EMIT User Guide

Monitoring Tool of MEASURE Platform

EMIT Overview

EMIT is a set of web services that make possible to manage MQTT clients. In fact, EMIT allows users to define some MQTT broker connections, to create some MQTT clients and to specify some MQTT callbacks i.e. the MQTT message processes attached to MQTT clients and launched on MQTT message reception. Moreover, EMIT makes possible to update MQTT client states i.e. to connect/disconnect from MQTT brokers, to subscribe/unsubscribe to given topics and to publish messages on given topics. Finally, EMIT makes possible to view the different MQTT client state updates and MQTT messages received and stored by some MQTT client callbacks.

EMIT is provided as a web application i.e. with a HTML frontend to these web services that embeds a JavaScript library that correspond to the HTTP clients of these web services. EMIT is also provided such HTTP clients as a Java library. The usage of such HTTP client is illustrated thanks to the web interface use cases below.

EMIT Installation

EMIT code base is available on GitHub at https://github.com/jeromerocheteau/emit. It consists of a set of Java projects managed by the means of Maven. It can be compiled, packaged and deployed thanks to the following commands:

- git clone https://github.com/jeromerocheteau/emit.git
- cd emit/emit-monitoring/
- mvn clean compile package install tomcat7:redeploy

EMIT runs on a Java application container such as Apache Jetty, Apache Tomcat, Apache TomEE, IBM Webspehere, RedHat Jboss, Oracle Glassfish, etc.

The deployment settings can be modified into the Maven project description (pom.xml file) in order to specify the application server onto the EMIT instance will be depoyed to and its credentials

Dependencies

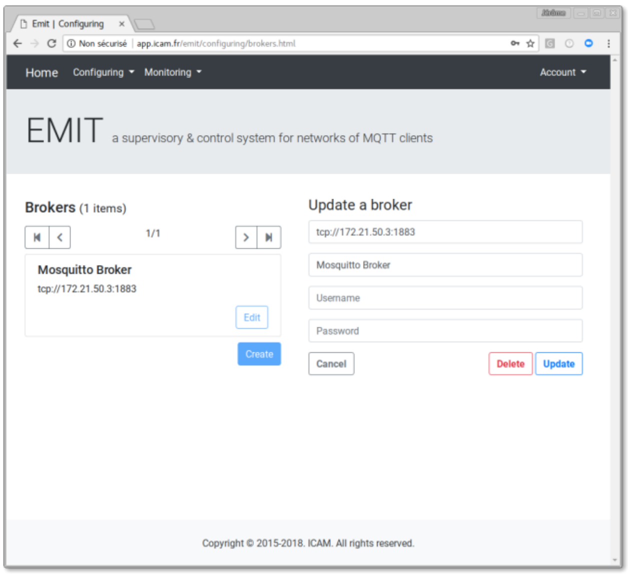
EMIT requires that the following dependencies are already installed as shared libraries within the Java application container:

- com.google.code.gson:gson:2.4
- org.mongodb:mongodb-driver:3.4.2
- org.mongodb:mongodb-driver-core:3.4.2
- org.mongodb:bson:3.4.2
- org.eclipse.paho:org.eclipse.paho.client.mqttv3:1.2.0
- com.github.jeromerocheteau:jdbc-servlet-api:1.0

An instance of EMIT is running at the URL http://app.icam.fr/emit and is accessible according to the following credentials username: measure@emit.icam.fr and password: m3@suR

Editing MQTT Clients

EMIT provides 2 main use cases in order to configure MQTT client networks: the first one consists in defining the connection settings to a given MQTT broker by specifying its URI and eventually its username/password credentials (see Illustration). EMIT also provides a use case to define MQTT clients merely by specifying its already registered MQTT broker (see Illustration).



