and management at
manufacturer stage

[Doc](#) | [Swagger-UI](#)

DX Dataflow API

Data processing
and dataflow management
(drivers and connectors)

[Doc](#) | [Swagger-UI](#)

DX Location API

Integration with ThingPark
geolocation solutions
(solvers and applications)

[Doc](#) | [Swagger-UI](#)

To create a dataflow, you need to know that each dataflow has its own parameters that you need to define and write down in the DX-API Swagger user interface in a JSON or XML format to create it. For each connector there are different parameters that you could use and for assist subscribers in the "Connectors Example Code" and "Connectors Templates" folders you will find examples codes with the only the mandatory information for the Dataflow connectors in JSON format. Remember that these connectors could have more parameters that you could find in the documentation, in other words, in this tutorial and with the example codes it will be shown how to create the most basic connectors.

Now with this examples codes you can change the required information and follow the next steps:

- Go to the DX API Dataflow
- Select the POST /bridgeDataflows request

POST /bridgeDataflows Bridge dataflow creation

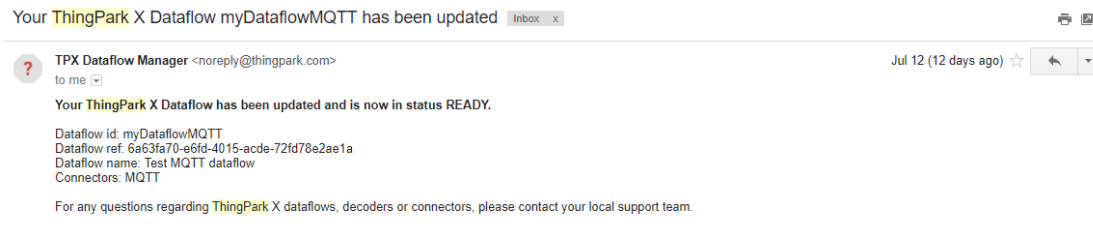
- Paste your code in the Value box as it shows the next image (Taking the MQTT example code, remember that you could use the blank templates and fill them with your information).

Parameter	Value	Description	Parameter Type	Data Type
bridgeDataflow	<pre>{ "id": "", "name": "", "bidirectional": false, "binder": { "classRef": "LRC_HTTP", "properties": { "deviceEUIList": "" } }, "driver": { "classRef": "" }, "connectors": [{ "classRef": "GenericMQTT", "properties": { "hostName": "HostServer:Port", "protocol": "", "accountPrefix": "", "login": "", "password": "" } }] }</pre>	Contents of the Bridge dataflow to create.	body	Model Example Value BridgeDataflow { ref (string, optional): Ref of the dataflow. Generated by the system upon creation., id (string, optional): Id of the dataflow. Note that this attribute can't be changed after creation., name (string, optional): Name of the dataflow., bidirectional (boolean, optional): If true, the dataflow will process both uplink messages and downlink messages. If false, the dataflow will only process uplink messages (with optimized performance). Default is false., state (string, optional): Read-only attribute to indicate the current state of the dataflow. Possible values are 'AWAITING_VALIDATION' (waiting to be validated for deployment), 'AWAITING_UPDATE' (waiting for the execution instance to be updated), 'READY' (deployed and ready to be used) or 'DEPLOY_ERROR' (could not be deployed due to an error). = ['AWAITING_VALIDATION', 'AWAITING_UPDATE', 'READY', 'DEPLOY_ERROR'], generatedInfo (DataflowGeneratedInfo, optional): Generated read-only dataflow information., binder (Processor, optional): Binder

Parameter content type: application/json

- Click the "Try it out!" button

- If you follow all the steps correctly your Dataflow will be waiting for the validation process. When the process is done you will receive the next email with the information of your connector:

