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

D5.4 - Best Practices for Experiments Design and Conduction

Document Id:	FIESTA-IoT-D54-15042018-Draft
File Name:	FIESTA-IoT-D54-15042018-Draft.pdf
Document reference:	Deliverable 5.4
Version:	Draft
Editor:	Mengxuan Zhao
Organisation:	EGM
Date:	15/04/2018
Document type:	Deliverable
Dissemination level:	PU

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Mengxuan Zhao	EGM	2018/01/15	ToC definition
	Rachit Agarwal	Inria	2018/02/01	Section: Handbook for Experimenters, Best Practices
	David Gomez	UC	2018/02/01	Section: Best Practices
	Flavio Cirillo	NEC	2018/02/13	Usage of FAQs for best practices creation (Section 3)
	Luis Sanchez	UC	2018/02/21	Contributions to Section 3 and Section 4
	Mengxuan Zhao	EGM	2018/02/27	Contribution to Lessons Learnt, Executive summary and Conclusion
	Elias Tragos	NUIG-Insight	2018/03/14	Comments, fixes, improvements
	Luis Sanchez	UC	2018/03/21	Contribution to Best Practices for Testbeds
V02	Mengxuan Zhao	EGM	2018/03/28	For internal review
V03	Tiago Teixeira	Unparallel	2018/04/03	QR
	Ronald Steinke	FOKUS	2018/04/05	QR
V04	Mengxuan Zhao Luis Sanchez Rachit Agarwal	EGM UC Inria	2018/04/06	Version for submission
V05	Martin Serrano	NUIG-Insight	2018/04/15	Circulated for Approval
Draft	Martin Serrano	NUIG-Insight	2018/04/15	EC Submitted

TABLE OF CONTENTS

TERMS AND ACRONYMS

AaaS	Annotator as a Service
API	Application Program Interface
DSL	Domain Specific Language
FAQ	Frequently asked questions
EEE	Experiment Execution Engine
FEDSpec	FIESTA-IoT Experiment Description
FIRE	Future Internet Research and Experimentation
FISMO	FIESTA-IoT Service Model Object
HTTP	Hypertext Transfer Protocol
IRI	Internationalized Resource Identifier
OC	Open Call
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language

1 EXECUTIVE SUMMARY

This deliverable summarizes the effort that the FIESTA-IoT consortium have made to help experimenters to efficiently get started with the semantic platform and design, deploy and execute their innovative experiments upon the platform. A rich handbook is created to describe every detail of the platform, and plenty of examples are provided in the handbook to complete the development guide.

The in-house experiments have been designed, developed and deployed on the FIESTA-IoT platform in parallel with the platform development, thus valuable feedback and lessons learnt have been collected and made available in the handbook. During the Open Call Experiment process, thanks to the interaction with external experimenters, we get extra feedback from the perspective of the external. These feedback and lessons learnt are communicated to the FIESTA-IoT experimenter community using different channels, such as the handbook, the FAQ section on the web page, the workshop tutorial materials, etc. The current deliverable gather these informations from various channels to present them in a single document. The feedback and lessons learnt consist of several topics, including thesteps of experiment design and deployment to interact with the FIESTA-IoT platform and the usage of the FIESTA-IoT tools. Best practices of using the FIESTA-IoT platform for experiments have been created from these feedback and lessons learnt.

2 HANDBOOK FOR EXPERIMENTERS

FIESTA-IoT created a guide (handbook) (FIESTA-IoT, 2018)] for experimenters to showcase the use of the platform correctly. The main idea behind the handbook is to provide a holistic vision of all the features that FIESTA-IoT platform currently supports. Following a “Getting Started”-like approach, the aim of the handbook is to guide external users (experimenters with respect to this deliverable) to correctly integrate their experiment with the FIESTA-IoT platform. For experimenters, the handbook clearly describes different ways how experimenters can interact with the FIESTA-IoT platform. It describes clearly the DSL (Domain Specific Language) experimenters need to follow to create their experiment along with examples. The handbook provides steps to register a new experiment, execute it based on the comfortability level of experimenters (novice or advanced) and retrieve the resultsets. The handbook list APIs available for advanced experimenters so that experimenters can create their own Experiment Execution Engine.

