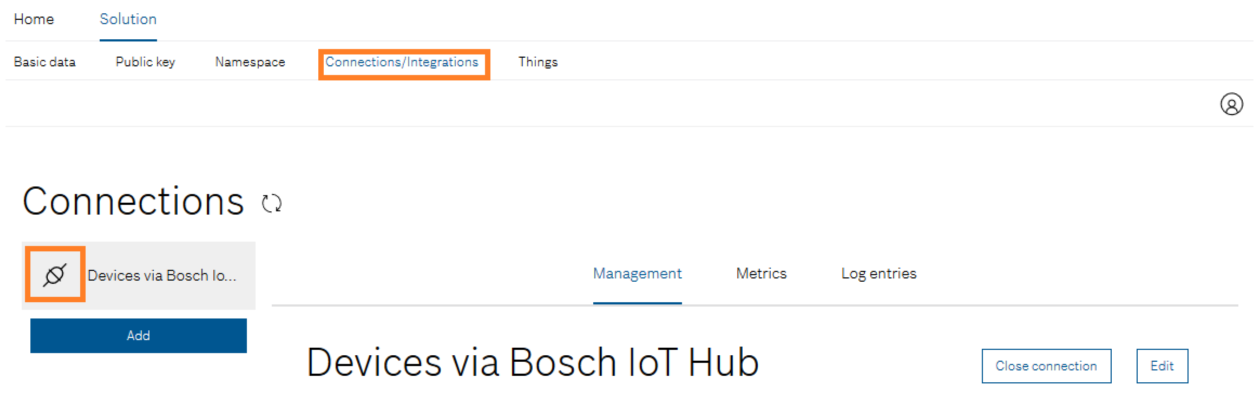
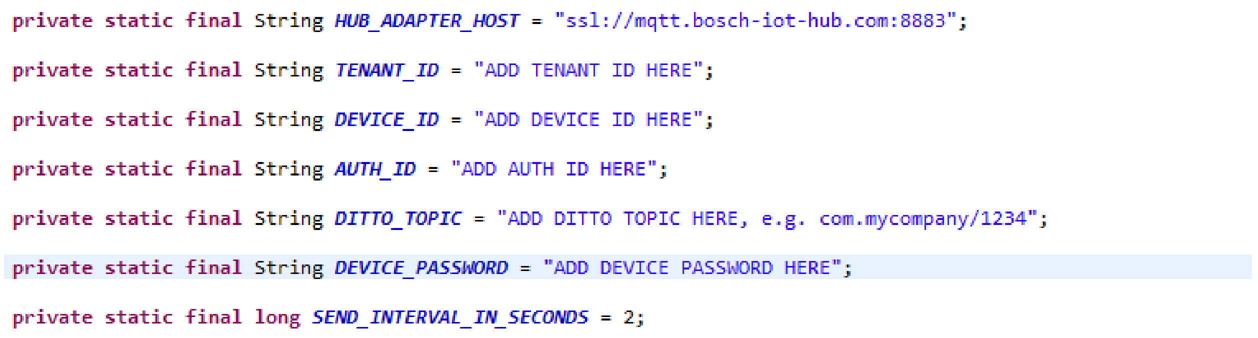
How to create a Java program that talks to the Bosch IoT Suite:

1. Book Asset Communication Package:
    a. https://docs.bosch-iot-suite.com/asset-communication/Subscribe-a-package-instance.html
    b. https://docs.bosch-iot-suite.com/asset-communication/First-configuration-steps.html
    c. https://docs.bosch-iot-suite.com/asset-communication/Device-provisioning.html
    d. Check the connection



