HORIZON 2020
The EU Framework Programme for Research and Innovation

FIESTA-IoT
www.fiesta-iot.eu

HORIZONS 2020 PROGRAMME

Research and Innovation Action – FIRE Initiative

Call Identifier:	H2020–ICT–2014–1
Project Number:	643943
Project Acronym:	FIESTA-IoT
Project Title:	Federated Interoperable Semantic IoT/cloud Testbeds and Applications

D3.6 Concept and Development for IoT Data Analytics and IoT Stream and Service Management

Document Id:	FIESTAIoT-WP3-D3.6-ConceptDevelopmentDataAnalyticsServiceManagement-V04.doc
File Name:	FIESTAIoT-WP3-D3.6-ConceptDevelopmentDataAnalyticsServiceManagement-V04.doc
Document reference:	Deliverable 3.6
Version:	V04
Editor:	Alireza Ahrabian, Tarek Elsaleh, Francois Carrez
Organisation:	UNIS
Date:	26 / 10 / 2017
Document type:	Report, Other
Dissemination level:	PU

Copyright © 2017 National University of Ireland - NUIG / Coordinator (Ireland), University of Southampton IT Innovation - ITINNOV (United Kingdom), Institut National de Recherche en Informatique & Automatique - INRIA, (France), University of Surrey - UNIS (United Kingdom), Unparallel

Innovation, Lda - UNINNOVA (Portugal), Easy Global Market - EGM (France), NEC Europe Ltd. NEC (United Kingdom), University of Cantabria UNICAN (Spain), Association Plate-forme Telecom - Com4innov (France), Research and Education Laboratory in Information Technologies - Athens Information Technology - AIT (Greece), Sociedad para el desarrollo de Cantabria – SODERCAN (Spain), Ayuntamiento de Santander – SDR (Spain), Korea Electronics Technology Institute KETI, (Korea).

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Alireza Ahrabian	UNIS	08/05/2017	Initial Document Template
V01	Alireza Ahrabian	UNIS	25/06/2017	Updated content for FIESTA-IoT Analytics. Updated table of contents for FIESTA-IoT Reasoning and Monitoring
V01	Rachit Agarwal	INRIA	22/08/2017	Platform Monitoring
V01	Hung Nguyen/ Elias Tragos	NUIG	17/09/2017	Reasoning
V02	Ronald Steinke/ Alireza Ahrabian	FOKUS/UNIS	27/09/2017	Testbed Monitoring and content merging
V03	Alireza Ahrabian	UNIS	06/10/2017	Document Ready for Review
V03	Alireza Ahrabian/ Hung Nguyen/ Elias Tragos/ Rachit Agarwal/ Ronald Steinke	UNIS/NUIG/ INRIA/FOKUS	23/10/2017	QR's: Paul Grace (ITINNOV) and Tiago Teixeira (UNPARALLEL) TR's: Nikos Kefalakis (AIT) and Jorge Lanza Calderón (UNICAN)
V04	Alireza Ahrabian, Tarek Elsaleh	UNIS	26/10/2017	Final version after reviewer comments

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

API	Application Programming Interface
CQELS	Continuous Query Evaluation over Linked Streams
GUI	Graphical User Interface
ICO	Internet Connected Object
IoT	Internet of Things
Jena	Framework to build semantic web applications
Jena TDB	Database to store semantic information called triple
KAT	Knowledge Acquisition Toolkit
LD4Sensors	A tool to link sensor data
LER	Linked Edit Rules
LOD	Linked Open Data
LOV	Linked Open Vocabularies
LOV4IoT	Linked Open Vocabularies for Internet of Things (LOV4IoT)
LSM	Linked Sensor Middleware
M3	Machine to Machine Measurement framework
SAO	Stream Annotation Ontology
S-LOR	Sensor-based Linked Open Rules
SDR	Semantic Data Repository
SSN	Semantic Sensor Networks

1 INTRODUCTION

1.1 Executive Summary

FIESTA-IoT facilitates the capture, storage and processing of data generated from a variety of testbeds in an interoperable manner. This document provides a description of the tools developed for processing data that is captured and stored by the FIESTA-IoT platform. Namely, the FIESTA-IoT Analytics, FIESTA-IoT Reasoning and FIESTA-IoT Monitoring tools.

The FIESTA-IoT Analytics tool uses the concept of historical data analysis as a web service, where the experimenter can select a variety of methods for the processing of data. This deliverable provides an explanation of the function for each technique along with a description of the tools parameters and error responses when making a request. FIESTA-IoT Reasoning provides a reasoning engine web service for obtaining actionable knowledge from streamed data sets. This is achieved by using a set of rules (i.e. if-then conditions) that are applied to the data. The tool provides the means for the experimenter to create rules along with rule execution and storage. This document provides a detailed explanation of the FIESTA-IoT Reasoning tool, where examples of the tools use are also provided.

Finally, this document provides a detailed description of the FIESTA-IoT monitoring tool. That is, the tool continuously collects information regarding testbed performance. The tool provides a means of visually evaluating testbed status and performance. Furthermore, this tool provides a convenient method of assessing malfunctions at the sensor level for a given testbed.

2 FIESTA-IOT ANALYTICS

In this section, we provide an explanation of the proposed tool in relation to the FIESTA-IoT platform.

2.1 Introduction

This project proposes to provide data analysis tools as a web service, to enable a wider range of data consumer access to advanced data analysis tools. These algorithms are made available as HTTP rest methods; a data consumer can then run these algorithms on their data set by making the appropriate REST call as described in this section. A description of the data analysis tools that will be provided along with the components interaction within the FIESTA-IoT architecture was presented in deliverable D3.5 [1]. It was identified that the following data analysis tools would be provided, namely: 1) pre-processing methods, 2) supervised learning, 3) unsupervised learning algorithms and finally 4) other techniques and methods (examples of techniques belonging to this class are spectral and dependence estimation methods). In the subsequent sections we provide an overview of the API for interacting with the component. Furthermore, a detailed description of methods and parameters that are provided by the proposed tool, along with error messages that may arise is provided. Finally, use case examples of the proposed tool are also presented.

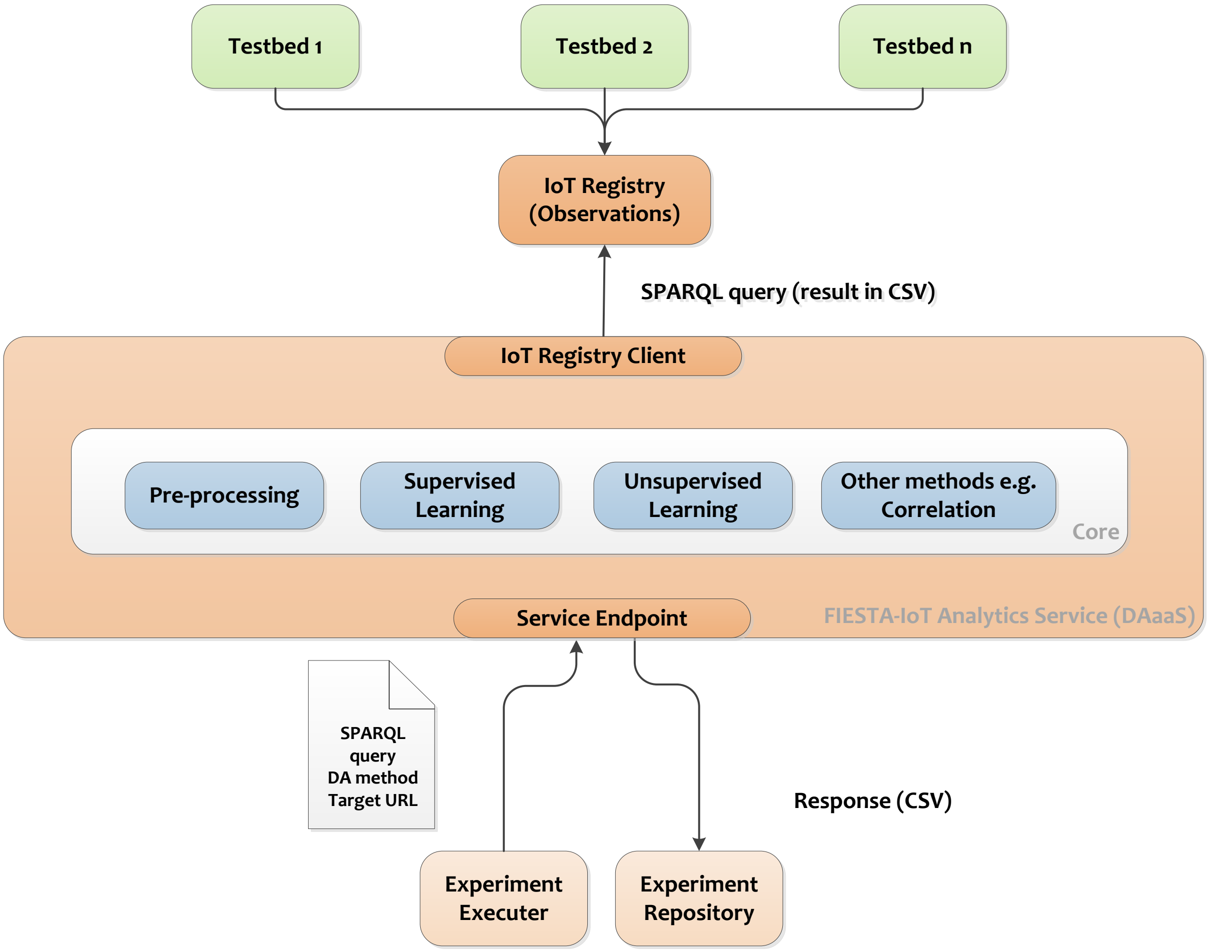


Figure 1: FIESTA-IoT Analytics Service Interaction

2.1 Component API Description

In order to invoke the proposed FIESTA-IoT Analytics service, a HTTP POST request must be made. The body of the request contains a JSON object that is shown in Figure 1. Namely, the list of methods and the corresponding parameters are provided along, with the SPARQL query and endpoint where the data can be obtained. A detailed description of the methods and the corresponding parameters are provided in the subsequent sections. It should be noted that a detailed explanation of the components interaction with the FIESTA-IoT platform can be found in Section 7.2 of deliverable 4.2 [2].

```
{
 "Method": ["Method 1","Method 2","Method 3"],
 "Parameters": ["Parameters 1", "Parameters 2", "Parameters 3"],
 "SPARQLquery":"SPARQL request string"
 "SPARQLendpoint":"SPARQL endpoint"
}
```

Figure 2: HTTP Request JSON Object

2.2 Methods and Parameters

This section provides a description of the functionality of the complete set of methods provided in the FIESTA-IoT Analytics tool, as well as the input parameters required for each method. Furthermore, a complete list of error messages is also presented.

2.2.1 Set of Methods

The corresponding list of methods and the corresponding functionality is presented below:

1. Pre-processing

Pre-processing techniques enable the data consumer to remove corrupted and noisy data points from the original raw time series data. The FIESTA-IoT Analytics tool provides two such methods, namely, digital filtering and outlier removal (please refer to Table 1 for list and description of the relevant methods).

Method	**Method Description**
Outlier	A method for removing outliers from de-trended signals. The method used is the winsorization technique.
FilterData	An implementation of a finite impulse response (FIR) digital filter.

Table 1: Pre-Processing Techniques List

2. Unsupervised Learning

This tool provides exploratory data analysis tools, namely unsupervised machine learning techniques to enable the experimenter to discover patterns of interest in the data set being analysed (please refer to Table 2 for list and description of the relevant methods).

Method	**Method Description**
KMeans	This technique clusters the data. That is, the data is grouped together based on a minimization of the distance between the data points and the centroids of the clusters.
PCA	Dimensionality reduction technique that can also be used for inferring the directions of variability in the data set.

Table 2: Unsupervised Learning Techniques List

3. Supervised Learning

Many data analysis problems require the experimenter to either determine a relationship between a set of input and output data points, or to obtain an estimate of the output data points given the input data points. To this end, both linear and nonlinear supervised learning techniques are provided (please refer to Table 3 for list and description of the relevant methods).

Method	**Method Description**
LinReg	This method is an implementation of linear regression. Linear regression finds a linear relationship between the input data set and output data points.
KNNreg	K-Nearest Neighbours is a location based non-parametric approach for estimating the relationship between the input and output data points.

Table 3: Supervised Learning Techniques List

4. Other Methods

Data analysis tools that are not applicable to the above categories, are listed in the *other methods* field. This tool provides spectral analysis and data dependency estimation tools for the experimenter. Spectral estimation tools are particularly useful for designing digital filters for removing noise, while data dependency estimation tools are particularly useful for linear regression.

Method	Method Description
FFT	The Fast Fourier Transform, a method for determining the spectral content of a set of data points.
Periodogram	Periodogram function obtains an estimate of the power of a set of data points for a given frequency. Particularly useful in cases where data is corrupted with unwanted noise.
Correlation	Estimates the linear dependence between sets of variables/sensors.

Table 4: Other Methods List

2.2.2 Methods and Parameters Description

In the previous section 2.2.1, a description of the methods included in the FIESTA-IoT Analytics tool was provided. In this section, the description of the list of methods and the corresponding input parameters (along with the parameter data type) is presented.

1. Pre-processing

Table 5 provides a description of the input parameters for the pre-processing methods.

Method	Parameters	Description
Outlier	*Thresh*	Value between **0** and **1**, selects the percentage of tail values to remove from the ordered time series data. Type: Float
FilterData	*Type*	Select between, **"*B*"** Bandpass Filter, **"*L*"** Lowpass filter and Highpass filter **"*H*"**. Type: String
	cutoff_1	For the respective filters is the first normalised cutoff frequency, between 0 and 0.5. Type: Float
	cutoff_2	For bandpass filter only, the second cutoff frequency. Type: Float
	numtaps	Filter length. Usually select 30. Type: Integer

Table 5: List of methods and parameters for pre-processing techniques.

2. Unsupervised Learning

Table 6 provides a description of the input parameters for the unsupervised learning methods.

Method	Parameters	Description
KMeans	*NumClusters*	The number of clusters to select. Type: Integer
	Mode	Select either, "***CL***" the cluster label assigned to each data point, ***"Cent"*** the estimated centroid for each

		cluster group or ***“Sil”*** to determine the silhouette coefficient. Type: String
PCA	*Mode*	Select either, ***“ExpVar”*** the explained variance for the different principal components, or ***“Comp”*** the principal component loadings that is the direction in the data corresponds to the highest variance. Type: String

Table 6: List of methods and parameters for unsupervised learning techniques.

3. Supervised Learning

In Table 7 a description of the input parameters for the supervised learning methods is provided.

Method	**Parameters**	**Description**
LinReg	*Type*	Select between, **“*Param*”** the estimated parameters of the regression model, and **“*Predict*”** the estimate of the output given the test data. Type: String
	Dependant	Select the column index corresponding to the dependent variable. Type: Integer
	Ratio	Select the ratio of the training data to test data. Value between 0 and 1. Type: Float
KNNreg	*Num*	Selects the number of nearest neighbours. Type: Integer
	Dependant	Select the column index corresponding to the dependent variable. Type: Integer
	Ratio	Select the ratio of the training data to test data. Value between 0 and 1. Type: Float

Table 7: List of methods and parameters for supervised learning techniques.

4. Other Methods

The corresponding methods do not require any parameters: FFT, Periodogram and Correlation.

2.2.3 Error Messages

The FIESTA-IoT Analytics tool has integrated into the response of a given data analysis query a set of error messages to enable both the experimenter and potentially the FIESTA-IoT platform to identify the origin of the error.

Error Message	**Description**
Incorrect JSON Format	The input JSON object has been specified incorrectly. Response Code: 400
Incorrect JSON labels	The JSON keys do not follow the format specified in Figure 1. Response Code: 400
Content In the Incorrect Format	The content type in the request header is incorrectly specified. It should be: application/json. Response Code: 400
userId Header not included	userId key and/or value not included in the request header. Response Code: 400
femoId Header not included	femoId key and/or value not included in the request header. Response Code: 400
jobId Header not included	jobId key and/or value not included in the request header. Response Code: 400
Unable to connect to SPARQL endpoint	Either invalid SPARQL endpoint address or a timeout owing to slow response of the SPARQL endpoint. Response Code: 400
Unable to retrieve data	Data was not retrieved from the SPARQL endpoint. Response Code: 200
Length of data too small	The number of data points are too small for the FIESTA-IoT Analytics tool. Response Code: 200
Either time stamp or data variable missing	For a given timestamp/data value column pair, either timestamp or the data value column is missing from the data requested for processing. Response Code: 400
Incorrect Method Name	Incorrectly specified method name: either technique name does not exist or spelling mistake. Response Code: 400
Incorrect Method Parameter Specification	Incorrectly specified method input parameters. Response Code: 400
Unable to store/or confirm storage of processed data	Storage of the processed data could not be confirmed. Response Code: 400
Incorrect Sequence of Methods	Certain combinations of data analysis methods are not allowed, owing to incompatibility with other techniques. Response Code: 400

No. of parameters not equal to the No. of methods	For a given method a parameter is missing or equivalently for a given parameter a method is missing. Response Code: 400
Error	A general error in the processing of the data points. Response Code: 400

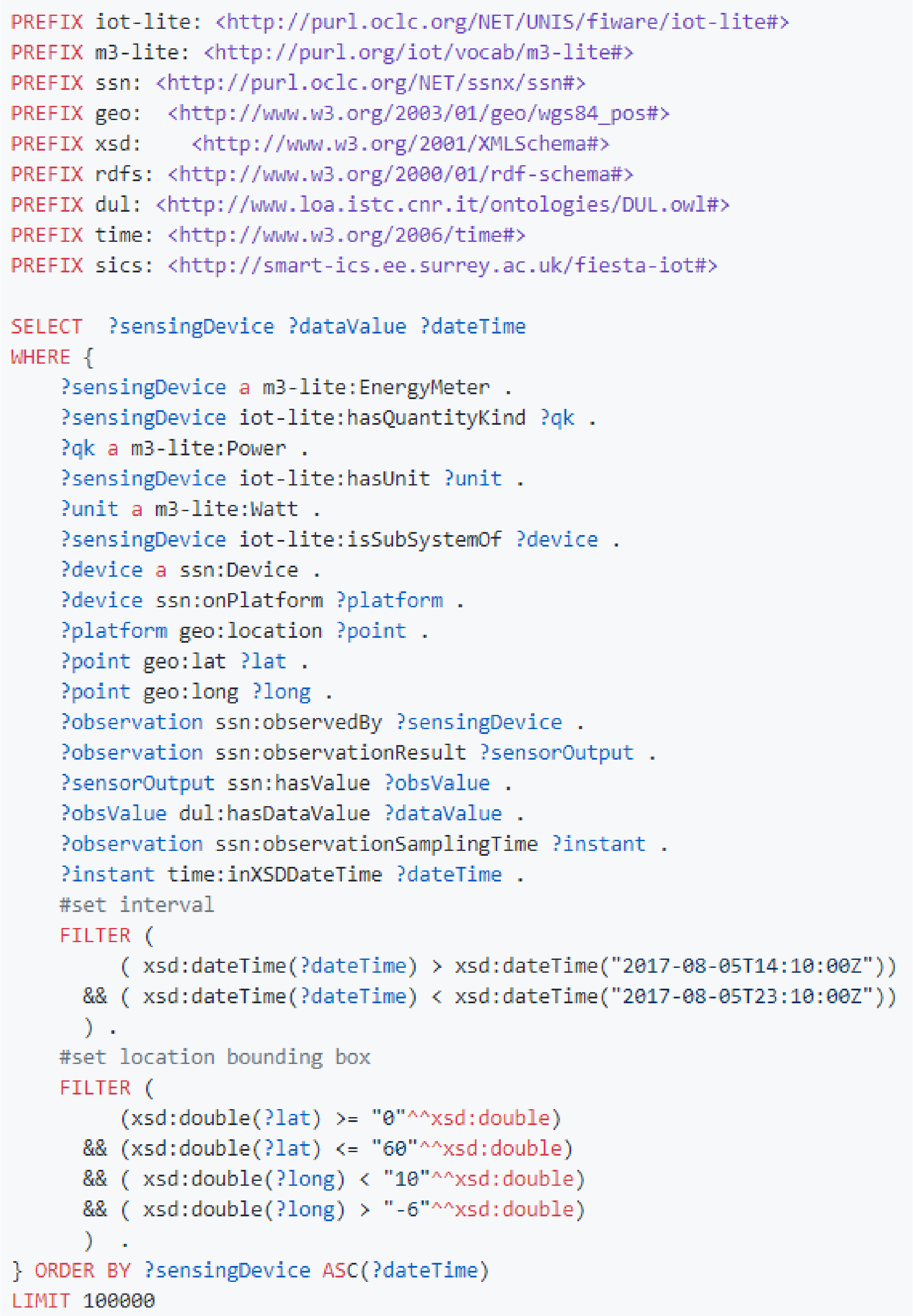
Table 8: List of error messages and corresponding descriptions.

2.3 FIESTA-IoT Analytics Use Case

In this section two use case examples of the FIESTA-IoT Analytics tool are provided.