One of the most important part of an experiment is the SPARQL query. The handbook provides recommendations on optimised queries that should be sent to the FIESTA-IoT platform without overloading it. These best practices/recommendations are mentioned in section 4. On top of these recommendations, the handbook lists various SPARQL queries that an experimenter can use to first understand the data stored in the FIESTA-IoT system and then use the recommendations to create experiment specific queries.

3 LESSONS LEARNT FROM EXPERIMENT DEVELOPMENT AND DEPLOYMENT

This section summarizes the lessons learnt from experiment development and deployment of the in-house and the open-call experiments.

3.1 Creating and managing an experiment when FIESTA-IoT tools are used.

3.1.1 Periodicity

- If the execution of the Service defined within the experiment is producing no data, it is very important to check the consistency of fed:startTime, fed:Periodicity and fed:stopTime;
- Some times there might be misunderstanding of the fed:Periodicity field. The fed:Periodicity is the time period that should pass between two consecutive execution of the same service defined within the experiment.
- FIESTA-IoT does not support a feature to specify an exact number of execution of the SPARQL but it is possible to have the same result by correctly tuning the fed:startTime, fed:Periodicity and fed:stopTime parameters

3.1.2 Experiment managing

- Before making any update to a service in an experiment it is necessary to delete the scheduled services through the Management console UI. Once deleted, the experiment can be updated according to the needs and the services can be rescheduled on the execution engine.

3.1.3 Historical data retrieving

- The initial design of the experiment managing specification aims at helping experimenters to interact with the available sensors and get the latest data periodically. For the experiments that need only historical data for massive data analytics, the FIESTA experiment management model turns to be heavy and unnecessary, while it requires some learning effort from the experimenter. From the feedback of one Open-Call experiment, in this case, it is more efficient to directly interact with the IoT-registry API which accepts the SPARQL queries without any extra upper layer complexity (an example is given in 3.4).

3.2 Creating the SPARQL

- The SPARQL is one of the critical points of an experiment. It is very common to be confused between classes and instance of an ontology. For instance the following example is wrong (check the highlighted text)

```
Prefix ssn: http://purl.oclc.org/NET/ssnx/ssn#
Prefix dul: http://www.loa.istc.cnr.it/ontologies/DUL.owl#
Prefix geo: http://www.w3.org/2003/01/geo/wgs84_pos#
Prefix m3-lite: http://purl.org/iot/vocab/m3-lite#
Prefix xsd: http://www.w3.org/2001/XMLSchema#
Prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
select distinct ?lat ?long ?dataValue
where {
    ?observ a ssn:Observation.
    ?observ ssn:observedBy m3-lite:SoilHumiditySensor.
    ?observ ssn:observedProperty m3-lite:SoilHumidity.
    ?observ geo:location ?point.
    ?point geo:lat ?lat.
    ?point geo:long ?long.
    ?observ ssn:observationResult ?output.
  ?output ssn:hasValue ?value.
     ?value dul:hasDataValue ?dataValue.
}
```

An observation is “observedBy” an instance of sensor, not by a class (m3-lite:SoilHumiditySensor”. The property should also be an instance of class “m3:soilHumidity”. A correct example would be:

```
Prefix ssn: <http://purl.oclc.org/NET/ssnx/ssn#>
Prefix dul: <http://www.loa.istc.cnr.it/ontologies/DUL.owl#>
Prefix geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
Prefix m3-lite: <http://purl.org/iot/vocab/m3-lite#>
Prefix xsd: <http://www.w3.org/2001/XMLSchema#>
Prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
select distinct ?lat ?long ?dataValue
where {
    ?observ a ssn:Observation.
    ?observ ssn:observedBy ?sensorID.
    ?sensorID a m3-lite:SoilHumiditySensor.
    ?observ ssn:observedProperty ?qkr.
    ?qkr a m3-lite:SoilHumidity.
    ?observ geo:location ?point.
    ?point geo:lat ?lat.
    ?point geo:long ?long.
    ?observ ssn:observationResult ?output.
    ?output ssn:hasValue ?value.
    ?value dul:hasDataValue ?dataValue.
}
```

