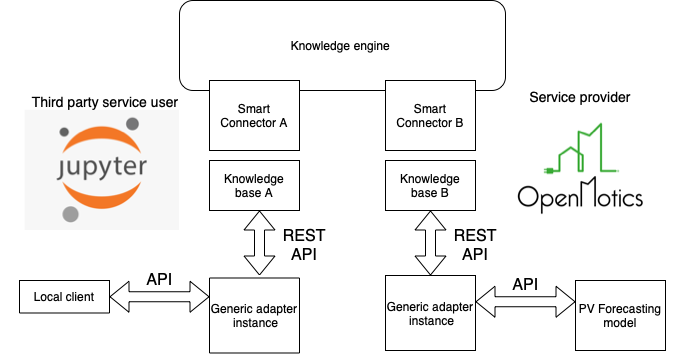
Demo

# Description of the files

* A Jupyter Notebook Whitegood Appliance Tester. This mocks, for our case, a whitegood appliance setting up the SIF from its side and sending a power sequence to a locally running generic adapter (localhost:9090). Steps 1-6 go through the whole sequence of setting up the required configuration (the part that should be very similar for all SSAs); step 7 actually sends a POST call, assuming that the service provider is sending long-polling calls to receive it.

<https://drive.google.com/file/d/1PiGxI0c6OWzNNAoZtiBEmhpTqSZ_pBDR/view?usp=drive_link>

* A figure detailing a possible test layout; the Jupyter Notebook logo represents the code in the notebook referred to in the previous point, the OpenMotics logo represents the service that could be running either locally or on a server somewhere.
* A \_bindingset and \_graphpattern file, in this case for BOSCH devices sending a power profile. This is just an example that hopefully clarifies a bit the relation between a binding (or binding set) and a graph pattern.
* Two excel files, one with a template for service SAREFization, where the different tabs are different steps in the process of going from an API call for an existing service to a graph pattern for that service. The second one is an example of this for the flexibility service of Inetum



# Deployment

## Generic adapter

Documents

[docs · master · InterConnect Public / Generic Adapter · GitLab (inesctec.pt)](https://gitlab.inesctec.pt/interconnect-public/generic-adapter/-/tree/master/docs)

Detailed instructions for creating/starting Generic Adapter

[docs/01\_preparations.md · master · InterConnect Public / Generic Adapter · GitLab (inesctec.pt)](https://gitlab.inesctec.pt/interconnect-public/generic-adapter/-/blob/master/docs/01_preparations.md)

### Docker

Generic Adapter container registry

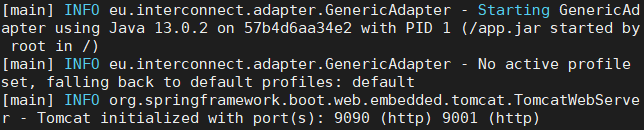
[Container Registry · InterConnect Public / Generic Adapter · GitLab (inesctec.pt)](https://gitlab.inesctec.pt/interconnect-public/generic-adapter/container_registry/313)

docker run your-tag

Knowledge Engine host is:<https://ke.interconnectproject.eu/rest/>docker run -e KE\_HOST=instance-of-knowledge-engine-you-want-to-use your-tag

**docker pull docker-registry.inesctec.pt/interconnect-public/generic-adapter:2.1**

**docker run -p 9001:9001 -p 9090:9090 --name Generic\_Adapter docker-registry.inesctec.pt/interconnect-public/generic-adapter:2.1**



