

Librairie LiveObjects



Halim Bendiabdallah
Janvier 2021

Sommaire

Introduction	2
Installation	4
Installation de l'environnement de développement	4
Création d'un compte LiveObjects	4
Chargement de l'exemple	6
Exécution de l'application de test	8
Paramétrage de l'application et exécution de l'application de test	8
Structure de l'application	10
Initialisation de la connectivité cellulaire	10
Publication d'un message vers la plateforme LiveObjects	11
Programmation et envoi d'une commande	11

Introduction

Le document décrit l'installation ainsi que l'utilisation de la librairie LiveObjects sur les plateformes Hardware de STMicroelectronics.

La librairie LiveObjects s'appuie sur XCubeCellular. Il s'agit d'un middleware proposé par STMicroelectronics qui permet de développer des applications TCP/IP (HTTP, MQTT...) de manière transparente sur la base d'API.

La librairie LiveObjects est compatible avec la version 5.2 de XCubeCellular. Elle utilise des librairies additionnelles tel que :

- Json
- MBedTLS
- STM32 Network Library

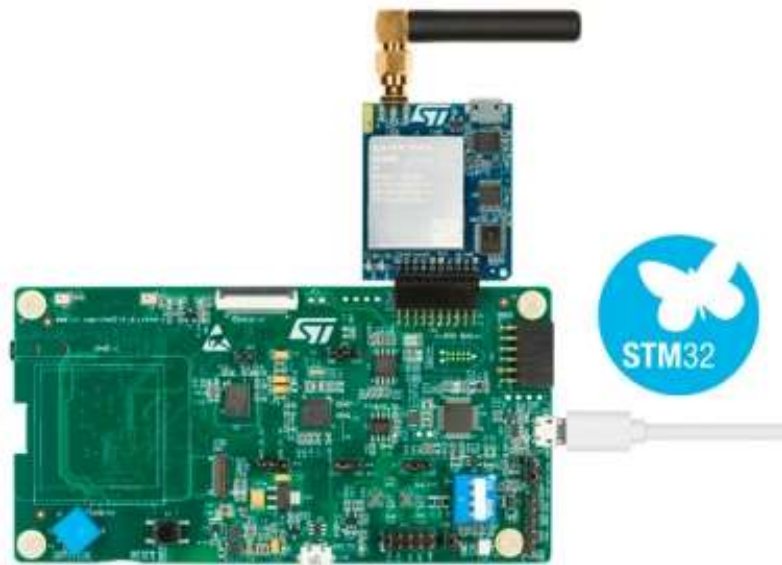
L'application s'exécute sur l'OS temps réel FreeRTOS (<https://fr.wikipedia.org/wiki/FreeRTOS>)

Les terminaux IoTs communiquent avec LiveObjects en MQTT. Cette pile est intégrée dans cette même bibliothèque.

Elle est validée sur les cartes suivantes :

- Discovery P-L496G-CELL02 + le module cellulaire SEQUANS GSM01Q
- STEVAL-STWINKT1 + le module cellulaire SEQUANS GSM01Q

Mais il est possible de porter la librairie sur d'autres plateformes STM32.



P-L496G-CELL02



STEVAL-STWINKT1

Installation

Installation de l'environnement de développement

L'environnement de développement utilisé est STM32CubeIDE. Il est basé sur l'IDE Eclipse. Il peut être téléchargé directement à l'adresse <https://www.st.com/en/development-tools/stm32cubeide.html>

La procédure d'installation est décrite pour l'ensemble des environnements dans le document intitulé **STM32CubeIDE Installation guide – User manuel** (https://www.st.com/resource/en/user_manual/dm00603964-stm32cubeide-installation-guide-stmicroelectronics.pdf)

La librairie peut être téléchargée depuis le Github Orange à l'adresse <https://github.com/Orange-OpenSource>. Le projet se nomme **LiveObjects-4-XCubeCellular** et est disponible sous licence BSD 2.

Pour un bon suivi du projet il est conseillé d'utiliser l'outil de gestion de version décentralisé **Git** (<https://fr.wikipedia.org/wiki/Git>).