2.3.1 Use Case Example 1 – Correlation Analysis

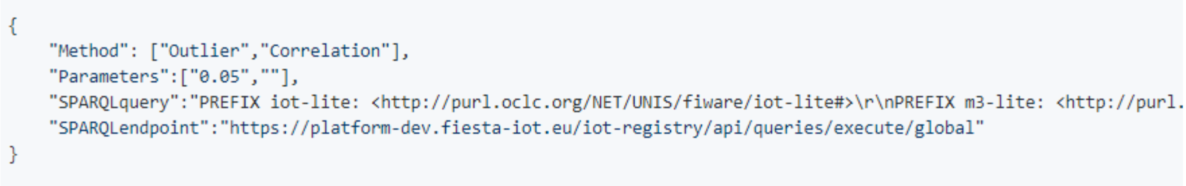
In the first use case, we illustrate the FIESTA-IoT Analytics tools performance in correlation analysis of data (using power data from the UNIS testbed, where an example of the SPARQL query request is shown in Figure 3) drawn from the IoT-Registry. Correlation enables the experimenter to determine dependencies/similarities that may exist between sensors. Such information can then be used to infer if sensors are measuring phenomena with similar dynamics. To this end, the FIESTA-IoT Analytics tool first applies an outlier removal algorithm (for suppressing artefacts that would reduce the effectiveness of carrying out correlation analysis), where then the correlation function is then called. The JSON body of the HTTP POST request sent to the FIESTA-IoT Analytics tool is shown in Figure 4.



```
PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>
PREFIX m3-lite: <http://purl.org/iot/vocab/m3-lite#>
PREFIX ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
PREFIX geo:  <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX xsd:    <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
PREFIX time: <http://www.w3.org/2006/time#>
PREFIX sics: <http://smart-ics.ee.surrey.ac.uk/fiesta-iot#>

SELECT  ?sensingDevice ?dataValue ?dateTime
WHERE {
    ?sensingDevice a m3-lite:EnergyMeter .
    ?sensingDevice iot-lite:hasQuantityKind ?qk .
    ?qk a m3-lite:Power .
    ?sensingDevice iot-lite:hasUnit ?unit .
    ?unit a m3-lite:Watt .
    ?sensingDevice iot-lite:isSubSystemOf ?device .
    ?device a ssn:Device .
    ?device ssn:onPlatform ?platform .
    ?platform geo:location ?point .
    ?point geo:lat ?lat .
    ?point geo:long ?long .
    ?observation ssn:observedBy ?sensingDevice .
    ?observation ssn:observationResult ?sensorOutput .
    ?sensorOutput ssn:hasValue ?obsValue .
    ?obsValue dul:hasDataValue ?dataValue .
    ?observation ssn:observationSamplingTime ?instant .
    ?instant time:inXSDDateTime ?dateTime .
    #set interval
    FILTER (
         ( xsd:dateTime(?dateTime) > xsd:dateTime("2017-08-05T14:10:00Z"))
      && ( xsd:dateTime(?dateTime) < xsd:dateTime("2017-08-05T23:10:00Z"))
      ) .
    #set location bounding box
    FILTER (
         (xsd:double(?lat) >= "0"^^xsd:double)
      && (xsd:double(?lat) <= "60"^^xsd:double)
      && ( xsd:double(?long) < "10"^^xsd:double)
      && ( xsd:double(?long) > "-6"^^xsd:double)
      )  .
} ORDER BY ?sensingDevice ASC(?dateTime)
LIMIT 100000
```

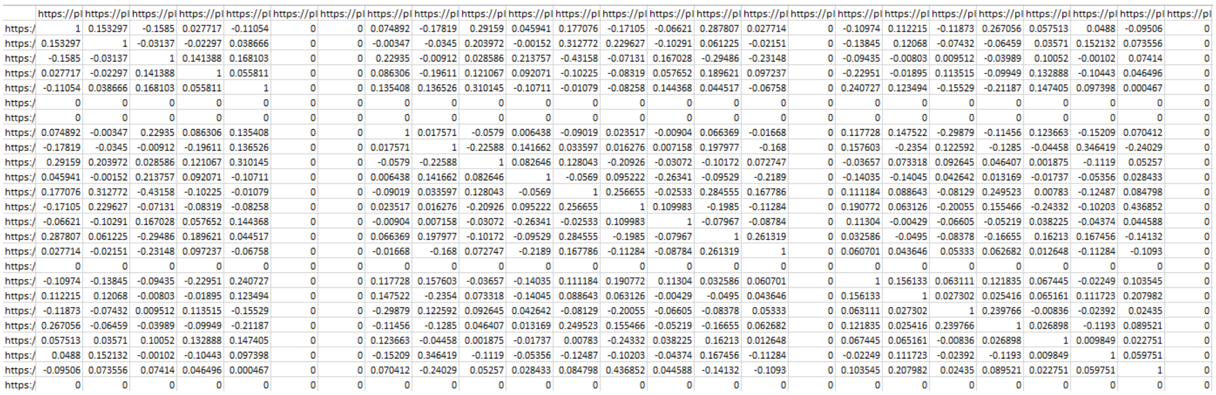
Figure 3: SPARQL query body for obtaining power sensor data and time stamps



```
{
    "Method": ["Outlier","Correlation"],
    "Parameters":["0.05",""],
    "SPARQLquery":"PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>\r\nPREFIX m3-lite: <http://purl.
    "SPARQLendpoint":"https://platform-dev.fiesta-iot.eu/iot-registry/api/queries/execute/global"
}
```

Figure 4: POST request body for FIESTA-IoT Analytics tool (SPARQL sentence shown in Figure 3)

The outcome of the FIESTA-IoT Analytics tool is shown in Figure 5. It should be noted that along the diagonal of the matrix, the output is equal to one (unless the data set is missing and the tool has set the correlation equal to zero) as it corresponds to the self correlation of the sensor. While the off-diagonal elements correspond to the correlation between different sensors. From Figure 5, it can be observed that most of the power sensors are uncorrelated with each other. This indicates that the activity of each sensor is largely independent of the neighbouring sensors, thereby indicating that activity profiles of each sensor are largely dissimilar.

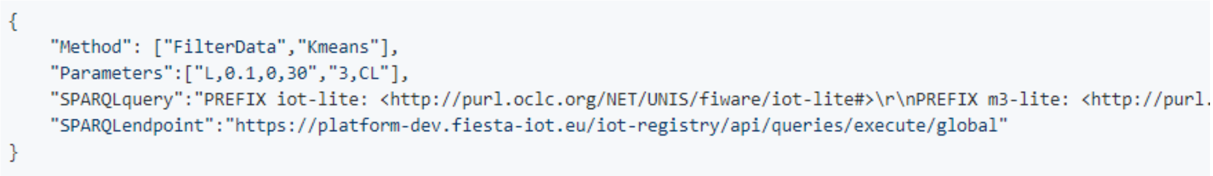


	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl	https://pl
https:/	1	0.153297	-0.1585	0.027717	-0.11054	0	0	0.074892	-0.17819	0.29159	0.045941	0.177076	-0.17105	-0.06621	0.287807	0.027714	0	-0.10974	0.112215	-0.11873	0.267056	0.057513	0.0488	-0.09506	0
https:/	0.153297	1	-0.03137	-0.02297	0.038666	0	0	-0.00347	-0.0345	0.203972	-0.00152	0.312772	0.229627	-0.10291	0.061225	-0.02151	0	-0.13845	0.12068	-0.07432	-0.06459	0.03571	0.152132	0.073556	0
https:/	-0.1585	-0.03137	1	0.141388	0.168103	0	0	0.22935	-0.00912	0.028586	0.213757	-0.43158	-0.07131	0.167028	-0.29486	-0.23148	0	-0.09435	-0.00803	0.009512	-0.03989	0.10052	-0.00102	0.07414	0
https:/	0.027717	-0.02297	0.141388	1	0.055811	0	0	0.086306	-0.19611	0.121067	0.092071	-0.10225	-0.08319	0.057652	0.189621	0.097237	0	-0.22951	-0.01895	0.113515	-0.09949	0.132888	-0.10443	0.046496	0
https:/	-0.11054	0.038666	0.168103	0.055811	1	0	0	0.135408	0.136526	0.310145	-0.10711	-0.01079	-0.08258	0.144368	0.044517	-0.06758	0	0.240727	0.123494	-0.15529	-0.21187	0.147405	0.097398	0.000467	0
https:/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
https:/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
https:/	0.074892	-0.00347	0.22935	0.086306	0.135408	0	0	1	0.017571	-0.0579	0.006438	-0.09019	0.023517	-0.00904	0.066369	-0.01668	0	0.117728	0.147522	-0.29879	-0.11456	0.123663	-0.15209	0.070412	0
https:/	-0.17819	-0.0345	-0.00912	-0.19611	0.136526	0	0	0.017571	1	-0.22588	0.141662	0.033597	0.016276	0.007158	0.197977	-0.168	0	0.157603	-0.2354	0.122592	-0.1285	-0.04458	0.346419	-0.24029	0
https:/	0.29159	0.203972	0.028586	0.121067	0.310145	0	0	-0.0579	-0.22588	1	0.082646	0.128043	-0.20926	-0.03072	-0.10172	0.072747	0	-0.03657	0.073318	0.092645	0.046407	0.001875	-0.1119	0.05257	0
https:/	0.045941	-0.00152	0.213757	0.092071	-0.10711	0	0	0.006438	0.141662	0.082646	1	-0.0569	0.095222	-0.26341	-0.09529	-0.2189	0	-0.14035	-0.14045	0.042642	0.013169	-0.01737	-0.05356	0.028433	0
https:/	0.177076	0.312772	-0.43158	-0.10225	-0.01079	0	0	-0.09019	0.033597	0.128043	-0.0569	1	0.256655	-0.02533	0.284555	0.167786	0	0.111184	0.088643	-0.08129	0.249523	0.00783	-0.12487	0.084798	0
https:/	-0.17105	0.229627	-0.07131	-0.08319	-0.08258	0	0	0.023517	0.016276	-0.20926	0.095222	0.256655	1	0.109983	-0.1985	-0.11284	0	0.190772	0.063126	-0.20055	0.155466	-0.24332	-0.10203	0.436852	0
https:/	-0.06621	-0.10291	0.167028	0.057652	0.144368	0	0	-0.00904	0.007158	-0.03072	-0.26341	-0.02533	0.109983	1	-0.07967	-0.08784	0	0.11304	-0.00429	-0.06605	-0.05219	0.038225	-0.04374	0.044588	0
https:/	0.287807	0.061225	-0.29486	0.189621	0.044517	0	0	0.066369	0.197977	-0.10172	-0.09529	0.284555	-0.1985	-0.07967	1	0.261319	0	0.032586	-0.0495	-0.08378	-0.16655	0.16213	0.167456	-0.14132	0
https:/	0.027714	-0.02151	-0.23148	0.097237	-0.06758	0	0	-0.01668	-0.168	0.072747	-0.2189	0.167786	-0.11284	-0.08784	0.261319	1	0	0.060701	0.043646	0.05333	0.062682	0.012648	-0.11284	-0.1093	0
https:/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
https:/	-0.10974	-0.13845	-0.09435	-0.22951	0.240727	0	0	0.117728	0.157603	-0.03657	-0.14035	0.111184	0.190772	0.11304	0.032586	0.060701	0	1	0.156133	0.063111	0.121835	0.067445	-0.02249	0.103545	0
https:/	0.112215	0.12068	-0.00803	-0.01895	0.123494	0	0	0.147522	-0.2354	0.073318	-0.14045	0.088643	0.063126	-0.00429	-0.0495	0.043646	0	0.156133	1	0.027302	0.025416	0.065161	0.111723	0.207982	0
https:/	-0.11873	-0.07432	0.009512	0.113515	-0.15529	0	0	-0.29879	0.122592	0.092645	0.042642	-0.08129	-0.20055	-0.06605	-0.08378	0.05333	0	0.063111	0.027302	1	0.239766	-0.00836	-0.02392	0.02435	0
https:/	0.267056	-0.06459	-0.03989	-0.09949	-0.21187	0	0	-0.11456	-0.1285	0.046407	0.013169	0.249523	0.155466	-0.05219	-0.16655	0.062682	0	0.121835	0.025416	0.239766	1	0.026898	-0.1193	0.089521	0
https:/	0.057513	0.03571	0.10052	0.132888	0.147405	0	0	0.123663	-0.04458	0.001875	-0.01737	0.00783	-0.24332	0.038225	0.16213	0.012648	0	0.067445	0.065161	-0.00836	0.026898	1	0.009849	0.022751	0
https:/	0.0488	0.152132	-0.00102	-0.10443	0.097398	0	0	-0.15209	0.346419	-0.1119	-0.05356	-0.12487	-0.10203	-0.04374	0.167456	-0.11284	0	-0.02249	0.111723	-0.02392	-0.1193	0.009849	1	0.059751	0
https:/	-0.09506	0.073556	0.07414	0.046496	0.000467	0	0	0.070412	-0.24029	0.05257	0.028433	0.084798	0.436852	0.044588	-0.14132	-0.1093	0	0.103545	0.207982	0.02435	0.089521	0.022751	0.059751	1	0
https:/	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 5: Correlation between sensors

2.3.2 Use Case Example 2 – Clustering

In the next use case, a clustering algorithm was applied to the data points obtained from the SPARQL query shown in Figure 3. Clustering data sets enables the experimenter to identify patterns of interest that may arise in the data. Figure 6 illustrates the POST request made to the FIESTA-IoT Analytics tool, where a Low pass digital filtering was first applied to reduce high frequency noise artefacts.



```
{
    "Method": ["FilterData","Kmeans"],
    "Parameters":["L,0.1,0,30","3,CL"],
    "SPARQLquery":"PREFIX iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#>\r\nPREFIX m3-lite: <http://purl.
    "SPARQLendpoint":"https://platform-dev.fiesta-iot.eu/iot-registry/api/queries/execute/global"
}
```

Figure 6: POST request for filtering and clustering

To determine the correct number of groups so as to cluster the data appropriately, a measure of fit known as the silhouette coefficient (SC) was used. The coefficient is between [-1,1], where 1 is generally a good fit, that is the clusters are well separated and -1 is a poor fit. Using this approach, it was determined that 4 clusters with a silhouette coefficient of 0.94 was to be used. Figure 7 illustrates the label for each time point, that is, the “state of activity” for each time point. To determine the magnitude of the activity level, the magnitude of the centroids were taken. The centroids of the respective clusters were: Cluster 1 = 2303, Cluster 2 = 2075, Cluster 3 = 2380 and Cluster 4 = 2572, where high and low activities can then be determined. From Figure 7 we can observe that the nominal activity of the sensors is in cluster 1. Low activities corresponding to cluster 2 were capture between samples 2-8, 10-13 and 160-180. While high activity clusters 3 and 4 where for the samples indices greater than 246.

Such analysis enables the experimenter to reduce the dimensionality of highly complex data sets, to make simple inferences.

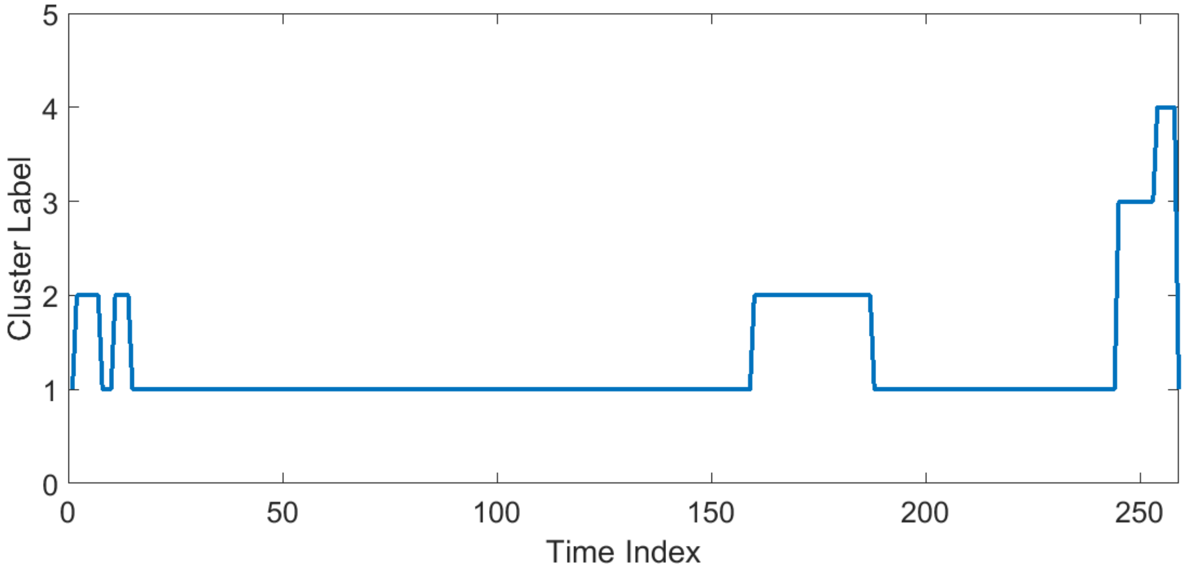


Figure 7: Data clustering

3 FIESTA-IOT REASONING

3.1 Introduction

The FIESTA-IoT Reasoning component is an implementation of a semantic reasoner to work on top of the FIESTA-IoT platform. A semantic reasoning engine is a rule based engine that is able to infer logical consequences from a set of IoT measurements. In doing so, the FIESTA-IoT Reasoner simplifies the creation of rules, which are generated and stored in a rule repository. This component provides a set of API services and a User Interface (UI) for experimenters, making it easy to design and execute rules base on the Apache Jena open source framework[1]. The reasoning module can be used by experimenters to create notifications or alerts based on the rules that they set for specific types of measurements coming from the FIESTA-IoT testbeds. For example, the experimenters might set rules in the form of expressions "if (condition) then (result)" as below:

- If (temperature) > (25degrees) then (notify_hot)
- If (speed) < (30km/h) then (notify_traffic)
- If (temperature) < (19degrees) and (humidity) > (60%) then (notify_unhealthy)

3.2 Architecture

In Figure 8 we provide an overview of the architecture of the FIESTA-IoT Reasoning engine. The central point of the architecture is the Reasoning Service Engine, which is the main component responsible for performing the reasoning services. This component is also connected with the FIESTA-IoT services and specifically with the IoT Registry (through its service endpoints) for getting the list of available sensors and quantity kinds for creating the rules and the measurements, upon which the rules will be executed.

The Reasoner Service Engine exposes three APIs for (i) Rule Creation, (ii) Rule Registration and (iii) Rule Execution, which are explained in detail in the next subsections. These APIs are connected with a MySQL database for storing the rules and the results of the executions.

The experimenters are provided with two options for using the Reasoning Engine: (i) through a web application user interface developed using AngularJS or (ii) thorough their own client application using REST APIs provided by the Reasoning Engine.

[1] https://jena.apache.org/index.html

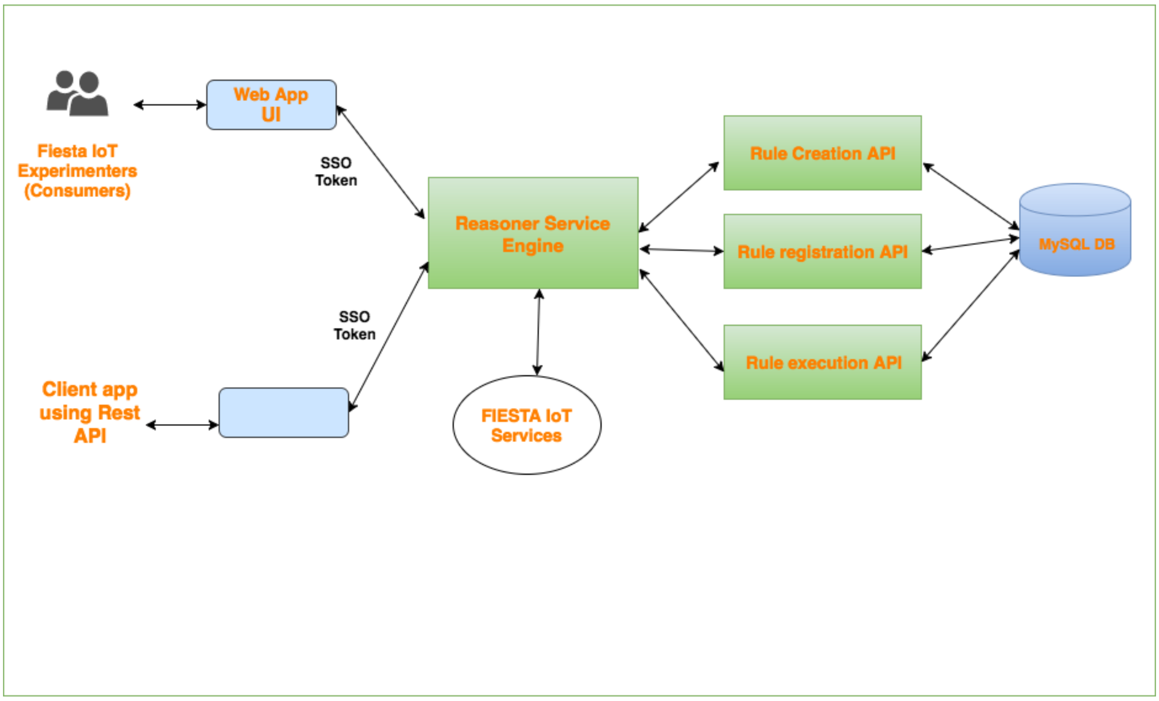


Figure 8: FIESTA-IoT Reasoning Architecture

In Figure 9 an example sequence diagram for creating a new rule is depicted. The first part shows the authentication procedure so that the experimenter can get a new token to be used for the next calls. When the experimenter wants to create a new rule, he sends through the API the new request to the reasoner service engine including all data with respect to the quantity kind, the sensor id (to whom the rule will be applied), etc. The Reasoner Service Engine requests then the validation that the information for the sensor and the quantity kind are ok. If this validation succeeds, the Reasoner Service Engine sends the data to the MySQL DB to add the new rule. The MySQL DB then replies with the validation (or error) message which is forwarded to the experimenter.