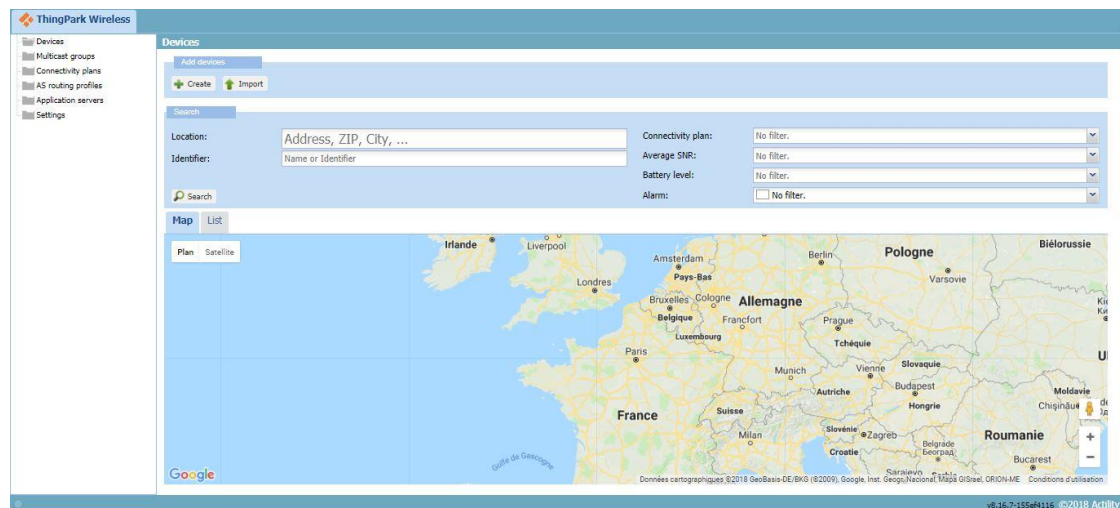


While the activation process is completed you can go to the Activity ThingPark interface or to the DX Core API and create a new AS routing profile that points to the next URL:

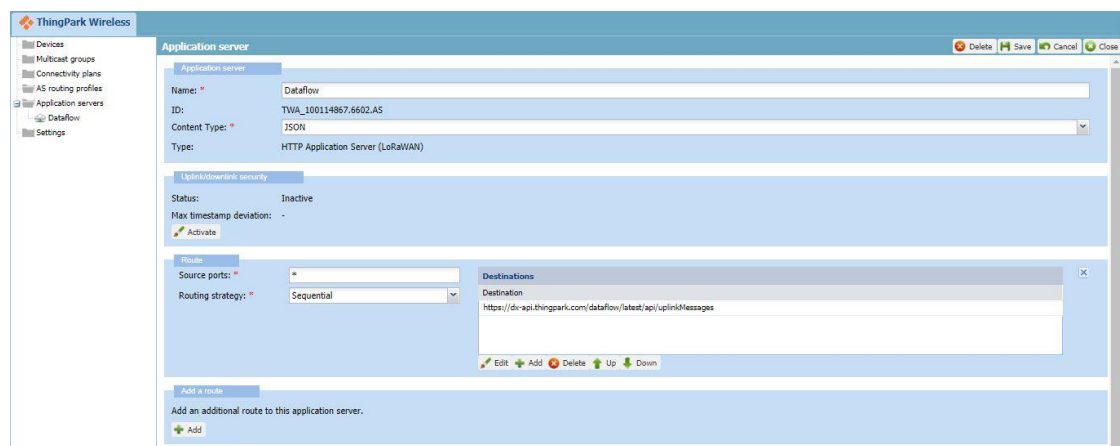
<https://dx-api.thingpark.com/dataflow/latest/api/uplinkMessages>

To achieve this, follow the next steps:

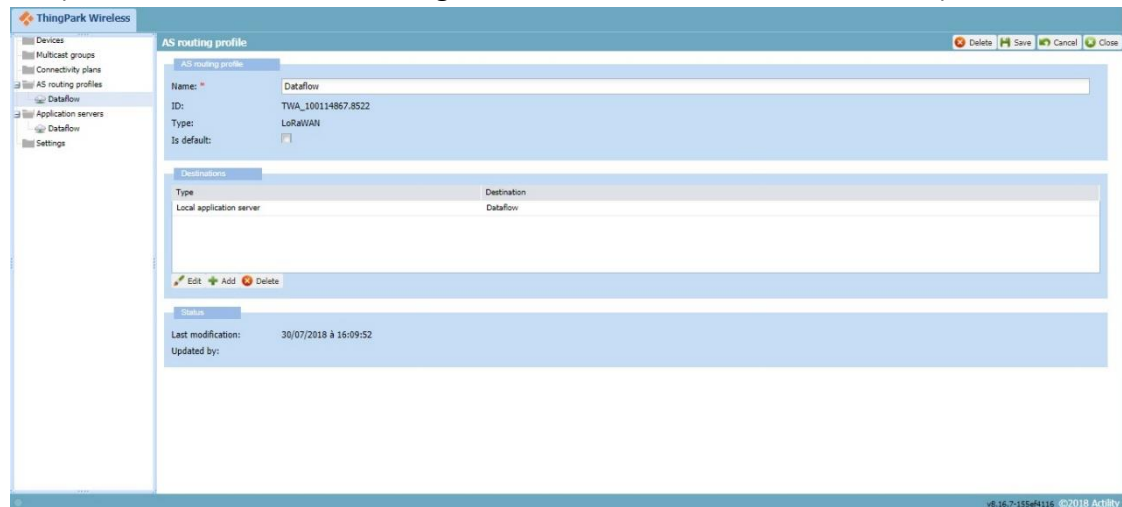
- Enter in your ThingPark Wireless user interface



