Scorpio Broker Documentation

Release 0.9

NEC Laboratories Europe GmbH

Dec 20, 2019

Contents

Scorpio is an NGSI-LD compliant context broker developed by NEC Laboratories Europe and NEC Technologies India.

CHAPTER 1

NGSI-LD

NGSI-LD is an open API and Datamodel specification for context management [published by ETSI](https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf).

CHAPTER 2

Building

Scorpio is developed in Java using SpringCloud as microservice framework and Apache Maven as build tool. Some of the tests require a running Apache Kafka messagebus (further instruction are in the Setup chapter). If you want to skip those tests you can run “mvn clean package -DskipTests” to just build the individual microservices.

Source code avilable at Github

https://github.com/ScorpioBroker/ScorpioBroker

CHAPTER 3

Setup

Scorpio requires two components to be installed.

3.1 Postgres

Please download the [Postgres DB](https://www.postgresql.org/) and the [Postgis](https://postgis.net) extension and follow the instructions on the websites to set them up.

Scorpio has been tested and developed with Postgres 10.

The default username and password which Scorpio uses is "ngb". If you want to use a different username or password you need to provide them as parameter when starting the StorageManager and the RegistryManager.

e.g.

```
java -jar Storage/StorageManager/target/StorageManager-<VERSIONNUMBER>-SNAPSHOT.jar \
    --reader.datasource.username=funkyusername --reader.datasource.
↪password=funkypassword
```

OR

```
java -jar Registry/RegistryManager/target/RegistryManager-<VERSIONNUMBER>-SNAPSHOT.
↪jar \
    --spring.datasource.username=funkyusername --spring.datasource.
↪password=funkypassword`
```

Don't forget to create the corresponding user ("ngb" or the different username you chose) in postgres. It will be used by the SpringCloud services for database connection. While in terminal, log in to the psql console as postgres user:

```
sudo -u postgres psql
```

Then create a database "ngb":

```
postgres=# create database ngb;
```

Create a user “ngb” and make him a superuser:

```
postgres=# create user ngb with encrypted password 'ngb';
postgres=# alter user ngb with superuser;
```

Grant privileges on database:

```
postgres=# grant all privileges on database ngb to ngb;
```

Also create an own database/schema for the Postgis extension:

```
postgres=# CREATE DATABASE gisdb;
postgres=# \connect gisdb;
postgres=# CREATE SCHEMA postgis;
postgres=# ALTER DATABASE gisdb SET search_path=public, postgis, contrib;
postgres=# \connect gisdb;
postgres=# CREATE EXTENSION postgis SCHEMA postgis;
```

3.2 Apache Kafka

Scorpio uses [Apache Kafka](https://kafka.apache.org/) for the communication between the microservices.

Scorpio has been tested and developed with Kafka version 2.12-2.1.0

Please download [Apache Kafka](https://kafka.apache.org/downloads) and follow the instructions on the website.

In order to start kafka you need to start two components:

Start zookeeper with

```
<kafkafolder>/bin/[Windows]/zookeeper-server-start.[bat|sh] <kafkafolder>/config/
↪zookeeper.properties
```

Start kafkaserver with

```
<kafkafolder>/bin/[Windows]/kafka-server-start.[bat|sh] <kafkafolder>/config/server.
↪properties
```

For more details please visit the Kafka website.

CHAPTER 4

Getting a docker container

The current maven build supports two types of docker container generations from the build using maven profiles to trigger it.

The first profile is called 'docker' and can be called like this

```
mvn clean package -DskipTests -Pdocker
```

this will generate individual docker containers for each micro service. The corresponding docker-compose file is *docker-compose-dist.yml*

The second profile is called 'docker-aaio' (for almost all in one). This will generate one single docker container for all components the broker except the kafka message bus and the postgres database.

To get the aaio version run the maven build like this

```
mvn clean package -DskipTests -Pdocker-aaio
```

The corresponding docker-compose file is *docker-compose-aaio.yml*

4.1 General remark for the Kafka docker image and docker-compose

The Kafka docker container requires you to provide the environment variable *KAFKA_ADVERTISED_HOST_NAME*. This has to be changed in the docker-compose files to match your docker host ip. You can use *127.0.0.1* however this will disallow you to run Kafka in a cluster mode.

For further details please refer to https://hub.docker.com/r/wurstmeister/kafka

4.2 Running docker build outside of Maven

If you want to have the build of the jars separated from the docker build you need to provide certain VARS to docker. The following list shows all the vars and their intended value if you run docker build from the root dir