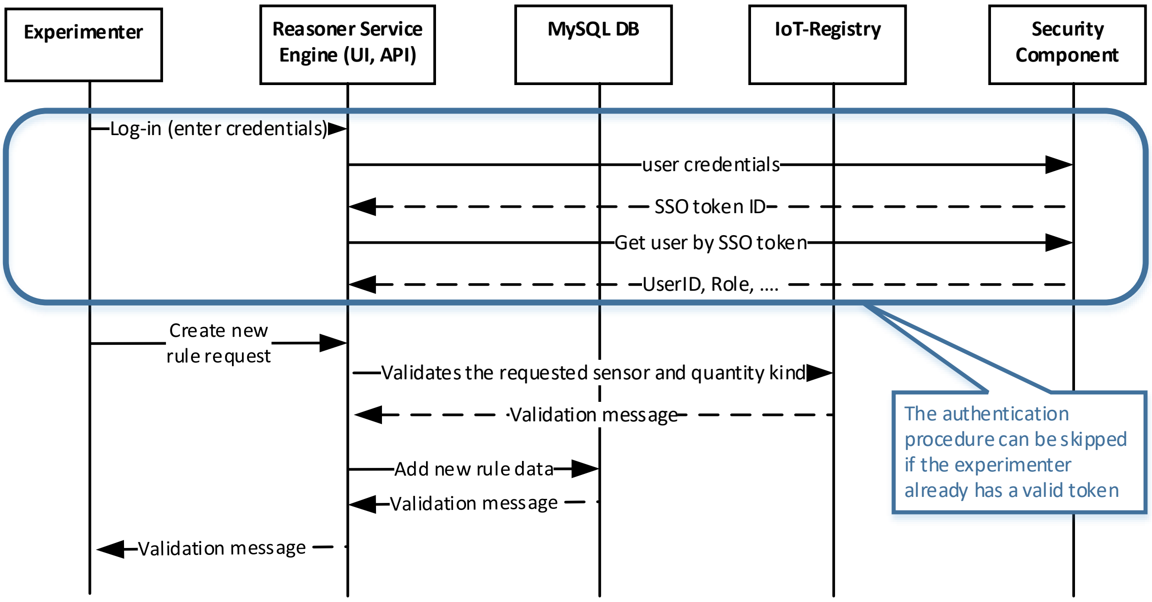


Figure 9: Example sequence diagram for creating a new reasoning rule

In Figure 10 an example sequence diagram for executing a rule is depicted. Same as before the authentication process precedes anything else. Then, the experimenter through the API sends a request for a new execution, by submitting the required data for the rule to be executed. The Reasoner Service Engine requests from the MySQL DB the list of registered rules, to validate the requested rule. Then the Reasoner Service Engine stores the information for the new execution to the MySQL DB. Next, the Reasoner Service Engine sends a request to the IoT-Registry for the observations of the selected sensor (for whom the rule is applied) and the selected time period. Finally, the response of the rule is sent to the experimenter.

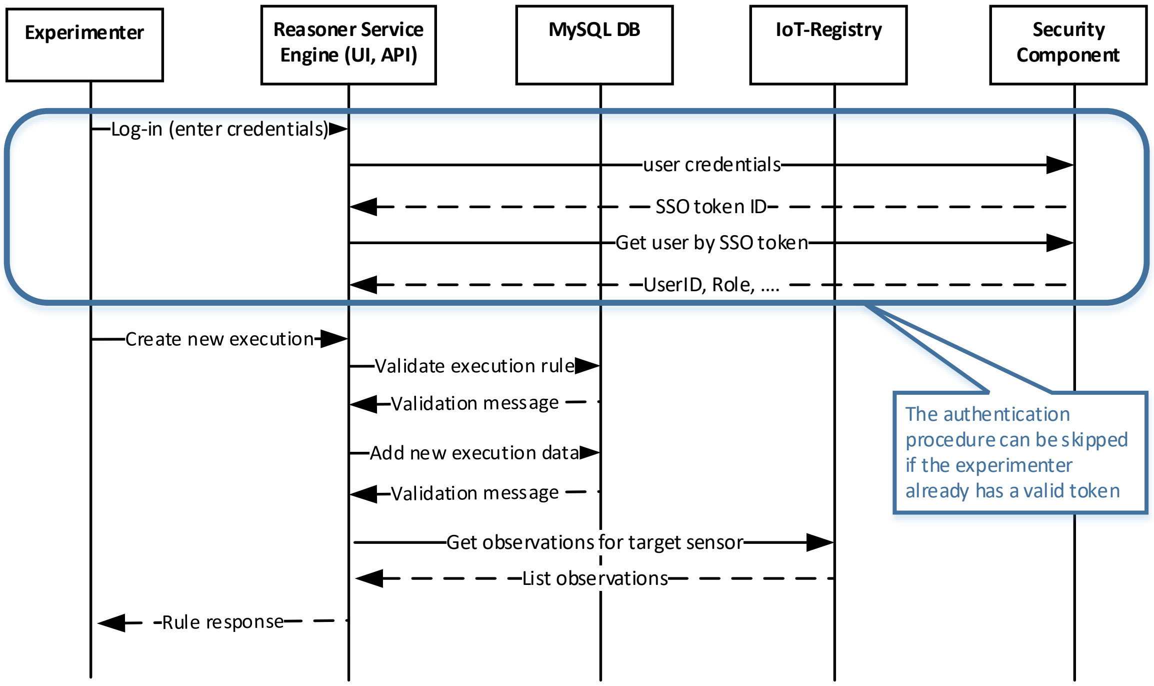
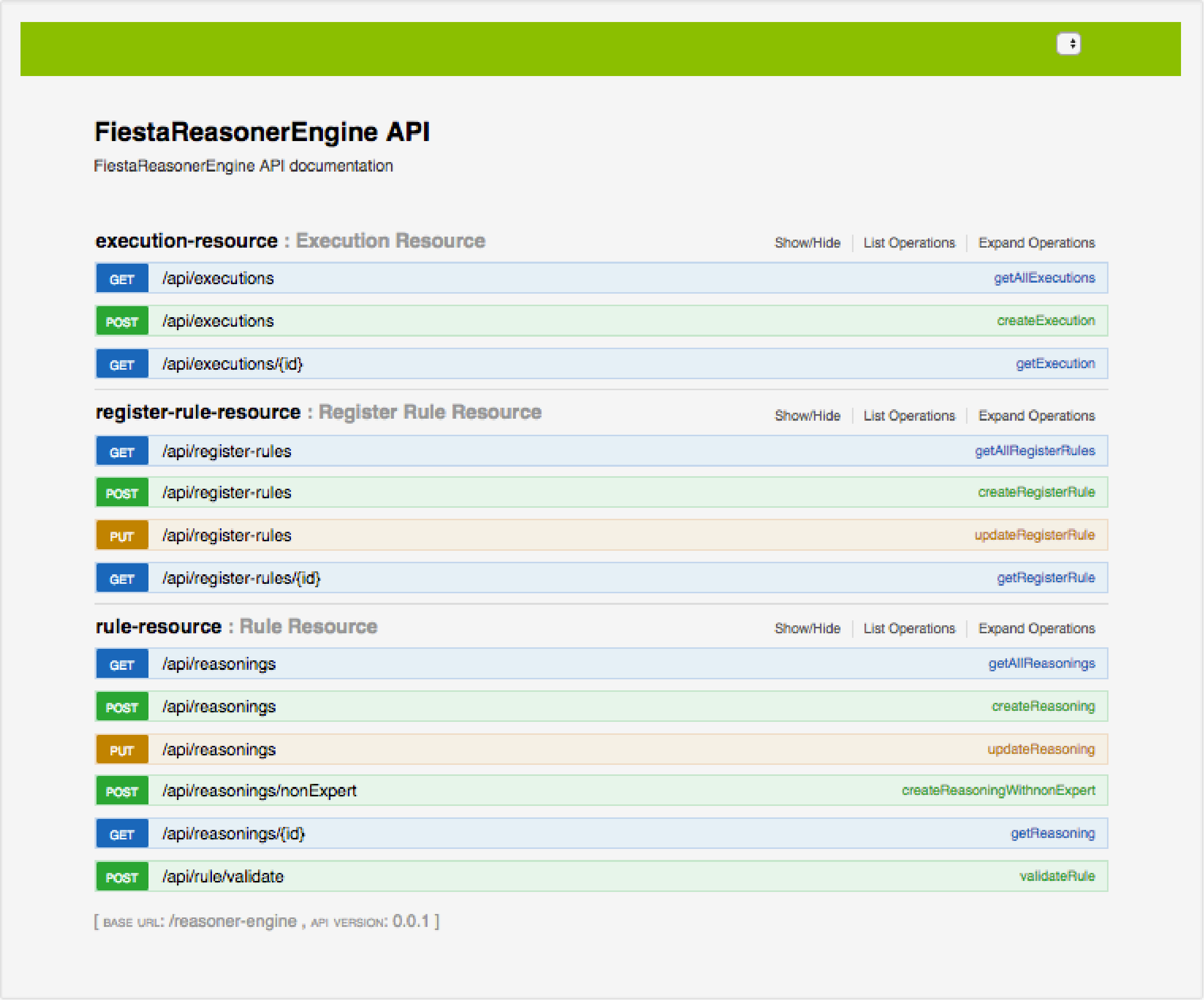


Figure 10: Example sequence diagram for creating and running a new rule execution.

The rest of the section below describes the implementation and usage of the Reasoning Services, while the UI tools for the Reasoning Engine will be described in Deliverable 4.8 [3]. The following sections present some example calls of the respective APIs. More detailed calls and the related response bodies are provided in the Annex.

3.3 Reasoning API

Currently, the FIESTA-IoT Reasoning module supports REST APIs for (i) Rule creation, (ii) Rule Registration and (iii) Rule Execution in the following three resources: (i) rule-resource, (ii) register-rule-resource and (iii) execution-resource, as presented in Figure 11. The documentation of the APIs can be found on the FIESTA-IoT portal, under the Help menu. Below, we describe the usage of these APIs.



FiestaReasonerEngine API

FiestaReasonerEngine API documentation

execution-resource : Execution Resource

GET	/api/executions	getAllExecutions
POST	/api/executions	createExecution
GET	/api/executions/{id}	getExecution

register-rule-resource : Register Rule Resource

GET	/api/register-rules	getAllRegisterRules
POST	/api/register-rules	createRegisterRule
PUT	/api/register-rules	updateRegisterRule
GET	/api/register-rules/{id}	getRegisterRule

rule-resource : Rule Resource

GET	/api/reasonings	getAllReasonings
POST	/api/reasonings	createReasoning
PUT	/api/reasonings	updateReasoning
POST	/api/reasonings/nonExpert	createReasoningWithnonExpert
GET	/api/reasonings/{id}	getReasoning
POST	/api/rule/validate	validateRule

[BASE URL: /reasoner-engine , API VERSION: 0.0.1]

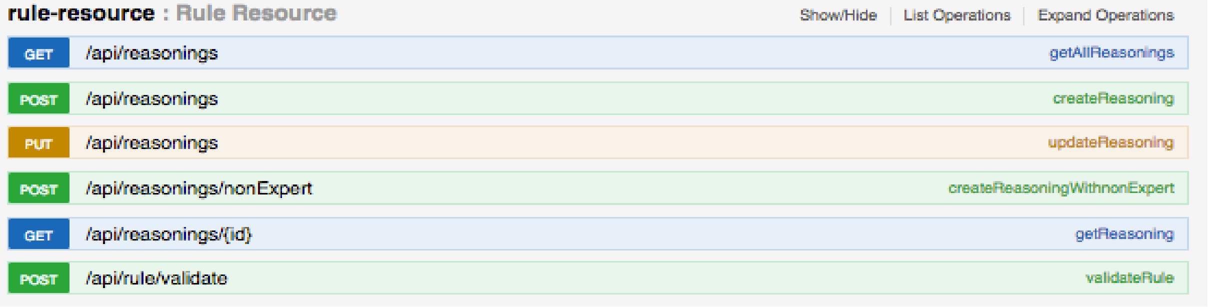
Figure 11: Reasoning API

All requests to the FIESTA-IoT Reasoning API must provide a header with:

- *Content-Type*: application/json
- *iPlanetDirectoryPro*: SSO Token obtained from the FIESTA IoT authentication API.

3.3.1 Rule Resource

The Rule Resource API provides several services that can be used by experimenters for creating, editing, updating and validating reasoning rules. Figure 12 presents the list of services currently supported in the rule-resource API.



rule-resource : Rule Resource

Show/Hide | List Operations | Expand Operations

GET	/api/reasonings	getAllReasonings
POST	/api/reasonings	createReasoning
PUT	/api/reasonings	updateReasoning
POST	/api/reasonings/nonExpert	createReasoningWithnonExpert
GET	/api/reasonings/{id}	getReasoning
POST	/api/rule/validate	validateRule

Figure 12: Rule resource API listing

3.3.1.1 Get all rules API

The "*getAllReasonings*" service (presented in Table 9) provides the function for experimenters to get the list of currently created rules, using parameters such as page, size, and sort. Since the list of created rules can be quite long in real deployments, the experimenter can select the rules according to the following parameters below to limit the number of rules.

Title	Get all Rules API
URL	*/api/reasonings*
Method	`GET`

URL Params	**Optional**
	`page=[alphanumeric]`
	sets the starting page number for the rules to be returned. For example, setting the page to "5", the reasoning engine will return the rules from number size*5 until size*6-1, depending on the size parameter below.
	size=`[alphanumeric]`
	sets the number of rules per page to be returned.
	`sort=[string]`
	sets the sorting criteria for the returned result set, i.e. ascending or descending.
	example:
	page=3
	size=30
	sort=asc

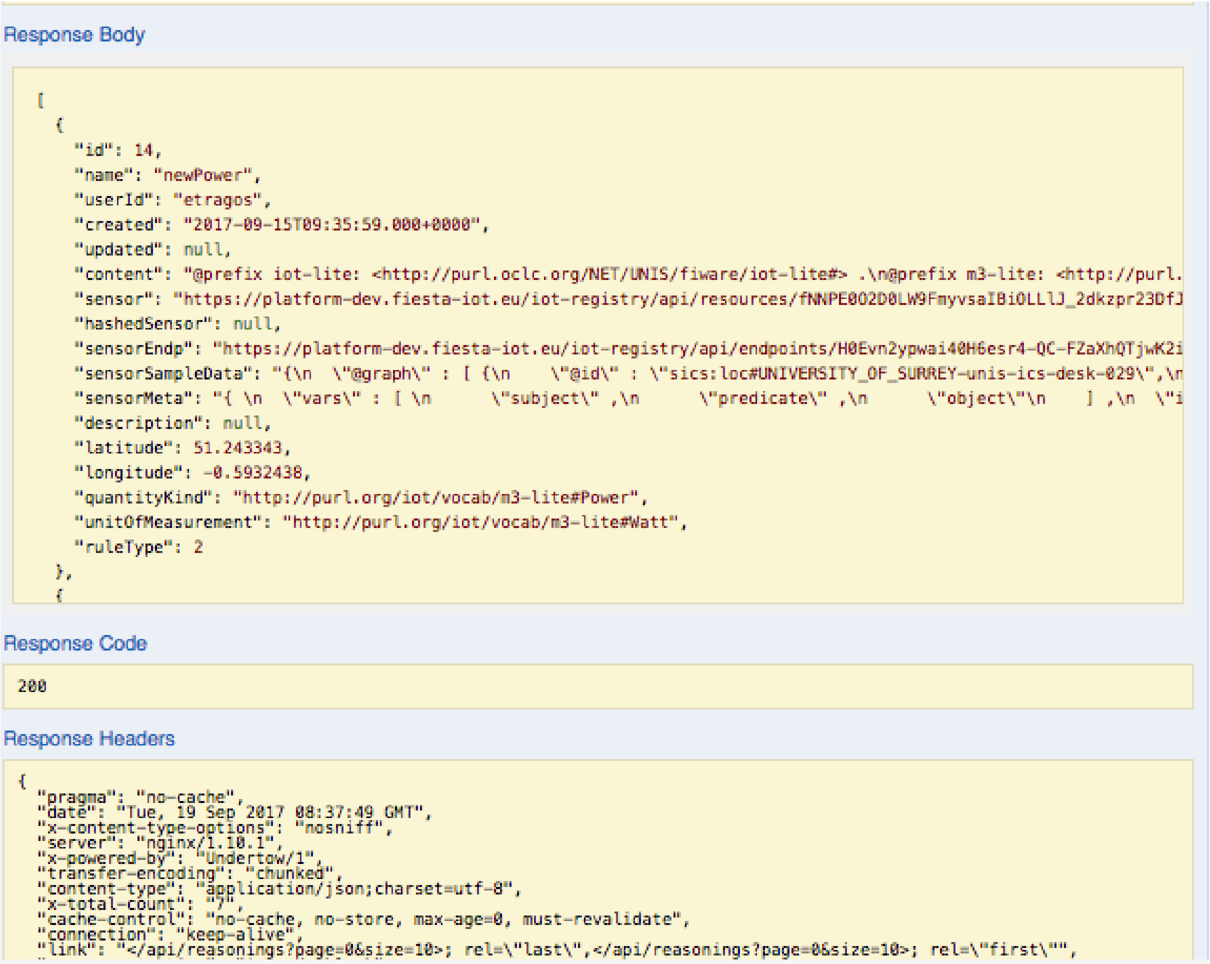
Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found

Sample Call	`curl -X GET --header 'Accept: application/json' 'https://platform.fiesta-iot.eu/reasoner-engine/api/reasonings?page=1&size=10&sort=asc'`
	
	`or`

	`https://platform.fiesta-iot.eu/reasoner-engine/api/reasonings?page=1&size=10&sort=asc`
Notes	This is where all uncertainties, commentary, discussion etc. can go. I recommend timestamping and identifying oneself when leaving comments here.

Table 9: Get All rules API

In Figure 13 an example of the response of the "*getAllReasonings*" service is provided. The response body includes a json array with the json strings presenting the parameters of each rule. The response code provides the http response which can be the verification that the request was successful or if there are any issues or errors.



```
[
  {
    "id": 14,
    "name": "newPower",
    "userId": "etragos",
    "created": "2017-09-15T09:35:59.000+0000",
    "updated": null,
    "content": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.
    "sensor": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/fNNPE0O2D0LW9FmyvsaIBiOLLlJ_2dkzpr23DfJ
    "hashedSensor": null,
    "sensorEndp": "https://platform-dev.fiesta-iot.eu/iot-registry/api/endpoints/H0Evn2ypwai40H6esr4-QC-FZaXhQTjwK2i
    "sensorSampleData": "{\n  \"@graph\" : [ {\n    \"@id\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-029\",\n
    "sensorMeta": "{ \n  \"vars\" : [ \n      \"subject\" ,\n      \"predicate\" ,\n      \"object\"\n    ] ,\n  \"i
    "description": null,
    "latitude": 51.243343,
    "longitude": -0.5932438,
    "quantityKind": "http://purl.org/iot/vocab/m3-lite#Power",
    "unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt",
    "ruleType": 2
  },
  {
```

```
200
```

```
{
  "pragma": "no-cache",
  "date": "Tue, 19 Sep 2017 08:37:49 GMT",
  "x-content-type-options": "nosniff",
  "server": "nginx/1.10.1",
  "x-powered-by": "Undertow/1",
  "transfer-encoding": "chunked",
  "content-type": "application/json;charset=utf-8",
  "x-total-count": "7",
  "cache-control": "no-cache, no-store, max-age=0, must-revalidate",
  "connection": "keep-alive",
  "link": "</api/reasonings?page=0&size=10>; rel=\"last\",</api/reasonings?page=0&size=10>; rel=\"first\"",
```

Figure 13: Get All rule example response

3.3.1.2 Get Rule by ID API

Experimenters can also query the reasoning engine to get a specific rule by providing the rule identification number. This can be done by using the "*getReasoning*" service of the rule-resource API, using as a parameter only the rule ID, as seen in Table 10. An example of the request for getting a rule by its id is also provided in the table. If the user is authenticated with the FIESTA-IoT platform, the request is as simple as accessing the URL https://platform-dev.fiesta-iot.eu/reasoner-engine/api/reasonings/{rule_id}.

