# Simio API Note: MQTT Steps

April 2020 (Dhouck)

Contents

[Simio API Note: MQTT Steps 1](#_Toc32842374)

[Overview 2](#_Toc32842375)

[Some Background Information on MQTT 3](#_Toc32842376)

[Doing MQTT with the Simio MQTT Steps API 4](#_Toc32842377)

[MQTT Steps Code Overview 5](#_Toc32842378)

[Running the Model 6](#_Toc32842379)

[Notes on Use 6](#_Toc32842380)

[The Experiment (multi-thread) Problem. 6](#_Toc32842381)

[Adding Logic 6](#_Toc32842382)

[TroubleShooting 8](#_Toc32842383)

# Overview

This API Note describes a User Extension that permits Simio to communicate using the popular IoT MQTT communications protocol.

The techniques discussed here provide the following:

1. A Simio Process Step that Publishes.
2. A Simio Subscribe Element that Fires an Event when a MQTT topic arrives.

For testing, an Mqtt Test utility is provided, as well as a Simio example model.

## Some Background Information on MQTT

MQTT is a lightweight Publish and Subscribe protocol that is used for IoT communications. Communications conversations each have a unique Topic, that is hierarchical and represented with slashes. A packet information conveyed about a topic is called a Payload.

For example, if we are communicating information about information in a plant, a Topic might be of the form Location/Machine/Information So, for example roughing-pulpit/rollstand-7/speed. If our client subscribes to this topic, it might get a Payload of “Setpoint=23,Average=22.4”. But you can see that it entirely up to the designers of the communications to specify the formats for the Topics and Payloads.

Note that the topics are case-sensitive, so it has become a convention to make the topics lower-case… but it is of course up to you.

The process responsible for brokering these conversations is called a MQTT Server. When a client wants to listen to a conversion, it Subscribes to a Topic. If it also wants to contribute to the conversation, it Publishes a Payload with that Topic.

The missing piece of information is how to communicate with the Server. This is done by referencing its address, which is a URL and Port. As you may infer from this, the underlying protocol is IP.

# Doing MQTT with the Simio MQTT Steps API

The implementation of MQTT for Simio involves:

1. A MqttPublishElement
2. A MqttSubscribeElement
3. A MqttPublish Step

The MQTT Subscribe Element has properties for the MQTT Server address (URL and Port) and a Topic. When a payload arrives with that topic, a Simio Event is fired.

A MQTT Publish Step references a MQTT Publish Element, which contains the MQTT Server information. When a Simio Entity enters the Step, the Payload associated with the Step is published under the given Topic.

In the example model, one of the Steps is place at the entrance of the Simio Server. When the Entity enters the server, it sends its name as the Payload with the topic of server1/enter.

## MQTT Steps Code Overview

The code for the Process Step is in the

# Running the Model

There are some moving parts in this, but the setup is rather straightforward:

This was tested with the popular MQTT Server called Mosquitto Server ( <https://mosquitto.org> ).

The MQTT client-side code for the steps was written using the NuGet package MQTTNet (by Christian Cratky)

The steps to get this running are:

Download and install the Mosquitto server as a service and start it.

Place the MqttSteps.DLL in your {username}/Documents/SimioUserExtensions folder.

Launch the MqttTest utility.

Open the Simio project.

Now do the following:

1. MqttTest utility: Subscribe to server1/enter and server1/exit
2. MqttTest utility: Set the Publish topic to entity/launch
3. Simio: Start the Simio simulation
4. MqttTest utility: Press Publish
5. Simio: Observe that an Entity is emitted from the source, and when it hits the Server1…
6. MqttTest utility: … Observer that payloads for server1/enter and server1/exit are logged.

# Notes on Use

## The Experiment (multi-thread) Problem.

## Adding Logic

# TroubleShooting

## Make sure the Mosquitto Server/Broker is running.

## If the Server is on a remote computer, check your firewall.

Check the Mosquitto Server using utilities such as MQTT Explorer (Windows)