EMIT a supervisory & control system for networks of MQTT clients

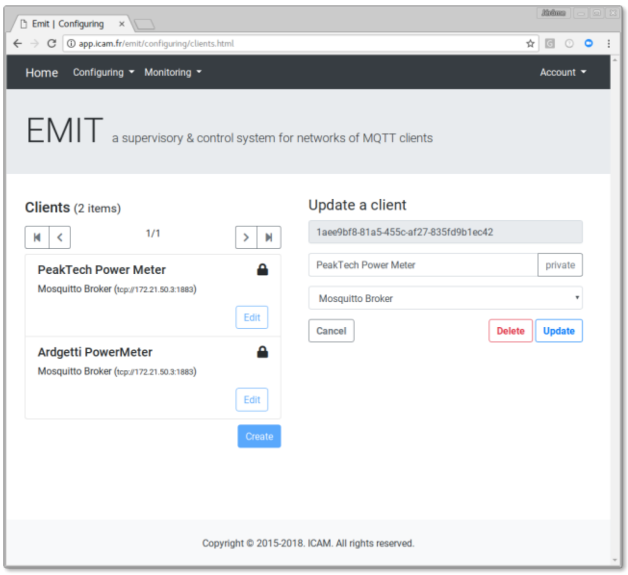
Brokers (1 items) 1/1

Mosquitto Broker
tcp://172.21.50.3:1883

Update a broker
tcp://172.21.50.3:1883
Mosquitto Broker
Username
Password

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 1: Editing MQTT Broker



EMIT a supervisory & control system for networks of MQTT clients

Clients (2 items) 1/1

PeakTech Power Meter
Mosquitto Broker (tcp://172.21.50.3:1883)

Ardgetti PowerMeter
Mosquitto Broker (tcp://172.21.50.3:1883)

Update a client
1aee9bf8-81a5-455c-af27-835fd9b1ec42
PeakTech Power Meter private
Mosquitto Broker

Copyright © 2015-2018. ICAM. All rights reserved.

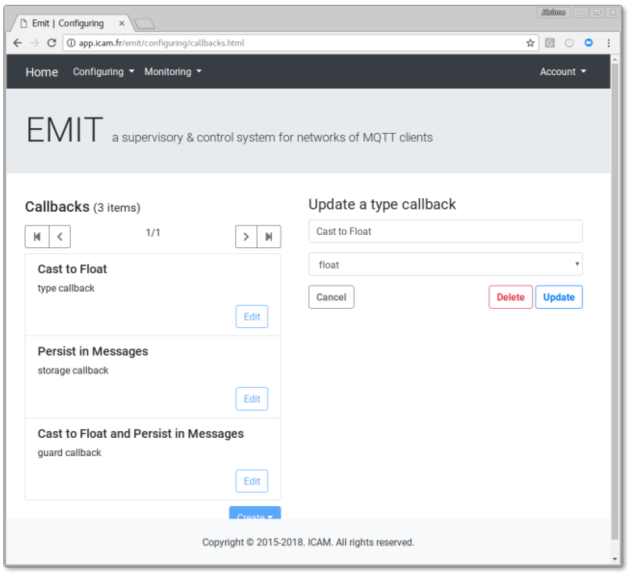
Illustration 2: Editing MQTT Client

Updating MQTT Client Callbacks

EMIT makes possible to define processes of MQTT message that are received by some MQTT clients. Such processes are called MQTT callbacks and EMIT provides several built-in MQTT callback edition use cases. Every MQTT callbacks returns a Boolean value according to the fact that the message process ends successfully or not. The 5 main MQTT callbacks are described below.

The 1st MQTT callback consists in specifying the data type of the MQTT message payload: users specify the data type from a built-in type selection list (see Illustration).

The 2nd MQTT callback consists in a MQTT topic filter or pattern matcher: user defines such pattern (see Illustration).