Title	Get Rule by ID (getReasoning)
URL	/api/reasonings/{id}
Method	`GET`

URL Params	**Required:**
	`id=[integer]`
	`    The rule id to be returned`
	
	example: id=12

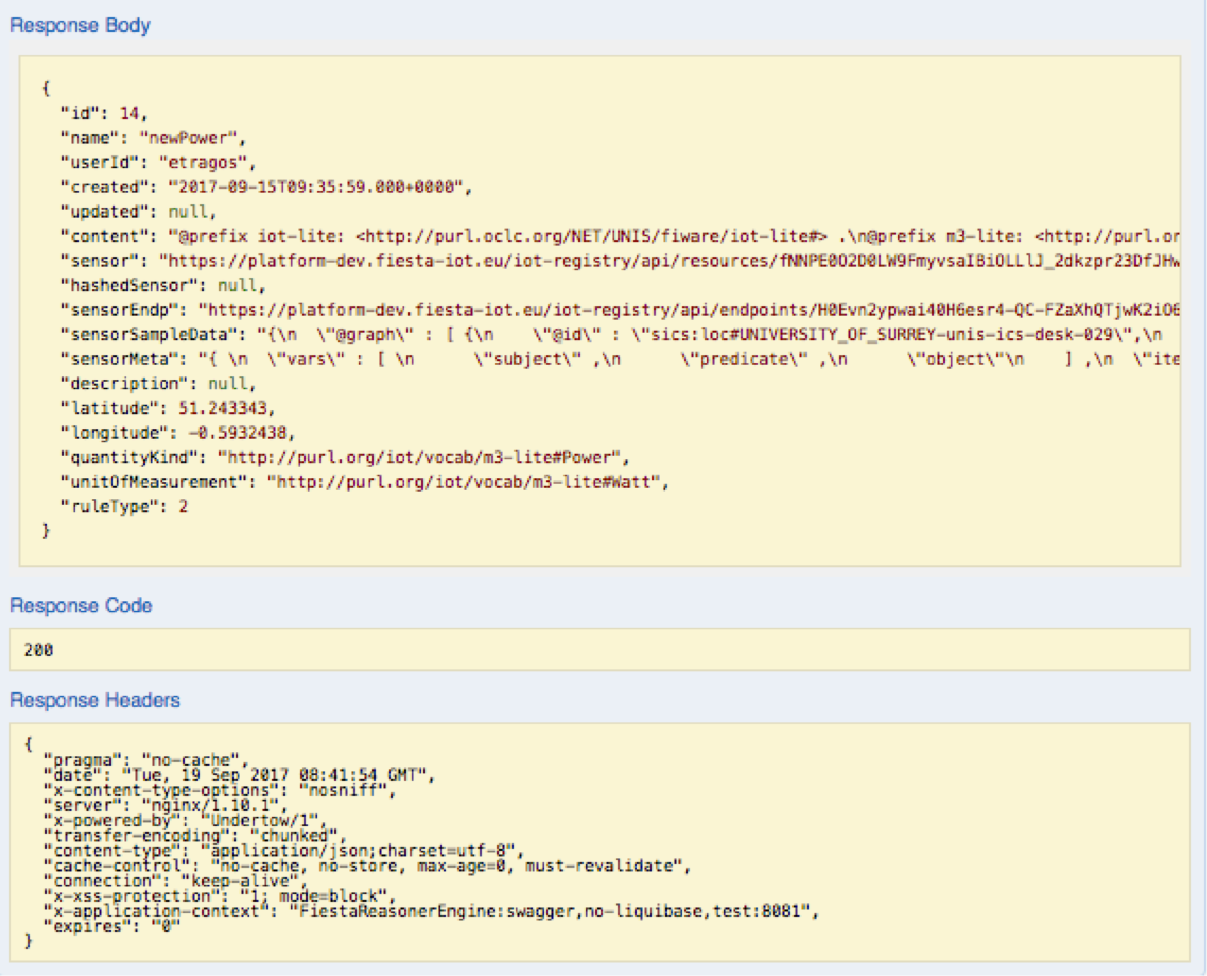
Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found

Sample Call	`curl -X GET --header 'Accept: application/json' 'https://platform.fiesta-iot.eu/reasoner-engine/api/reasonings/3'`
	
	`OR`
	
	`https://platform.fiesta-iot.eu/reasoner-engine/api/reasonings/3`

Notes	-

Table 10: Get rule by ID API

Figure 14 shows the results of the execution of the query for getting rule by its id. As it can be seen, this time the result is a single json string with the parameters of the rule.



Response Body

```
{
  "id": 14,
  "name": "newPower",
  "userId": "etragos",
  "created": "2017-09-15T09:35:59.000+0000",
  "updated": null,
  "content": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.or
  "sensor": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/fNNPE0O2D0LW9FmyvsaIBiOLLlJ_2dkzpr23DfJHw
  "hashedSensor": null,
  "sensorEndp": "https://platform-dev.fiesta-iot.eu/iot-registry/api/endpoints/H0Evn2ypwai40H6esr4-QC-FZaXhQTjwK2iO6
  "sensorSampleData": "{\n  \"@graph\" : [ {\n    \"@id\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-029\",\n
  "sensorMeta": "{ \n  \"vars\" : [ \n      \"subject\" ,\n      \"predicate\" ,\n      \"object\"\n    ] ,\n  \"ite
  "description": null,
  "latitude": 51.243343,
  "longitude": -0.5932438,
  "quantityKind": "http://purl.org/iot/vocab/m3-lite#Power",
  "unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt",
  "ruleType": 2
}
```

Response Code

```
200
```

Response Headers

```
{
  "pragma": "no-cache",
  "date": "Tue, 19 Sep 2017 08:41:54 GMT",
  "x-content-type-options": "nosniff",
  "server": "nginx/1.10.1",
  "x-powered-by": "Undertow/1",
  "transfer-encoding": "chunked",
  "content-type": "application/json;charset=utf-8",
  "cache-control": "no-cache, no-store, max-age=0, must-revalidate",
  "connection": "keep-alive",
  "x-xss-protection": "1; mode=block",
  "x-application-context": "FiestaReasonerEngine:swagger,no-liquibase,test:8081",
  "expires": "0"
}
```

Figure 14: Get rule by ID sample response

3.3.1.3 Create rule

When an experimenter creates a rule, basically he creates a template for the rule, defining the name, description, the quantity kind that the rule should check and an example of sensor from which the rule will check its measurements.

As discussed in Section 3.2, there are two ways for experimenters to create a reasoning rule, as semantic experts and as non-semantic experts. This was considered as mandatory in FIESTA-IoT in order to simplify the process of creating rules even for users that do not know about RDF and SPARQL queries. Thus, in the reasoning API we provide two services, the "*createReasoning*" for the semantic experts and the "*createReasoningWithnonExpert*" for the non-semantic experts.

In Table 11 the description of the service for creating a rule is provided. The service is similar for the semantic and the non-semantic experts, since the latter is mainly used for the Reasoning Tool to be described in D4.8. The experimenters have to define some parameters for:

- *Content*: this is the main text describing the rule in a SPARQL query format.
- *Description*: this is the description of the rule.
- *Sensor*: this is an example URI of a sensor to be used for the rule.
- *Latitude*, *Longtitude*: example location details for the sensor.
- *quantityKind*: the modality of the sensor, i.e. temperature, humidity, power, etc.
- *unitOfMeasurement*: the measurement unit of the sensor, i.e. RH, *degreesCelsius*, Watt, etc.
- *reasoning*: the json string describing the rule to be created.

An example of the Content parameter for defining a rule for a power sensor “if value > 0.1 then notify_high” is also shown in the table.

Title	Create new Rule (createReasoning)
URL	/api/reasonings
Method	`POST`

URL Params	**Required:**
	`reasoning=[string]`
	`This is the json string that provides the description of the rule to be created. The json string should be in the following form:`
	`{`
	`"content": "string",`
	`"description": "string",`
	`"id": 0,`
	`"latitude": 0,`
	`"longitude": 0,`
	`"name": "string",`
	`"quantityKind": "string",`
	`"sensor": "string",`
	`"unitOfMeasurement": "string"`
	`}`
	`And content should be in the form of:`
	
	`@prefix iot-lite:`
	`<http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .`
	`@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .`
	`@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .`
	`@prefix geo:  <http://www.w3.org/2003/01/geo/wgs84_pos#> .`
	`@prefix xsd:    <http://www.w3.org/2001/XMLSchema#> .`
	`@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .`
	`@prefix dul:`
	`<http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .`
	`@prefix time: <http://www.w3.org/2006/time#> .`
	`@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .`
	`@prefix reasoning: <https://fiesta-iot.eu/reasoning#>`
	`.(?observation rdf:type ssn:Observation),`
	`(?observation ssn:observedProperty ?observedProperty),`
	`(?observedProperty rdf:type m3-lite:Power),`
	`(?observation ssn:observationResult ?sensorOutput),`
	`(?sensorOutput ssn:hasValue ?obsValue),`
	`(?obsValue dul:hasDataValue ?dataValue),`
	`(?obsValue iot-lite:hasUnit ?unit),`
	`(?unit rdf:type m3-lite:Watt),`
	`greaterThan(?dataValue, “0.1”^^xsd:double) ->`
	`(?observation reasoning:announce`

	“notify_high”^^xsd:string).
Success Response	**Example:** **Code:** 201 Created
Error Response	**Example:** Code: 400 Bad request Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found Code: 500 Internal server error!

Sample Call	`curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \`
	`"content": "%40prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> . \`
	`%40prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> . \`
	`%40prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> . \`
	`%40prefix geo:  <http://www.w3.org/2003/01/geo/wgs84_pos#> . \`
	`%40prefix xsd:    <http://www.w3.org/2001/XMLSchema#> . \`
	`%40prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . \`
	`%40prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> . \`
	`%40prefix time: <http://www.w3.org/2006/time#> . \`
	`%40prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> . \`
	`%40prefix reasoning: <https://fiesta-iot.eu/reasoning#> .(?observation rdf:type ssn:Observation), \`
	`(?observation ssn:observedProperty ?observedProperty), \`
	`(?observedProperty rdf:type m3-lite:Power), \`
	`(?observation ssn:observationResult ?sensorOutput), \`
	`(?sensorOutput ssn:hasValue ?obsValue), \`
	`(?obsValue dul:hasDataValue ?dataValue), \`
	`(?obsValue iot-lite:hasUnit ?unit), \`
	`(?unit rdf:type m3-lite:Watt), \`
	`greaterThan(?dataValue, “0.1”^^xsd:double) -> (?observation reasoning:announce “notify_high”^^xsd:string).  \`
	`", \`
	`"description": "string", \`
	`"id": 0, \`
	`"latitude": 0, \`
	`"longitude": 0, \`
	`"name": "string", \`
	`"quantityKind": "string", \`
	`"sensor": "string", \`
	`"unitOfMeasurement": "string" \`
	`}' 'https://platform.fiesta-iot.eu/reasoner-engine/api/reasonings'`

Notes	-

Table 11: Rule creation for Semantic Expert API

3.3.1.4 Update Rule API

Experimenters may also need to change some parameters in the rules they have created at some point. For this, the Reasoning Engine provides a service for updating the rules by using the “*updateReasoning*” service using a PUT command and changing the content of the rule, as seen in Table 12.

Title	Update a Rule (updateReasoning)
URL	/api/reasonings
Method	`PUT`

URL Params	**Required:**
	reasoning=[string]
	This is the json string that provides the updated description of the rule to be created. The json string should be in the following form:
	{
	"content": "string",
	"description": "string",
	"id": 0,
	"latitude": 0,
	"longitude": 0,
	"name": "string",
	"quantityKind": "string",
	"sensor": "string",
	"unitOfMeasurement": "string"
	}
	And content should be in the form of:
	@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .
	@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .
	@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .
	@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
	@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
	@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .
	@prefix time: <http://www.w3.org/2006/time#> .
	@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
	@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .(?observation rdf:type ssn:Observation),
	(?observation ssn:observedProperty ?observedProperty),
	(?observedProperty rdf:type m3-lite:Power),
	(?observation ssn:observationResult ?sensorOutput),
	(?sensorOutput ssn:hasValue ?obsValue),
	(?obsValue dul:hasDataValue ?dataValue),
	(?obsValue iot-lite:hasUnit ?unit),
	(?unit rdf:type m3-lite:Watt),
	greaterThan(?dataValue, “0.1”^^xsd:double) -> (?observation reasoning:announce “notify_high”^^xsd:string).

Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 400 Bad request Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found Code: 500 Internal server error!

Sample Call	`curl -X POST --header 'Content-Type: application/json' --header`
	`'Accept: application/json' -d '{ \`
	`   "content": "%40prefix iot-lite:`
	`<http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> . \`
	` %40prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> . \`
	` %40prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> . \`
	` %40prefix geo:  <http://www.w3.org/2003/01/geo/wgs84_pos#> . \`
	` %40prefix xsd:    <http://www.w3.org/2001/XMLSchema#> . \`
	` %40prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . \`
	` %40prefix dul:`
	`<http://www.loa.istc.cnr.it/ontologies/DUL.owl#> . \`
	` %40prefix time: <http://www.w3.org/2006/time#> . \`
	` %40prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .`
	`\`
	` %40prefix reasoning: <https://fiesta-iot.eu/reasoning#>`
	`.(?observation rdf:type ssn:Observation), \`
	` (?observation ssn:observedProperty ?observedProperty), \`
	` (?observedProperty rdf:type m3-lite:Power), \`
	` (?observation ssn:observationResult ?sensorOutput), \`
	` (?sensorOutput ssn:hasValue ?obsValue), \`
	` (?obsValue dul:hasDataValue ?dataValue), \`
	` (?obsValue iot-lite:hasUnit ?unit), \`
	` (?unit rdf:type m3-lite:Watt), \`
	` greaterThan(?dataValue, “0.1”^^xsd:double) -> (?observation`
	`reasoning:announce “notify_high”^^xsd:string).  \`
	` ", \`
	`   "description": "string", \`
	`   "id": 0, \`
	`   "latitude": 0, \`
	`   "longitude": 0, \`
	`   "name": "string", \`
	`   "quantityKind": "string", \`
	`   "sensor": "string", \`
	`   "unitOfMeasurement": "string" \`
	` }' 'https://platform.fiesta-iot.eu/reasoner-`
	`engine/api/reasonings'`

Notes	-

Table 12: Update rule API

3.3.1.5 Rule validation

When experimenters want to create or update some rules, to ensure that they are in the correct format before they are executed, they must be validated, using the “*validateRule*” service as seen in Table 13. This service takes two parameters for the **rule id** and the **sensor id** and then provides the validation response that can be true or false and a message string describing the result of the validation (i.e. what error occurred).

Title	Validate a Rule (validateRule)
URL	/api/rule/validate
Method	`POST`

URL Params	**Required:**
	`validateRequest=[string]`
	`This is the json string that provides the information for the rule to be validated. It should be in the following form:`
	`{`
	`"rule": "string",`
	`"sensorId": "string"`
	`}`
	`rule: the id of the rule to be validated`
	`sensorId: the id of the sensor for which the rule will be applied.`

Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 400 Bad request Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found Code: 500 Internal server error!

Sample Call	`curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \`
	`"rule": "10", \`
	`"sensorId": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/x1AlxibeGRXJDPUbYHcB9Wol22kDiTEwzjR1t445JQfIPuv0YJivjsrb14DRkpj7mVw5_Ax4eVEsDr1PMu0AJxoj0uQFEZhf743kKon7QVRc-DmsGDO9E6fxBK6Oc9pd" \`
	`}' 'https://platform.fiesta-iot.eu/reasoner-engine/api/rule/validate'`

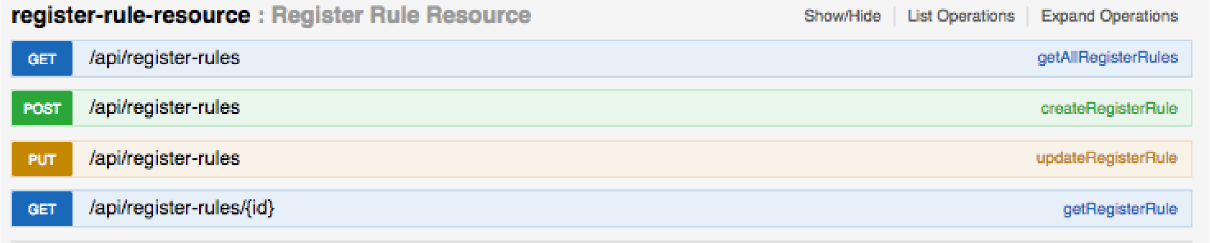
Notes	-

Table 13: Rule validation API

3.3.2 Register Rule Resource

The Register Rule Resource API provides several services that can be used by experimenters for registering a rule that they have created previously so that they can then execute it on their experiment. By registering a rule, the experimenter defines specifically for which sensor (or set of sensors) the rule will execute. The Register Rule

Resource API provides services for retrieving the list of registered rules, for registering a new rule, for updating a rule registration and for getting a specific registered rule. In Figure 15 the list of services currently supported in the rule-resource API are presented.



register-rule-resource : Register Rule Resource

GET	/api/register-rules	getAllRegisterRules
POST	/api/register-rules	createRegisterRule
PUT	/api/register-rules	updateRegisterRule
GET	/api/register-rules/{id}	getRegisterRule

Figure 15: List all api in register rule resource

3.3.2.1 Get all registered rules API

The "*getAllRegisterRules*" service (presented in Table 14) provides the function for experimenters to retrieve the full list of currently registered rules in a similar way like the "*getAllRules*" service, using parameters such as page, size, and sort. However, here, by accessing this service, the experimenter will only retrieve the list of rules he has registered and not the rules registered by other experimenters. Since the list of registered rules can be quite long in real deployments, the experimenter can select the registered rules according to the following parameters below to limit the number of rules, same as in the "*getAllRules*" service described in Section 3.3.1.1. The table also shows an example of the request to get the list of registered rules starting from page 5 with a page size of "10" and sorted descending according to their id.

Title	Get all registered Rules (getAllRegisterRules)
URL	/api/register-rule
Method	`GET`

URL Params	**Optional:**
	`page=[integer]`
	sets the starting page number for the registered rules to be returned. For example, setting the page to "5", the reasoning engine will return the registered rules from number size*5 until size*6-1, depending on the size parameter below.
	size=`[integer]`
	sets the number of the registered rules per page to be returned.
	`sort=[string]`
	sets the sorting criteria for the returned resultset, i.e. ascending or descending.
	example:
	page=3
	size=30
	sort=asc

Success Response	**Example:** **Code:** 200 OK

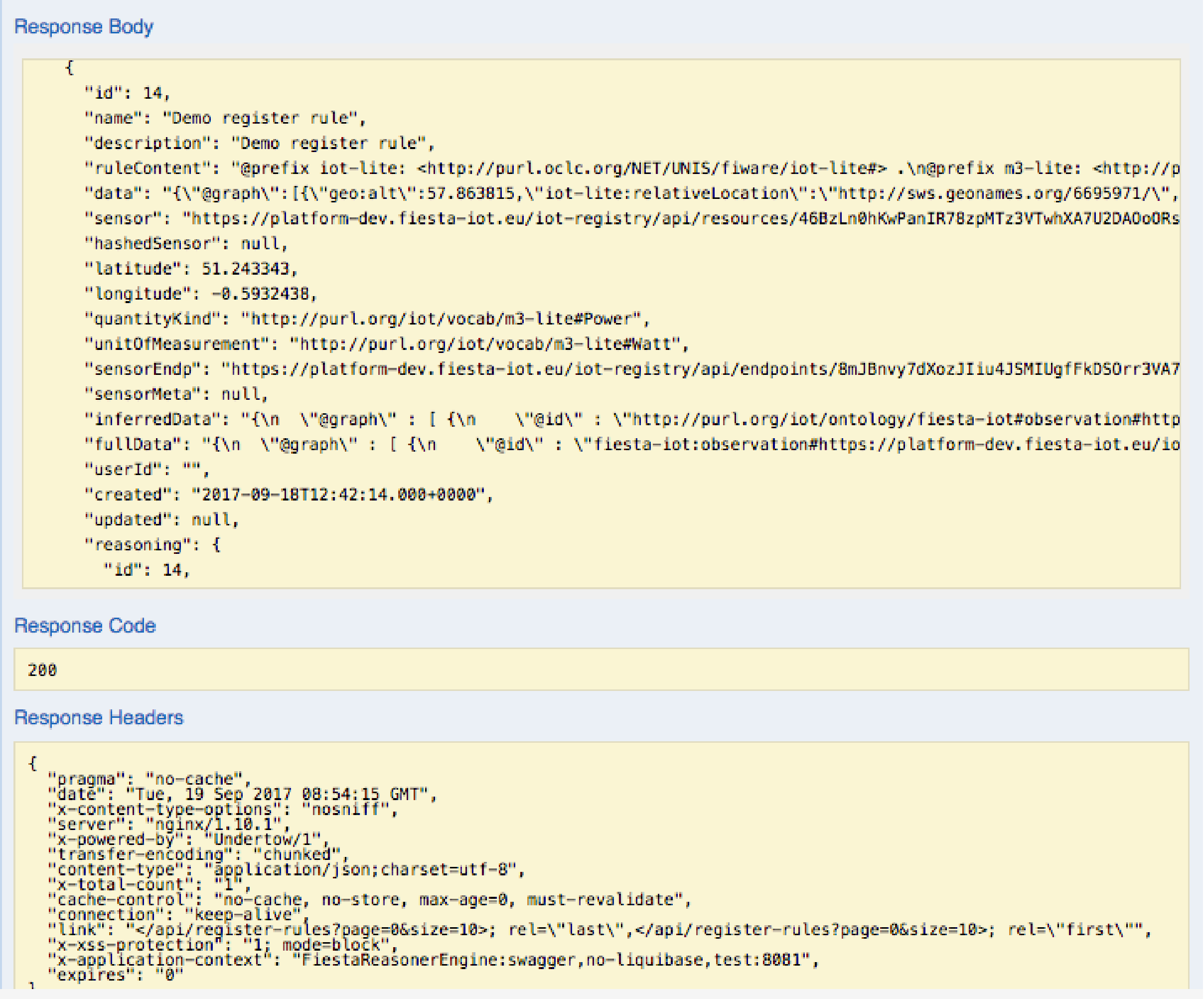
Error Response	**Example:** Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found

Sample Call	`curl -X GET --header 'Accept: application/json' 'https://platform.fiesta-iot.eu/reasoner-engine/api/register-rules?page=5&size=10&sort=desc'`
	OR
	`https://platform.fiesta-iot.eu/reasoner-engine/api/register-rules?page=5&size=10&sort=desc`

Notes	-

Table 14: Get all register rule API

Figure 16 shows the response of the previous request, which can be either a json array containing the descriptions of the registered rules or a single json string.



```
    {
      "id": 14,
      "name": "Demo register rule",
      "description": "Demo register rule",
      "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://p
      "data": "{\"@graph\":[{\"geo:alt\":57.863815,\"iot-lite:relativeLocation\":\"http://sws.geonames.org/6695971/\",
      "sensor": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/46BzLn0hKwPanIR78zpMTz3VTwhXA7U2DAOoORs
      "hashedSensor": null,
      "latitude": 51.243343,
      "longitude": -0.5932438,
      "quantityKind": "http://purl.org/iot/vocab/m3-lite#Power",
      "unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt",
      "sensorEndp": "https://platform-dev.fiesta-iot.eu/iot-registry/api/endpoints/8mJBnvy7dXozJIiu4JSMIUgfFkDSOrr3VA7
      "sensorMeta": null,
      "inferredData": "{\n  \"@graph\" : [ {\n    \"@id\" : \"http://purl.org/iot/ontology/fiesta-iot#observation#http
      "fullData": "{\n  \"@graph\" : [ {\n    \"@id\" : \"fiesta-iot:observation#https://platform-dev.fiesta-iot.eu/io
      "userId": "",
      "created": "2017-09-18T12:42:14.000+0000",
      "updated": null,
      "reasoning": {
        "id": 14,
```

```
200
```

```
{
  "pragma": "no-cache",
  "date": "Tue, 19 Sep 2017 08:54:15 GMT",
  "x-content-type-options": "nosniff",
  "server": "nginx/1.10.1",
  "x-powered-by": "Undertow/1",
  "transfer-encoding": "chunked",
  "content-type": "application/json;charset=utf-8",
  "x-total-count": "1",
  "cache-control": "no-cache, no-store, max-age=0, must-revalidate",
  "connection": "keep-alive",
  "link": "</api/register-rules?page=0&size=10>; rel=\"last\",</api/register-rules?page=0&size=10>; rel=\"first\"",
  "x-xss-protection": "1; mode=block",
  "x-application-context": "FiestaReasonerEngine:swagger,no-liquibase,test:8081",
  "expires": "0"
```

Figure 16: Get All register rule sample response

3.3.2.2 Get register rule by id API

Experimenters can also query the reasoning engine to get a specific registered rule by providing the rule registration identification number. This can be done by using the "*getRegisterRule*" service of the register-rule-resource API, using as a parameter only the registration ID, as seen in Table 15.