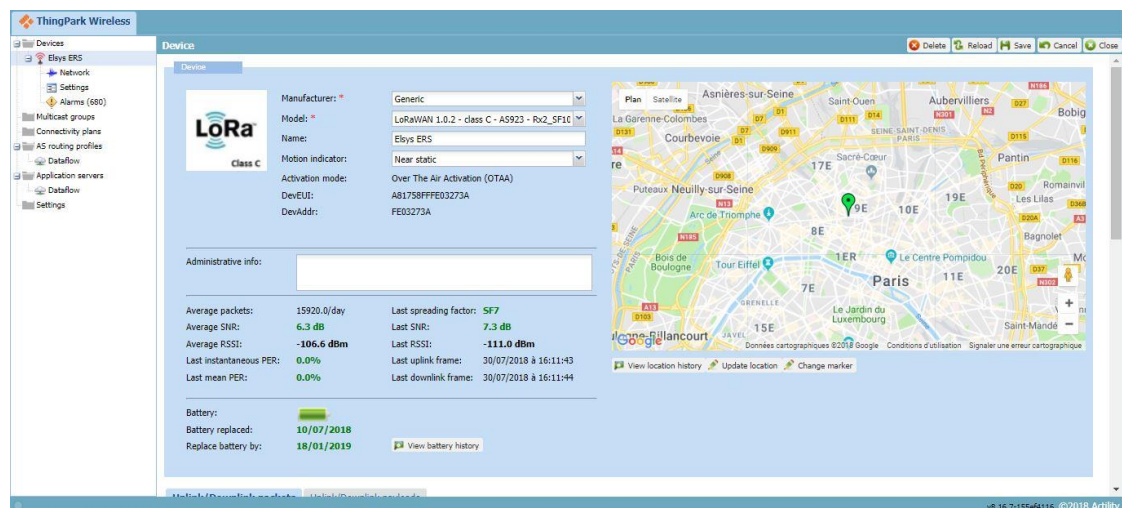
- Create a new HTTP application server with the URL given above (In this tutorial the application server will be called Dataflow, but you can choose the name of your preference)



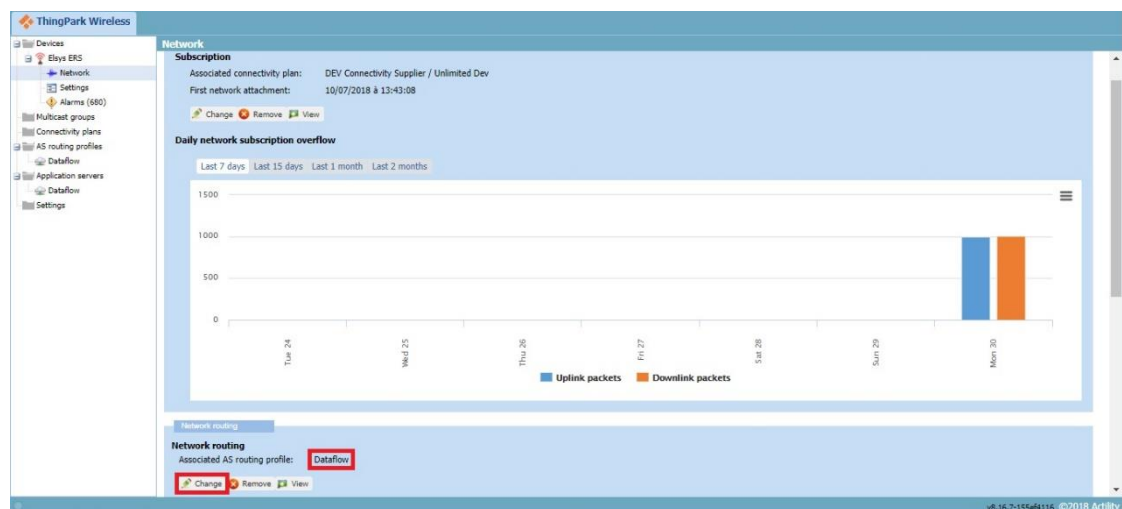
- Create a new AS Routing Profile and associate the previous application server created (In this tutorial the AS Routing Profile will be called Dataflow as well)



- Open your device network settings going to Devices -> List -> Edit (Pencil tool)



- Go to Network settings and associate your new AS Routing Profile



That's it your AS Routing Profile is ready! Your Dataflow is now ready and sending your device information.