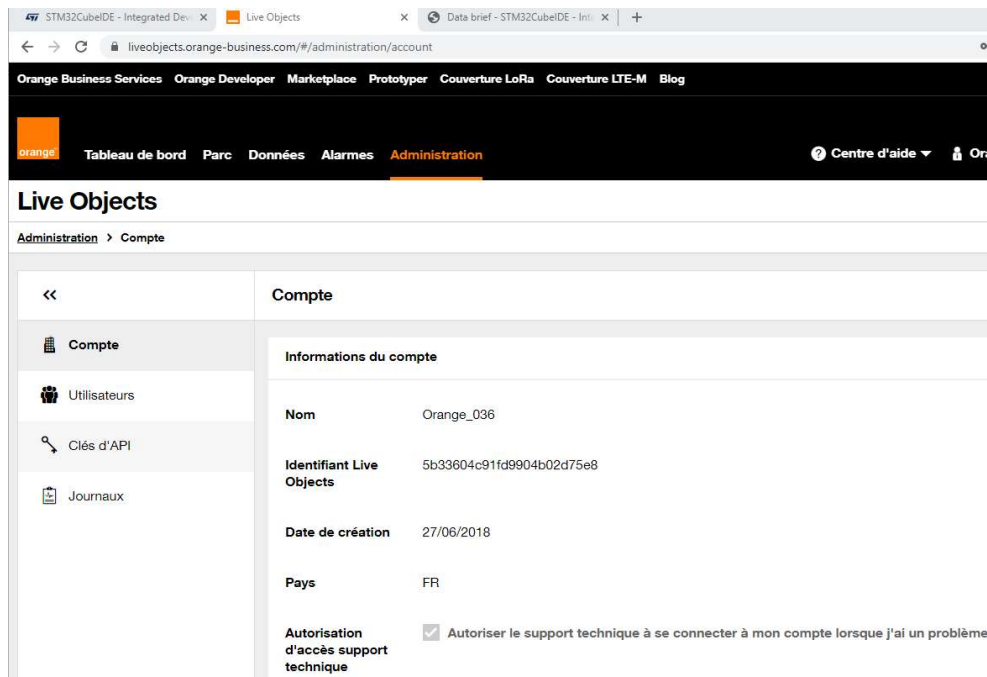
Pour des raisons de facilité d'intégration et d'exécution, le projet embarque à la fois la librairie LiveObjets mais aussi le middleware XCubeCellular. Contrairement à celle proposée sur le site de ST, celle-ci n'embarque que les éléments indispensables au fonctionnement sur LiveObjects.

Création d'un compte LiveObjects

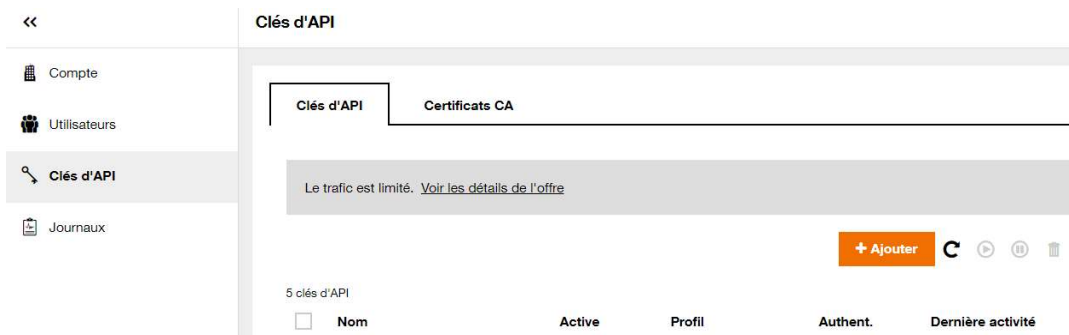
Le développement d'une application nécessite la création d'un compte LiveObjects. L'enregistrement à l'adresse <https://liveobjects.orange-business.com/#/liveobjects> est gratuit pour un nombre restreint d'objets.

Après enregistrement, s'assurer de bien sauvegarder la clé Master affichée à l'écran. Celle-ci n'est plus consultable par la suite.

- Les objets connectés utilisent principalement le MQTT pour s'interfacer avec la plateforme. Pour cela on doit créer une clé pour dialoguer à travers ce protocole. Aller sur le menu Administrateur et cliquer ensuite sur clés d'API à gauche de l'écran.



- Cliquer ensuite sur le bouton Ajouter



- Saisir les différents champs tels que le nom, la description, etc... puis sélectionner « Equipement MQTT » et cliquer sur créer.

Information * champ obligatoire

Nom *

Description

Valide à partir de ?