### Binaries

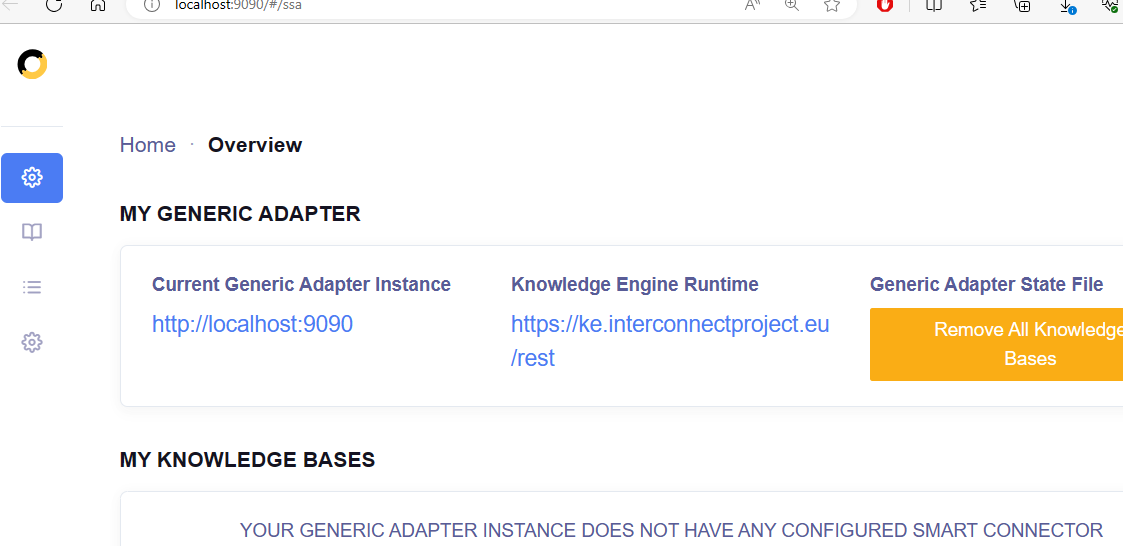
In application.properties change port (if you want) and change Knowledge Engine host (ke.host). (1). Also, if you want, you can override **read timeout**, **connect timeout** and **thread pool size** by changing values of connect.timeout, read.timeout and server.tomcat.threads.max respectively in same file, regarding Suggestion [#61](https://gitlab.inesctec.pt/interconnect/generic-adapter/-/issues/61)

Open GenericAdapter.java, right click -> Run As -> Java Application. If you go to<http://localhost:9090/> (in my case) you will get "Greetings from Generic Adapter!" message. (2)

If you get an error like it is depicit below, then, you also need to include knowledge-engine repository in your workspace. After that, you need to build that project via "mvn clean install" command. When build is finished, check version of KE and replace it in pom.xml file, line 23.

### Test whether the adapter is running

Open [localhost](http://localhost:9090/) in the browser.

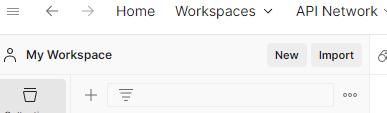


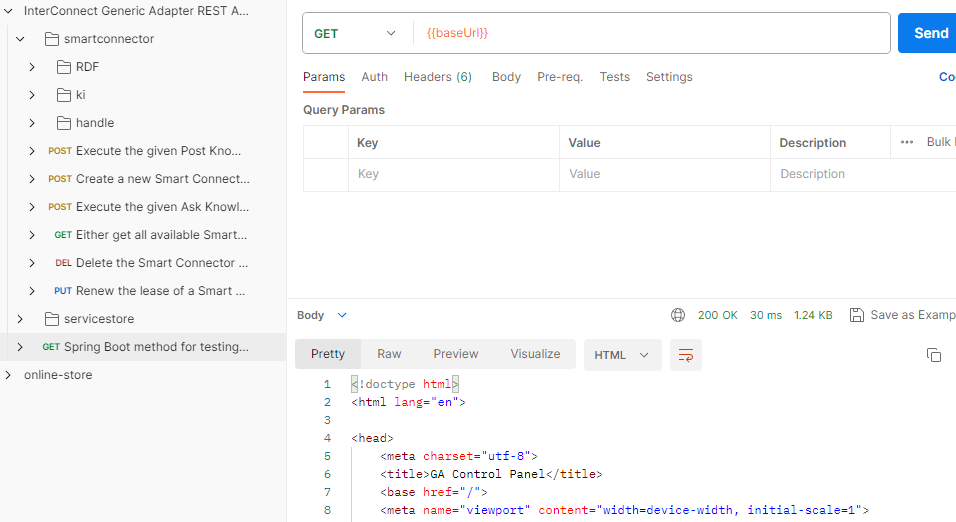
### Postman Collection

The Postman collection describes the available endpoints.