MQTT Connector with CloudMQTT

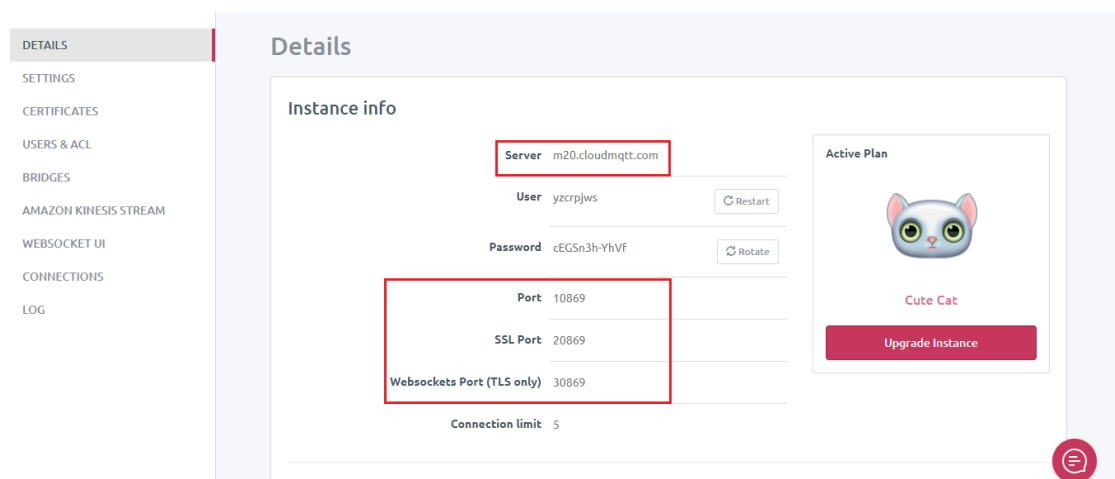
To create a new MQTT dataflow here will be explained how to connect with a free third party MQTT broker called CloudMQTT.

Prerequisites:

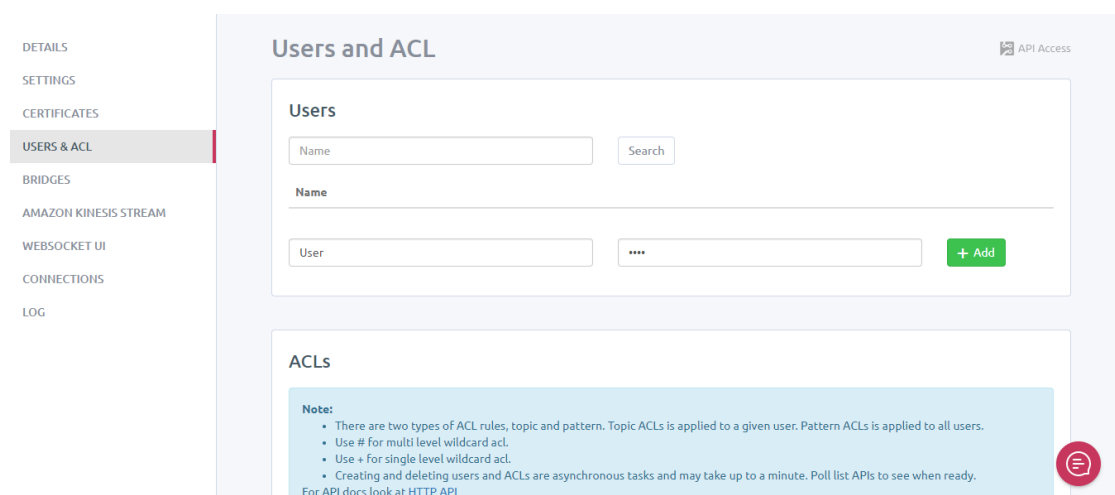
- ThingPark Wireless application in one of the following network servers: Dev1, IoT, POC.
- Device and Gateway registered in Activity ThingPark Wireless application.
- CloudMQTT account (could be a free or paid account).

Configuration:

- Create an instance and go to details where you will find your MQTT host and the ports to connect using different protocols, see next image.



- Create a user which will be the credentials to connect with Activity with this MQTT Broker.



- Finally, create a topic from where you expect to receive data, you could use a wild card topic using the “#” symbol and give the Read and Write access to the user.

