Artifact Abstract: MBP: Not just an IoT platform

Ana Cristina Franco da Silva
IPVS, University of Stuttgart
Stuttgart, Germany
franco-da-silva@ipvs.uni-stuttgart.de

Jan Schneider
IPVS, University of Stuttgart
Stuttgart, Germany
st117301@stud.uni-stuttgart.de

Pascal Hirmer
IPVS, University of Stuttgart
Stuttgart, Germany
hirmer@ipvs.uni-stuttgart.de

***Abstract*—This artifact contains the Multi-purpose Binding and Provisioning Platform (MBP), an open-source IoT platform developed for easy binding, provisioning, and management of IoT environments. In the following, a guide is introduced how to set up and use the MBP.**

***Index Terms*—Internet of Things, IoT environments**

I. MBP Setup

In this section, we describe different ways to set up the MBP IoT platform: using Docker, Windows, or Ubuntu 16.04. The MBP is available on Github. For this artifact submission, we created a branch[1], which should be used for evaluation. Furthermore, a Docker installation[2] is provided on Github. The MBP platform has the following dependencies and software requirements:

- Git, Java 8 (or up), Apache Maven
- Apache Tomcat Application Server 8 (or up)
- MongoDB 4.2.2 (standard port: 27017)
- InfluxDB 1.7.9 (standard port: 8086)
- Mosquitto MQTT Broker 1.6.8 (standard port: 1883)
- Web Browsers: Chrome or Firefox

The easiest way to set up the MBP is using **Docker**:

1) Clone the Docker repository[2].
2) Build the Docker image: In the repository folder run
   `docker build -t mbp --build-arg branch=percom_artifact full/`
3) Run the image as container:
   `docker run -it --name mbp -p 80:80 -p 1883:1883 mbp:latest`
4) The MBP can be accessed on your Browser through the container's HTTP standard port 80.

The following steps set up the MBP on **Windows**:

1) Install the dependencies manually: Java 8, Apache Tomcat, MongoDB, InfluxDB, Mosquitto Broker
2) Clone the MBP repository[1]
3) Build .war file using Maven: In the repository folder run `mvn clean install`
4) Rename the resulting .war file to `MBP.war` and copy it into Tomcat's `webapps` folder
5) Start Tomcat by running `/bin/startup.bat`
6) Access the MBP on your Browser through `http://localhost:8080/MBP`

As an alternative, you can also use the already provided .war file in the folder `packaged` of the MBP repository[1].

On **Ubuntu 16.04**, the setup of the MBP only requires the following steps:

1) Clone the MBP repository[1]
2) In the repository folder, run the Shell script `install.sh`.
3) Access the MBP on your Browser through `http://localhost:8080/MBP`

II. Using the MBP

After successful setup, the MBP can be accessed through the browser and can be used to set up IoT environments. In the following, we describe the usage of the MBP. Furthermore, a Quick Start guide is available on Github[3].

A. User Creation, Login, and Home Page

On the login page, you can already log in using the default user *admin*. The password is *admin*. As an alternative, you can create a new user account by selecting *Register*. Here, you can provide your user name and password. Once logged in, you will be forwarded to the MBP's home page, which is depicted in Figure 1 of the attached paper. On this page, the number of currently registered devices, sensors, actuators, operators, and IoT environment models are shown. Initially, they should all be zero. Furthermore, a short tutorial guides you through creation of an IoT environment. In the following, we explain how to use the core functionalities of the MBP.

B. Registration of Devices

First, we register an IoT device in the MBP. This is done through the Tab *Devices*. By pressing the + button, a new device can be created. For this, you should either create a virtual machine, simulating an IoT device (e.g., using VirtualBox) or add a Raspberry Pi or another ssh accessible device, such as a Laptop or PC to your network, which can also serve as IoT devices. Next, give your device a name, a type, enter the correct IP address (make it static to be sure), the user name to access it through SSH, a password if existent, and the SSH RSA key to access the device. By selecting *Register*, the device is created. In the device list, it should now appear as

This work is partially funded by the BMWi project IC4F (01MA17008)

[1]https://github.com/IPVS-AS/MBP/tree/percom_artifact

[2]https://github.com/IPVS-AS/MBP-Docker

[3]https://github.com/IPVS-AS/MBP/wiki/Quick-Start

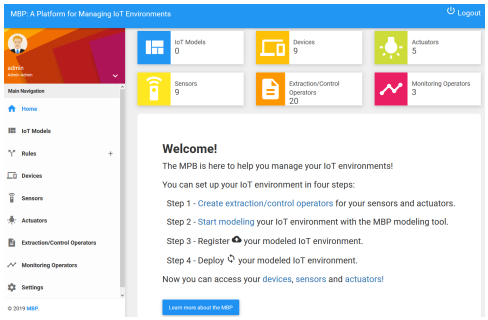

Fig. 1. Login page of the MBP

Available. If the state is *Unavailable* or *No SSH*, the device is not reachable in the network or the SSH RSA key is incorrect.