- In order to have a optimal SPARQL and minimize the checks the triple store needs to do, it is often possible to omit the type check of the sensor. In the example, we know in advance that all sensors of type “m3-lite:SoilHumiditySensor” have the quantityKind “m3:soilHumidity”, so we can omit the sensor type check:

```
Prefix ssn: http://purl.oclc.org/NET/ssnx/ssn#
Prefix dul: http://www.loa.istc.cnr.it/ontologies/DUL.owl#
Prefix geo: http://www.w3.org/2003/01/geo/wgs84_pos#
Prefix m3-lite: http://purl.org/iot/vocab/m3-lite#
Prefix xsd: http://www.w3.org/2001/XMLSchema#
Prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
select distinct ?lat ?long ?dataValue
where {
    ?observ a ssn:Observation.
    ?observ ssn:observedProperty ?qkr.
    ?qkr a m3-lite:SoilHumidity.
    ?observ geo:location ?point.
    ?point geo:lat ?lat.
```

```
      ?point geo:long ?long.
      ?observ ssn:observationResult ?output.
    ?output ssn:hasValue ?value.
        ?value dul:hasDataValue ?dataValue.
}
```

- It is worth to note that if a limit of number of observations to be returned within the SPARQL of the FEDSpec is specified, and the SPARQL is doing an historical query with the time interval inchanged, then the resultset might be always the same.

3.3 Security system

3.3.1 Authorization

- There are many reasons for which a call to an API resource results to a “401 Unauthorized”.
  - A first reason might be that the token is not valid. In order to check its validity, try the following:

    ```
    curl    --verbose    --request    POST    --header    "Content-Type:
    application/json"    --header    "iplanetDirectoryPro:    <TOKEN>"
    https://platform.fiesta-
    iot.eu/openam/json/users?_action=idFromSession
    ```

    and check whether an HTTP 200 OK is returned.

  - Another reason might be that the requested resource does not allow certain HTTP method. Most of the resources exposed by the API allows only the GET method, used for retrieving data. One exception is the “`queries/execute/{resource/observations/global}`” resources, to which it is meant to be sent a SPARQL query. Check the training platform for more information on the API[1]
  - A “401 Unauthorized” might be returned also when a brand new token is provided. This might be due to cookies, for example Postman makes use of the cookies and it might happen that an old and not valid cookie is used for IoT-Registry call even if a new token has been provided within the header. In this case it is better to remove the cookies[2]

3.3.2 Authentication

- The FIESTA-IoT authentication API does not allow cross-origin requests (CORS) from browser based scripts, and javascript technologies like Ajax will then not comply with this security policy. It is better (in the first instance) to use an alternative method to authenticate and get a token via the API, i.e. a method not running in the browser. Another approach is to develop a proxy. That is a service that runs the API calls and the browser client then interacts with this service.

[1] http://moodle.fiesta-iot.eu/mod/book/view.php?id=104&chapterid=42

[2] https://www.getpostman.com/docs/postman/sending_api_requests/cookies.

3.3.3 Session

- To prevent exhaustion of server resources, users are only allowed a fixed number of sessions. Once over this limit is reached, a previous session is selected and invalidated. When handling with multiple sessions from the same account, each logged-in session should be logged out after completion of use.
- Removing the token with a logout action is not mandatory but highly benefitial. It is helpful to make the logout request if the token will not be used for a period of time.

  The call that to be executed is as follows:

```
POST /openam/json/sessions/?_action=logout HTTP/1.1

Host: platform.fiesta-iot.eu
Content-Type: application/json
iPlanetDirectoryPro: AQIC5wM2LY4SfcwEw_Jg.........

{}
```

  The body of the POST request can be an empty JSON document and you only have to include the header with the token you want to remove.

3.3.4 Resources Access

- In order to protect the system for malignous misusage, all the resources within FIESTA-IoT platform are protected by the authentication procedure. It is not possible to do any HTTP call to any of the resources, without executing the authentication procedure.
- It is not possible to access testbeds resource with an experiment account, only a testbed administrator role can access to https://platform.fiesta-iot.eu/iot-registry/api/testbeds resources
- When accessing the portal a blank page might appear. This is an authentication problem. First thing is to be sure to have granted some role within the FIESTA-IoT platform. If this condition is satisfied, then often the problem resides on an old invalid cookie. It is worth to try to clear cookies from the browser.
- Authentication is a key element of the whole FIESTA-IoT platform, thus any request need to carry a valid authentication token. It is better to check the FIESTA-IoT training material[3] that contains complete information about the FIESTA-IoT Platform security framework. Note that before being able to access any of the FIESTA-IoT Platform services (e.g. AaaS) it is mandatory to pass through authentication and authorization.