Title	Get all registered Rules (getAllRegisterRules)
URL	/api/register-rule
Method	**`GET`**

URL Params	**Optional:**
	`page=[integer]`
	sets the starting page number for the registered rules to be returned. For example, setting the page to "5", the reasoning engine will return the registered rules from number size*5 until size*6-1, depending on the size parameter below.
	size=`[integer]`
	sets the number of the registered rules per page to be returned.
	`sort=[string]`
	sets the sorting criteria for the returned resultset, i.e. ascending or descending.
	example:
	page=3
	size=30
	sort=asc

Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found

Sample Call	`curl -X GET --header 'Accept: application/json' 'https://platform.fiesta-iot.eu/reasoner-engine/api/register-rules?page=5&size=10&sort=desc'`
	`OR`
	`https://platform.fiesta-iot.eu/reasoner-engine/api/register-rules?page=5&size=10&sort=desc`

Notes	-

Table 15: Get rule by ID

3.3.2.3 Register rule API

For registering a rule, the experimenters should access the "*createRegisterRule*" service (see Table 16), where they must define the json string of the rule registration. In the json string, the experimenters must define the rule that they want to register and provide details regarding the sensor they are checking with the rule.

Title	Register new Rule (createReasoning)
URL	/api/register-rules
Method	**`POST`**

URL Params	**Required:**
	`registerRule=[string]`
	`This is the json string that provides the description of the rule to be registered. The json string should be in the following form:`
	`{`
	`"description": "string",`
	`"latitude": 0,`
	`"longitude": 0,`
	`"name": "string",`
	`"quantityKind": "string",`
	`"ruleId": 0,`
	`"sensor": "string",`
	`"unitOfMeasurement": "string"`
	`}`
	With the following sub-parameters
	- `"ruleID": string`
	`the id of the rule that was previously created and needs to be registered.`
	- `"Name": string`
	`the name of the rule to be registered.`
	- `"Description": string`
	`this is the description of the rule registration.`
	- `"Sensor": string`
	`this is an example URI of a sensor to be used for the rule.`
	- `"Latitude", "Longtitude": example location details for the sensor.`
	- `"quantityKind": string`
	`the modality of the sensor, i.e. temperature, humidity, power, etc.`
	- `"unitOfMeasurement": string`
	`the measurement unit of the sensor, i.e. RH, degreesCelsius, Watt, etc.`

Success Response	**Example:** **Code:** 201 Created
Error Response	**Example:** Code: 400 Bad request Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found Code: 500 Internal server error!

Sample Call	curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \\ "description": "string", \\ "latitude": 0, \\ "longitude": 0, \\ "name": "string", \\ "quantityKind": "string", \\ "ruleId": 0, \\ "sensor": "string", \\ "unitOfMeasurement": "string" \\ }' 'https://platform.fiesta-iot.eu/reasoner-engine/api/register-rules'
Notes	-

Table 16: Register rule API

3.3.2.4 Update register rule API

Similar as with the rule creation, the Reasoning Engine includes also a service (see Table 17) for updating the registered rules, via accessing the “updateRegisterRule” service and defining the new content of the rule registration, i.e. the new sensor.

Title	Update a registered Rule (updateRegisterRule)
URL	/api/rgister-rules
Method	**PUT**

URL Params	**Required:**
	registerRule=[string]
	This is the json string that provides the description of the updated registered rule. The json string should be in the following form:
	{
	"description": "string",
	"latitude": 0,
	"longitude": 0,
	"name": "string",
	"quantityKind": "string",
	"ruleId": 0,
	"sensor": "string",
	"unitOfMeasurement": "string"
	}
	With the following sub-parameters
	- "ruleID": string
	the id of the rule that was previously created and needs to be registered.
	- "Name": string
	the name of the rule to be registered.
	- "Description": string
	this is the description of the rule registration.
	- "Sensor": string
	this is an example URI of a sensor to be used for the rule.
	- "Latitude", "Longtitude": example location details for the sensor.
	- "quantityKind": string
	the modality of the sensor, i.e. temperature, humidity, power, etc.

	- "unitOfMeasurement": string the measurement unit of the sensor, i.e. RH, degreesCelsius, Watt, etc.
Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 400 Bad request Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found Code: 500 Internal server error!

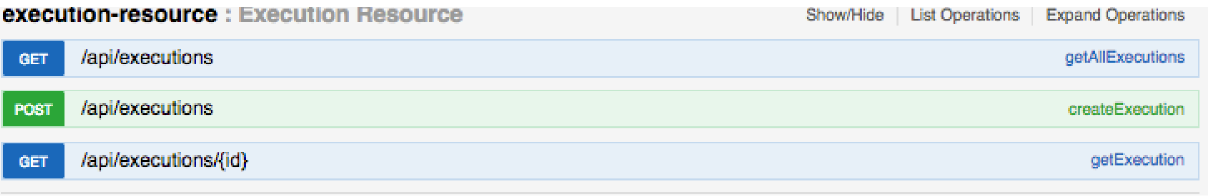
Sample Call	`curl -X PUT --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \`
	`"description": "string", \`
	`"id": 0, \`
	`"latitude": 0, \`
	`"longitude": 0, \`
	`"name": "string", \`
	`"quantityKind": "string", \`
	`"ruleId": 0, \`
	`"sensor": "string", \`
	`"unitOfMeasurement": "string" \`
	`}' 'https://platform.fiesta-iot.eu/reasoner-engine/api/register-rules'`

Notes	-

Table 17: Update register rule API

3.3.3 Execution Resource

After creating and registering a rule, the next step is to execute the rule to see the reasoning results. For this, the Reasoning Engine provides the Execution Resource API with services for retrieving previous executions, creating a new execution and retrieving a specific previous execution, as seen in Figure 17



execution-resource : Execution Resource

GET /api/executions getAllExecutions

POST /api/executions createExecution

GET /api/executions/{id} getExecution

Figure 17: Rule execution API

3.3.3.1 Get All executions

The "*getAllExecutions*" service (presented in Table 18) provides the function for experimenters to retrieve the full list of previous executions of their registered rules in a similar way like the "*getAllRules*" service, using parameters such as page, size, and sort. However, here, by accessing this service, the experimenter will only retrieve the list of their own previous executions and not the executions of other experimenters. Since the list of previous executions can be quite long in real deployments, the experimenter can select the previous executions to retrieve according to the following parameters below in order to limit the number of results, same as in the "*getAllRules*" service described in Section 3.3.1.1.

Title	Get all previous executions of Rules (getAllExecutions)
URL	. /api/executions
Method	**`GET`**

URL Params	**Optional:**
	`page=[integer]`
	sets the starting page number for the executions to be returned. For example, setting the page to "5", the reasoning engine will return the previous executions from number size*5 until size*6-1, depending on the size parameter below.
	size`=[integer]`
	sets the number of the executions per page to be returned.
	`sort=[string]`
	sets the sorting criteria for the returned resultset, i.e. ascending or descending.
	example:
	page=3
	size=30
	sort=asc

Success Respons e	**Example:** **Code:** 200 OK
Error Respons e	**Example:** Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found

Sample Call	`curl -X GET --header 'Accept: application/json' 'https://platform.fiesta-iot.eu/reasoner-engine/api/executions?page=5&size=10&sort=asc'`
	
	`OR`
	
	`https://platform.fiesta-iot.eu/reasoner-engine/api/executions?page=5&size=10&sort=asc`

Notes	-

Table 18: Get All executions API

3.3.3.2 Get Specific Execution

Experimenters can also query the reasoning engine to get a specific previous execution by providing the execution identification number. This can be done by using the "*getExecution*" service of the execution-resource API, using as a parameter only the execution ID, as seen in Table 19.

Title	Get a specific Execution (getExecution)
URL	/api/executions/{id}
Method	`GET`

URL Params	**Required:**
	`id=[integer]`
	`     The id of the specific execution to be returned`
	
	example:
	id=13

Success Response	**Example:** **Code:** 200 OK
Error Response	**Example:** Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found

Sample Call	`curl -X GET --header 'Accept: application/json' 'https://platform.fiesta-iot.eu/reasoner-engine/api/executions/13'`
	
	OR
	
	`https://platform.fiesta-iot.eu/reasoner-engine/api/executions/13`

Notes	-

Table 19: Get detail execution API

3.3.3.3 Execute Rule

The last action an experimenter must perform to execute a rule is to access the "*createExecution*" service and create a new execution, by providing a textual description in a json format of the execution, setting the required parameters (see Table 20).

If the experimenter wants to get the reasoning results only on the latest value, then the started/ended should be the same value and should be set to the current date/time. Otherwise, setting the started and ended at different values, the rule will be executed to the list of measurements within these times.

The response of an example execution can be seen in Figure 18, where the response body includes the results of the execution, containing also the inference results.

Title	Create a new Execution (createExecution)
URL	/api/executions
Method	`POST`

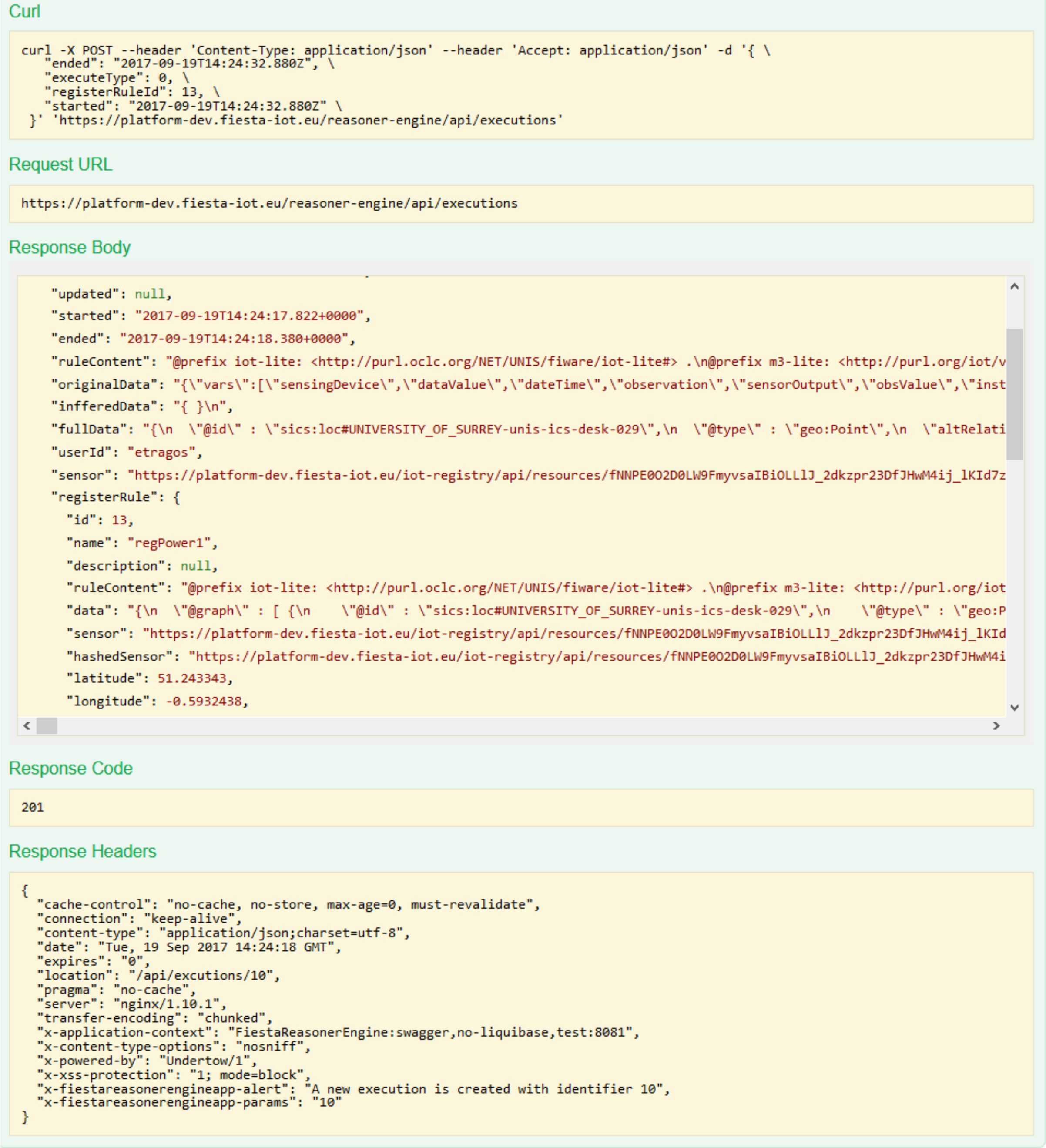
URL Params	**Required:**
	`execution=[string]`
	`This is the json string that provides the description of the new execution to be created. The json string should be in the following form:`
	`{`
	`  "started": "2017-10-23T14:04:26.542Z"`
	`  "ended": "2017-10-23T14:04:26.542Z",`
	`  "executeType": 0,`
	`  "registerRuleId": 0,`
	`}`
	With the following sub-parameters
	- `"started": datestring`
	`the starting date of the dataset that will be checked by this rule.`
	- `"ended": datestring`
	`the ending date of the dataset that will be checked by this rule.`
	- `"executeType": integer`
	`can be “1” if the experimenter wants to get results only on the latest value or “2” if the experimenter wants the results on a time series.`
	- `"registerRuleId": integer`
	`The id of the registered rule that will be executed.`

Success Response	**Example:** **Code:** 201 Created
Error Response	**Example:** Code: 400 Bad request Code: 401 Unauthorized Code: 403 Forbidden Code: 404 Not Found Code: 500 Internal server error!

Sample Call	`curl -X POST --header 'Content-Type: application/json' --header 'Accept: */*' -d '{ \`
	`  "ended": "2017-10-23T14:04:26.542Z", \`
	`  "executeType": 1, \`
	`  "registerRuleId": 15, \`
	`  "started": "2017-10-23T14:04:26.542Z" \`
	`}' 'https://platform.fiesta-iot.eu/reasoner-engine/api/executions'`

Notes	-

Table 20: Execute rule API



```
curl -X POST --header 'Content-Type: application/json' --header 'Accept: application/json' -d '{ \
   "ended": "2017-09-19T14:24:32.880Z", \
   "executeType": 0, \
   "registerRuleId": 13, \
   "started": "2017-09-19T14:24:32.880Z" \
 }' 'https://platform-dev.fiesta-iot.eu/reasoner-engine/api/executions'
```

```
https://platform-dev.fiesta-iot.eu/reasoner-engine/api/executions
```

```
    "updated": null,
    "started": "2017-09-19T14:24:17.822+0000",
    "ended": "2017-09-19T14:24:18.380+0000",
    "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/v
    "originalData": "{\"vars\":[\"sensingDevice\",\"dataValue\",\"dateTime\",\"observation\",\"sensorOutput\",\"obsValue\",\"inst
    "inferedData": "{ }\n",
    "fullData": "{\n  \"@id\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-029\",\n  \"@type\" : \"geo:Point\",\n  \"altRelati
    "userId": "etragos",
    "sensor": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/fNNPE0O2D0LW9FmyvsaIBiOLLlJ_2dkzpr23DfJHwM4ij_lKId7z
    "registerRule": {
      "id": 13,
      "name": "regPower1",
      "description": null,
      "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot
      "data": "{\n  \"@graph\" : [ {\n    \"@id\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-029\",\n    \"@type\" : \"geo:P
      "sensor": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/fNNPE0O2D0LW9FmyvsaIBiOLLlJ_2dkzpr23DfJHwM4ij_lKId
      "hashedSensor": "https://platform-dev.fiesta-iot.eu/iot-registry/api/resources/fNNPE0O2D0LW9FmyvsaIBiOLLlJ_2dkzpr23DfJHwM4i
      "latitude": 51.243343,
      "longitude": -0.5932438,
```

```
201
```

```
{
  "cache-control": "no-cache, no-store, max-age=0, must-revalidate",
  "connection": "keep-alive",
  "content-type": "application/json;charset=utf-8",
  "date": "Tue, 19 Sep 2017 14:24:18 GMT",
  "expires": "0",
  "location": "/api/excutions/10",
  "pragma": "no-cache",
  "server": "nginx/1.10.1",
  "transfer-encoding": "chunked",
  "x-application-context": "FiestaReasonerEngine:swagger,no-liquibase,test:8081",
  "x-content-type-options": "nosniff",
  "x-powered-by": "Undertow/1",
  "x-xss-protection": "1; mode=block",
  "x-fiestareasonerengineapp-alert": "A new execution is created with identifier 10",
  "x-fiestareasonerengineapp-params": "10"
}
```

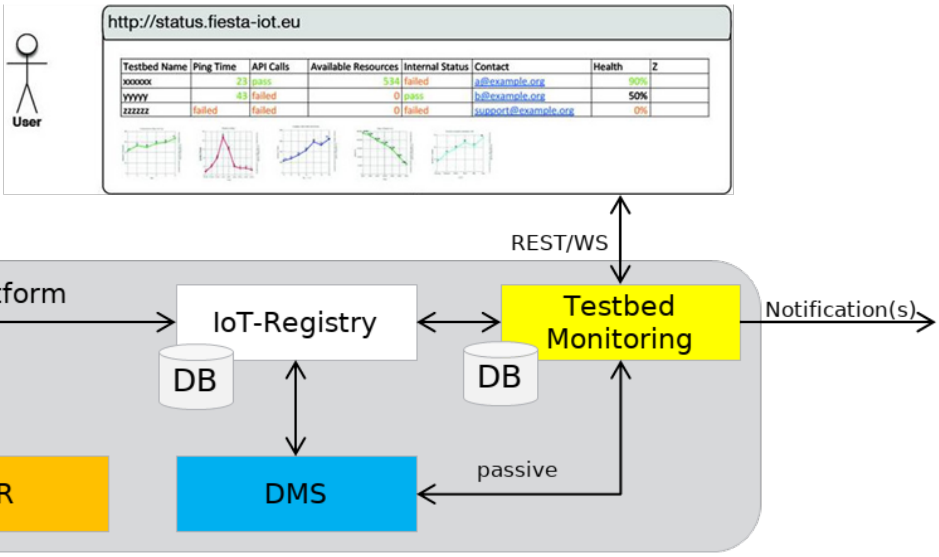
Figure 18: Example Execute rule response

4 FIESTA-IOT MONITORING

4.1 Testbed Monitoring

To have a way to quickly overview the testbeds, their overall situation and the data that is provided by them, a tool for monitoring the testbed data was implemented.

