mycontroller-org / 2mqtt Public

MQTT Bridge

Apache-2.0 license

10 stars 2 forks

Star Notifications

Code Issues Pull requests Actions Projects Security Insights

main Go to file

jkandasa ... 3 weeks ago

View code

2mqtt

golangci-lint passing | publish container images passing | publish executables passing

2mqtt is MQTT bridge. You can convert the serial, ethernet to MQTT

Supported Providers

README.md

- `ethernet` to `MQTT`
- `raw` - sends the serial,ethernet messages to mqtt as is
  - `serial` to `MQTT`
  - `ethernet` to `MQTT`
  - `http` to `MQTT`

Download

Container images

- Docker Hub

- Quay.io

Docker Run

```
docker run --detach --name 2mqtt \
    --volume $PWD/config.yaml:/app/config.yaml \
    --device /dev/ttyUSB0:/dev/ttyUSB0 \
    --env  TZ="Asia/Kolkata" \
    --restart unless-stopped \
    docker.io/mycontroller/2mqtt:1.4
```

Download Executables

- Released versions
- Pre Release - `main` branch executables

Configuration

You can have more than one adapter configurations

Provider options: `mysensors_v2` and `raw`

```
logger:
  mode: development   # logger mode: development, production
  encoding: console   # encoding options: console, json
  level: info         # log levels: debug, info, warn, error, fatal

adapters:   # you can have more than one adapter
  - name: adapter1          # name of the adapter
    enabled: false          # enable or disable the adapter, default disabled
    reconnect_delay: 20s    # reconnect automatically, if there is a failure on the c
    provider: mysensors_v2  # provider type, options: mysensors_v2, raw
    source: # source is the device, to be converted to MQTT, based on the type, confi
      type: serial              # source device type: serial
      port: /dev/ttyUSB0        # serial port
      baud_rate: 115200         # serial baud rate
      transmit_pre_delay: 10ms  # waits and sends a message, to avoid collision on th
    mqtt: # mqtt broker details
      broker: tcp://192.168.10.21:1883  # broker url: supports tcp, mqtt, tls, mqtts
      insecure: false                   # enable/disable insecure on tls connection
      username:                         # username of the broker
      password:                         # password of the broker
      subscribe: in_rfm69/#             # subscribe a topic, should include `#` at th
      publish: out_rfm69                # publish on this topic, can add many topics
      qos: 0                            # qos number: 0, 1, 2
```

```
    transmit_pre_delay: 0s
    reconnect_delay: 5s
```

Source device configuration

Based on the source type the configurations will be different.

Serial

```
source:
  type: serial              # source device type
  port: /dev/ttyUSB0        # serial port
  baud_rate: 115200         # serial baud rate
  transmit_pre_delay: 10ms  # waits and sends a message, to avoid collision on the so
  message_splitter: # message splitter byte, default '10'
```

Ethernet

```
source:
  type: ethernet                    # source device type
  server: tcp://192.168.10.21:5003  # ethernet server address with port
  transmit_pre_delay: 10ms          # waits and sends a message, to avoid collision o
  message_splitter: # message splitter byte, default '10'
```

HTTP

```
source:
  type: http                      # source device type
  listen_address: "0.0.0.0:8080"  # listening address and port
  is_auth_enabled: true           # enable/disable basic authentication
  username: hello                 # username of basic authentication
  password: hello123              # password of basic authentication
```

for http source, on mqtt the payload (json string) will be as follows,

```
{
  "method":"POST",
  "remoteAddress":"192.168.0.1:57112",
  "host":"my-secret-host.com:8080",
  "path":"/hello",
  "body":"say hello",
  "queryParameters":{
    "q1": ["v1"],
```

```
    "q2": ["v1", "v2"]
  },
  "headers":{
    "Accept-Encoding":["gzip"],
    "Cache-Control":["no-cache"],
  },
  "timestamp":"2022-05-27T08:01:55.806281887+05:30"
}
```

Special note on message_splitter

NOTE: Applicable for `serial` and `ethernet` devices

- `message_splitter` is a reference char to understand the end of message on serial and ethernet device read
- This special char will be included while writing to the device.
- supports only one char, should be supplied in byte, ie: `0` to `255`, extended ASCII chars

Quick references

For complete details refer the extended ASCII table

- `0` - Null char
- `3` - End of Text
- `4` - End of Transmission
- `8` - Back Space
- `10` - Line Feed
- `13` - Carriage Return

Script support

In `raw` provider we can add script to support custom specification. If we leave the script part empty, works without formatting.