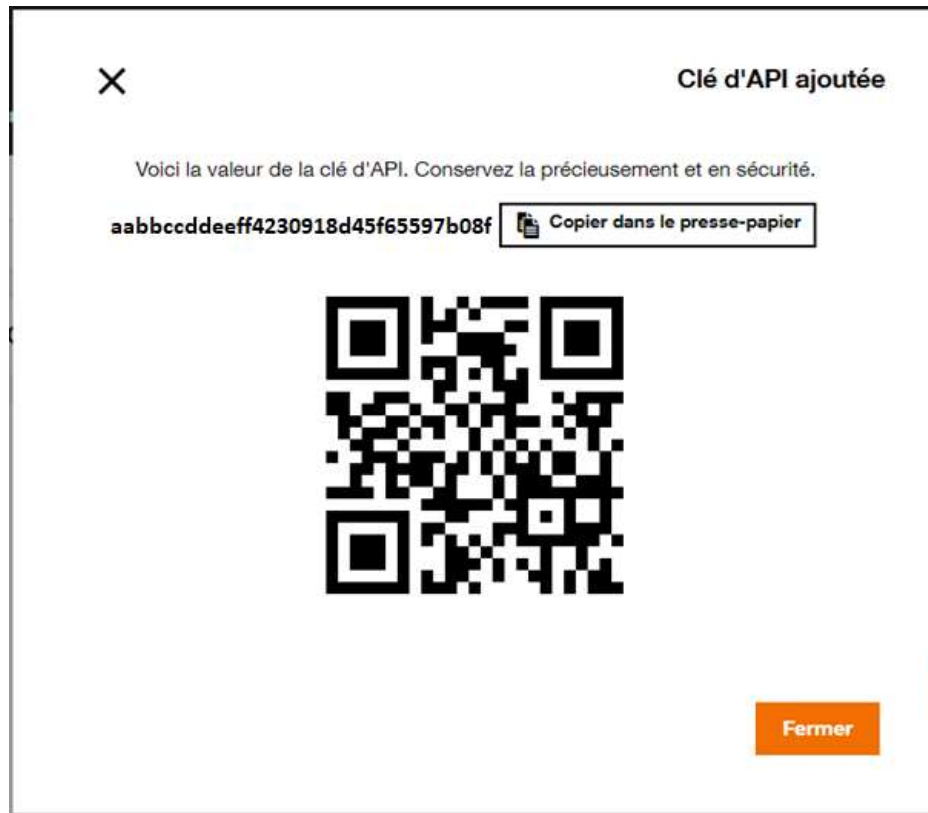
Valide jusqu'au ?

Profil ☒ Equipement MQTT ☐ Application ☐ Connecteur externe ☐ Personnalisé

Rôles

Nom	Description	Lecture	Ecriture

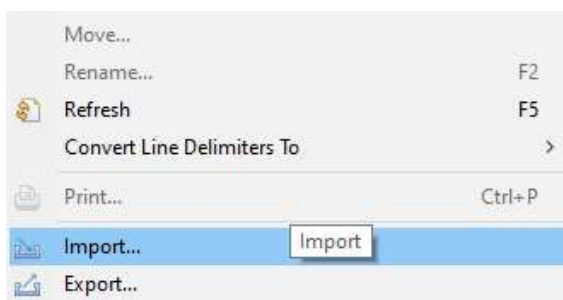
- La clé est ensuite générée. Il est nécessaire de la stocker ou de l'enregistrer pour son utilisation future au sein de l'application embarquée.



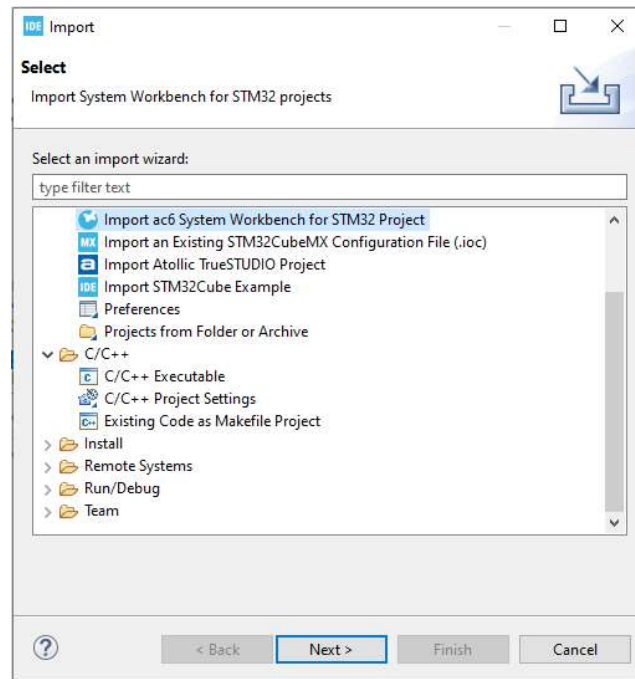
Chargement de l'exemple

La librairie est livrée avec un exemple d'application basique de connexion et d'échanges de données avec la plateforme LiveObjects.