The Testbed Monitoring Tool shall continuously collect information from the testbeds to monitor them or, being more precise, the data of the testbed that is stored in the FIESTA-IoT platform. For this it collects data from different components but mainly from the IoT-Registry. As the queries can be time consuming it is crucial to do these in proper time intervals. The gathered data will be analysed in different ways. For the basic overall status, the latest observations of every sensor are calculated and visualized to get an overall overview. Other data are the trend of every sensor to visualize them on the one hand but also to find any malfunctions of them.



http://status.fiesta-iot.eu

Testbed Name	Ping Time	API Calls	Available Resources	Internal Status	Contact	Health	Z
xxxxxx	23	pass	534	failed	a@example.org	90%	
yyyyy	43	failed	0	pass	b@example.org	50%	
zzzzzz	failed	failed	0	failed	support@example.org	0%	

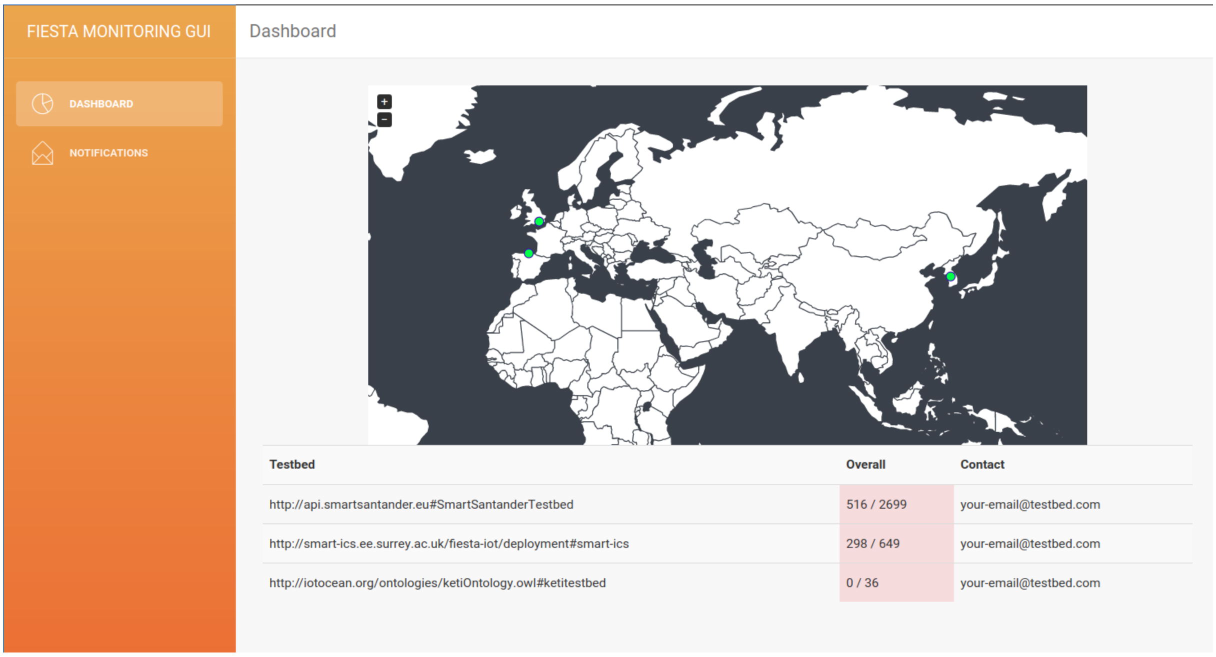
Figure 19: The Testbed Monitoring Tool in the FIESTA-IoT platform

For this, the collected data will be stored in a database. But the data will be stored without semantic information and only for a smaller time frame that is proper to calculate trends. The stored data is used to provide the information explained before. The information can be obtained mainly via a GUI that will visualize the data to give a quick impression of the situation. Additionally, some information can be retrieved via REST interface to get the collected or calculated data raw. Another possibility to get informed is via a notification system. This allows the user to trigger notifications when the state of one or more testbeds has changed, either from bad to good for experimenters or the opposite for testbed providers.

The motivation to establish such a system is to give the users of the FIESTA-IoT platform an overview of the health state of testbeds before starting an experiment. An experiment can last a very long time. So, it would be worth to get an experience of the data that will be consumed during this time. Also, the state of the gathered data during the experiment can be checked after the experiment, if the data is relevant and there were no gaps in the data stream. But also, the testbed owners can check that their data is transferred to the platform successfully. Often testbed owners have their own systems to check the health of their equipment, but here the data as they are provided to other users of this data is checked.

4.1.1 Provided Features

The Testbed Monitoring is integrated into the portal and can be found under the tools section. It is integrated using an *iframe* and started as separate instance. It uses the role management of the security solution to determine the role of the user to show different views dependent of the role of the user, either registered user, experimenter, testbed owner or FIESTA administrator.



Testbed	Overall	Contact
http://api.smartsantander.eu#SmartSantanderTestbed	516 / 2699	your-email@testbed.com
http://smart-ics.ee.surrey.ac.uk/fiesta-iot/deployment#smart-ics	298 / 649	your-email@testbed.com
http://iotocean.org/ontologies/ketiOntology.owl#ketitestbed	0 / 36	your-email@testbed.com

Figure 20: The Testbed Monitoring Dashboard

The visual part of the Testbed Monitoring consists of a dashboard which lists all testbeds with some basic information like the number of sensors having an observation in the last 2 hours and the total number of registered sensors. The dashboard can be seen in Figure 20. Additionally, all testbeds are shown in a map and their locations are marked. When clicking on a testbed, a detailed list of all registered sensors is shown. The sensors will be shown with the latest observation value and the time it was made. Also, the type of the sensor is shown. If a sensor is clicked more detailed information will be shown. A graph that shows the trend of the measured quantity over the time and other information like the typical time interval is shown.

The testbed management view is only visible to FIESTA-IoT administrators. Here a new registered testbed can be added for monitoring. This is needed when a testbed was added to the platform and all sensors were registered and are working properly. Also, testbeds can be removed again from monitoring.

The notification system can be used to select one or more testbeds and define a trigger when a notification to a user shall be sent. This can help to postpone an experiment to a situation when all involved testbeds are in a state which is acceptable for experimenting. Also, testbed owners can define some triggers when they want to get notified when there is a problem in their provided data.

The analysis of data is done in the background and uses the gathered data. It will shape the data for visualizing and querying, but also tries to calculate and determine some trends to detect a change in the delivery of testbed data. This will be done in a long-term manner and can detect a frequency drift or changes in the accuracy of the sensor.

4.1.2 Integration into the platform

The Testbed Monitoring is running as a Flask App[2] and so is separated from the components in the *Wildfly* container. In order integrate it into the portal with the help of an *iframe*, an *nginx* proxy was setup to make it available under the same namespace of the portal. The website is using a REST API and websockets to exchange data between the backend and the website in the browser. The *nginx* instance will bypass both communication ways. It also assures that the header information for the security component is still available. By integrating the monitoring into the portal namespace, the access by users is secured and users need to register to use the service.

The security component of the platform is used to get the role of the visiting user. This is needed to adapt the view, as not every role can see everything.

The backend of the Testbed Monitoring queries the IoT-Registry locally on the platform. It will use the local open port and accesses the service directly to query the needed data. As the Testbed Monitoring is placed in the platform it can also query other components like the EEE to get other necessary information or the DMS to passively observe specific data.

4.1.3 Monitoring Tests and Collected Quantities

The Testbed Monitoring will query the IoT-Registry component of the platform mainly for gathering data. It will collect the registered testbeds and the corresponding resources that are the sensors that will provide the observations. The observations will be collected and stored per sensor.

The process of gathering data is done as follows. First the IoT-Registry will be checked for the registered testbeds. All testbeds will be added as not activated. This process will be repeated every day to check for newly added or removed testbeds. The querying of resources is also done daily. It will query the IoT-Registry for all sensors that have a type, a quantity kind, a unit, and a location and are connected to an activated testbed. Newly found resources will be added, removed ones will be deleted. The values of the attributes of the sensor are converted into a leaner format, cutting away some of the semantic notation. The gathering of observations will be done where every x hours the observations of the last x hours will be retrieved. The observations will be converted, prepared and stored per sensor. For every sensor only data of the last month is stored. This data will be used to do the analytics and to calculate quality statements for every sensor.

The stored data will be used for the visualization in the GUI and to feed the API. Also, the analytics tasks use the gathered data and generate additional data for every sensor and testbed that can be consumed by the GUI and the API.

For the GUI, the latest observation per sensor and the latest values for a specific sensor can be directly taken from the prepared dataset, also the location per sensor is already available. Calculated are the number of active sensors per testbed and the combined location of all sensors of a testbed for example.

[2] http://flask.pocoo.org/

For the GUI, the latest observation per sensor and the latest values for a specific sensor can be directly taken from the prepared dataset, also the location per sensor is already available. Calculated are the number of active sensors per testbed and the combined location of all sensors of a testbed for example.

4.1.4 Components

The Components of the Testbed Monitoring can be categorized into three parts, the background tasks, the GUI and the API. The background tasks will query the IoT-Registry for data stored in the platform as well as preparing them for the usage in the GUI. Also, the background tasks will do the analytics. The REST API will provide collected and calculated data from the Monitoring that can be used by other components and will be used by the GUI to add on-demand actions like showing graphs or activating testbeds. The GUI is the main part that will be used by Experimenters and Testbed Owners. It consists of four main parts, the dashboard, the Detailed View, the Testbed Management and the Notification System.

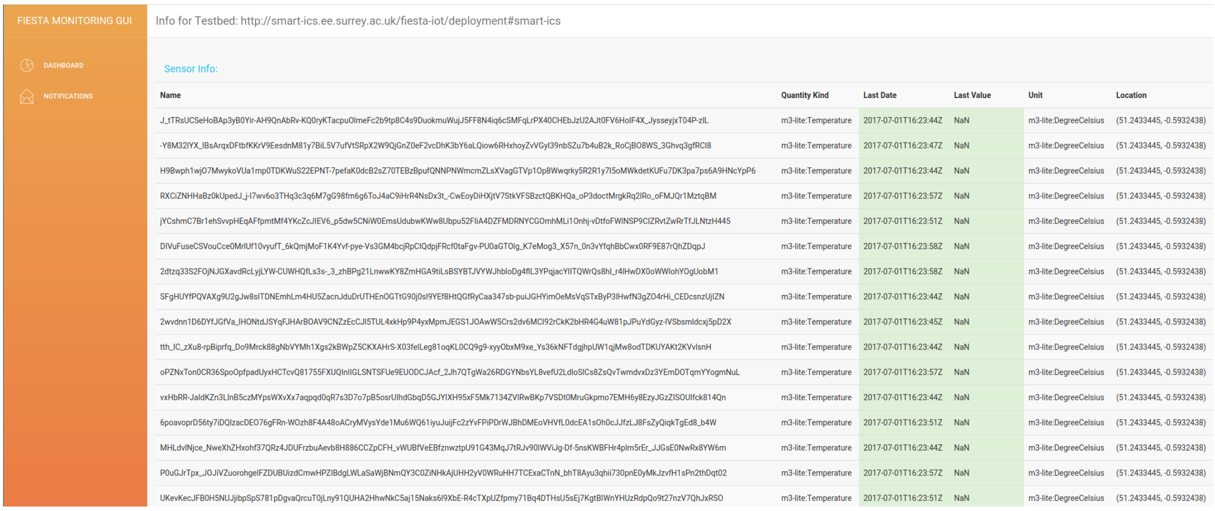
4.1.4.1 Dashboard

The Dashboard is the main entry for the GUI. It can be used by any user and will show the testbeds activated for monitoring in a table and their overall status.

In Figure 20 can be seen the dashboard. It consists of two parts, the table of listed testbeds and a map where the location of the testbeds is marked. If a testbed is clicked, a detailed view of the testbed is opened.

4.1.4.2 Detailed View

The Detailed View shows a list of the registered sensors that belong to the testbed.



Info for Testbed: http://smart-ics.ee.surrey.ac.uk/fiesta-iot/deployment#smart-ics

Sensor Info:

Name	Quantity Kind	Last Date	Last Value	Unit	Location
J_tTRsUCSeHoBAp3yB0Yir-AH9QnAbRv-KQ0ryKTacpuOlmeFc2b9tp8C4s9DuokmuWujJ5FF8N4iq6cSMFqLrPX40CHEbJzU2AJt0FV6HolF4X_JysseyjxT04P-zIL	m3-lite:Temperature	2017-07-01T16:23:44Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
-Y8M32lYX_lBsArqxDFtbfKKrV9EesdnM81y7BiL5V7ufVtSRpX2W9QjGnZ0eF2vcDhK3bY6aLQiow6RHxhoyZvVGyl39nbSZu7b4uB2k_RoCjBO8WS_3Ghvq3gfRCl8	m3-lite:Temperature	2017-07-01T16:23:47Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
H9Bwph1wjO7MwykoVUa1mp0TDKWuS22EPNT-7pefaK0dcB2sZ70TEBzBpufQNNPNWmcmZLsXVagGTVp1Op8Wwqrky5R2R1y7I5oMWkdetKUFu7DK3pa7ps6A9HNcYpP6	m3-lite:Temperature	2017-07-01T16:23:44Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
RXCiZNHHaBz0kUpedJ_j-I7wv6o3THq3c3q6M7gG98fm6g6ToJ4aC9iHrR4NsDx3t_-CwEoyDiHXjtV7StkVFSBzctQBKHQa_oP3doctMrgkRq2lRo_oFMJQr1MztqBM	m3-lite:Temperature	2017-07-01T16:23:57Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
jYCshmC7Br1ehSvvpHEqAFfpmtMf4YKcZcJIEV6_p5dw5CNiW0EmsUdubwKWw8Ubpu52FIiA4DZFMDRNYCGOmhMLi1Onhj-vDtfoFWINSP9ClZRvtZwRrTfJLNtzH445	m3-lite:Temperature	2017-07-01T16:23:51Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
DIVuFuseCSVouCce0MrlUf10vyufT_6kQmjMoF1K4Yvf-pye-Vs3GM4bcjRpCIQdpjFRcf0taFgv-PU0aGTOlg_K7eMog3_X57n_0n3vYfqhBbCwx0RF9E87rQhZDqpJ	m3-lite:Temperature	2017-07-01T16:23:58Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
2dtzq33S2FOjNJGXavdRcLyjLYW-CUWHQfLs3s-_3_zhBPg21LnwwKY8ZmHGA9tiLsBSYBTJVYWJhbloDg4flL3YPqjacYlITQWrQs8hl_r4lHwDX0oWWlohYOgUobM1	m3-lite:Temperature	2017-07-01T16:23:58Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
SFgHUYfPQVAXg9U2gJw8slTDNEmhLm4HU5ZacnJduDrUTHEnOGTtG90j0sl9YEf8HtQGfRyCaa347sb-puiJGHYimOeMsVqSTxByP3lHwfN3gZO4rHi_CEDcsnzUjIZN	m3-lite:Temperature	2017-07-01T16:23:44Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
2wvdnn1D6DYfJGfVa_lHONtdJSYqFJHArBOAV9CNZzEcCJI5TUL4xkHp9P4yxMpmJEGS1JOAwW5Crs2dv6MCl92rCkK2bHR4G4uW81pJPuYdGyz-lVSbsmldcxj5pD2X	m3-lite:Temperature	2017-07-01T16:23:45Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
tth_IC_zXu8-rpBiprfq_Do9Mrck88gNbVYMh1Xgs2kBWpZ5CKXAHrS-X03felLeg81oqKL0CQ9g9-xyyObxM9xe_Ys36kNFTdgjhpUW1qjMw8odTDKUYAKt2KVvlsnH	m3-lite:Temperature	2017-07-01T16:23:44Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
oPZNxTon0CR36SpoOpfpadUyxHCTcvQ81755FXUQInlIGLSNTSFUe9EUODCJAcf_2Jh7QTgWa26RDGYNbsYL8vefU2LdloSlCs8ZsQvTwmdvxDz3YEmDOTqmYYogmNuL	m3-lite:Temperature	2017-07-01T16:23:57Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
vxHbRR-JaldKZn3LInB5czMYpsWXvXx7aqpqd0qR7s3D7o7pB5osrUlhdGbqD5GJYIXH95xF5Mk7134ZVlRwBKp7VSDt0MruGkpmo7EMH6y8EzyJGzZlSOUlfck814Qn	m3-lite:Temperature	2017-07-01T16:23:44Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
6poavoprD56ty7iDQlzacDEO76gFRn-WOzh8F4A48oACryMVysYde1Mu6WQ61iyuJuijFc2zYvFPiPDrWJBhDMEoVHVfL0dcEA1sOh0cJJfzLJ8FsZyQiqkTgEd8_b4W	m3-lite:Temperature	2017-07-01T16:23:51Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
MHLdvlNjce_NweXhZHxohf37QRz4JDUFrzbuAevb8H886CCZpCFH_vWUBfVeEBfznwztpU91G43MqJ7tRJv90lWViJg-Df-5nsKWBFHr4plm5rEr_JJGsE0NwRx8YW6m	m3-lite:Temperature	2017-07-01T16:23:44Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
P0uGJrTpx_JOJiVZuorohgelFZDUBUizdCmwHPZlBdgLWLaSaWjBNmQY3C0ZiNHkAjUHH2yV0WRuHH7TCExaCTnN_bhT8Ayu3qhii730pnE0yMkJzvfH1sPn2thDqt02	m3-lite:Temperature	2017-07-01T16:23:57Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)
UKevKecJFB0H5NUJjibpSpS781pDgvaQrcuT0jLny91QUHA2HhwNkC5aj15Naks6l9XbE-R4cTXpUZfpmy71Bq4DTHsU5sEj7KgtBlWnYHUzRdpQo9t27nzV7QhJxRSO	m3-lite:Temperature	2017-07-01T16:23:51Z	NaN	m3-lite:DegreeCelsius	(51.2433445, -0.5932438)

Figure 21: The Testbed Monitoring Detailed View

All resources of the testbed are listed with their respective observed quantity kind, the latest observation with time, value and unit, and the location of the sensor as seen in

Figure 21. When a sensor is clicked, a graph is shown that lists the latest values of this sensor.

4.1.4.3 Testbed Management

The Testbed Management is available for FIESTA administrators only. Here testbeds can be (de)activated for monitoring. As data will be collected only when the testbeds are activated, this can be helpful to only collect data when the testbed and all resources are registered and the workflow is established. This can be helpful as the collected data will be used in analytics tasks and not proper working resources could lead to false interpretation of the data.

4.1.4.4 Notification

The notification system can be used to get notified when the state of one or more testbeds have changed their overall status. This can be used by experimenters when a testbed which will be involved in an experiment is not performing well enough or when resources which are used in an experiment are not sending data to the FIESTA-IoT platform anymore or have been deregistered. Also, testbed owners could register for notifications to get feedback for the process of transferring the sensed data into the platform.

4.1.5 Separation to other components

The Testbed Monitoring will separate from similar components in the platform, the FIESTA-IoT Analytics and the Platform Monitoring that is introduced in the next section. The Analytics Service will work on the data and enhances it, while the Testbed Monitoring tries to analyze the quality of data by comparing observations on a bigger timescale. It tries to determine frequency drifts and to find sensors that are sensing probably not correctly by analyzing sensors measuring the same phenomena in the same region. The Platform Monitoring will monitor the components of the testbed itself. It tries to determine components that behave different from normal operation.

4.2 FIESTA-IoT Platform Monitoring

FIESTA-IoT Platform Monitoring is performed in 2 ways:

- Monitoring Java Virtual Machine (JVM)
- Monitoring logs generated by various components.

In this deliverable, we will not be detailing how monitoring of the JVM is done as this was already explained in Deliverable 4.7 [4]. However, in this deliverable, we focus on how logs generated by various components can be monitored. The logs provide better understanding of execution of component’s functionality and are important to detect bugs in the component. These logs can be analysed for better understanding and bug free FIESTA-IoT Platform. To perform analysis on the logs we use the Graylog[3] tool.

The main idea behind Graylog is to have all the logs generated by different components in a central repository, to analyse the logs and to discover and resolve issues faster. It also allows system administrators to perform queries on the stored/collected logs.

[3] https://www.graylog.org

Graylog offers a web client where system administrators can monitor the performance of the system based on the collected logs. To do so, Graylog offers wide range of functionalities such as search on large-scale log based data, dashboard for quick visualizations of metrics, triggers and alerts and collector that enables easy configuration of the technology used to ship logs to Graylog. Beyond the above-mentioned functionalities, Graylog offers secured access to the logs as logs can hold critical information and REST APIs to access stored information. Further, Graylog architecture follows support for cluster deployment[4].

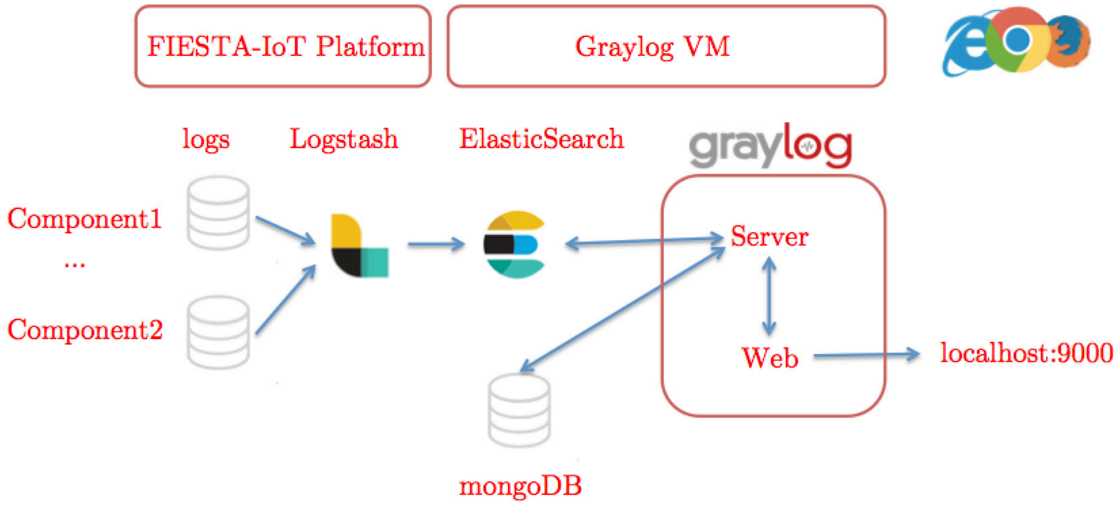


Figure 22: Graylog Architecture.