ACLs

Notes:

- There are two types of ACL rules, topic and pattern. Topic ACLs is applied to a given user. Pattern ACLs is applied to all users.
- Use # for multi level wildcard acl.
- Use + for single level wildcard acl.
- Creating and deleting users and ACLs are asynchronous tasks and may take up to a minute. Poll list APIs to see when ready. For API docs look at HTTP API

Type: Pattern Pattern: # Read/Write: ☒ Read Access? ☒ Write Access? + Add

- With this configuration you are ready to create and test a MQTT Dataflow! Save the information obtained in the previous steps: Server URL, port protocol, port number, user name and password.

Now you can go to the DX-API Dataflow application and follow the next steps:

- Select the POST /bridgeDataflows request

POST /bridgeDataflows Bridge dataflow creation

- Create your JSON/XML code filling the blank spaces in the following text:

[CloudMQTT Example](#)

- Paste your code in the POST /bridgeDataflows request (Here it will be use the blank example code with the infomation previously gathered and one Elsys sensor)

Response Content Type: application/json ▼

Parameters

Parameter	Value	Description	Parameter Type	Data Type
bridgeDataflow	<pre>{ "id": "MyMQTT", "name": "My MQTT Connector", "bidirectional": false, "binder": { "classRef": "LRC_HTTP", "properties": { "deviceEUIlist": "A81758FFFE03273B" } }, "driver": { "classRef": "Elsys" }, "connectors": [{ "classRef": "GenericMQTT", "properties": { "hostName": "m20.cloudMQTT.com:10869", "protocol": "TCP", "accountPrefix": "test", "login": "User", "password": "*****" } }] }</pre>	Contents of the Bridge dataflow to create.	body	Model Example Value BridgeDataflow { ref (string, optional): Ref of the dataflow. Generated by the system upon creation., id (string, optional): Id of the dataflow. Note that this attribute can't be changed after creation., name (string, optional): Name of the dataflow., bidirectional (boolean, optional): If true, the dataflow will process both uplink messages and downlink messages. If false, the dataflow will only process uplink messages (with optimized performance). Default is false., state (string, optional): Read-only attribute to indicate the current state of the dataflow. Possible values are 'AWAITING_VALIDATION' (waiting to be validated for deployment), 'AWAITING_UPDATE' (waiting for the execution instance to be updated), 'READY' (deployed and ready to be used) or 'DEPLOY_ERROR' (could not be deployed due to an error). = [AWAITING_VALIDATION, 'AWAITING_UPDATE', 'READY', 'DEPLOY_ERROR'], generatedInfo (DataflowGeneratedInfo, optional): Generated read-only dataflow information., binder (Processor, optional): Binder

Parameter content type: application/xml ▼

- Click the “Try it out!” button
- Now your connector will be waiting for the validation process. When the process is done you will receive an email with a change of status to READY for your connector

Don't forget creating your Dataflow AS routing profile that points to the next URL:

<https://dx-api.thingpark.com/dataflow/latest/api/uplinkMessages>

Set this routing profile (As it is shown above) in your devices and that's it! Your Dataflow is now ready and sending your device information. Go to the WebSocket UI on CloudMQTT to see the messages incoming from Actility.

3. DISPLAY YOUR DEVICE DATA

Here it will be shown how to display data using an IoT dashboard, or in other words, a graphical user interface where users can manage their IoT solution for data collection, processing, visualization and device management.

3.1.1 Freeboard and MQTT Example

In this tutorial it will be explained how to display device data using a free IoT dashboard called Freeboard.

Freeboard is a simple web panel that shows the information of the different IoT devices that you have connected in real time. And, can fully function in the browser as a static web application without needing a server, acting as a MQTT client.

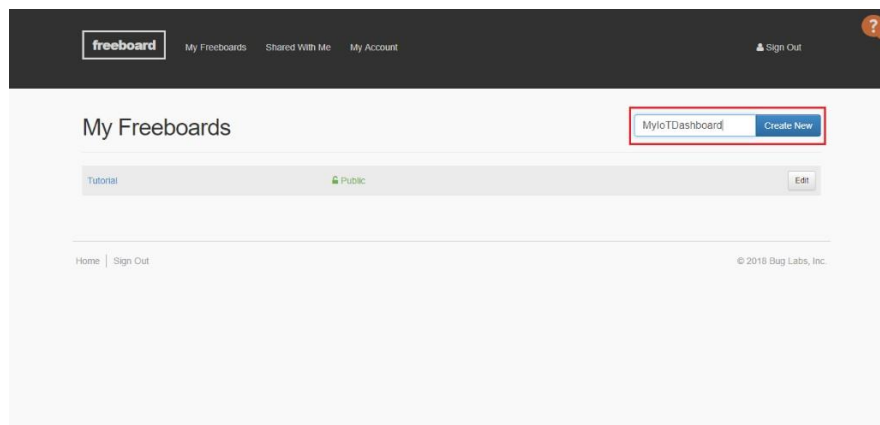
In this tutorial Freeboard it is going to be used as a MQTT client connected to a generic MQTT broker.