EMIT a supervisory & control system for networks of MQTT clients

Callbacks (3 items) 1/1

Cast to Float
type callback

Persist in Messages
storage callback

Cast to Float and Persist in Messages
guard callback

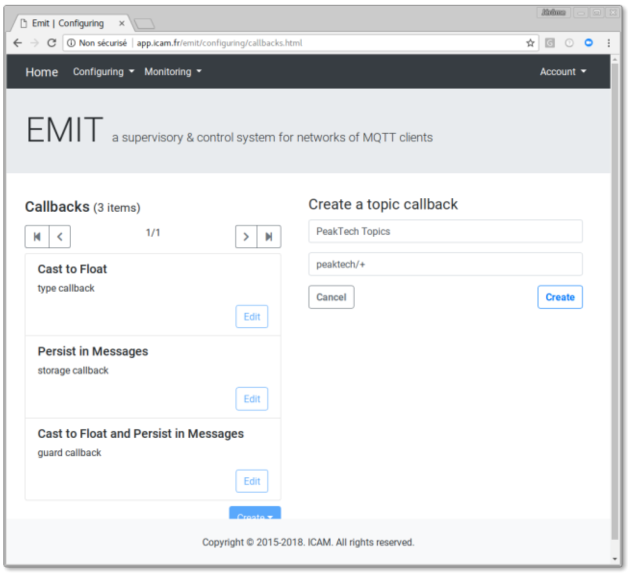
Update a type callback

Cast to Float

float

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 3: Editing MQTT Type Callback



EMIT a supervisory & control system for networks of MQTT clients

Callbacks (3 items) 1/1

Cast to Float
type callback

Persist in Messages
storage callback

Cast to Float and Persist in Messages
guard callback

Create a topic callback

PeakTech Topics

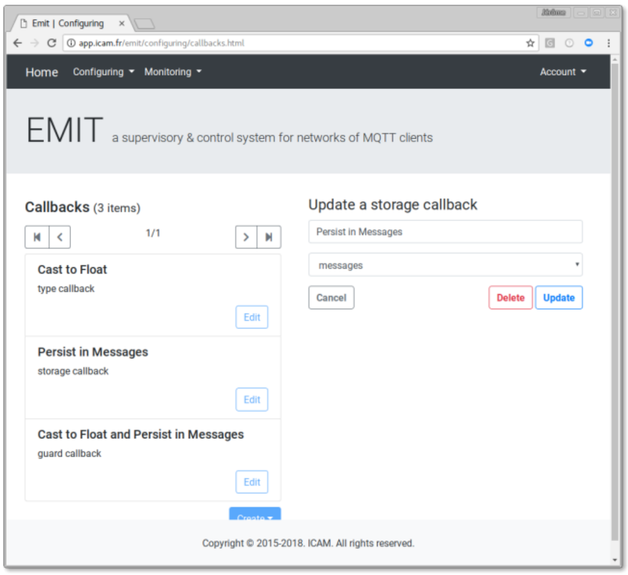
peaktech/+

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 4: Editing MQTT Topic Callback

The 3rd MQTT callack consists in persisting messages within a database: users speficy the collection that messages will be stored into by selecting this collection between the *messages* collection and the *failures* one (see Illustration).

The 4th MQTT callback consists in verifying a condition over MQTT message payloads: users define the value, its type and the comparison operator among this operator set { =, ≠, < , >, ≤, ≥ } (see Illustration). The MQTT feature callback returns true if and only if the condition is satisfied by the MQTT message payload.



EMIT a supervisory & control system for networks of MQTT clients

Callbacks (3 items) 1/1

Cast to Float
type callback

Persist in Messages
storage callback

Cast to Float and Persist in Messages
guard callback

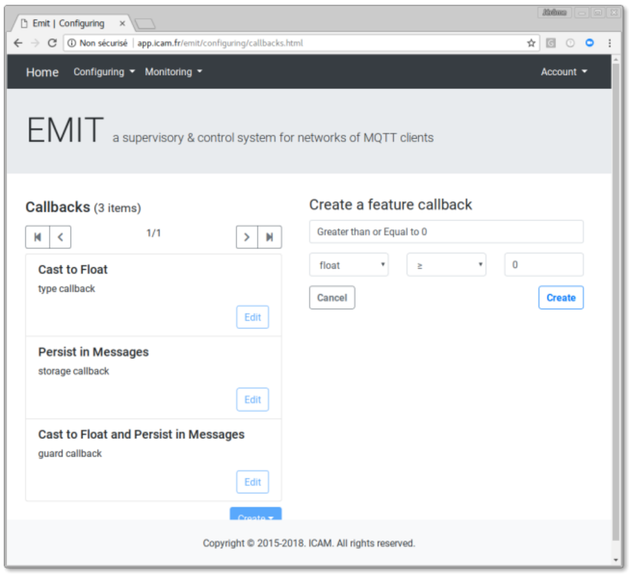
Update a storage callback

Persist in Messages

messages

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 5: Editing MQTT Storage Callback