`2mqtt` support limited `JavaScript` support along with goja

Configuration file with raw provider and a script support

```
logger:
  mode: development   # logger mode: development, production
  encoding: console   # encoding options: console, json
  level: info         # log levels: debug, info, warn, error, fatal

adapters:   # you can have more than one adapter
```

```
  - name: adapter1          # name of the adapter
    enabled: false          # enable or disable the adapter, default disabled
    reconnect_delay: 20s    # reconnect automatically, if there is a failure on the c
    provider: raw  # provider type, options: mysensors_v2, raw
    source: # source is the device, to be converted to MQTT, based on the type, confi
      type: serial              # source device type: serial
      port: /dev/ttyUSB0        # serial port
      baud_rate: 115200         # serial baud rate
      transmit_pre_delay: 10ms  # waits and sends a message, to avoid collision on th
    mqtt: # mqtt broker details
      broker: tcp://192.168.10.21:1883  # broker url: supports tcp, mqtt, tls, mqtts
      insecure: false                   # enable/disable insecure on tls connection
      username:                         # username of the broker
      password:                         # password of the broker
      subscribe: receive_data/#         # subscribe a topic, should include `#` at th
      publish: publish_data             # publish on this topic, can add many topics
      qos: 0                            # qos number: 0, 1, 2
      transmit_pre_delay: 0s
      reconnect_delay: 5s
    formatter_script: # script used to perform custom formatting
      to_mqtt: |
        // your multiline javascript
        // to perform formatting
        // read examples for more details
      to_source: |
        // your multiline javascript
        // to perform formatting
        // read examples for more details
```

to_mqtt

data received from `source` device and posts to `mqtt` . can be `serialPort` , `http` , `ethernet` . for details refer source device options section.

you will receive the following variables on your script

- `raw_data` - data received from `source` device

once the format done, at the end of script, you have to submit the result in two ways
in both way, your response should be assigned into `result` variable *WITHOUT* `let` , `var` or `const` .

1. if you do not want to change the `mqtt` parameters, like `topic` , `QoS` , etc, just assign the response into `result` . example: `result="hello"`
2. if you need to change `mqtt` parameters dynamically, assign key/value into `result` variable. supported keys,

- `data` - your response data should passed on the mqtt publish

- `mqtt_topic` - if you want to change the topic dynamically. NOTE: this topic will be appended(suffix) along with global topic (`adapters[].mqtt.publish`)
- `mqtt_qos` - you can modify the QoS
- `ignore` - if you think, no need to proceed further with this data and do not want to send it to `mqtt` add `ignore: true` . This message will be dropped.

examples:

simple string return

```
// want to append "_modified" at the end of raw data
data=raw_data + "_modified"

// return
result=data
```

return as object

```
// want to add mqtt topic dynamically
// assume the raw_data is "hello;mqtt/secret/topic"
const dataArray = raw_data.split(";")

// return
result = {
  data: dataArray[0], // ie: hello
  mqtt_topic: dataArray[1], // ie: mqtt/secret/topic
}
```

ignore the data. do not proceed further

```
// assume the raw_data is "ignore_me"

// ignoreMe is "true", if the message is "ignore_me"
const ignoreMe = raw_data == "ignore_me"

// return
result = {
  data: raw_data,
  ignore: ignoreMe, // is true. this message will not send to mqtt
}
```

to_source

data received from `mqtt` passed to formatter script and posts to `source` device.

you will receive the following variables on your script

- `raw_data` - data received from mqtt subscription
- `mqtt_topic` - topic of the received message
- `mqtt_qos` - qos of the received message

once the format done, at the end of script, you have to submit the result in two ways
in both way, your response should be assigned into `result` variable *WITHOUT* `let`, `var` or `const`.

1. just assign the into `result`. example: `result="hello"`
2. assign key/value into `result` variable. supported keys,

- `data` - your response data should passed to `source` device
- `ignore` - if you think, no need to proceed further with this data and do not want to send it to `source` device add `ignore: true`. This message will be dropped.

examples

simple string return

```
// want to include mqtt topic in the message
// assume:
//    raw_data is "hello"
//    mqtt_topic is "mqtt/secret/topic"

const finalData = raw_data + ";" + mqtt_topic

// return
result = finalData // ie: hello;mqtt/secret/topic
```

return as object

```
// want to include mqtt topic in the message
// assume:
//    raw_data is "hello"
//    mqtt_topic is "mqtt/secret/topic"

const finalData = raw_data + ";" + mqtt_topic

// return
result = {
  data: finalData, // ie: hello;mqtt/secret/topic
}
```

ignore the data. do not proceed further

```
// assume the raw_data is "ignore_me"

// ignoreMe is "true", if the message is "ignore_me"
const ignoreMe = raw_data == "ignore_me"

// return
result = {
  data: raw_data,
  ignore: ignoreMe, // is true. this message will not send to mqtt
}
```

Releases 6

v1.4 Latest
on Feb 20

+ 5 releases

Packages

No packages published

Languages

- **Go** 90.9%
- **Shell** 7.6%
- **Dockerfile** 1.5%