A successful deployment of Graylog stack requires mongoDB[5], ElasticSearch[6], Logstash[7] and Java8+. Its collector can be run on various platforms. Within FIESTA-IoT ecosystem, Graylog is only available to platform administrators and its stack is deployed as shown in the previous Figure 22.

With respect to the FIESTA-IoT platform that has multiple components running on various technologies (Wildfly, OpenAM and MySQL), Graylog provide a way to monitor the performance of FIESTA-IoT platform and easily browse through errors occurring if any.

4.2.1 Install and Run

Below we provide in brief installation guide for all the components and requirements related to successful deployment of Graylog stack. Once installed, the administrator should first ssh to the Graylog VM using

```
    ssh -L 9000:localhost:9000 <USERNAME>@<HOST_IP>
```

[4] http://docs.graylog.org/en/latest/pages/architecture.html

[5] https://www.mongodb.org

[6] https://www.elastic.co/products/elasticsearch

[7] https://www.elastic.co/products/logstash

Then only they can access the Graylog web using http://localhost:9000. This is done for security purposes.

4.2.1.1 Installations on Graylog VM

The following Table 21 lists requirements for the smooth working of Graylog server and the accompanying web.

Requirements	Version
Java	8+

Table 21 Requirements

In case java is not installed use following commands:

```
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

Once Java is installed MongoDB and ElasticSearch should be installed before proceeding to install Graylog server and Graylog Web. Note that these installations should be done on Graylog VM.

4.2.1.1.1 MongoDB

MongoDB can be installed using following command

```
    sudo apt-get install mongodb-server
```

The above commands will install the latest version of mongoDB. Once installed its configuration file is present in `/etc` folder under the name `mongo.conf`. Admins can update the configuration depending on the needs. We in FIESTA-IoT use basic default configuration. Again, if the configuration is changed it is advised to restart the server using

```
    sudo service mongodb restart
```

4.2.1.1.2 ElasticSearch

After successful installation of mongoDB, ElasticSearch can be installed using

```
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -
echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main"
| sudo tee -a /etc/apt/sources.list.d/elastic-5.x.list
sudo apt-get update && sudo apt-get install elasticsearch
sudo su
cd /var/lib/elasticsearch/
mkdir data
mkdir logs
sudo chown elasticsearch:elasticsearch ../elasticsearch
cd elasticsearch/
```

```
sudo chown elasticsearch:elasticsearch *
```

The above commands will install the latest version (v5.5.1 at the time of writing of this document) and will create `elasticsearch.yml` in the `/etc/elasticsearch/` folder. Administrators are advised to change the `elasticsearch.yml` with following entries:

```
cluster.name: graylog
node.name: graylogFiesta
path.data: /var/lib/elasticsearch/data
path.logs: /var/lib/elasticsearch/logs
network.host: 0.0.0.0
http.port: 9200
discovery.zen.ping.unicast.hosts: ["<HOSTIP>:9200"]
```

This is the minimal configuration that is needed for the successful installation. Note that here the `<HOSTIP>` is the IP address of the machine. We are further not going into details of each entry as they are well explained in the ElasticSearch documentation. Again, if the configuration is successfully updated, it is advised to restart the server using

```
    sudo service elasticsearch restart
```

This will restart the `graylogFiesta` ElasticSearch node.

4.2.1.1.3 Graylog Server + Web

After successful installation of mongoDB and ElasticSearch, Graylog server and its web component can be installed using:

```
wget        https://packages.graylog2.org/repo/packages/graylog-2.3-
repository_latest.deb
sudo dpkg -i graylog-2.3-repository_latest.deb
sudo apt-get update && sudo apt-get install graylog-server
```

This will install Graylog server version 2.3 and the Graylog web component. Once installed, it is essential that Graylog now connect to mongoDB and ElasticSearch. The configuration of this connection can be found in `server.conf` file located in `/etc/graylog/server/`. The minimum configuration that one should perform is listed below:

```
is_master = true
node_id_file = /etc/graylog/server/node-id
password_secret = <SECRET>
root_username = <USERNAME>
root_password_sha2 = <PASSWORD>
root_timezone = Europe/Paris
plugin_dir = /usr/share/graylog-server/plugin
rest_listen_uri = http://127.0.0.1:9000/api/
web_listen_uri = http://127.0.0.1:9000/
elasticsearch_hosts = http://<ESBINDINGHOST>:9200
rotation_strategy = count
elasticsearch_max_docs_per_index = 20000000
```

```
elasticsearch_max_number_of_indices = 20
retention_strategy = delete
elasticsearch_shards = 4
elasticsearch_replicas = 0
elasticsearch_index_prefix = graylog
allow_leading_wildcard_searches = false
allow_highlighting = false
elasticsearch_analyzer = standard
output_batch_size = 500
output_flush_interval = 1
output_fault_count_threshold = 5
output_fault_penalty_seconds = 30
processbuffer_processors = 5
outputbuffer_processors = 3
processor_wait_strategy = blocking
ring_size = 65536
inputbuffer_ring_size = 65536
inputbuffer_processors = 2
inputbuffer_wait_strategy = blocking
message_journal_enabled = true
message_journal_dir = /var/lib/graylog-server/journal
lb_recognition_period_seconds = 3
mongodb_uri = mongodb://localhost/graylog
mongodb_max_connections = 1000
mongodb_threads_allowed_to_block_multiplier = 5
content_packs_dir = /usr/share/graylog-server/contentpacks
content_packs_auto_load = grok-patterns.json
proxied_requests_thread_pool_size = 32
```

Once this is done, the administrator should restart the server using

```
      sudo service graylog-server restart
```

This will restart the Graylog server that also hosts the web component. Note that here <USENAME>, <PASSWORD> and a <SECRET> should be provided along with the ElasticSearch binding IP <ESBINDINGHOST>. This <ESBINDINGHOST> should be same as that ElasticSearch publishes.

4.2.1.2 Installations on FIESTA-IoT Platform machine

Logstash components should be installed on the VM where FIESTA-IoT platform is running so that logs can be pushed to the Graylog VM. Following provides a detailed guide towards installation and configuration of *Logstash* on the VM.

To install Logstash use following:

```
wget -qO - https://packages.elastic.co/GPG-KEY-elasticsearch | sudo
apt-key add -
echo "deb https://packages.elastic.co/elasticsearch/2.x/debian stable
main" | sudo tee -a /etc/apt/sources.list.d/elasticsearch-2.x.list
```

```
echo   "deb   https://packages.elastic.co/logstash/2.4/debian   stable
main" | sudo tee -a /etc/apt/sources.list
sudo apt-get update && sudo apt-get install logstash
```

The above commands will create logstash directory under `/opt/logstash` and will create a `conf.d` folder under `/etc/logstash`.

Once conf.d is created, we now create a configuration file in the `/etc/logstash/conf.d/`. We should name this file as `logstash.conf`. This file should contain:

```
input {
    file {
        type => "<TYPE1>"
        path => [ "<PATH>/<LOGFILENAME1>.log" ]
        codec => multiline {
            pattern => "^%{TIMESTAMP_ISO8601}|^%{MONTHDAY}-
%{MONTH}-%{YEAR}[T ]%{TIME}?|^%{TIME}|^Hibernate?"
            negate => true
            what => "previous"
            }
        }
…
}
filter {
    if [type] == "<TYPE1>" {
        grok {
            match => [ "message", "%{TIMESTAMP_ISO8601:timestamp}
%{LOGLEVEL:loglevel}  %{GREEDYDATA:msg}"]
        }
    }
…
}
output {
         gelf {
            chunksize => 1420
            host => "<HOSTIP>"
            level => "INFO"
            port => 9200
            sender => "Platform"
            }
}
```

Note that here <TYPE1> should be a component name from where the logs are to be fetched. <PATH>/<LOGFILENAME1> should be the absolute path of the log file. Further, in case multiple components are producing logs, the administrator should configure the “input” and “filter”. They should add another “file” block and another “if” block in the configuration above. Once this is done, the administrator should restart the Logstash service using

```
        sudo service logstash restart
```

After a successful start, administrators can check the Logstash logs at `/var/log/logstash/`. In order to check the syntactical errors in the configuration file, administrators can use following command

```
opt/logstash/bin/logstash -f /etc/logstash/conf.d/logstash.conf –t
```

Note that the above will install Logstash version 2.4 this version is not latest version.

4.2.2 Graylog web Dashboard

Once all the dependencies are installed, as said before the admin should first ssh to the Graylog VM using

```
        ssh –L 9000:localhost:9000 <USERNAME>@<HOST_IP>
```

then only they can access the Graylog web using http://localhost:9000. Further, they need <USERNAME> and <PASSWORD> (set in the `server.conf` file of Graylog) for authentication.

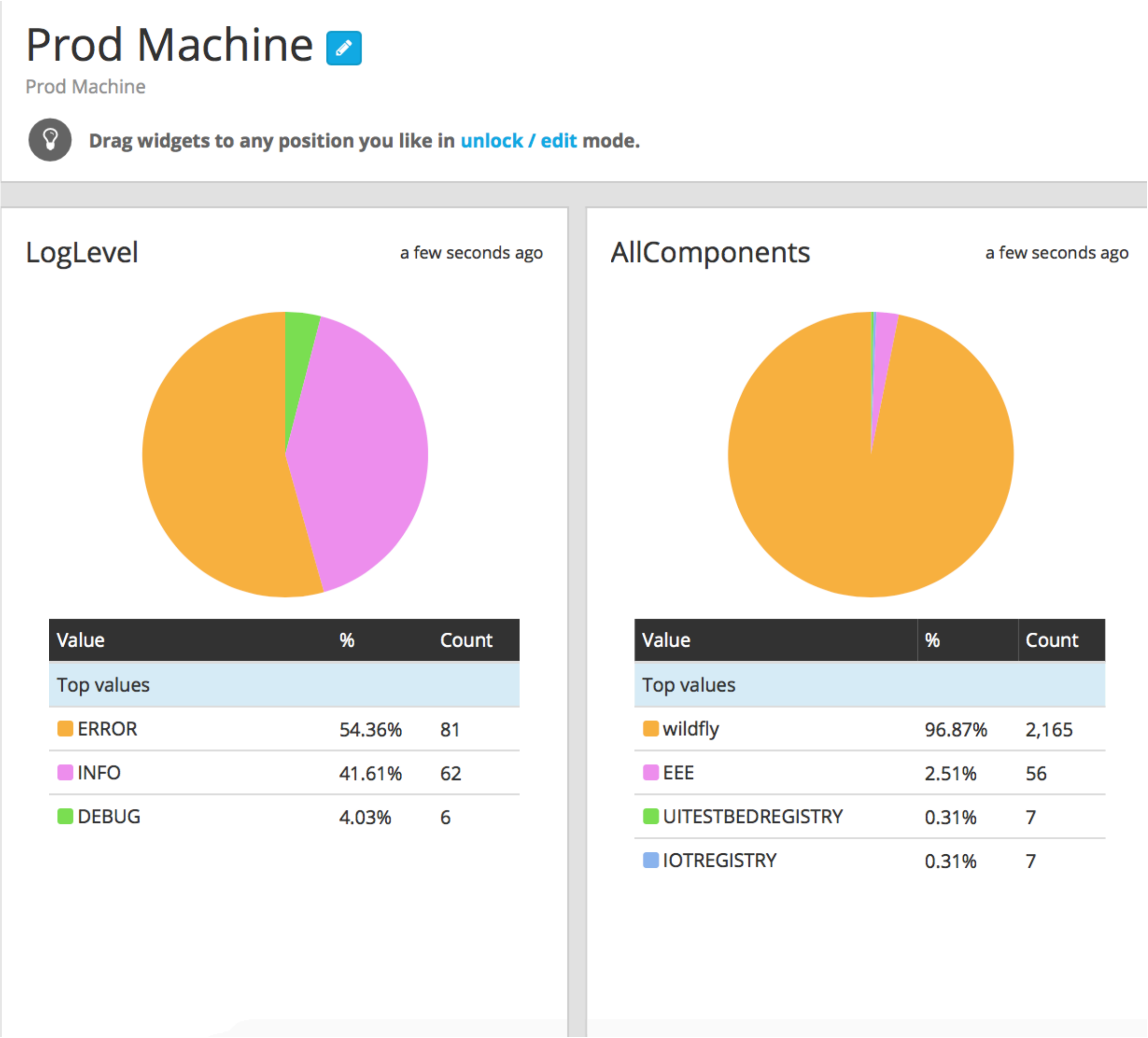
After the successful login, the very first time user should further configure the Graylog server. They need to go to http://localhost:9000/system/inputs and launch a new input. They should select “`Gelf UDP`” as input type and then provide necessary information making sure that the bind address input is the `<ESBINDINGHOST>` that usually is the IP on which ElasticSearch is publishing and the port is 9200. Once this is done, the user should select “`Manage Extractor`” and then select “`add extractor`” that are custom `grok patterns` for the custom FIESTA-IoT component specific message formats. A sample custom `grok pattern` that is

```
%{TIMESTAMP_ISO8601}      \[%{DATA:thread}\]      %{LOGLEVEL:LogLevel}
%{DATA:package} - %{GREEDYDATA:message}
```

Such `grok patterns` are used to extract information from the log message for analysis purposes. After the configuration is done, administrators can see the parsed log messages and write queries. A sample query looks like

