**GW Device Protocols**

IoT Platform

*September 19th, 2014*

**CONFIDENTIAL**

© Telefónica, S.A.

Not to be copied, distributed or referred to without the prior written consent of Telefónica. All rights reserved.

The information contained herein and all related data or information are proprietary and confidential to Telefónica and are provided in confidence. All use, disclosure, copying, transfer, reproduction, display, distribution, or creation of derivative works, unless expressly authorized in the written by Telefónica, is strictly prohibited.

Information in this document is subject to change without notice and does not represent a commitment on the part of Telefónica.

**Signature control table**

|  |  |  |
| --- | --- | --- |
| Edited by | Reviewed by | Approved by |
| Antonio José López |  |  |

**Version control**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Changed sections | Author |
| 1.0 | 19/09/2014 |  | Antonio José López |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Contents**

[1 Introduction 6](#_Toc398906775)

[1.1 Expected audience 6](#_Toc398906776)

[2 Device Integration Process Overview 7](#_Toc398906777)

[2.1 Device Model definition 7](#_Toc398906778)

[2.2 Device provisioning 7](#_Toc398906779)

[2.3 Available protocols 8](#_Toc398906780)

[3 Ultralight 2.0 (UL2.0) 8](#_Toc398906781)

[3.1 Measures 8](#_Toc398906782)

[3.2 Commands 9](#_Toc398906783)

[3.2.1 Command Response 10](#_Toc398906784)

[3.2.2 Predefined Ping command 10](#_Toc398906785)

[4 MQTT 11](#_Toc398906786)

[4.1 MQTT Protocol Specification 11](#_Toc398906787)

[4.2 MQTT Implementation 11](#_Toc398906788)

[4.2.1 Measures 11](#_Toc398906789)

[4.2.2 Commands 12](#_Toc398906790)

[5 SensorML 13](#_Toc398906791)

[6 Glossary 14](#_Toc398906792)

[7 References 15](#_Toc398906793)

# Introduction

This document is related to Telefonica IoT Platform.

The goal of this guide is to expose, in a simple and easy manner, the different device protocols available in its Gateway component.

## Expected audience

We assume in this document that you are a device developer (maker) aiming to work with Telefonica IoT Platform

# Device Integration Process Overview

This section describes the overall process a partner or integrator needs to follow in order to integrate devices or sensors into the IoT Platform, by using its Gateway component (SBC)

In order to do so, both SBC Portal [5] or SBC APIs [6] can be used.

First step is to model the device, defining the list of sensors included, based on the phenomenon to be measured, units,..

Once a model is created, new devices can be provisioned in the platform before they start sending the measures to the Platform.

A complete workshop with sensor integration examples can be found in the document Technical Workshop [2].

## Device Model definition

The Sensor Model contains all device specifications, including the protocol implemented to receive commands (if any), sensors included (detailing measured phenomena and the alias to represent it), and the generic user properties (any parameter used to define device status or property).

Sensor model definition steps are defined in SBC Web Portal customer document [5], section 2.2, or via API in SBC REST APIs document [6], section 6.4.

## Device provisioning

Once a Model is defined and provisioned in the Platform, new devices can be provisioned following that model.

As a reference, there are two resources describing the device in the Platform. The *Device* itself, describing the physical elements, and the *Asset*, describing the environment/business data. This way, the Device can be replaced with another one, keeping the Asset resource with no change, so northbound apps consuming its data will not detect any difference (no id change even the deviceID changes, as they are using the assetID).

As a reference, sensor provisioning steps are defined in SBC Web Portal customer document [5], section 2.1,  or via API in SBC REST APIs document [6], section 6.2. Devices can also be provisioned via bulk files as described in SBC Web Portal customer document [5], section 2.1.4.

## Available protocols

Device protocols currently supported in the platform to receive measures are SensorML, UL2.0 or MQTT, but new ones can be supported by developing plugins.

