目次

始めに

OpenHABの設定ファイルの記述方法を本ドキュメントで示す。Linuxにおける標準では、`/etc/openhab` ディレクトリ配下にサブディレクトリが用意されているので、そこに各種ファイルを記述していく。

例として、Switchbot温湿度計から取得できるデータをOpenHABでどう管理していくかを定義するファイルの記述方法を示す。

環境構築

OpenHABのインストール方法は、本ドキュメントと同じプロジェクト配下にある`reTerminal_setup.md` を別途参照。

インストール後、OpenHABのGUIから各種必要となるバインディング（アドオン）の追加をする必要がある。バインディングの追加に関しては、GUIからの追加が無難だと公式からも声明がされている。 # **MQTTバインディングとJSONPath Transformationアドオンは必須**

例えば、Tapo電球を操作する場合、Tapoアドオンの追加が必要となる。

また、VSCodeで作業を行う場合、拡張機能のOpenHABを入れることを推奨する。

Thingsファイル

Thingsファイルでは、「**何**で繋がった**どのモノ**」をOpenHABに接続するかを定義する。OpenHABに機器を追加するときには、常にThingsファイルの記述から始める。モノはバインディングを通じてopenHABに接続されるため、追加するモノをつなげるバインディングの選択が必要となる。

温湿度計の室温データは、ESPを介してローカルのMQTTブローカにサブスクライブされているため、「**MQTTブローカ**で繋がった**温湿度計の室温データ**」を追加することになる。

これをファイルで記述すると、以下のようになる。

```
Bridge mqtt:broker:broker [host="127.0.0.1", secure=false,
username="admin", password="admin"] {
    //SbMeter1_Temp
    Thing topic Meter1_Temp_Status_device "Meter1_Temp_Status_device" {
    Channels:
      Type string : Meter1_Temp_Status_device
```

```
    [stateTopic="haudi/hc1.0/ul3fqMC4uNtvOOjGP5WCcw/temphumid/h1_r1_meter1/status" ,
    transformationPattern="JSONPATH:$.temp"]
        }
    }
```

1行目で、MQTTによってこの機器が接続されていることを定義している。何で接続されているかを定義する際には、`Bridge`を最初につける。
`Bridge`の次の3つの記述は、MQTTで接続する際には必須の記述となる。: `mqtt:broker:broker`
次の角括弧で囲まれた箇所で、使用しているMQTTブローカに関する情報を記述する。上の例では、ローカルのMQTTブローカにユーザ名とパスワードを指定して接続をしている。

ここまでが、MQTTに関する記述となっている。

次に、扱うデータを定義する。データを定義する際には`Thing`をつける。
`Thing`の次の3つの記述で、**デバイス**の**室温データ**と分かるようにラベルを付けている。: `topic Meter1_Temp_Status_device "Meter1_Temp_Status_device"`
次の波括弧で囲まれた場所で扱うデータの詳細を定義している。扱うデータの詳細は**チャンネル**という単位で管理を行う。扱うデータの詳細を定義する際には、`Channels:`を最初につける。チャンネルは、後述の**Items**とThingをつなげる役割を果たし、ItemsとThingsで相互にデータを渡すことが可能となる。
次の記述で、扱うデータがどういったデータなのかを管理する。上の例では、**デバイス**の**室温データ**と分かるようにラベルを付け、そのトピック
を`"haudi/hc1.0/ul3fqMC4uNtvOOjGP5WCcw/temphumid/h1_r1_meter1/status"`指定している。# **haudi プロトコルに準拠**
最後の`transformationPattern="JSONPATH:$.temp"`という記述によって、取得されたJSONデータから、温度データのみを引き抜いている。

Itemsファイル

Itemsファイルでは、Thingsファイルで用意した、データのために受け皿の役割となるItemsを定義する。
Itemsには状態があり、後述ruleやWebAppを通じて使用される。

ファイルでは、以下のような単純な記述になる。

```
String Meter1_Temp_Status_device
{channel="mqtt:topic:broker:Meter1_Temp_Status_device:Meter1_Temp_Status_device"}
```

Rulesファイル

Rulesファイルでは、その名の通りルールとしてある条件で一連のコマンドが実行される。例として、温湿度計の温度が変化した場合に、前述のItemの値をその時の値に動的に変化させるといったことが可能となる。

これをファイルで記述すると以下のようになる。

```

```

```
rule "HumidSensor_OHtoUI"
when
    Item Meter1_Humid_Status_device changed
then
    var time = new DateTimeType().format("%1$tY-%1$tm-%1$tdT%1$tT%1tZ")
    Meter1_Humid_Command_ui.sendCommand('{"text":'+
Meter1_Humid_Status_device.state + ',"ts":"' + time + '"}')
end
```

Itemとして、温湿度計の温度が変化した場合に、５行目以降のコマンドが実行され、タイムスタンプとともにデータが送られるようになっている。