C. Registration of Extraction/Control Operators

Extraction and control operators are pieces of source code, usually scripts implemented, e.g., in Python or Shell, that accesses sensors and actuators of IoT devices, and communicate with the MBP platform to transfer sensor data or to control actuators. In order to set up a very small demo scenario, we provide an extraction operator in our repository[4]. This extraction operator does not require actual IoT hardware, since it simulates accessing a temperature sensor and sends random values to the MBP platform. This makes the artifact evaluation more easy, however, if you have available, e.g., a Raspberry pi and a temperature sensor, feel free to adjust our script with your configuration. To register this extraction operator, select the Tab *Extraction/Control Operators* of the MBP, select +, give it a name, select the Unit °C and drop the content of the operator folder in the file drop zone (install.sh, start.sh, stop.sh, and temperature_stub.py). By selecting *Register*, the extraction operator is created.

Next, we create a control operator, which is responsible to activate an actuator. The same steps as above are taken, however, another operator script is used, which you can find also in our repository[5]. This script also does not require a physical actuator, it logs activation events to a text file on the device's file system.

D. Registration and Deployment of Sensors/Actuators

Now, all required steps are done to create a sensor and bind it to the MBP platform. For this, select the Tab *Sensors*, select the +, give it a name, select the type *temperature*, and select the previously created extraction operator and device. After selection of *Register*, the newly created sensor should appear in the list of sensors as *Ready*. Next, click on the newly created sensor, which redirects you to a detailed view. Here, sensor data and basic statistics can be viewed. In order to bind the sensor, select *Deploy Sensor*, which is only available if the device has the state *Available*. During deployment, the MBP connects to your device, deploys the extraction operator on it, and runs its *install.sh* script to install necessary dependencies. After deployment, the button *Start sensor* should appear, which will start extraction of data from the sensor. After selecting this button, a new view will appear, showing the live sensor data in a diagram. Historical values can be depicted after refreshing the historical sensor values view (top right corner of the diagram).

After that, repeat the same steps by registering an actuator in the Tab *Actuator*, using the control operator. Create, deploy and start the actuator accordingly.

E. Creation of Rules

After the sensor is bound, we can define a simple rule, which will be continuously evaluated by a CEP system. To do so, select the Tab *Rules*. Then, select *Conditions* and select + to add a new condition. Give it a name and select *Next*. On the right, under sensors, drag our created sensor and drop it in the zone in the middle. Click on the sensor and toggle the field *Filter condition* to *On*. Now, click on the "=" until the ">" icon appears. As a value, enter the number 20. Hence, if the sensor value is greater than 20 °C, the condition evaluates to true. You also have the possibility to create very complex conditions, however, for this example, we keep it simple. Now, select *Next* and you will be shown a CEP query, which will be used to evaluate the condition. After selecting *Finish*, change to the Tab *Actions*. Here, we define what should happen if the condition evaluates to true. In our case, we want to activate the actuator. Select +, give the Action a name and select *Actuator action* as Type. Next, select the previously created actuator and select *Register*. To create the rule based on Condition and Action, select the Tab *Definitions* tab, select + and choose the previously created condition and action. As soon as the sensor sends values greater than 20 °C, the rule will be activated, which you can also see in the table.

F. Modeling of IoT Environments

Another functionality is modeling IoT environments graphically. To do so, select the Tab *IoT Models*, which opens a modeling tool. Select, e.g., the *Raspberry Pi* from the palette (under Device types) and drag and drop it onto the canvas. Click on the Raspberry Pi and enter its information on the right. Next, select a Temperature sensor from the palette (under Sensor types), drag and drop it to the canvas and create an edge from the Raspberry Pi to the sensor by clicking on the box and dragging a line. Then, select the sensor, give it a name and select the corresponding Extraction Operator from the Adapter drop-down menu. Next, select *Register components* (the small cloud icon on the top right), which will create the device and sensor entities. Select *Deploy model* on the top right to bind the sensor. You can access the sensor data through the sensor's detailed view as shown before.

[4]https://github.com/IPVS-AS/MBP/tree/percom_artifact/resources/adapter-scripts/temperature_stub

[5]https://github.com/IPVS-AS/MBP/tree/percom_artifact/resources/adapter-scripts/actuator_stub