EMIT a supervisory & control system for networks of MQTT clients

Callbacks (3 items) 1/1

Cast to Float
type callback

Persist in Messages
storage callback

Cast to Float and Persist in Messages
guard callback

Create a feature callback

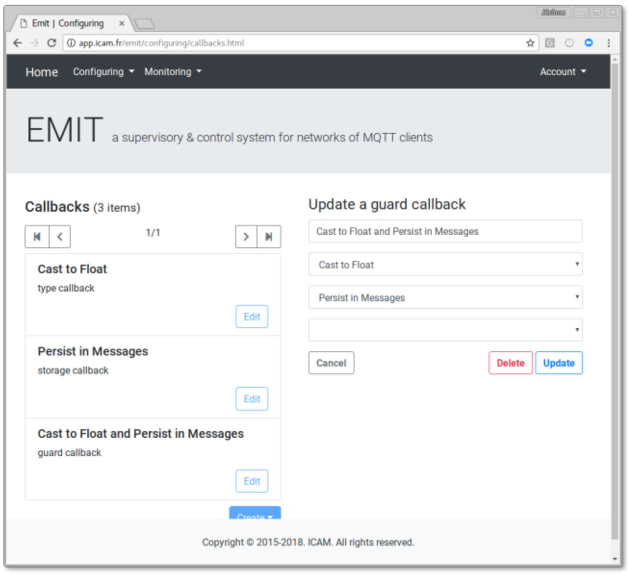
Greater than or Equal to 0

float ≥ 0

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 6: Editing MQTT Feature Callback

The 5th and last MQTT callback consists in a composite callback as it makes possible to test a first MQTT callback and to dispatch the process flow to a second callback if the first callback ended successfully and, eventually, to a third callback otherwise. Users have then to select the *test*, *success* and *failure* callbacks from the selection list of the already defined MQTT callbacks (see Illustration).


Emit | Configuring

app.icam.fr/emit/configuring/callbacks.html

Home Configuring Monitoring Account

EMIT a supervisory & control system for networks of MQTT clients

Callbacks (3 items)

1/1

Cast to Float
type callback
Edit

Persist in Messages
storage callback
Edit

Cast to Float and Persist in Messages
guard callback
Edit

Create

Update a guard callback

Cast to Float and Persist in Messages

Cast to Float

Persist in Messages

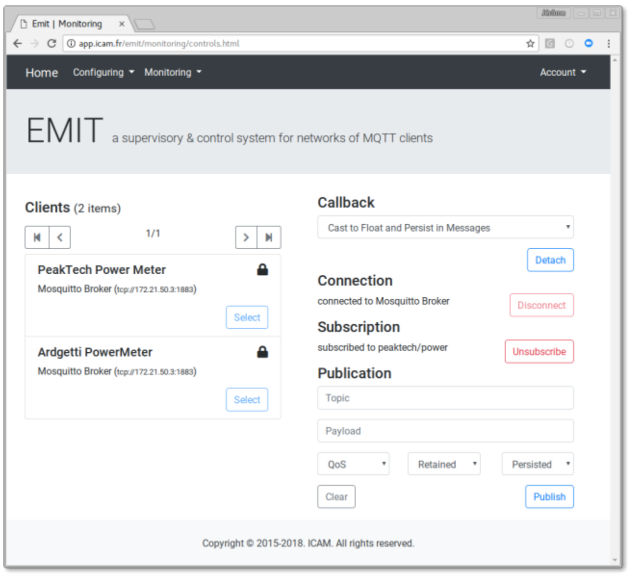
Cancel Delete Update

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 7: Editing MQTT Guard Callback

Updating MQTT Client States

The main EMIT use case consists in updating the different MQTT client states. In fact, EMIT makes possible to connect or disconnect a MQTT client from its MQTT broker. EMIT makes possible to subscribe or unsubscribe to a given topics from its related MQTT broker. EMIT makes also possible to publish a message to a given topic to its related broker. In addition, EMIT makes possible to attach or detach a MQTT callback to or from a MQTT client (see Illustration).