3.3.5 Accounts and roles

- The “testbed admin” role is not granted if certain milestones are not passed, such as being able to correctly annotate observations and resource descriptions, as well

[3] http://moodle.fiesta-iot.eu/mod/book/view.php?id=106

as to have interoperable TPS. Checking the FIESTA-IoT training tool[4] for more information might be handful.

- The FIESTA-IoT platform and the FIESTA-IoT ticketing system are using two disjoint Identity Manager servers. For that reason in order to use the ticket system it is necessary to create a new account on it (the same username and password can be used for your convenience). The advantage to use the ticketing system is to more easily track the status of the issues.

3.4 Data Access

- Sensors produce data on their own schedule. It is not possible to act on that parameter (i.e. controlling the time between two consecutive observations from a sensor).
- An example for making a SPARQL to the IoT-Registry can be found below (the {{iPlanetDirectoryPro}} needs to be set accordingly).

```
curl -X POST \
http://localhost:8080/iot-registry/api/queries/execute/global \
-H 'accept: application/json' \
-H 'cache-control: no-cache' \
-H 'content-type: text/plain' \
-H 'iplanetdirectorypro: {{iPlanetDirectoryPro}}' \
-d 'Prefix ssn: http://purl.oclc.org/NET/ssnx/ssn#
Prefix iotlite: http://purl.oclc.org/NET/UNIS/fiware/iot-lite#
Prefix dul: http://www.loa.istc.cnr.it/ontologies/DUL.owl#
Prefix geo: http://www.w3.org/2003/01/geo/wgs84_pos#
Prefix time: http://www.w3.org/2006/time#
Prefix m3-lite: http://purl.org/iot/vocab/m3-lite#
Prefix xsd: http://www.w3.org/2001/XMLSchema#
Prefix rdf:  http://www.w3.org/1999/02/22-rdf-syntax-ns#

select ?lat ?long ?dataValue ?time ?sensorID
where {
    ?observ a ssn:Observation.
    ?observ ssn:observedBy ?sensorID.
    ?observ ssn:observedProperty ?qkr.
    ?qkr rdf:type m3-lite:AirTemperature.
    ?observ geo:location ?point.
    ?point geo:lat ?lat.
    ?point geo:long ?long.
    ?observ ssn:observationResult ?output.
    ?output ssn:hasValue ?value.
    ?value dul:hasDataValue ?dataValue.
    ?observ ssn:observationSamplingTime ?t.
    ?t time:inXSDDateTime ?time.

    FILTER (
        (xsd:double(?lat) >= "4.2E1"^^xsd:double)
        && (xsd:double(?lat) <= "4.8E1"^^xsd:double)
        && ( xsd:double(?long) >= "-3.9E0"^^xsd:double)
        && ( xsd:double(?long) <= "-3.6E0"^^xsd:double)
        )
}
```

[4] http://moodle.fiesta-iot.eu/mod/book/view.php?id=96&chapterid=93

```
order by desc(?time)
limit 10'
```

Other examples can be found on the FIESTA-IoT training platform[5] and curl command generated from the Postman tool[6]

- It is not possible to request data coming from a particular sensor by specifying the sensor identifier valid within the domain of a specific deployment. This is due to the fact that sensor identifiers are transformed into FIESTA-IoT namespace (the process of transformation involves a hashing mechanism so the identifiers are not human friendly). In general when an experimenter approaches FIESTA-IoT Platform it should be made in a testbed-agnostic way. This is not looking, for instance, for SmartSantander devices but for FIESTA-IoT devices (which includes devices from various testbeds).
  Discovery of sensors in the majority of the cases is based on their position or their phenomena that they observe.
- Due to system overload, it is possible that the execution of SPARQL queries is responded with a server error (i.e. 500 HTTP responose code). When developing an experiment, it is important to take this into account and retry the query until it is successfully answered.
- Some of the sensors and actuators available are exposed by services (typically RESTful based services) which let the experimenter get the observations that they have produced. When the nature of the experiment allows it, accessing data using these services is always more efficient than executing a SPARQL query that would provide the same information.
- New releases of the IoT-Registry have disabled (by default) the inference engine. The reason behind this is that using the inference feature of the semantic engine of the IoT-Registry by default makes all queries, even those that would not need it, to take much longer (at least two or three orders of magnitude) making a typical query which is now being resolved in some few seconds to take some minutes. Nonetheless, there is a way to keep using this engine.