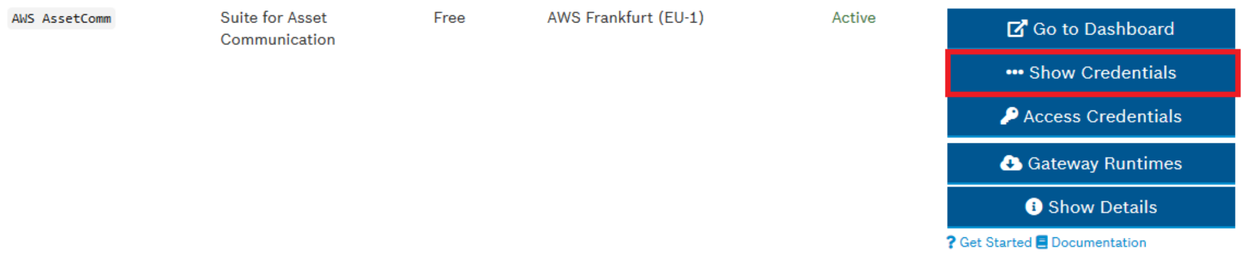
2. Use Vorto to get the Bosch IoT Suite Plugin for a device. We use the Raspberry Pi one here:
    a. https://vorto.eclipse.org/#/details/org.eclipse.vorto.tutorials:RaspberryPi:1.0.0
    b. Download the Bosch IoT Suite Plugin
    c. Device provisioning
        i. Download Postman (https://www.getpostman.com/downloads/)
        ii. Download Source Code
        iii. Create your Thing
            1. https://github.com/eclipse/vorto/blob/master/docs/tutorials/create_thing.md
            2. Save Device_ID, Auth_ID, password
3. Application

a. Configure App



```java
private static final String HUB_ADAPTER_HOST = "ssl://mqtt.bosch-iot-hub.com:8883";

private static final String TENANT_ID = "ADD TENANT ID HERE";

private static final String DEVICE_ID = "ADD DEVICE ID HERE";

private static final String AUTH_ID = "ADD AUTH ID HERE";

private static final String DITTO_TOPIC = "ADD DITTO TOPIC HERE, e.g. com.mycompany/1234";

private static final String DEVICE_PASSWORD = "ADD DEVICE PASSWORD HERE";

private static final long SEND_INTERVAL_IN_SECONDS = 2;
```

i. Device_ID, Auth_ID, password from Device provisioning
ii. Tentant_ID: from Asset Communication package