Download the file from Gitlab [InterConnect GenericAdapter Postman Collection.json · master · InterConnect Public / Generic Adapter · GitLab (inesctec.pt)](https://gitlab.inesctec.pt/interconnect-public/generic-adapter/-/blob/master/InterConnect%20GenericAdapter%20Postman%20Collection.json)

and import it ot Postman

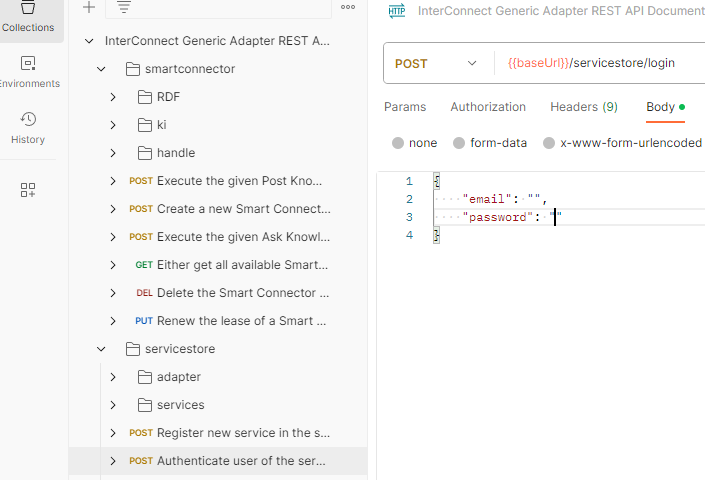




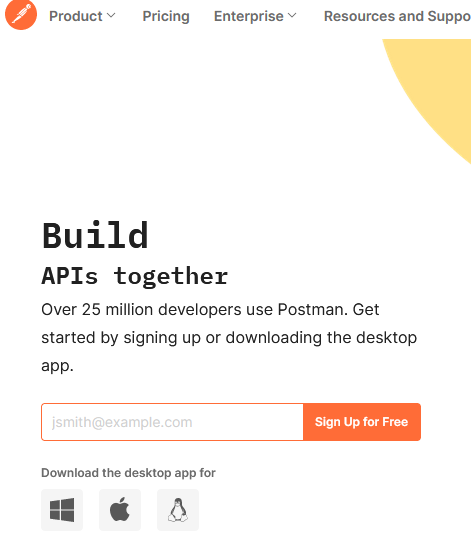


To change the base url

To authenticate to service store



Note: To download postman go to <https://www.postman.com/>



## Local Client - Jupyter notebook

The notebook is available here

<https://drive.google.com/file/d/1PiGxI0c6OWzNNAoZtiBEmhpTqSZ_pBDR/view?usp=drive_link>

“bsh\_powerprofile\_graphpattern.txt” is used for registering the service.