When invoking a service, this needs to be added ?inference=true as a query string in the URL.

For the moment inference will remain disabled unless there is no other way to get the requested information.

In this sense, the first thing that is necessary is to exactly know which is the information to discover.

- In this respect, even if inference is not enabled, the queries can be adapted to get the same result. The key idea is to use the Object Properties within the SPARQL. For example, when interested in all the Sensing Devices the Object Properties referred to this class can be leveraged. By adding into the SPARQL the following statement *?sensor iot-lite:hasQuantityKind ?qk.* it will find all the nodes in the graph that has this object property. These nodes will be precisely all the sensing devices. There are different solutions exploiting the same concept. For example:

```
?o a ssn:Observation.
?o ssn:observedBy ?sensor.
```

[5] (http://moodle.fiesta-iot.eu/pluginfile.php/666/mod_folder/content/0/Material/Postman/OC1-TS-iot-registry-api.json?forcedownload=1

[6] https://www.getpostman.com/docs/postman/sending_api_requests/generate_code_snippets

```
?sensor iot-lite:hasQuantityKind ?qk.
```

A SPARQL with these statements will gather sensing devices that had produced an observation (so not only registered sensors but active registered sensors).

3.5 FIESTA-IoT tools

3.5.1 AaaS (Annotator as a Service)

- The AaaS is a service provided by the FIESTA-IoT platform for easing the annotation of your testbeds, resources and observations. Good documentation can be found within the FIESTA-IoT training platform[7]
- In order to optimize the traffic with the AaaS, it is possible to send multiple observations as an array, like following:

```
    {
        "observations": [{
            "observed_by": "http://device01",
            "location": {
                "lat": 1.5,
                "lon": 1.2000000476837158
            },
            "quantity_kind": "qk1",
            "value": "value1",
            "format": "float",
            "unit": "uom",
            "timestamp": "2017-07-27T07:29:00.66Z"
        }, {
            "observed_by": "http://device02",
            "location": {
                "lat": 1.5,
                "lon": 1.2000000476837158
            },
            "quantity_kind": "qk2",
            "value": "value2",
            "format": "float",
            "unit": "uom",
            "timestamp": "2017-07-27T07:29:00.66Z"
        }]
    }
```

- All the available output types (set with the accept header) for the basic AaaS are available: *application/ld+json; application/rdf+xml; application/n-triples; application/n-quads; application/turtle; text/plain; text/csv; text/rdf+n3; application/n3; text/n3; text/n-quads; text/nquads*

  It is good to note that for certain output types, some constraints are needed to be followed, like application/rdf+xml requires to have the observed_by field set with a valid IRI.If the values of a JSON request to the AaaS is containing some errors (like typo) but all the field names are correct, the response from the AaaS might still be rejected, with an error, by the testbed component. It is better to use the

[7]http://moodle.fiesta-iot.eu/pluginfile.php/672/mod_folder/content/0/Presentations/01-FIESTA-TPS_and_AaaS.pdf?forcedownload=1

Ontology Validator in order to be sure that everything is in the correct form, also the response from the AaaS.

- The timestamp is a required field and cannot be omitted within the RDF of the observations and it cannot be omitted when requesting the AaaS for creating the RDF.
- The attribute "type" is part of the description of the sensing devices (as "qk" or "uom"). This attribute is not compulsory and if it is not included the sensor is annotated as ssn:SensingDevice (i.e. the generic Class for the sensors).
  This is perfectly correct, that's why letting the testbed providers decide whether or not to go the extra mile and set this "type" attribute in order to make the annotated resource description more specific.
  When using the "type" attribute the correct class of the M3-Lite taxonomy needs to be used. For example, for a sensor measuring the temperature of the water "type":"WaterThermometer" should be used, similarly, for a sensor measuring ReactivePower "type":"EnergyMeter" is the right one. In case there is not an appropriate class in M3-Lite (note that it has to be subClass of SensingDevice), simply communicate with the FIESTA-IoT team.

3.5.2 EDR (Experiment Data Receiver)

- The Experiment Data Receiver exposes only the "ExperimentServer/store/" resource.

3.5.3 Postman Collection

- The collections are often using variables like {{path}} or {{iPlanetDirectoryPro}} that need to be set. In this sense, the values can be directly hardcoded by replacing them with appropriate value (i.e. {{path}} as https://platform.fiesta-iot.eu). But it is handier to have a look at the Postman documentation[8] where it is explained how to use variables in collections and how to automate things. This is of great help for automatically set the OpenAM Security Token in all your requests.