Ouvrez l'environnement de développement STM32CubeIDE, puis cliquez sur File->Import



La fenêtre de sélection de type de projet apparaît

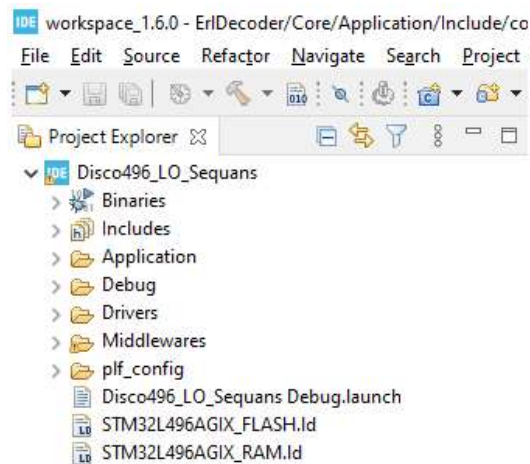


Cliquer sur le bouton Directory “**Import ac6 System Workbench for STM32 Project**” puis sélectionner le répertoire :

“\STM32CubeExpansion_CELLULAR_V5.2.0\Projects\STM32L496G_Discovery\Demonstrations\Cellular\IDE\STM32CubeIDE\Disco496_LO_Sequans”.

Pour la version STWINKT1:

“\STM32CubeExpansion_CELLULAR_V5.2.0\Projects\STWINKT1\Demonstrations\Cellular\IDE\STM32CubeIDE\STWINKT1_SEQUANS”.

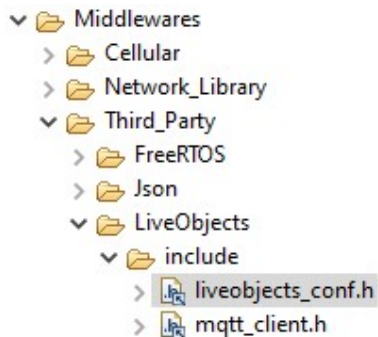


Le projet apparaît dans l’explorateur de projet comme ci-dessus.

Exécution de l'application de test

Paramétrage de l'application et exécution de l'application de test

Tous les paramètres de connexion à LiveObjects se trouvent définis dans un seul fichier qui porte le nom **liveobjects_conf.h**