```
        type:EEE AND LogLevel:ERROR
```

Graylog besides querying provide quick visualizations that can be incorporated within a dashboard. Figure 23 shows a sample dashboard with 2 visualizations, one for LogLevel and all components that are publishing the logs. Nevertheless, more visualizations are also possible (like histograms).



Prod Machine

Prod Machine

Drag widgets to any position you like in unlock / edit mode.

LogLevel

a few seconds ago

Value	%	Count
Top values		
🟧 ERROR	54.36%	81
🟪 INFO	41.61%	62
🟩 DEBUG	4.03%	6

AllComponents

a few seconds ago

Value	%	Count
Top values		
🟧 wildfly	96.87%	2,165
🟪 EEE	2.51%	56
🟩 UITESTBEDREGISTRY	0.31%	7
🟦 IOTREGISTRY	0.31%	7

Figure 23: Graylog Quick Access Dashboard

5 SUMMARY

This document has presented a set of tools, namely: FIESTA-IoT Analytics, FIESTA-IoT Reasoning and FIESTA-IoT Monitoring as web services that an experimenter can consume. This document has described the implementation, functionality and use cases for such tools.

The FIESTA-IoT Reasoning component enables the inference of data. This was achieved by allowing the experimenter to either create a set of rules or to select from a set of semantically stored rules to process data. The functionality of the tool was demonstrated with examples. This document provided a comprehensive description of the tools functionality and use cases. The FIESTA-IoT Analytics tool allowed the experimenter to select a set of pre-processing and machine learning algorithms to analyze historical data sets. This document provided a detailed description of the available algorithms and error messages when interacting with the component. Furthermore, this document provided two use case examples that demonstrated the FIESTA-IoT Analytics tool potential in analyzing data obtained from the IoT-Registry. Finally, this document provided a detailed description of the implementation and functionality of the FIESTA-IoT Monitoring tool. Namely, the tool captures information relevant to testbed performance, where a portal provides a convenient dashboard for a user to evaluate current testbed performance.

This document has addressed the core challenge of providing a set of tools for both processing data as well as performance monitoring of IoT testbeds. For each tool, future work will seek to improve functionality as follows. Namely, for the FIESTA-IoT Analytics tool, we will attempt to increase the number of data analysis tools available to the experimenter, along with a feedback mechanism where the experimenter can evaluate the performance of specific algorithms. For the FIESTA-IoT Reasoning tool, emphasis will be placed on developing complex event processing functionalities. While for the FIESTA-IoT Monitoring tool, mechanisms for using performance data to provide rapid and active feedback to the testbeds will be considered.

REFERENCES

[1] FIESTA-IoT, “Deliverable 3.5: Concept and Development for IoT Data Analytics and IoT Stream and Service Management”, 2016

[2] FIESTA-IoT, “Deliverable 4.2: EaaS Model Specification and Implementation V2”, 2017

[3] FIESTA-IoT, “Deliverable 4.8: Infrastructure for Submitting and Managing IoT Experiments”, 2017.

[4] FIESTA-IoT, “Deliverable 4.7: Infrastructure for Submitting and Managing IoT Experiments V1”, 2016

APPENDIX I REASONING EXAMPLES

A1.1 Rule resource API

A1.1.1 Create rule

Title	Create rule

Sample Call	{
	"name":"Demo new rule 3333",
	"content":"@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .\n\n(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"25\"^^xsd:double) -> (?observation reasoning:announce \"dangerous_notify\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"23\"^^xsd:double) -> (?observation reasoning:announce \"lowpower_notify\"^^xsd:string).",
	"sensor":"https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I",
	"description":"demo new rule",
	"latitude":51.243343,
	"longitude":-0.5932438,
	"quantityKind":"http://purl.org/iot/vocab/m3-lite#Power",
	"unitOfMeasurement":"http://purl.org/iot/vocab/m3-lite#Watt"
	
	}

Response body	Response status code: 201 { "id": 27, "name": "Demo new rule 3333", "userId": "hungnguyen", "created": "2017-09-27T09:02:05.179+0000", "updated": null, "content": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .\n\n(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"25\"^^xsd:double) -> (?observation reasoning:announce \"dangerous_notify\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit

	rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"23\"^^xsd:double) -> (?observation reasoning:announce \"lowpower_notify\"^^xsd:string).", "sensor": "https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I", "description": null, "latitude": 51.243343, "longitude": -0.5932438, "quantityKind": "http://purl.org/iot/vocab/m3-lite#Power", "unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt", "ruleType": 1, "nonExpertOriginalRules": null }

Response body	Response status code: 400
	{
	"result": false,
	"message": "Can not get sensor information given by sensor ID:https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I"
	}

Response body	Response status code: 500 Internal Server error

A1.1.2 Update rule

Title	Update rule
Sample Call	{ “id”:18, “name”:”Demo new rule 4444111”, “content”:”@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .\n\n(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \”25\”^^xsd:double) -> (?observation reasoning:announce \”dangerous_notify\”^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \”21\”^^xsd:double) -> (?observation reasoning:announce \”lowpower_notify\”^^xsd:string).”, “sensor”:”https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I”, “description”:”demo edit rule”,

	“latitude”:51.243343, “longitude”:-0.5932438, “quantityKind”:”http://purl.org/iot/vocab/m3-lite#Power”, “unitOfMeasurement”:”http://purl.org/iot/vocab/m3-lite#Watt”, “ruleType”:1 }

Response body	Response status code: 200
	{
	"id": 27,
	"name": "Demo new rule 3333",
	"userId": "hungnguyen",
	"created": "2017-09-27T09:02:05.179+0000",
	"updated": "2017-09-27T09:02:05.179+0000",
	"content": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .\n\n(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"25\"^^xsd:double) -> (?observation reasoning:announce \"dangerous_notify\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"23\"^^xsd:double) -> (?observation reasoning:announce \"lowpower_notify\"^^xsd:string).",
	"sensor": "https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I",
	"description": null,
	"latitude": 51.243343,
	"longitude": -0.5932438,
	"quantityKind": "http://purl.org/iot/vocab/m3-lite#Power",
	"unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt",
	"ruleType": 1,
	"nonExpertOriginalRules": null
	}

Response body	Response status code: 400 x-fiestareasonerengineapp-error → Can not update rule invalid rule id! x-fiestareasonerengineapp-error →You can not update this rule because you don not have permission !

Response body	Response status code: 500 Internal Server error

A1.1.3 Validate rule

Title	Validate rule

Sample Call	{
	"rule":"@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .\n\n(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"25\"^^xsd:double) -> (?observation reasoning:announce \"dangerous_notify\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"23\"^^xsd:double) -> (?observation reasoning:announce \"lowpower_notify\"^^xsd:string).",
	"sensorId":"https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I"
	}

Response body	Response status code: 200
	{
	"result": true,
	"message": "Rule validation is valid!"
	}
	
	{
	"result": false,
	"message": "Can not get sensor information given by sensor ID:https://platform.fiesta-iot.eu/iot-registry/api/resources/VsnDY_ipIeAhy2eCc5jxNRqGyBVsIwso2bO-8KCr7GKnfKLgda8TdXItkjaADUHLb6VnxSxvR7MddDzbM9fR-Crr9BuuRehd9QCZYPKVzsuaAvFxz6BhRc_PTWFEzu2I"
	}
	
	{
	"result": false,
	"message": "org.apache.jena.reasoner.rulesys.Rule$ParserException: Malformed rule\nAt '@. '"
	}

A1.2 Register rule resource

A1.2.1 Rule registration

Title	Rule registration

Sample Call	{
	"name":"Demo register new rule ABC",
	"description":"Demo register new rule ABC",
	"sensor":"https://platform.fiesta-iot.eu/iot-registry/api/resources/Ur7Q-GLgxiLsfK4ZhXffEryue052DxDQzb8jxqKMPyLJZUiTr-ZpAj1ZK_hi302o5gp8V6Fe1a2jEzg_STnJkUCQHp8f7qAg1DiohqUnfcll3289LvfcuRmXiDPfZROl",
	"quantityKind":"http://purl.org/iot/vocab/m3-lite#Power",
	"unitOfMeasurement":"http://purl.org/iot/vocab/m3-lite#Watt",
	"latitude":51.243343,
	"longitude":-0.5932438,
	"ruleId":20
	}

Response body	Response status code: 201 { "id": 23, "name": "Demo register new rule ABC", "description": "Demo register new rule ABC", "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"high_notification\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"low_notification\"^^xsd:string).", "sensor": "https://platform.fiesta-iot.eu/iot-registry/api/resources/Ur7Q-GLgxiLsfK4ZhXffEryue052DxDQzb8jxqKMPyLJZUiTr-ZpAj1ZK_hi302o5gp8V6Fe1a2jEzg_STnJkUCQHp8f7qAg1DiohqUnfcll3289LvfcuRmXiDPfZROl", "latitude": 51.243343, "longitude": -0.5932438, "quantityKind": "http://purl.org/iot/vocab/m3-lite#Power", "unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt", "userId": "hungnguyen",

	"created": "2017-09-27T09:09:45.405+0000", "updated": null, "ruleId": 20 }
Respons e body	Response status code: 400 x-fiestareasonerengineapp-error →Can not get sensor endpoint by given sensor ID, Please try again sensor! !
Respons e body	Response status code: 500 Internal Server error

A1.2.2 Updating rule registration

Title	Updating rule registration

Sample Call	{
	"id":20,
	"name":"Demo register edit rule",
	"description":"Demo register edit rule",
	"sensor":"https://platform.fiesta-iot.eu/iot-registry/api/resources/tRRAK2lA6S5GEca2qPQD6hWzOn-kLp82OXHnXItm16LbPlSitapxvtgEcrxPmWuDG-vqcW8xUTwrYj13_jt-t01DzPKZA6v1VYA_UVR77ihfGV9LONi8Tm0Ccv3rzBXR",
	"quantityKind":"http://purl.org/iot/vocab/m3-lite#Power",
	"unitOfMeasurement":"http://purl.org/iot/vocab/m3-lite#Watt",
	"latitude":51.243343,
	"longitude":-0.5932438,
	"ruleId":20
	}

Response body	Response status code: 200 { "id": 20, "name": "Demo register edit rule", "description": "Demo register edit rule", "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"high_notification\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue

	?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"low_notification\"^^xsd:string).", "sensor": "https://platform.fiesta-iot.eu/iot-registry/api/resources/tRRAK2lA6S5GEca2qPQD6hWzOn-kLp82OXHnXltm16LbPlSitapxvtgEcrxPmWuDG-vqcW8xUTwrYj13_jt-t01DzPKZA6v1VYA_UVR77ihfGV9LONi8Tm0Ccv3rzBXR", "latitude": 51.243343, "longitude": -0.5932438, "quantityKind": "http://purl.org/iot/vocab/m3-lite#Power", "unitOfMeasurement": "http://purl.org/iot/vocab/m3-lite#Watt", "userId": "hungnguyen", "created": "2017-09-25T11:44:55.000+0000", "updated": "2017-09-27T09:18:32.929+0000", "ruleId": 20 }
Response body	Response status code: 400 x-fiestareasonerengineapp-error →Not found any register rule given by this ID!
Response body	Response status code: 500 Internal Server error

A1.3 Execution resource

A1.3.1 Get specific execution

Title	Get specific execution

Sample Call	{
	"created": "2017-10-23T14:04:26.551Z",
	"ended": "2017-10-23T14:04:26.552Z",
	"fullData": "string",
	"id": 0,
	"infferedData": "string",
	"originalData": "string",
	"registerRule": {
	"created": "2017-10-23T14:04:26.552Z",
	"data": "string",
	"description": "string",
	"fullData": "string",
	"hashedSensor": "string",
	"id": 0,
	"inferredData": "string",
	"latitude": 0,
	"longitude": 0,
	"name": "string",
	"quantityKind": "string",

	"reasoning": {
	"content": "string",
	"created": "2017-10-23T14:04:26.552Z",
	"description": "string",
	"hashedSensor": "string",
	"id": 0,
	"latitude": 0,
	"longitude": 0,
	"name": "string",
	"quantityKind": "string",
	"ruleType": 0,
	"sensor": "string",
	"sensorEndp": "string",
	"sensorMeta": "string",
	"sensorSampleData": "string",
	"unitOfMeasurement": "string",
	"updated": "2017-10-23T14:04:26.552Z",
	"userId": "string"
	},
	"ruleContent": "string",
	"sensor": "string",
	"sensorEndp": "string",
	"sensorMeta": "string",
	"unitOfMeasurement": "string",
	"updated": "2017-10-23T14:04:26.552Z",
	"userId": "string"
	},
	"ruleContent": "string",
	"sensor": "string",
	"started": "2017-10-23T14:04:26.552Z",
	"status": true,
	"type": 0,
	"updated": "2017-10-23T14:04:26.552Z",
	"userId": "string"
	}

Response body	Response status code: 200 { "id": 10, "status": true, "created": "2017-09-25T12:14:22.000+0000", "updated": null, "started": "2017-09-25T12:14:20.000+0000",

	"ended": "2017-09-25T12:14:22.000+0000", "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"high_notification\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"low_notification\"^^xsd:string).", "originalData": "{\"vars\":[\"sensingDevice\",\"dataValue\",\"dateTime\",\"observation\",\"sensorOutput\",\"obsValue\",\"instant\"],\"items\":[]}", "infferedData": "{ }\n", "fullData": "{\n \"@id\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-120\",\n \"@type\" : \"geo:Point\",\n \"altRelative\" : \"2\",\n \"relativeLocation\" : \"http://sws.geonames.org/6695971/\",\n \"geo:alt\" : 57.863815,\n \"geo:lat\" : 51.2433445,\n \"geo:long\" : -0.5932438,\n \"@context\" : {\n \"long\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#long\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"lat\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#lat\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"alt\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#alt\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"relativeLocation\" : {\n \"@id\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#relativeLocation\"\n },\n \"altRelative\" : {\n \"@id\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#altRelative\"\n },\n \"onemtom\" : \"http://www.onem2m.org/ontology/Base_Ontology/base_ontology#\",\n \"qudt\" : \"http://data.qudt.org/qudt/owl/1.0.0/unit.owl#\",\n \"iot-lite\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#\",\n \"qu\" : \"http://purl.org/NET/ssnx/qu/qu#\",\n \"owl\" : \"http://www.w3.org/2002/07/owl#\",\n \"ns\" : \"http://creativecommons.org/ns#\",\n \"xsd\" : \"http://www.w3.org/2001/XMLSchema#\",\n \"fiesta-iot\" : \"http://purl.org/iot/ontology/fiesta-iot#\",\n \"rdfs\" : \"http://www.w3.org/2000/01/rdf-schema#\",\n \"ssn\" : \"http://purl.oclc.org/NET/ssnx/ssn#\",\n \"geo\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#\",\n \"sics\" : \"http://smart-ics.ee.surrey.ac.uk/fiesta-iot/\",\n \"terms\" : \"http://purl.org/dc/terms/\",\n \"rdf\" : \"http://www.w3.org/1999/02/22-rdf-syntax-ns#\",\n \"dcterms\" : \"http://purl.org/dc/terms/\",\n \"dul\" : \"http://www.loa.istc.cnr.it/ontologies/DUL.owl#\",\n \"time\" : \"http://www.w3.org/2006/time#\",\n \"mthreelite\" : \"http://purl.org/iot/vocab/m3-lite#\",\n \"vann\" : \"http://purl.org/vocab/vann/\",\n \"dc\" : \"http://purl.org/dc/elements/1.1/\"\n }\n}\n", "userId": "hungnguyen", "sensor": "https://platform.fiesta-iot.eu/iot-registry/api/resources/tRRAK2lA6S5GEca2qPQD6hWzOn-kLp82OXHnXltm16LbPlSitapxvtgEcrxPmWuDG-vqcW8xUTwrYj13_jt-t01DzPKZA6v1VYA_UVR77ihfGV9LONi8Tm0Ccv3rzBXR", "executeType": 2, "registerRuleId": 20 }
Response body	Response status code: 404 Not found

Response body	Response status code: 500 Internal Server error

A1.3.3 Execute rule

Title	Execute a rule

Sample Call	Execute current time
	{
	"registerRuleId":20,
	"started": "null",
	"ended": "null",
	"executeType":1
	}
	Execute within a specific time period (must be less than 5 days)
	{
	"registerRuleId":20,
	"started":"2017-09-20T23:00:00.000Z",
	"ended":"2017-09-26T23:00:00.000Z",
	"executeType":2
	}

Response body	Response status code: 201 { "id": 17, "status": true, "created": "2017-09-27T09:34:28.009+0000", "updated": null, "started": "2017-09-27T09:34:24.910+0000", "ended": "2017-09-27T09:34:24.910+0000", "ruleContent": "@prefix iot-lite: <http://purl.oclc.org/NET/UNIS/fiware/iot-lite#> .\n@prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#> .\n@prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#> .\n@prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#> .\n@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .\n@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .\n@prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#> .\n@prefix time: <http://www.w3.org/2006/time#> .\n@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .\n@prefix reasoning: <https://fiesta-iot.eu/reasoning#> .(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\ngreaterThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"high_notification\"^^xsd:string).(?observation rdf:type ssn:Observation),\n(?observation ssn:observedProperty ?observedProperty),\n(?observedProperty rdf:type m3-lite:Power),\n(?observation ssn:observationResult ?sensorOutput),\n(?sensorOutput ssn:hasValue ?obsValue),\n(?obsValue dul:hasDataValue ?dataValue),\n(?obsValue iot-lite:hasUnit ?unit),\n(?unit rdf:type m3-lite:Watt),\nlessThan(?dataValue, \"20\"^^xsd:double) -> (?observation reasoning:announce \"low_notification\"^^xsd:string).",

	"originalData": "{\"@graph\":[{\"geo:alt\":57.863815,\"iot-lite:relativeLocation\":\"http://sws.geonames.org/6695971/\",\"geo:lat\":51.2433445,\"@type\":\"geo:Point\",\"geo:long\":-0.5932438,\"@id\":\"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-105\",\"iot-lite:altRelative\":\"2\"},{\"observedBy\":\"sics:resource/sc-sics-sp-012-power\",\"observationResult\":\"sics:sensorOutput#j0XVeZvdyW\",\"@type\":\"ssn:Observation\",\"observationSamplingTime\":\"sics:timeInterval#UTC_OPR6ay621D\",\"observedProperty\":\"sics:observationProperty#Power\",\"location\":\"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-105\",\"@id\":\"sics:observation#j0XVeZvdyW\"},{\"@type\":\"mthreelite:Power\",\"@id\":\"sics:observationProperty#Power\"},{\"hasUnit\":\"sics:unit#Watt\",\"@type\":\"ssn:ObservationValue\",\"dul:hasDataValue\":35.881526,\"@id\":\"sics:observationValue#j0XVeZvdyW\"},{\"@type\":\"mthreelite:EnergyMeter\",\"@id\":\"sics:resource/sc-sics-sp-012-power\"},{\"@type\":\"ssn:SensorOutput\",\"hasValue\":\"sics:observationValue#j0XVeZvdyW\",\"@id\":\"sics:sensorOutput#j0XVeZvdyW\"},{\"@type\":\"time:Instant\",\"@id\":\"sics:timeInterval#UTC_OPR6ay621D\",\"inXSDDateTime\":\"2017-09-27T09:34:00Z\"},{\"@type\":\"mthreelite:Watt\",\"@id\":\"sics:unit#Watt\"}],\"@context\":{\"qudt\":\"http://data.qudt.org/qudt/owl/1.0.0/unit.owl#\",\"observedBy\":{\"@type\":\"@id\",\"@id\":\"http://purl.oclc.org/NET/ssnx/ssn#observedBy\"},\"iot-lite\":\"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#\",\"owl\":\"http://www.w3.org/2002/07/owl#\",\"ns\":\"http://creativecommons.org/ns#\",\"xsd\":\"http://www.w3.org/2001/XMLSchema#\",\"observedProperty\":{\"@type\":\"@id\",\"@id\":\"http://purl.oclc.org/NET/ssnx/ssn#observedProperty\"},\"rdfs\":\"http://www.w3.org/2000/01/rdf-schema#\",\"long\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#double\",\"@id\":\"http://www.w3.org/2003/01/geo/wgs84_pos#long\"},\"ssn\":\"http://purl.oclc.org/NET/ssnx/ssn#\",\"geo\":\"http://www.w3.org/2003/01/geo/wgs84_pos#\",\"terms\":\"http://purl.org/dc/terms/\",\"observationSamplingTime\":{\"@type\":\"@id\",\"@id\":\"http://purl.oclc.org/NET/ssnx/ssn#observationSamplingTime\"},\"dcterms\":\"http://purl.org/dc/terms/\",\"mthreelite\":\"http://purl.org/iot/vocab/m3-lite#\",\"vann\":\"http://purl.org/vocab/vann/\",\"lat\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#double\",\"@id\":\"http://www.w3.org/2003/01/geo/wgs84_pos#lat\"},\"onemtom\":\"http://www.onem2m.org/ontology/Base_Ontology/base_ontology#\",\"altRelative\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#string\",\"@id\":\"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#altRelative\"},\"observationResult\":{\"@type\":\"@id\",\"@id\":\"http://purl.oclc.org/NET/ssnx/ssn#observationResult\"},\"qu\":\"http://purl.org/NET/ssnx/qu/qu#\",\"alt\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#double\",\"@id\":\"http://www.w3.org/2003/01/geo/wgs84_pos#alt\"},\"fiesta-iot\":\"http://purl.org/iot/ontology/fiesta-iot#\",\"hasValue\":{\"@type\":\"@id\",\"@id\":\"http://purl.oclc.org/NET/ssnx/ssn#hasValue\"},\"relativeLocation\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#string\",\"@id\":\"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#relativeLocation\"},\"sics\":\"http://smart-ics.ee.surrey.ac.uk/fiesta-iot/\",\"hasUnit\":{\"@type\":\"@id\",\"@id\":\"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#hasUnit\"},\"rdf\":\"http://www.w3.org/1999/02/22-rdf-syntax-ns#\",\"location\":{\"@type\":\"@id\",\"@id\":\"http://www.w3.org/2003/01/geo/wgs84_pos#location\"},\"dul\":\"http://www.loa.istc.cnr.it/ontologies/DUL.owl#\",\"time\":\"http://www.w3.org/2006/time#\",\"inXSDDateTime\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#dateTime\",\"@id\":\"http://www.w3.org/2006/time#inXSDDateTime\"},\"hasDataValue\":{\"@type\":\"http://www.w3.org/2001/XMLSchema#double\",\"@id\":\"http://www.loa.istc.cnr.it/ontologies/DUL.owl#hasDataValue\"},\"dc\":\"http://purl.org/dc/elements/1.1/\"}}", "infferedData": "{\n \"@id\" : \"http://smart-ics.ee.surrey.ac.uk/fiesta-iot/observation#j0XVeZvdyW\",\n \"announce\" : \"high_notification\",\n \"@context\" : {\n \"announce\" : {\n \"@id\" : \"https://fiesta-iot.eu/reasoning#announce\"\n }\n }\n}\n", "fullData": "{\n \"@graph\" : [{\n \"@id\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-105\",\n \"@type\" : \"geo:Point\",\n \"altRelative\" : \"2\",\n \"relativeLocation\" : \"http://sws.geonames.org/6695971/\",\n \"geo:alt\" : 57.863815,\n \"geo:lat\" : 51.2433445,\n \"geo:long\" : -0.5932438\n }, {\n \"@id\" : \"sics:observation#j0XVeZvdyW\",\n \"@type\" : \"ssn:Observation\",\n \"observationResult\" : \"sics:sensorOutput#j0XVeZvdyW\",\n \"observationSamplingTime\" : \"sics:timeInterval#UTC_OPR6ay621D\",\n \"observedBy\" : \"sics:resource/sc-sics-sp-012-power\",\n \"observedProperty\" : \"sics:observationProperty#Power\",\n \"location\" : \"sics:loc#UNIVERSITY_OF_SURREY-unis-ics-desk-105\",\n \"announce\" : \"high_notification\"\n }, {\n \"@id\" : \"sics:observationProperty#Power\",\n \"@type\" : \"mthreelite:Power\"\n }, {\n \"@id\" : \"sics:observationValue#j0XVeZvdyW\",\n \"@type\" : \"ssn:ObservationValue\",\n \"hasUnit\" : \"sics:unit#Watt\",\n \"dul:hasDataValue\" : 35.881526\n }, {\n \"@id\" : \"sics:resource/sc-sics-sp-012-power\",\n \"@type\" :

	\"mthreelite:EnergyMeter\"\n }, {\n \"@id\" : \"sics:sensorOutput#j0XVeZvdyW\",\n \"@type\" : \"ssn:SensorOutput\",\n \"hasValue\" : \"sics:observationValue#j0XVeZvdyW\"\n }, {\n \"@id\" : \"sics:timeInterval#UTC_OPR6ay621D\",\n \"@type\" : \"time:Instant\",\n \"inXSDDateTime\" : \"2017-09-27T09:34:00Z\"\n }, {\n \"@id\" : \"sics:unit#Watt\",\n \"@type\" : \"mthreelite:Watt\"\n }],\n \"@context\" : {\n \"announce\" : {\n \"@id\" : \"https://fiesta-iot.eu/reasoning#announce\"\n },\n \"long\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#long\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"lat\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#lat\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"alt\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#alt\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"relativeLocation\" : {\n \"@id\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#relativeLocation\"\n },\n \"altRelative\" : {\n \"@id\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#altRelative\"\n },\n \"location\" : {\n \"@id\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#location\",\n \"@type\" : \"@id\"\n },\n \"observedProperty\" : {\n \"@id\" : \"http://purl.oclc.org/NET/ssnx/ssn#observedProperty\",\n \"@type\" : \"@id\"\n },\n \"observedBy\" : {\n \"@id\" : \"http://purl.oclc.org/NET/ssnx/ssn#observedBy\",\n \"@type\" : \"@id\"\n },\n \"observationSamplingTime\" : {\n \"@id\" : \"http://purl.oclc.org/NET/ssnx/ssn#observationSamplingTime\",\n \"@type\" : \"@id\"\n },\n \"observationResult\" : {\n \"@id\" : \"http://purl.oclc.org/NET/ssnx/ssn#observationResult\",\n \"@type\" : \"@id\"\n },\n \"hasDataValue\" : {\n \"@id\" : \"http://www.loa.istc.cnr.it/ontologies/DUL.owl#hasDataValue\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#double\"\n },\n \"hasUnit\" : {\n \"@id\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#hasUnit\",\n \"@type\" : \"@id\"\n },\n \"inXSDDateTime\" : {\n \"@id\" : \"http://www.w3.org/2006/time#inXSDDateTime\",\n \"@type\" : \"http://www.w3.org/2001/XMLSchema#dateTime\"\n },\n \"hasValue\" : {\n \"@id\" : \"http://purl.oclc.org/NET/ssnx/ssn#hasValue\",\n \"@type\" : \"@id\"\n },\n \"onemtom\" : \"http://www.onem2m.org/ontology/Base_Ontology/base_ontology#\",\n \"qudt\" : \"http://data.qudt.org/qudt/owl/1.0.0/unit.owl#\",\n \"iot-lite\" : \"http://purl.oclc.org/NET/UNIS/fiware/iot-lite#\",\n \"qu\" : \"http://purl.org/NET/ssnx/qu/qu#\",\n \"owl\" : \"http://www.w3.org/2002/07/owl#\",\n \"ns\" : \"http://creativecommons.org/ns#\",\n \"xsd\" : \"http://www.w3.org/2001/XMLSchema#\",\n \"fiesta-iot\" : \"http://purl.org/iot/ontology/fiesta-iot#\",\n \"rdfs\" : \"http://www.w3.org/2000/01/rdf-schema#\",\n \"ssn\" : \"http://purl.oclc.org/NET/ssnx/ssn#\",\n \"geo\" : \"http://www.w3.org/2003/01/geo/wgs84_pos#\",\n \"sics\" : \"http://smart-ics.ee.surrey.ac.uk/fiesta-iot/\",\n \"terms\" : \"http://purl.org/dc/terms/\",\n \"rdf\" : \"http://www.w3.org/1999/02/22-rdf-syntax-ns#\",\n \"dcterms\" : \"http://purl.org/dc/terms/\",\n \"dul\" : \"http://www.loa.istc.cnr.it/ontologies/DUL.owl#\",\n \"time\" : \"http://www.w3.org/2006/time#\",\n \"mthreelite\" : \"http://purl.org/iot/vocab/m3-lite#\",\n \"vann\" : \"http://purl.org/vocab/vann/\",\n \"dc\" : \"http://purl.org/dc/elements/1.1/\"\n }\n}\n", "userId": "hungnguyen", "sensor": "https://platform.fiesta-iot.eu/iot-registry/api/resources/1NiQTwHfIgfqKODCcZpmYgtwvYqIZQ0Ycj2wnEoFkfgS-xeviqhbKlcHUQQpttfkTglWwwFwjU6CVXwT3Fh5ONrUE27If0dzZwm1nFeSYdEcRB08HPWy5MMfSBbN6r3y", "executeType": 1, "registerRuleId": 24 }
Response body	Response status code: 500 Internal Server error