Prerequisites to display data in Freeboard:

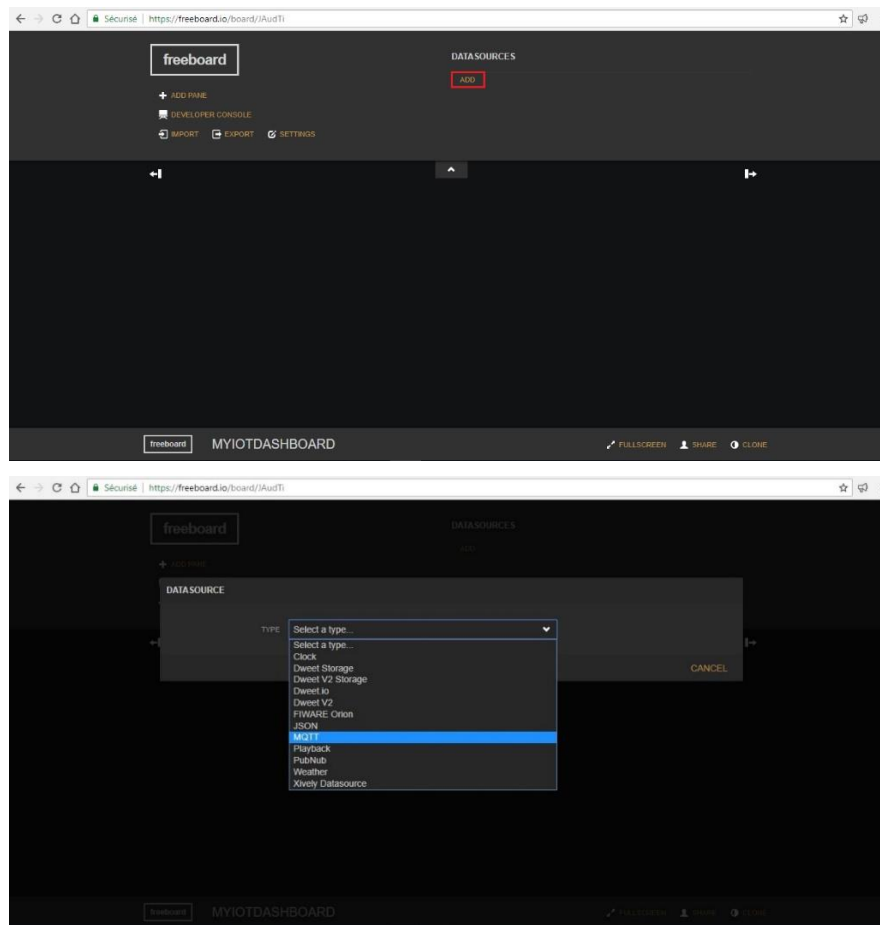
- ThingPark Wireless application in one of the following network servers: Dev1, IoT, POC.
- Device and Gateway registered in Actility ThingPark Wireless application.
- MQTT broker connected with Actility ThingPark Wireless through a dataflow and AS routing profile previously created.

Freeboard configuration

- Create a free trial account in <https://freeboard.io/>
- Log in and create a new dashboard in <https://freeboard.io/account/>



- Create a “datasource” inside your new dashboard and select the protocol or network server from where you want to retrieve the device information. In this case will be MQTT protocol.



- Put the information of your MQTT broker which will send your device to Freeboard.

TYPE: **MQTT**

NAME: **YourPreferredName**

TOPIC: **accountPrefix/things/deviceEU/uplink**

For IBM quickstart, replace only DEVICEID with the device id string found in the upper right corner. For Watson IoT platform, replace DEVICEID with your specific device mac address or with '*/' to listen to all devices in your organization. For all other MQTT servers, enter your own topic search string.