- Saisissez le mot de passe généré précédemment (clé d'API) à l'occurrence **MQTTCLIENT_DEFAULT_PASSWORD**
- Saisissez le nom de l'objet (clientID) à l'occurrence **MQTTCLIENT_DEFAULT_CLIENTID**
- Modifier le StreamID si besoin à l'occurrence **MQTT_DEFAULT_STREAM_ID**
- Entrer le ou les certificats si vous utilisez MBEDTLS


```
23 #ifndef __cplusplus
24 extern "C" {
25 #endif
26
27 /* Includes -----*/
28 /* Exported constants -----*/
29
30 /* -----*/
31 /* MQTT Parameters TO BE DEFINED
32 /* These parameters can also be updated by the setup menu */
33 /* -----*/
34 /* URL of MQTT dashboard: TO BE DEFINED - to get from the MQTT server account */
35 #define MQTTCLIENT_DEFAULT_SERVER_NAME ((uint8_t *)"liveobjects.orange-business.com") /* MQTT
36 #define MQTTCLIENT_DEFAULT_USERNAME ((uint8_t *)"json+device") /* MQTT
37 #define MQTTCLIENT_DEFAULT_PASSWORD ((uint8_t *)"aabbccddeeff4230918d45f65597b08f") /* MQTT
38
39 /* -----*/
40 /* Other MQTT parameters
41 /* These parameters can also be updated by the setup menu */
42 /* -----*/
43 #define MQTTCLIENT_DEFAULT_CLIENTID ((uint8_t *)"seqv2") /* MQTT client id
44 #define MQTTCLIENT_DEFAULT_SYNCHRO_PERIOD ((uint8_t *)"60000") /* MQTT synchronization per
45 #define MQTT_DEFAULT_STREAM_ID ((uint8_t *)"Data")
46 #define MQTTCLIENT_DEFAULT_ENABLED ((uint8_t *)"1") /* MQTTclient enabled by de
47 #define MQTTCLIENT_DEFAULT_PARAM_NB (7U) /* number of MQTTclient set
48 #define MQTTCLIENT_KEEP_ALIVE 3000U /* MQTT keep-alive duration
49
50 #if (USE_MBEDTLS == 1)
51 #define MQTTCLIENT_DEFAULT_SERVER_PORT 8883
52 #else
53 #define MQTTCLIENT_DEFAULT_SERVER_PORT 1883
54 #endif /* (USE_MBEDTLS == 1) */
55
56 #if (USE_MBEDTLS == 1)
57 #define MQTTCLIENT_ROOT_CA \
58 { \
59 45, 45, 45, 45, 45, 66, 69, 71, 73, 78, 32, 67, 69, 82, 84, 73, 70, 73, 67, 65, 84, 69,
60 45, 45, 45, 45, 45, 10, 77, 73, 73, 68, 114, 122, 67, 67, 65, 112, 101, 103, 65, 119, 73,
61 66, 65, 103, 73, 81, 67, 68, 118, 103, 86, 112, 66, 67, 82, 114, 71, 104, 100, 87, 114, 74,
62 87, 90, 72, 72, 83, 106, 65, 78, 66, 103, 107, 113, 104, 107, 105, 71, 57, 119, 48, 66, 65,
```

Compiler et exécuter le programme. Celui-ci, après connexion, enverra l'IMEI, l'IMSI ainsi que le nom du module au cloud LiveObjects sous le format JSON.



Il est possible de vérifier le statut de la connexion ainsi que les données envoyées sur le tableau de bord LiveObjects.







<input type="checkbox"/>	Auto-created device (mqtt / urn:lo:nsid:mqtt:liv eboosterV2)	urn:lo:nsid:mqtt:liv / eboosterV2	MQTT	 Hors ligne	06/02/2020 18:20:33
<input type="checkbox"/>	Auto-created device (mqtt / urn:lo:nsid:mqtt:se seqv2)	urn:lo:nsid:mqtt:se / qv2	MQTT	 En ligne	01/02/2021 12:46:45



Du 00 : 00 : 00  Au 23 : 59 : 59 


Sélectionner un filtre ci-dessous 

Filtres

2 réponses  

Date	Source	Stream	Valeur	Tags
01/02/2021 12:45:45	clientId : seqv2 deviceId : urn:lo:nsid:mqtt:seqv2	 Data 	{ "imei": "123456789", "opname": "sequans001", "imsi": "987654321" }	- 
26/01/2021 21:09:50	clientId : seqv2 deviceId : urn:lo:nsid:mqtt:seqv2	 Data 	{ "imei": "123456789", "opname": "sequans001", "imsi": "987654321" }	- 

 1 

20 

Structure de l'application

L'application s'exécute sur FreeRTOS. L'ensemble des librairies se trouve dans le répertoire projet « Third_Party » lui-même présent dans « Midllewares ».

On y trouve le code source de:

- L'OS FreeRTOS
- De la librairie opensource Json-parser
- De MbedTLS
- De la librairie LiveObjects

A noter que la librairie LiveObjects intègre sa propre pile MQTT. En effet celle-ci est intrinsèque à la librairie.

Les librairies de ST se trouvent dans les répertoires « Cellular » et « Network_Library ».

Il est recommandé de ne pas modifier l'ensemble de ces fichiers sous peine de créer des dysfonctionnements de l'application final.

L'application est placée dans le répertoire « Application/User ». On y retrouve le fichier « main.c » qui instancie le Thread principal qui, lui-même instancie celui qui gère l'exécution de la partie cellulaire (liveobjects_freertos.c).

Initialisation de la connectivité cellulaire

Nous nous intéresserons par la suite au processus présent dans le fichier liveobjects_freertos.c

L'ensemble des APIs LiveObjects est défini dans le fichier mqtt_liveobjects.h. Celui-ci est documenté sous Doxygen.