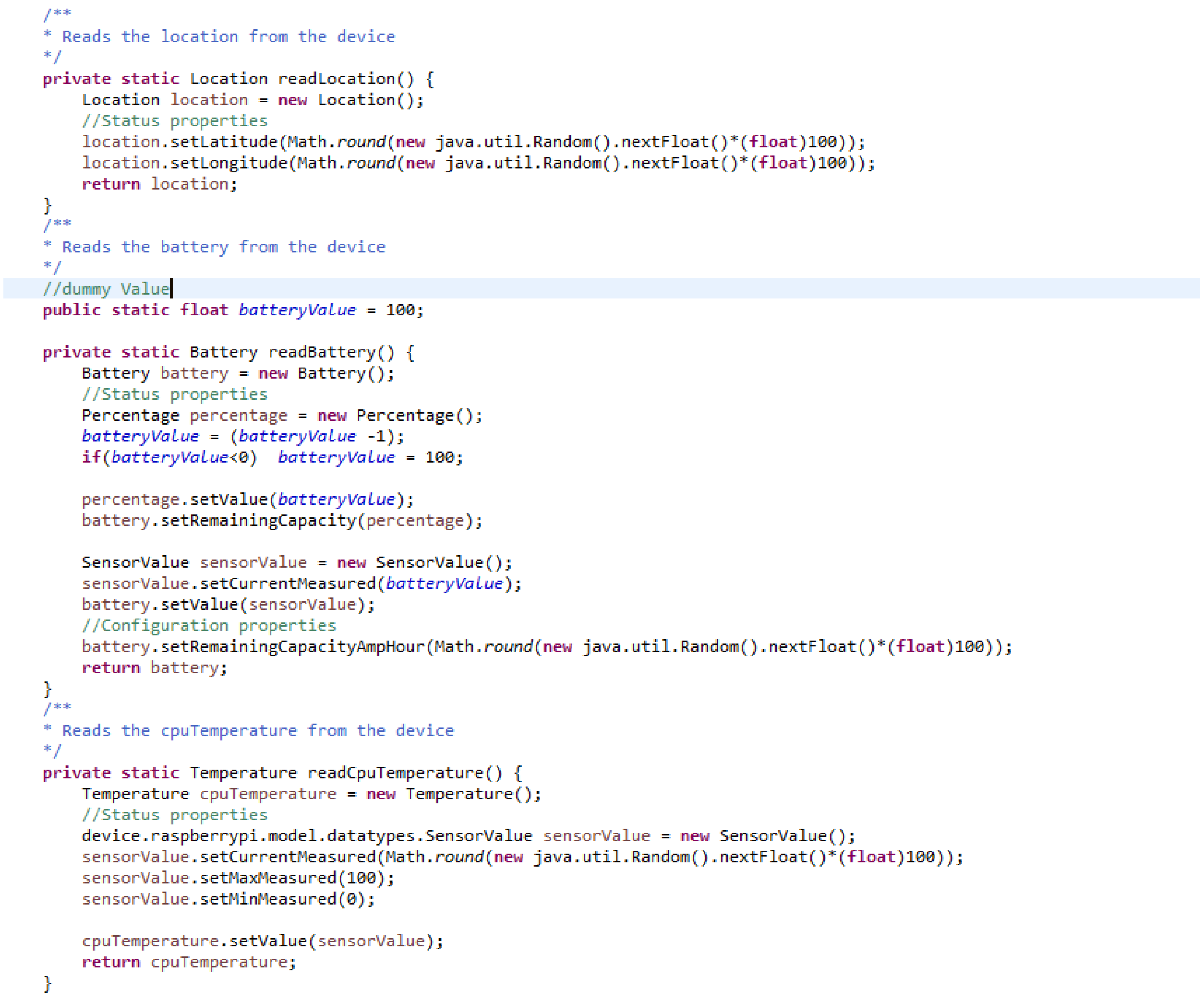


AWS AssetComm	Suite for Asset Communication	Free	AWS Frankfurt (EU-1)	Active

iii. Ditto_Topic = namespace/device_ID

b. Download https://docs.bosch-iot-hub.com/cert/iothub.crt
i. Java: <project>/src/main/resources/iothub.crt

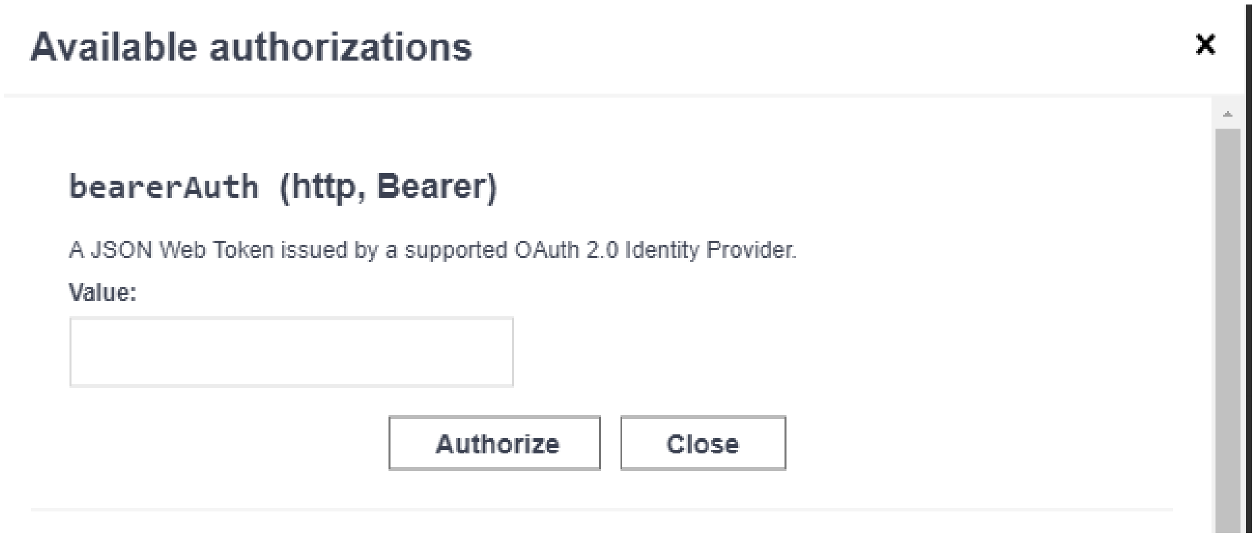
c. Change the code (example Code)



```java
/**
* Reads the location from the device
*/
private static Location readLocation() {
    Location location = new Location();
    //Status properties
    location.setLatitude(Math.round(new java.util.Random().nextFloat()*(float)100));
    location.setLongitude(Math.round(new java.util.Random().nextFloat()*(float)100));
    return location;
}
/**
* Reads the battery from the device
*/
//dummy Value
public static float batteryValue = 100;

private static Battery readBattery() {
    Battery battery = new Battery();
    //Status properties
    Percentage percentage = new Percentage();
    batteryValue = (batteryValue -1);
    if(batteryValue<0)  batteryValue = 100;

    percentage.setValue(batteryValue);
    battery.setRemainingCapacity(percentage);

    SensorValue sensorValue = new SensorValue();
    sensorValue.setCurrentMeasured(batteryValue);
    battery.setValue(sensorValue);
    //Configuration properties
    battery.setRemainingCapacityAmpHour(Math.round(new java.util.Random().nextFloat()*(float)100));
    return battery;
}
/**
* Reads the cpuTemperature from the device
*/
private static Temperature readCpuTemperature() {
    Temperature cpuTemperature = new Temperature();
    //Status properties
    device.raspberrypi.model.datatypes.SensorValue sensorValue = new SensorValue();
    sensorValue.setCurrentMeasured(Math.round(new java.util.Random().nextFloat()*(float)100));
    sensorValue.setMaxMeasured(100);
    sensorValue.setMinMeasured(0);

    cpuTemperature.setValue(sensorValue);
    return cpuTemperature;
}
```