Complete documentation regarding new plugins implementation is listed in Codec Framework docs [4].

# Ultralight 2.0 (UL2.0)

The Ultralight 2.0 is an HTTP protocol implemented as an alternative to SensorML, reducing the overhead to the minimum to allow low bandwith devices sending measures and receiving commands to/from the IoT Platform.

## Measures

To send a measure, the device should send an HTTP POST request to the following URL:

POST http://<HOST>:<PORT>/d?k=<api-key>&i=<deviceID>&t=<timestamp>&ip=<command\_URL>&getCmd=1

Body: <measures\_data>

Where the query parameters are:

* k: service apikey (mandatory)
* i: device identifier (mandatory)
* t: measure timestamp, ISO format (UTC) (optional)
* ip: device command URL (optional) (HTTP endpoint)
* getCmd: parameter to indicate receiving pending commands (optional) (see 3.2 section)

Measures data, included as a query parameter in the POST request body, must follow the following format:

<alias>|<value>

Or

<timestamp>|<alias>|<value>

Where:

* <alias>: Device sensor identifier as specified in the model
* <value>: Device sensor measure value.
  + In the location specific case, follow the convention <latitude>/<longitude>
* <timestamp>: Measure timestamp

In order to send multiple measures in the same request, measures should be separated by a hash symbol:

<alias1>|<value1>**#**<alias2>|<value2>**#**...

Or

<timestamp>|<alias1>|<value1>**#**<alias2>|<value2>**#**...

Example:

POST http://<HOST>:<PORT>/?k=abcdfefeff1232df&i=1234

Body: t|1#loc|45.6789/23.3246#c|4657

## Commands

Commands received by the Platform (see [6] to learn how to send commands from your application) will be sent to the device using pooling or push strategies:

* **Pooling**: when a device sends a measure as explained above, it will include an extra query parameter getCmd=1, so the Platform will send the command in the response body
* **Push**: When the device has a Command URL predefined, the Platform will invoque a POST as soon as a command is received for this device. This URL can be updated following multiple ways
  + Updating the device “command\_url” parameter using SBC Portal or APIs
  + Updating the url while sending a measure by using the ip query parameter

Commands format that should be processed by the device, independently of the reception strategy, is:

<device>@<command\_name>|<param1>|<param2>|…

where

* device: the device name
* command\_name: the command name
* param: command parameter as value or <param\_name>=<value>

Example of command received by the platform (by using the Portal or APIs)

<paid:command dest="device1234" name=“setConfig"> <paid:cmdParam name=“firmwareVersion"> <swe:Text> <swe:value>2.1</swe:value> </swe:Text> </paid:cmdParam> </paid:command>

Example of command delivered to the device:

device1234@setConfig|firmwareVersion=2.1

### Command Response

Device should response to the command request by using the following body format:

<device>@<command\_name>|<param>|<param>

where

* device: the device name (same received in the request)
* command\_name: the command name (same received in the request)
* param: command response parameter as value or <param\_name>=<value>

Example:

device1234@set|processed|OK

In case the device is using the pooling strategy, it should POST the command response above as soon as it is executed.

### Predefined Ping command

The Platform web has a device diagnostic method in order to check that the device is working correctly. Device supporting push commands should implement the ping command as follows

Command request:

<idDevice>@ping

Required command response:

<idDevice>@ping|ping OK

# MQTT

## MQTT Protocol Specification

MQTT (Message Queue Telemetry Transport) is a publish-subscribe based light weight messaging protocol for use on top of the TCP/IP protocol. It is designed for connections with remote locations where a "small code footprint" is required and/or network bandwidth is limited. The Publish-Subscribe messaging pattern requires a [message broker](http://en.wikipedia.org/wiki/Message_broker). The broker is responsible for distributing messages to interested clients based on the topic of a message

The specification does not specify the meaning of "small code foot print" or the meaning of "limited network bandwidth".

