VERSION 1.0
DECEMBER 14, 2017

MQTT CONNECTOR PROFESSIONAL

APPLICATION NOTE 01

Publish data example

4EACH S.R.O.
WWW.4EACH.CZ

Copyright © 2017 4Each s.r.o.

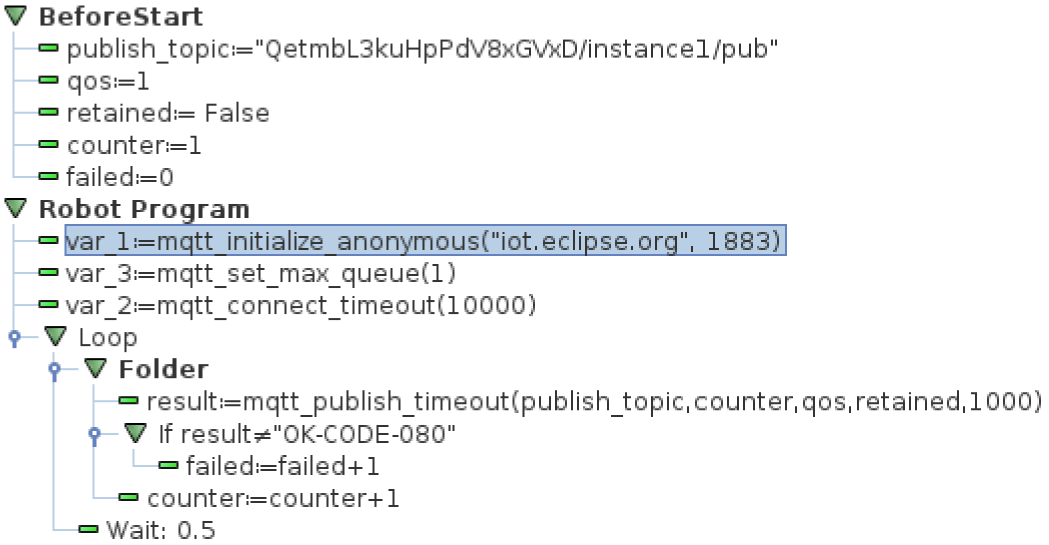
APPLICATION NOTE #1

In this series of application notes, we will go through basic usage of MQTT Connector Professional URCap extension. We will cover messaging fundamentals, like message sending and subscribing.

In this tutorial, we will learn how to send message to broker.

Usage of this URCap is realized through calling each function using URScript. This can be done using "Script Code" control, or as expression for variable assignment. Later has significant advantage, because you can easily keep track of the returned value. In most cases, return value consist of status code, which can be compared to gain the ability to react to unexpected errors, such as loss of connection. Consult status code chart for more details.

PROGRAM BODY



```
BeforeStart
  publish_topic:="QetmbL3kuHpPdV8xGVxD/instance1/pub"
  qos:=1
  retained:= False
  counter:=1
  failed:=0
Robot Program
  var_1:=mqtt_initialize_anonymous("iot.eclipse.org", 1883)
  var_3:=mqtt_set_max_queue(1)
  var_2:=mqtt_connect_timeout(10000)
  Loop
    Folder
      result:=mqtt_publish_timeout(publish_topic,counter,qos,retained,1000)
      If result≠"OK-CODE-080"
        failed:=failed+1
      counter:=counter+1
    Wait: 0.5
```

NOTE DESCRIPTION

Before start sequence initializes variables, to be later used in program body.

- publish_topic – MQTT topic designation as string, under which messages are published
- qos – Quality of service level. Accepted values are 0, 1, and 2 as integer
- retained – Boolean value specifies whether published message is retained
- counter – Variable integer value to be published
- failed - Variable integer value to keep track of failed message send attempts

ROBOT PROGRAM

The main program body is as following (note that robot program loops forever property is disabled)

mqtt_initialize_anonymous initializes connection details. In this instance, anonymous access is used. Function takes two parameters, MQTT broker **address** and **port**. Consult API documentation for details about access using credentials.

mqtt_set_max_queue – sets maximum outgoing messages queue length. This is important in case of connection loss, because every queued message is send after connection is restored. This can cause temporary network overload. Parameter with value 0 sets unlimited queue length.

Copyright © 2017 4Each s.r.o.

mqtt_connect_timeout – opens network connection between robot and MQTT broker. Function takes one parameter – **timeout**, which ensures maximum blocking duration of function call. Time is defined in milliseconds, and changes in 100ms increments.

Afterwards, program jumps into endless loop. In every iteration, following happens:

mqtt_publish_timeout – sends message to broker. Function takes 5 parameters:

- topic – topic under which the message is published
- message – payload to be send
- qos – quality of service
- retained – sets retained flag of the message
- timeout - maximum blocking duration of function call

For topic, qos and retained we will use already initialized variables. As message payload counter variable is used.

After broker response or timeout period (whichever comes first), program jumps to IF clause. It checks the publish result. If it was not successful, failed counter is incremented. In every case, counter variable is incremented.

FOOTNOTES

- Program doesn’t check whether broker is connected or not. It tries to connect, and then loops forever. However, if connection does become available in the future, it connects automatically, i.e. it performs automatic reconnect.
- Program never stops by itself. It needs to be stopped.
- Return values from function are stored in variables, but never used. They are set like this for clarity and ease of debugging thanks to variables lookup table in Polyscope.