d. Run Maven Clean + Maven Install

4. Run your Code
5. Monitor your Thing
   a. https://apidocs.bosch-iot-suite.com/?urls.primaryName=Bosch%20IoT%20Things%20-%20API%20v2
   b. Authorize with your OAuth 2.0 Token



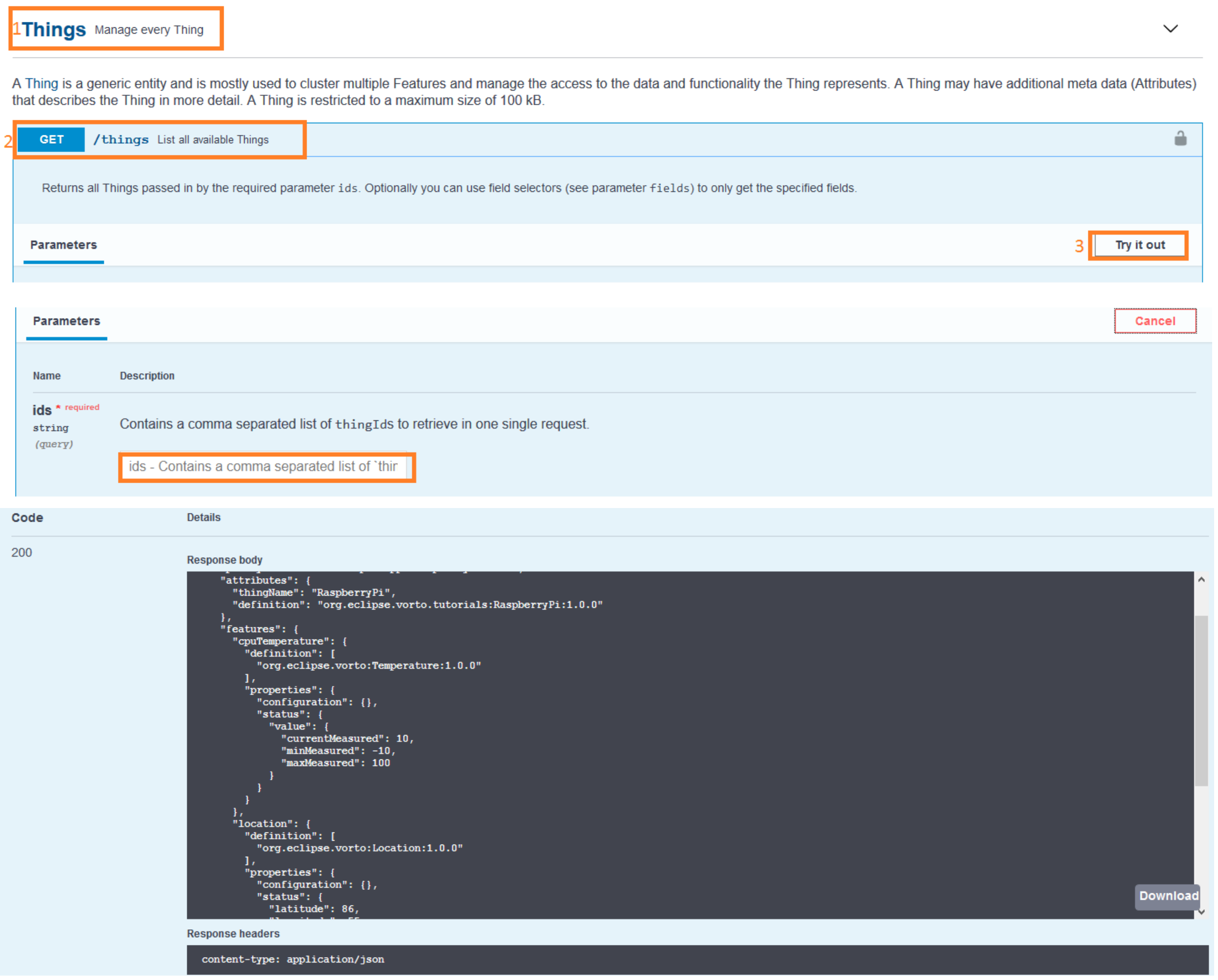
Available authorizations

bearerAuth (http, Bearer)

A JSON Web Token issued by a supported OAuth 2.0 Identity Provider.

Value:

c. Search your thing



A Thing is a generic entity and is mostly used to cluster multiple Features and manage the access to the data and functionality the Thing represents. A Thing may have additional meta data (Attributes) that describes the Thing in more detail. A Thing is restricted to a maximum size of 100 kB.

Returns all Things passed in by the required parameter `ids`. Optionally you can use field selectors (see parameter `fields`) to only get the specified fields.

Name	Description
ids * required string (query)	Contains a comma separated list of `thingIds` to retrieve in one single request. ids - Contains a comma separated list of `thir

Code	Details
200	Response body

```
    "attributes": {
      "thingName": "RaspberryPi",
      "definition": "org.eclipse.vorto.tutorials:RaspberryPi:1.0.0"
    },
    "features": {
      "cpuTemperature": {
        "definition": [
          "org.eclipse.vorto:Temperature:1.0.0"
        ],
        "properties": {
          "configuration": {},
          "status": {
            "value": {
              "currentMeasured": 10,
              "minMeasured": -10,
              "maxMeasured": 100
            }
          }
        }
      },
      "location": {
        "definition": [
          "org.eclipse.vorto:Location:1.0.0"
        ],
        "properties": {
          "configuration": {},
          "status": {
            "latitude": 86,
```

Response headers

```
content-type: application/json
```

6. Optional
    a. Transfer application to your Raspberry Pi
    b. Build a runnable jar file
    c. Add this code to your pom.xml

```
<project ...>

...
    <build>
            <plugins>
                    <plugin>
                            <groupId>org.apache.maven.plugins</groupId>
                            <artifactId>maven-jar-plugin</artifactId>
                            <version>2.2</version>
                            <!-- nothing here -->
                    </plugin>
                    <plugin>
                            <groupId>org.apache.maven.plugins</groupId>
                            <artifactId>maven-assembly-plugin</artifactId>
                            <version>2.2-beta-4</version>
                            <configuration>
                                    <descriptorRefs>
                                            <descriptorRef>jar-with-dependencies</descriptorRef>
                                    </descriptorRefs>
                                    <archive>
                                            <manifest>

        <mainClass>device.raspberrypi.RaspberryPiApp</mainClass>
                                            </manifest>
                                    </archive>
                            </configuration>
                            <executions>
                                    <execution>
                                            <phase>package</phase>
                                            <goals>
                                                    <goal>single</goal>
                                            </goals>
                                    </execution>
                            </executions>
                    </plugin>
            </plugins>
        </build>

    </project>
```

    d. Test your jar file
        i. java -jar raspberrypi-app-0.0.1-SNAPSHOT-jar-with-dependencies.jar
    e. Transfer your jar file to your raspberry pi

        i. E.g with puTTY
    f. Run your application
7. Use the demo Dashboard to monitor https://github.com/eclipse/vorto/blob/master/docs/tutorials/create_webapp_dashboard.md

Finish line. Create your own application

1) Create your own Vorto Model
    a) https://github.com/eclipse/vorto/blob/master/docs/tutorials/describe_device-in-5min.md
2) Build your application