## Local Client on Google Colab

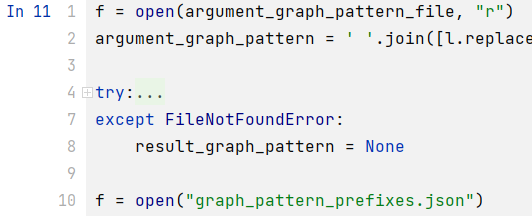
Google Colab allows to run Jupyter Notebooks with installing anything

# Experiments

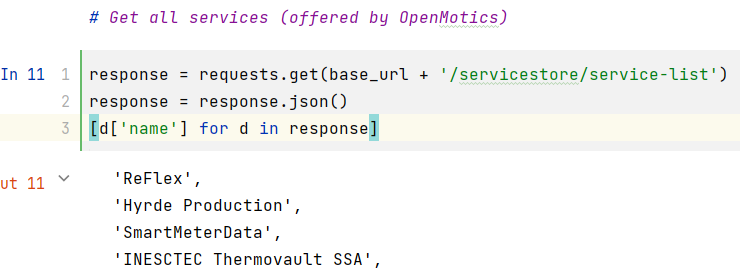
# Questions

“graph\_pattern\_prefixes.json” doesn't exist. Does “bsh\_powerprofile\_bindingset.json” do the same job?

FileNotFoundError: [Errno 2] No such file or directory: '**graph\_pattern\_prefixes.json**'

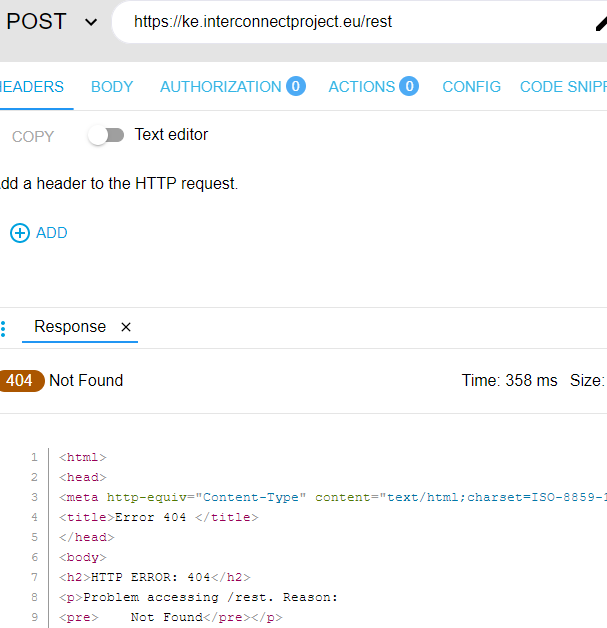


Looks like In 11 gets the services of all providers. Is it Correct?



How to check whether the Knowledge Engine is running?

<https://ke.interconnectproject.eu/rest/>

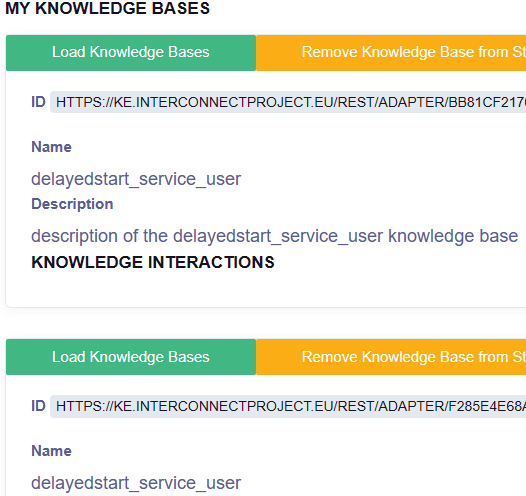


What is the purpose of “graph\_pattern\_prefixes”?

How to produce a Knowledge Base Id?

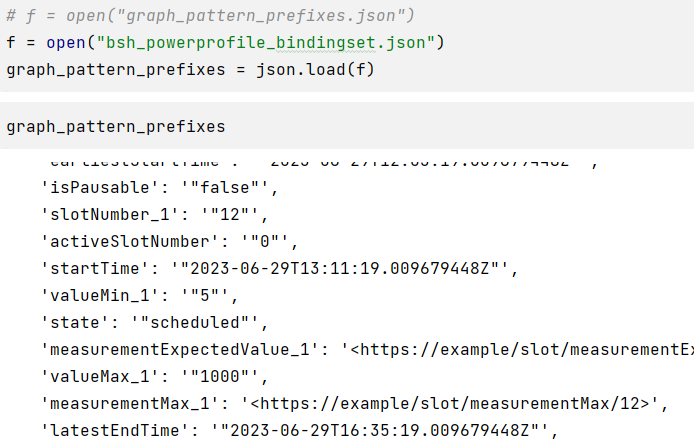
Look like the Knowledge Base Id is generated when the new services is registered to Knowledge engine





