Tutorial

Streaming over MQTT

Prerequisites

- Installed: Curl
- Installed: Mosquitto (MQTT-client)
- Optionally: jq
- Alternative: Virtual Machine containing the tools
- Available: An API-Key (will be provided)

Goal

Learn how to subscribe and publish data on one of the DSH public datastreams via MQTT

Steps

1. Get an API-key
2. Use API-key to request a REST token
3. Use REST token to request an MQTT token
4. Use MQTT token to subscribe to a datastream
5. Use MQTT token to publish on a datastream

API-keys on DSH

Are used to *identify* entities that can manage DSH MQTT access for a group of things.

API-keys on DSH

Are used to *identify* entities that can manage DSH MQTT access for a group of things.

Each of these entities can use this API-key to:

- onboard *multiple* things to DSH MQTT
- give things access to a *limited set* of streams
- control on a *per thing basis* what streams a thing can access (out-of-scope)

API-keys on DSH

Are used to *identify* entities that can manage DSH MQTT access for a group of things.

Each of these entities can use this API-key to:

- onboard *multiple* things to DSH MQTT
- give things access to a *limited set* of streams
- control on a *per thing basis* what streams a thing can access (out-of-scope)

we call these entities *tenants*

Get an API-key

Along with the API-key you will also receive:

- the name of a DSH deployment (`PLATFORM`)
- the name of the tenant (`TENANT`)
- other info you do not need for this tutorial

Export the needed values now, so you have them available in the next commands.

```
export PLATFORM=...
export TENANT=...
export API_KEY=...
```

REST tokens

- can be requested over the *REST API* of DSH on the `/auth/v0/token` endpoint
- this endpoint requires an *API-key* and the name of your *tenant*

Command

```
curl -X POST \
  "https://api.$PLATFORM.kpn-dsh.com/auth/v0/token" \
  -H "apikey: $API_KEY" \
  -d '{"tenant": "'$TENANT'"}' > rest-token.txt
```

Output

- This command will result in a *REST token*
- This is a *JWT* containing a set of *REST claims*
- These claims describe
  - what *other* REST APIs you can access
  - which *actions* are allowed on those APIs
- The `datastreams/v0/mqtt/token` claim should be in the token, otherwise it will be impossible to request an MQTT token

Contents of a JWT

Execute

```
cat rest-token.txt
```

to see the token and navigate to https://jwt.io and replace the data in the encoded form on the webpage by the contents of the JWT (REST token) you received in the previous step.

Contents of a JWT

Execute

```
cat rest-token.txt
```

to see the token and navigate to https://jwt.io and replace the data in the encoded form on the webpage by the contents of the JWT (REST token) you received in the previous step.

Did you just blindly give away credentials to an unknown website?

Contents of a JWT

Command-line alternative for Linux:

```
cat rest-token.txt | \
sed "s/[^.]*\.\([^.]*\)\.[^.]*/\1===/;s/\(\(....\)*\).*/\1/" | \
base64 -d | \
jq .
```

Command-line alternative for macOS:

```
cat rest-token.txt | \
sed "s/[^.]*\.\([^.]*\)\.[^.]*/\1===/;s/\(\(....\)*\).*/\1/" | \
base64 -D | \
jq .
```

MQTT tokens

- can be requested over the *REST API* of DSH at `/datastreams/v0/mqtt/token`
- this requires a *REST token* with the `datastreams/v0/mqtt/token` claim
- required to request the token for a named thing

```
export THING_ID=...
```

Command

```
curl -X POST \
"https://api.$PLATFORM.kpn-dsh.com/datastreams/v0/mqtt/token" \
-H "Authorization: Bearer `cat rest-token.txt`" \
-d '{"id":"'$THING_ID'"}' > mqtt-token.txt
```

Output

- This command will result in an *MQTT token*
- This is a *JWT* containing a set of *claims*
- They describe:
  - what other REST APIs you can access
  - what actions are allowed on those APIs

Some remarks

- The mqtt thing id **must be unique** (within your tenant) since only one connection with this id is allowed.
- Not all REST tokens allow all thing ids (some are bound to one specific THING_ID)

Inspect the JWT

Use jwt.io or use the command-line alternative for Linux:

```
cat mqtt-token.txt | \
sed "s/[^.]*\.\([^.]*\)\.[^.]*/\1===/;s/\(\(....\)*\).*/\1/" | \
base64 -d | \
jq .
```

Use jwt.io or use the command-line alternative for macOS:

```
cat mqtt-token.txt | \
sed "s/[^.]*\.\([^.]*\)\.[^.]*/\1===/;s/\(\(....\)*\).*/\1/" | \
base64 -D | \
jq .
```

Subscribe

- We will all subscribe to *dshdemoshared*, a stream created specifically for this tutorial.
- If you inspect the mqtt token, you will see that the allowed pattern for the subscription is `+/#`
  - `+` => a topic level element is allowed here, but no mqtt wildcards
  - `#` => wildcards are allowed here
  - `word` => you need to copy this verbatim

Command

On Linux, execute the following command:

```
mosquitto_sub -h mqtt.$PLATFORM.kpn-dsh.com -p 8883 \
-t "/tt/dshdemoshared/$THING_ID/#" --capath /etc/ssl/certs/ \
-d -P "`cat mqtt-token.txt`" -u $THING_ID -v
```

On macOS, use `--cafile iso --capath`:

```
mosquitto_sub -h mqtt.$PLATFORM.kpn-dsh.com -p 8883 \
-t "/tt/dshdemoshared/$THING_ID/#" \
--cafile /usr/local/etc/openssl/cert.pem \
-d -P "`cat mqtt-token.txt`" -u $THING_ID -v
```

Notes

- The username (`-u ...`) is not required for the DSH; it gets overruled by the `THING_ID` in the token. We put it here because some versions of the mosquitto client expect it to be there.
- `--capath` or `--cafile` is where the mosquitto client can find the SSL root certificates on your system, required to be able to connect to our mqtt protocol-adapter over SSL

Publish

- To see any data on our subscription, we need to publish some first
- Open a new terminal, we are going to use it to periodically send a message to DSH over mqtt
  - Ensure the required environment variables are available

Command

On Linux, execute the following command:

```
while sleep 1; do date "+$THING_ID%S"; done | \
mosquitto_pub -h mqtt.$PLATFORM.kpn-dsh.com \
-p 8883 -t "/tt/dshdemoshared/$THING_ID/" \
--capath /etc/ssl/certs/ \
-d -P "`cat mqtt-token.txt`" -u $THING_ID -l
```

On macOS, execute the following command:

```
while sleep 1; do date "+$THING_ID%S"; done | \
mosquitto_pub -h mqtt.$PLATFORM.kpn-dsh.com \
-p 8883 -t "/tt/dshdemoshared/$THING_ID/" \
--cafile /usr/local/etc/openssl/cert.pem \
-d -P "`cat mqtt-token.txt`" -u $THING_ID -l
```

Notes

- What do you see? Why?
- Try replacing `sleep 1` by `true`; what happens?

Notes

- What do you see? Why?
- Try replacing `sleep 1` by `true`; what happens?
- To allow the platform to scale, publish rate over MQTT is limited to 10 msgs/sec.

Next

The next tutorial (Streaming over Kafka) will explain how you can process data sent over mqtt in bulk on DSH.