Emit | Monitoring

app.icam.fr/emit/monitoring/controls.html

Home Configuring Monitoring Account

EMIT a supervisory & control system for networks of MQTT clients

Clients (2 items)

1/1

PeakTech Power Meter
Mosquitto Broker (tcp://172.21.50.3:1883)
Select

Ardgetti PowerMeter
Mosquitto Broker (tcp://172.21.50.3:1883)
Select

Callback

Cast to Float and Persist in Messages

Detach

Connection
connected to Mosquitto Broker
Disconnect

Subscription
subscribed to peaktech/power
Unsubscribe

Publication

Topic

Payload

QoS Retained Persisted

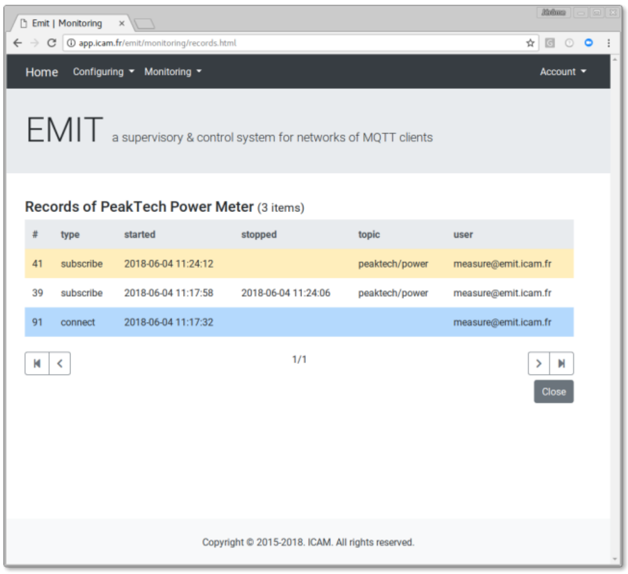
Clear Publish

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 8: Updating MQTT Client States

Viewing MQTT Client State Updates & Messages

EMIT provides 2 other use cases: EMIT makes possible to retrieve the different MQTT client state updates (see Illustration) and EMIT makes possible to retrieve the MQTT messages persisted into the embedded database engine by the means of MQTT storage callbacks that have been attached to MQTT clients (see Illustration).



EMIT a supervisory & control system for networks of MQTT clients

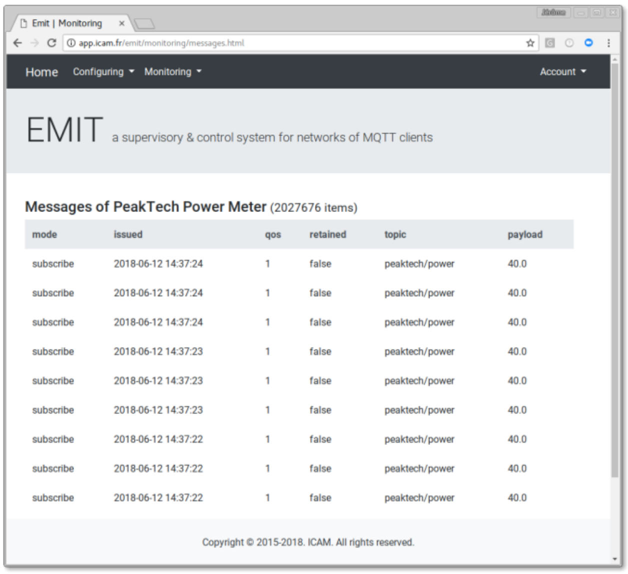
Records of PeakTech Power Meter (3 items)

#	type	started	stopped	topic	user
41	subscribe	2018-06-04 11:24:12		peaktech/power	measure@emit.icam.fr
39	subscribe	2018-06-04 11:17:58	2018-06-04 11:24:06	peaktech/power	measure@emit.icam.fr
91	connect	2018-06-04 11:17:32			measure@emit.icam.fr

1/1

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 9: Viewing MQTT Client State Updates



EMIT a supervisory & control system for networks of MQTT clients

Messages of PeakTech Power Meter (2027676 items)

mode	issued	qos	retained	topic	payload
subscribe	2018-06-12 14:37:24	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:24	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:24	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:23	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:23	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:23	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:22	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:22	1	false	peaktech/power	40.0
subscribe	2018-06-12 14:37:22	1	false	peaktech/power	40.0

Copyright © 2015-2018. ICAM. All rights reserved.

Illustration 10: Viewing MQTT Client Messages