We only saw exchanging graph patterns. How to we send/receive actual data?

graph\_pattern\_prefixes consists the actual data



Then is send to '/smartconnector/ki/register-post-react'

body = {"knowledgeInteractionType": "PostKnowledgeInteraction",

"knowledgeInteractionName": "react-ki",

"communicativeAct": {

"requiredPurposes": ["https://www.tno.nl/energy/ontology/interconnect#InformPurpose"],

"satisfiedPurposes": ["https://www.tno.nl/energy/ontology/interconnect#InformPurpose"]},

"argumentGraphPattern": argument\_graph\_pattern,

"resultGraphPattern": result\_graph\_pattern,

"prefixes": **graph\_pattern\_prefixes**}

response = requests.post(base\_url + '/smartconnector/ki/register-post-react',

headers={"KnowledgeBaseId": knowledgebase\_id},

json=body)

How can we query data with parameters? Should the graph pattern support it (ex. earliestStartDate, latestEndDate)?

The above answer provides an example.

How often do we have to register KB / KI?

Each new service should be registered.

When we ask for data, how do we get the response?

Obviously sending requests. To clarify that, let's make an example with at least 2 services. One could represent the DSO and the other the client/boiler.

Suppose we measure 3 types of data. This means that we must have 3 KBs or 3 KIs?

Should be one per service

{'powerSequence': '<http://example.org/spine-ssa/devices/SIEMENS-WT47XM40-68A40E31F2EA/powerSequences/188040199>',

'valueExpectedValue\_1': '"450"',

'deviceId': '<http://example.org/spine-ssa/devices/SIEMENS-WT47XM40-68A40E31F2EA>',

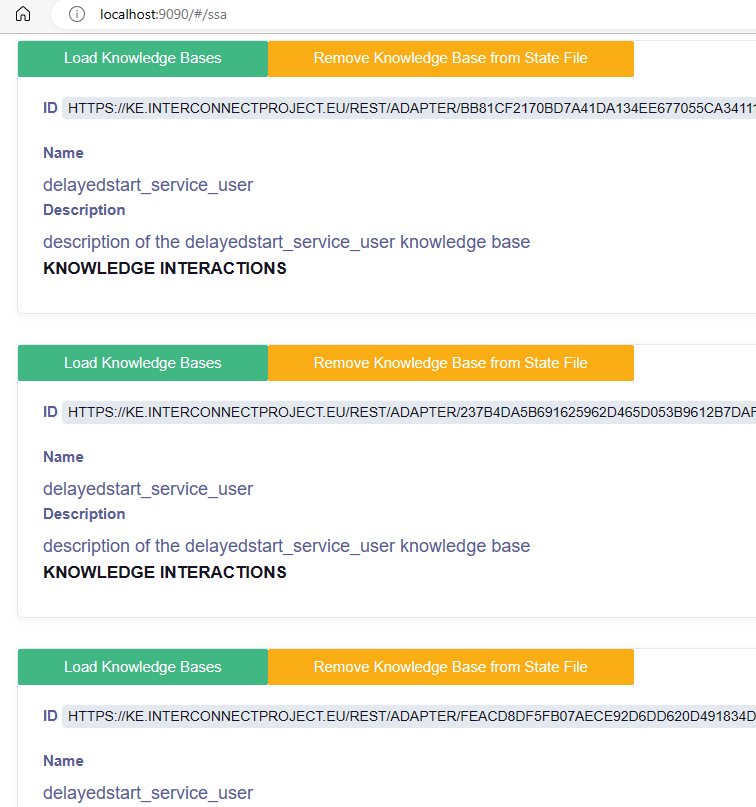
'associatedDevice': '<http://example.org/spine-ssa/devices/SIEMENS-WT47XM40-68A40E31F2EA/powerSequences/188040199/associatedDevice>',