For more information about MQTT 3.1 Specification, see [7]

## MQTT Implementation

As shown above, MQTT specs does not specify the messages format, so we have followed our UL2.0 simple conventions to implement topics and payloads.

IoT Platform GW acts as the MQTT Broker, so devices publish measures as MQTT Clients, and Subscribe to commands as MQTT Consumers.

### Measures

Devices can send individual or multiple measures. For single measures, the following topic must be used:

Topic: <api-key>/<device-id>/<alias>

Payload: <measure\_value>

Where…

Example:

Topic: abadfwfe12323dfsdf/1234/t

Payload: 25

For multiple measures, device must use the following:

Topic: <api-key>/<device-id>/multi

Payload: <alias1>|<value1>#<alias2>|<value2>#... (UL2.0 measures format)

Example:

Topic: abadfwfe12323dfsdf/1234/multi

Payload: t|25#c|15

### Commands

Following the same approach used for the Measures implementation, Commands follow the UL2.0 conventions.

In order to get pending commands, devices must publish a message without payload, using the topic:

<api-key>/<device-id>/cmdget

In order to receive commands, the topic to subscribe is:

<api-key>/<device-id>/cmd/+

The plus symbol means that there will be another level of topic in MQTT. An example of a message received:

822asijn7jwb9kn367fjz235/id234/cmd/set\_time

Payload format for commands will follow same format used in multi-measures, but will begin with the command ID which uniquely identify commands and responses.

cmdid|<cmdid>#<param1>|<value1>#..

An example may be:

cmdid|82ndsj28924hnsrh2932424#param1|value1#

To send responses to commands, devices have to publish an MQTT message using the topic:

Topic: <api-key>/<device-id>/cmdexe/<cmd>

Payload: cmdid|<cmdid>#<res\_param1>|<res\_param1\_value>#...

Where cmd is the name of the command. Response payload must include command ID.

An example of response to previous “set\_time” command :

Topic: 822asijn7jwb9kn367fjz235/id234/cmdexe/set\_time

Payload: cmdid|82ndsj28924hnsrh2932424#result|OK#

# SensorML

SensorML provides a common model for the description of devices and systems (resources), easily adaptable to any implementation needs. Additionally, it allows describing characteristics and capacities of the resources, parameters, location information, etc. It is a standard defined by the Open Geospatial Consortium (OGC).

SensorML is a description model to specify the features and capabilities of the sensor systems. SensorML provides a rich collection of metadata that can be mined and used for discovery of sensor systems and observation processes. This metadata includes identifiers, classifiers, constraints (time, legal, and security), capabilities, characteristics, contacts, and references, in addition to inputs, outputs, parameters, and system location.

Observation and Measurements (O&M) was defined also by OGC. O&M is a model to represent measurements coming from devices. It covers simple and complex models to describe these observations.

The complete SensorML specification can be found in [8]

# Glossary

|  |  |
| --- | --- |
| API: | Application Programing Interface. Unified Interface to Access to a particular software module. |
| HTTPS: | Secure http access. |
| IoT: | Internet of Things. |
| M2M: | Machine to Machine, applicatios in which both the communication endpoints are machines. |
| MQTT: | Message Queuing Telemetry Transport. Lightweight message Exchange protocol, aimed to sensor communications. |
| REST: | Representational State Transfer. Web interface based on XML and http messages. |
| SBC: | Smart Business Control. Device Communication Gateway for the Telefonica IoT Platform. |

# References

|  |  |
| --- | --- |
|  | QuickGuide-SBC-2.5.pdf |
|  | Technical Workshop.pdf |
|  | GWInterface-EN.2.5.pdf |
|  | Codec Framework docs |
|  | WebPortal\_customer\_en V2.5.pdf |
|  | RESTAPI 2.5.pdf |
|  | <http://mqtt.org/documentation> |
|  | <http://www.opengeospatial.org/standards/sensorml> |
|  |  |
|  |  |
|  |  |
|  |  |