- BUILD_DIR_ACS = Core/AtContextServer
- BUILD_DIR_SCS = SpringCloudModules/config-server
- BUILD_DIR_SES = SpringCloudModules/eureka
- BUILD_DIR_SGW = SpringCloudModules/gateway
- BUILD_DIR_HMG = History/HistoryManager
- BUILD_DIR_QMG = Core/QueryManager
- BUILD_DIR_RMG = Registry/RegistryManager
- BUILD_DIR_EMG = Core/EntityManager
- BUILD_DIR_STRMG = Storage/StorageManager
- BUILD_DIR_SUBMG = Core/SubscriptionManager
- JAR_FILE_BUILD_ACS = AtContextServer-${project.version}.jar
- JAR_FILE_BUILD_SCS = config-server-${project.version}.jar
- JAR_FILE_BUILD_SES = eureka-server-${project.version}.jar
- JAR_FILE_BUILD_SGW = gateway-${project.version}.jar
- JAR_FILE_BUILD_HMG = HistoryManager-${project.version}.jar
- JAR_FILE_BUILD_QMG = QueryManager-${project.version}.jar
- JAR_FILE_BUILD_RMG = RegistryManager-${project.version}.jar
- JAR_FILE_BUILD_EMG = EntityManager-${project.version}.jar
- JAR_FILE_BUILD_STRMG = StorageManager-${project.version}.jar
- JAR_FILE_BUILD_SUBMG = SubscriptionManager-${project.version}.jar
- JAR_FILE_RUN_ACS = AtContextServer.jar
- JAR_FILE_RUN_SCS = config-server.jar
- JAR_FILE_RUN_SES = eureka-server.jar
- JAR_FILE_RUN_SGW = gateway.jar
- JAR_FILE_RUN_HMG = HistoryManager.jar
- JAR_FILE_RUN_QMG = QueryManager.jar
- JAR_FILE_RUN_RMG = RegistryManager.jar
- JAR_FILE_RUN_EMG = EntityManager.jar
- JAR_FILE_RUN_STRMG = StorageManager.jar
- JAR_FILE_RUN_SUBMG = SubscriptionManager.jar

CHAPTER 5

Starting of the components

After the build start the individual components as normal Jar files.

Start the SpringCloud services by running

```
java -jar SpringCloudModules/eureka/target/eureka-server-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar SpringCloudModules/gateway/target/gateway-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar SpringCloudModules/config-server/target/config-server-<VERSIONNUMBER>-
↪SNAPSHOT.jar
```

Start the broker components

```
java -jar Storage/StorageManager/target/StorageManager-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar Core/QueryManager/target/QueryManager-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar Registry/RegistryManager/target/RegistryManager-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar Core/EntityManager/target/EntityManager-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar History/HistoryManager/target/HistoryManager-<VERSIONNUMBER>-SNAPSHOT.jar
java -jar Core/SubscriptionManager/target/SubscriptionManager-<VERSIONNUMBER>-
↪SNAPSHOT.jar
java -jar Core/AtContextServer/target/AtContextServer-<VERSIONNUMBER>-SNAPSHOT.jar
```

5.1 Changing config

All configurable options are present in application.properties files. In order to change those you have two options. Either change the properties before the build or you can override configs by add *–<OPTION_NAME>=<OPTION_VALUE)* e.g.

java -jar Storage/StorageManager/target/StorageManager-<VERSIONNUMBER>-SNAPSHOT.jar –reader.datasource.username=funkyusername –reader.datasource.password=funkypassword

CHAPTER 6

Basic interaction

By default the broker runs on port 9090 the base URL for interaction with the broker would be than *http://localhost:9090/ngsi-ld/v1/* For a detail explaination about the API please look the ETSI spec.

Generally speaking you can Create entities by sending an HTTP POST request to *http://localhost:9090/ngsi-ld/v1/entities* with a payload like this

```
{
    "id": "urn:ngsi-ld:testunit:123",
    "type": "AirQualityObserved",
    "dateObserved": {
        "type": "Property",
        "value": {
            "@type": "DateTime",
            "@value": "2018-08-07T12:00:00Z"
        }
    },
    "NO2": {
        "type": "Property",
        "value": 22,
        "unitCode": "GP",
        "accuracy": {
            "type": "Property",
            "value": 0.95
        }
    },
    "refPointOfInterest": {
        "type": "Relationship",
        "object": "urn:ngsi-ld:PointOfInterest:RZ:MainSquare"
    },
    "@context": [
        "https://schema.lab.fiware.org/ld/context",
        "https://uri.etsi.org/ngsi-ld/v1/ngsi-ld-core-context.jsonld"
    ]
}
```

In the given example the @context is in the payload therefor you have to set the ContentType header to application/ld+json

To receive entities you can send an HTTP GET to

http://localhost:9090/ngsi-ld/v1/entities/<entityId>

or run a query by sending a GET like this

```
http://localhost:9090/ngsi-ld/v1/entities/?type=Vehicle&limit=2
Accept: application/ld+json
Link: <http://<HOSTNAME_OF_WHERE_YOU_HAVE_AN_ATCONTEXT>/aggregatedContext.jsonld>;␣
↪rel="http://www.w3.org/ns/json-ld#context";type="application/ld+json"
```

For more detailed explaination on NGSI-LD or JSON-LD. Please look at the [ETSI Specification](https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf) or visit the [JSON-LD website](https://json-ld.org/).

CHAPTER 7

Troubleshooting

7.1 Missing JAXB dependencies

When starting the eureka-server you may facing the **java.lang.TypeNotPresentException: Type javax.xml.bind.JAXBContext not present** exception. It's very likely that you are running Java 11 on your machine then. Starting from Java 9 package *javax.xml.bind* has been marked deprecated and was finally completely removed in Java 11.

In order to fix this issue and get eureka-server running you need to manually add below JAXB Maven dependencies to *ScorpioBroker/SpringCloudModules/eureka/pom.xml* before starting:

```
...
<dependencies>
        ...
        <dependency>
                <groupId>com.sun.xml.bind</groupId>
                <artifactId>jaxb-core</artifactId>
                <version>2.3.0.1</version>
        </dependency>
        <dependency>
                <groupId>javax.xml.bind</groupId>
                <artifactId>jaxb-api</artifactId>
                <version>2.3.1</version>
        </dependency>
        <dependency>
                <groupId>com.sun.xml.bind</groupId>
                <artifactId>jaxb-impl</artifactId>
                <version>2.3.1</version>
        </dependency>
        ...
</dependencies>
...
```