SERVER: **YourMQTTBrokerHostServer**

For IBM quickstart use 'quickstart.internetofthings.ibmcloud.com', or type '[OrganizationID].messaging.internetofthings.ibmcloud.com' for Watson IoT, or enter any other MQTT server available

PORT: **MQTTBrokerTLSPort**

Typically either 8883 or 443 for secure, or 1883 for insecure communication

USE ENCRYPTION: **YES** ☒

Use TLS encryption to connect to the MQTT Server securely (recommended)

CLIENT ID: **YourPreferredClientId**

For IBM quickstart, use default 'quickstart'. For Watson IoT, enter your Organization ID. For all other MQTT servers, set a clientId which will be passed as 'a:clientId:ApiKey:Timestamp'

API KEY/USERNAME: **MQTTBrokerUser**

Not required for IBM quickstart, required for Watson IoT Platform connections

API AUTH TOKEN/PASSWORD: **UserPassword**

Not required for IBM quickstart, required for Watson IoT Platform connections

JSON MESSAGES?: **YES** ☒

If the messages on your topic are in JSON format they will be parsed so the individual fields can be used in freeboard widgets

Now your connection to Freeboard is ready and you could choose the widget that fit the most to your data!

Freeboard example with CloudMQTT

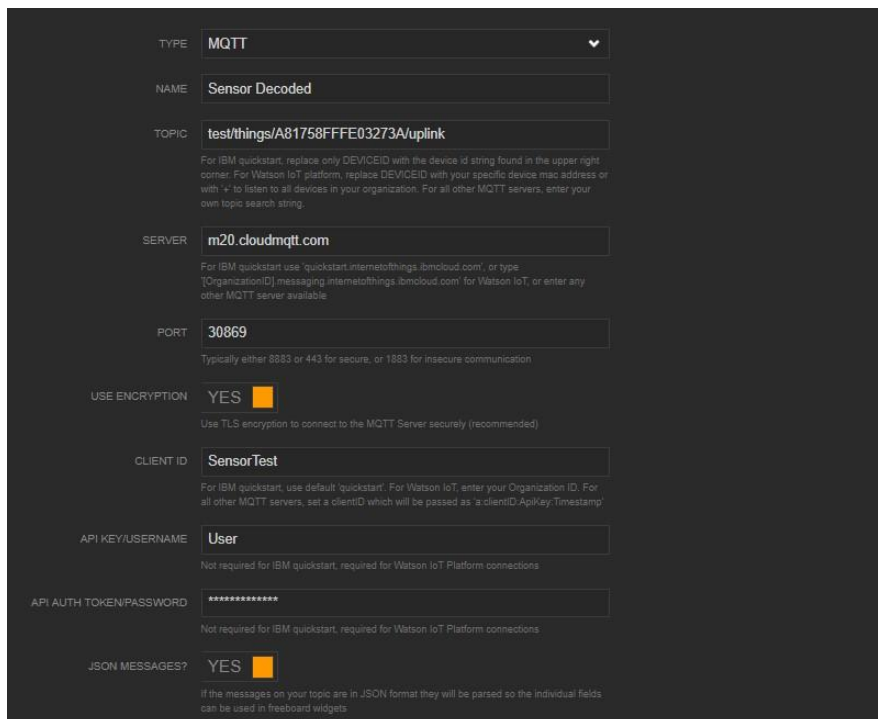
In the next image you will find an example of the information that is need in Freeboard to connect with CloudMQTT following the previous explanation and taking the information of the **MQTT Connector with CloudMQTT** example created in the DX-API Dataflow Swagger UI tutorial above.

[CloudMQTT Example](#)

Also, don't forget to create a new user and password for the Freeboard connection in CloudMQTT. Then retrieve the information needed from CloudMQTT: Server URL, port protocol, port name and new user credentials (Name and password).

With the information from your MQTT broker follow the same steps explained above:

- Go to the Datasource creation in the right corner of Freeboard
- Click "add"
- Put the information needed, take as an example the template displayed in the next image