'sequenceRemoteControllable': '"true"',

'slot\_1': '<https://example/slot/6>',

# Notes

Look like every time when the client code is executed, new service registration was created



The role of the generic adapter is to provide abstraction layer between the devices and the knowledge base

Interconnect DSO Interface

[serviceStoreService Details (interconnectproject.eu)](https://store.interconnectproject.eu/ServiceStore/#/service/service-details/2053747)

E-REDES

**Primary Knowledge Engine URL**

https://ke-pt.interconnectproject.eu/rest/

**Service Adapter Binding**

Bebda4180c06bcade9b1fdd070789ed471baef536602fafd446a2b8716c155ad559b6aa446499f5b4cec54e2ec5d9bdaea0b68c7a55d4b0b5efede327354d47b

FlexibilitySSA

[project-SSAs/INESCTEC-Flexibility/FlexibilitySSA · main · InterConnect Public / service-specifc-adapters · GitLab](https://gitlab.inesctec.pt/interconnect-public/service-specifc-adapters/-/tree/main/project-SSAs/INESCTEC-Flexibility/FlexibilitySSA)

# Tasks

Tasks:

Study about the graph, etc.

Define our data for the communication with: Boiler -> service > Generic ->KE

Define UI

Create our service to connect with Generic Adapter and Fiware

Define the architecture

Tasks 2:

* Study about FIWARE MQTT [FIWARE/tutorials.IoT-over-MQTT: :orange\_book: FIWARE 203: Provisioning Ultralight with an alternative transport: IoT over MQTT (github.com)](https://github.com/FIWARE/tutorials.IoT-over-MQTT)
* Find how DSO sends events to the boiler? Probably have to create a new service. Chack for DSO service. There are Flexibility service available. [project-SSAs/3E-flexibility-service/eee\_new/SSA\_base\_class.py · main · InterConnect Public / service-specifc-adapters · GitLab (inesctec.pt)](https://gitlab.inesctec.pt/interconnect-public/service-specifc-adapters/-/blob/main/project-SSAs/3E-flexibility-service/eee_new/SSA_base_class.py)
* Study about [docs/03\_ask\_answer.md · master · InterConnect Public / Generic Adapter · GitLab (inesctec.pt)](https://gitlab.inesctec.pt/interconnect-public/generic-adapter/-/blob/master/docs/03_ask_answer.md) and try to implement something, see the next.
* Simple mockup interface to show the workflow. One service sends a message and other lissent and performs some action. Covers our basic use case.
* Complex interface (more abstract)

PyCharm professional supports Jupyter notebooks

[Run and debug Jupyter notebook code cells | PyCharm Documentation (jetbrains.com)](https://www.jetbrains.com/help/pycharm/running-jupyter-notebook-cells.html#variables)

**Data sources**

Measurements

1. Temperature

{

id: iot\_sensor\_eadbed22-a59b-4728-ad8f-193fc45031e2|msr\_temp|1693226728,

type: 'Measurement\_Temperature',

value: float

}

2. Consumption

{

id: iot\_sensor\_eadbed22-a59b-4728-ad8f-193fc45031e2|msr\_cons|1693226728,

type: 'Measurement\_Consumption',

value: float

}

3. Content (lt)

{

id: iot\_sensor\_eadbed22-a59b-4728-ad8f-193fc45031e2|msr\_cont|1693226728,

type: 'Measurement\_Content',

value: float

}

4. Set measurement interval

5. Set threshold

Interconnect: Service Registration Flow