Au démarrage de la tâche du Thread LiveObjects, on crée une instance LiveObjects par l'appel de la fonction `lo_create()`. Si elle réussit, elle retourne un handle qui sera utilisé pour tous les appels des API.

Il faut ensuite affecter la configuration par l'appel de `lo_set_default_config(handle)`; Cette fonction affecte au handle tous les paramètres de connexion définis dans le fichier de configuration « liveobjects_conf.h »

Il est néanmoins possible de les définir ad hoc grâce à la fonction `lo_set_default_config`. Suite à cela, nous pouvons nous connecter au réseau par l'appel de la fonction `lo_connect_mqtt(handle)`. Cette fonction initialise le module, accroche le réseau cellulaire puis initie une connexion MQTT vers LiveObjects. Dans le cas où elle réussit, on définit les différents callback et subscriber.

Dans notre exemple, on souhaite être notifié de la réception des commandes :

```
lo_set_command_callback(handle, lo_command_callback);  
lo_command_subscribe(handle);
```



Publication d'un message vers la plateforme LiveObjects

Pour le bon fonctionnement de l'envoi et de la réponse d'un message MQTT, il est indispensable de vérifier de manière périodique que la connexion est toujours active grâce à la fonction `lo_is_connected(handle)` et d'appeler ensuite la méthode `lo_synchronize(handle)` qui permet un traitement des échanges MQTT et le cas échéant de déclencher les callbacks.

Dans notre exemple on publie le message Json qui comprend l'IMEI, l'IMSI ainsi que le nom du device par les procédures suivantes :

```
sprintf(buffer, (const char*)
"{\"opname\":\"sequans002\", \"imei\":\"%s\", \"imsi\":\"%s\"}", cellular_info.imei,
cellular_sim_info.imsi);

lo_publish_data(handle, (const char*) buffer);
```

Programmation et envoi d'une commande

Il est possible, dans notre exemple, de recevoir des commandes provenant de LiveObjects.

Depuis le site LiveObjects, aller sur la liste des devices puis sélectionner l'équipement sur lequel la commande doit être envoyée



<input checked="" type="checkbox"/>	Auto-created device (mqtt / seqv2)	urn:lo:nsid:mqtt:seqv2	/	MQTT	Hors ligne	06/02/2021 19:29:54
<input type="checkbox"/>	Auto-created device (mqtt / mart-charger)	urn:lo:nsid:mqtt:mart-charger	/	MQTT	Hors ligne	11/12/2020 14:41:04

L'écran de supervision de l'équipement apparaît. On remarquera une liste d'action possible à gauche de l'écran dont celui d'envoi de commandes (Downlink).

Cliquer sur cet élément pour rentrer dans le menu de saisie.

<<

↑ Auto-created device (mqtt / seqv2) - urn:lo:nsid:mqtt:seqv2

Supervision

Identité

Commandes (Downlink)

Firmwares

Configurations

Journal

Statut de l'équipement

Dernière communication
06/02/2021 19:29:54

Aucune règle d'alarme associée à cet équipement.

+ Ajouter une règle silent machine

Hors ligne

Interface - MQTT

MQTT Client ID
seqv2

Dernière adresse IP
92.184.104.124 (port : 49153)

Puis ensuite sur le bouton « Ajouter une commande »

Enregistrer une nouvelle commande
* champ obligatoire

MQTT

Requête *

Reset

Argument

state

true

(1)

(1): Une valeur booléenne (true, false) sera fournie comme telle en argument de la commande. Si vous souhaitez la fournir en tant que chaîne de caractères, il faut l'entourer de " ".

Durée max de l'état PENDING *

0

x

Jours

0

x

Heures

1

x

Minutes

?

Niveau d'acquiescement *

Aucun

Applicatif (Equipement a répondu)

?

Délai d'acquiescement

1

Minutes

?

Politique en cas d'échec

☐ Réessayer

1

fois

?

Annuler

Valider

La commande est composée du nom de la requête (balise json « req »), puis de l'argument (dans notre cas de « state » avec comme valeur « true »). Il est aussi obligatoire de définir la durée maximum de Pending.

Suite à cela la validation publiera le message en MQTT. L'application de test recevra la commande. La callback `lo_command_callback(int64_t cid, const char* req, const json_value* arg)` est appelé. On pourra exécuter les tâches associées à la requête (publié dans la variable *req* et *arg*)

On n'oubliera pas d'appeler la fonction `lo_publish_ack_command` pour valider ou invalider la commande.