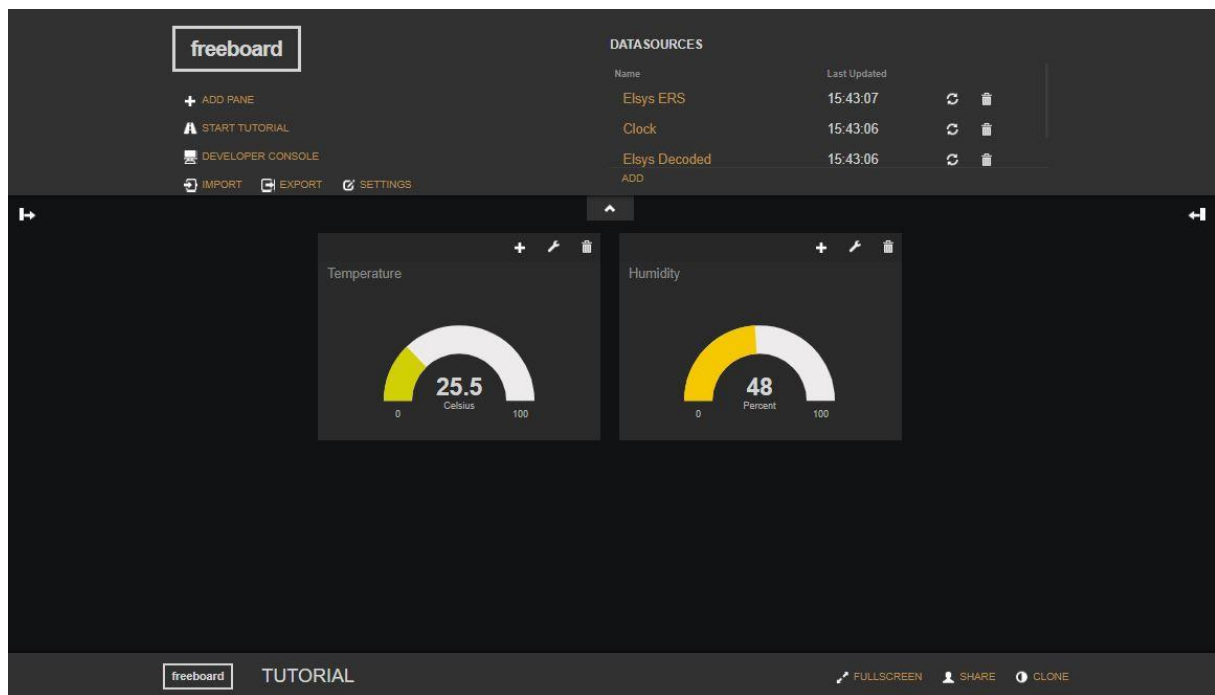
The image shows a configuration form for an MQTT connection in Freeboard. The form has the following fields and values:

- TYPE:** MQTT (selected from a dropdown)
- NAME:** Sensor Decoded
- TOPIC:** test/things/A81758FFFE03273A/uplink
For IBM quickstart, replace only DEVICEID with the device id string found in the upper right corner. For Watson IoT platform, replace DEVICEID with your specific device mac address or with '+' to listen to all devices in your organization. For all other MQTT servers, enter your own topic search string.
- SERVER:** m20.cloudmqtt.com
For IBM quickstart use 'quickstart.internetofthings.ibmcloud.com', or type '[OrganizationID].messaging.internetofthings.ibmcloud.com' for Watson IoT, or enter any other MQTT server available
- PORT:** 30869
Typically either 8883 or 443 for secure, or 1883 for insecure communication
- USE ENCRYPTION:** YES ☒
Use TLS encryption to connect to the MQTT Server securely (recommended)
- CLIENT ID:** SensorTest
For IBM quickstart, use default 'quickstart'. For Watson IoT, enter your Organization ID. For all other MQTT servers, set a clientId which will be passed as 'a:clientId:ApiKey:Timestamp'
- API KEY/USERNAME:** User
Not required for IBM quickstart, required for Watson IoT Platform connections
- API AUTH TOKEN/PASSWORD:** *****
Not required for IBM quickstart, required for Watson IoT Platform connections
- JSON MESSAGES?:** YES ☒
If the messages on your topic are in JSON format they will be parsed so the individual fields can be used in freeboard widgets

- Click "Save" and that's it!

After following these steps, you will have connected Freeboard with CloudMQTT and you are ready to display the device decoded data!

See an example using two-gauge widgets in the next image:



4. CONCLUSION

Congratulations! With this guide you learn how to retrieve and display data from Actility networks servers. Nevertheless, there is more that you can do! As making a more secure connection with the use of certificates in your connector parameters, or even send data to your devices! Go to our DX-API documentation and you will found out how to achieve this. If you have any question about the documentation or need more information don't hesitate and contact us through the next link:

<https://www.actility.com/contact/>

© 2017 ACTILITY SA. All rights reserved.

Portions of this documentation and of the software herein described are used by permission of their copyright owners.

Actility, ThingPark, are registered trademarks of Actility SA or its subsidiaries and may also be registered in other countries.

Other denoted product names of Actility SA or other companies may be trademarks or registered trademarks of Actility SA or its subsidiaries, or their respective owners.

Headquarters

Actility Lannion,
Actility S.A 4 rue Ampère BP 30225
22300 Lannion France

www.actility.com