3.5.4 Ticketing System

- The FIESTA-IoT platform and the FIESTA-IoT ticketing system are using two disjoint Identity Manager servers. For that reason in order to use the ticket system it is necessary to create a new account on it (the same username and password can be used for your convenience). The advantage to use the ticketing system is to more easily track the status of the issues.

3.5.5 TPI Configuration View

- When a resource is scheduled it is removed from the available resource list of the first tab. So if the resource is not visible it is worth to check the other tabs.

[8] http://blog.getpostman.com/2014/02/20/using-variables-inside-postman-and-collection-runner/

3.6 Testbed integration

- As directly experienced, in order to not overload the system, there are a couple of expedients for testbeds that have implemented the get-based TPS:
    - As far as possible, do not set frequencies smaller than 5 minutes.
    - In case of a long list of devices, try to split them in several schedules. If the TPS takes too long to compile the response to the getLastObservations query, the connection is timed-out by the platform. The timeout is set to 60 seconds. Typically, the time that the TPS takes to answer the request is proportional to the number of sensorIDs included in the request. Thus, it is better to avoid including large number of sensors per schedule. The actual strategy used to split the sensors in different schedules is left to the testbed providers (e.g. one schedule per type of sensor) as they should select the way that ease the management that they might have to do internally.
- A testing virtual machine, as playground, is available for easing the integration of testbeds. The system is a mirror of the one that you will have on the production machine so it should put you in the best position to, once gotten familiar with it, make the definitive integration on the real system that FIESTA-IoT experimenters are actually using. The playground will only be opened to testbed providers so that there are no worries about messing things up and the system can easily be reset without many collateral damages. All in all, the testbed providers need to simply go to the playground area[9] and start interacting with the system to train themselves on the tasks and procedures defined to integrate a testbed in the FIESTA-IoT Platform. In terms of security credentials, the same username and password can be used since the playground is using the same user management system as the production machine. Indeed, the security tokens to be used on playground.fiesta-iot.eu have to be obtained from platform.fiesta-iot.eu.
- The taxonomy of the FIESTA-IoT ontology can be always expanded in order to cover all the possible IoT testbeds. In order to do so we invite to contact us by specifying, for each of the sensor you think not matching any of the already available quantity kind, the following:

  - sensor name
  - sensor description
  - subclass of
  - Quantity Kind (QK)
  - QK description
  - subclass of

  For example:
  - sensor name : Oxidation Reduction Potential (ORP) Sensor
  - sensor description: Measures the Water Oxidation Reduction Potential (ORP) is the tendency of a chemical species to acquire electrons and thereby be reduced.
  - subclass of: ssn:SensingDevice
  - Quantity Kind (QK): Voltage
  - QK description: An electromotive force or potential difference expressed in

[9] https://playground.fiesta-iot.eu

Volts
- subclass of: Quantity Kind

- The FIESTA-IoT Platform works with stored data as linked data. The IRIs of the device are their identifiers and then we have links among them which are traversed to discover them. When registering devices it is mandatory that each device has a different IRI. If a device is registered using the same IRIs of a previously registered one, then when the registration is made, no new node is actually registered in the linked graph which is the FIESTA-IoT Platform repository. However, registering a device using the same IRI but indicating the hasDeployment to a different testbed IRI causes the addition, not the replacement, in the repository. The device will then “belong" to the two testbeds.
- When using the AaaS for annotating multiple observations, the TPS should not call the AaaS for each of the observations retrieved from the internal testbe service. The correct way for handling this on the TPS when receiving the getLastObservations request is to iterate over the list of sensorIDs included in the request, and for each of them obtain the observation from your internal testbed service and store it on an array. When it finishes going from the first to the last in the list, takes the resulting array and call the AaaS API for annotating the whole list of observations. The result is then put in the response.
  The output of the AaaS will result in a correct JSON-LD:
  ```
  { "@graph":[{observation from device1}, {observation from device2}, ...,
  {observation from deviceN}],"@context":{}}
  ```
  This will also minimize the calls to the AaaS API.
- If a sensor has not produced a new observation since the last time it was queried (e.g. it is broken or polling frequency is faster than observation generation), then the already provided observation should not be reported again. A different behaviour might harm the production platform.
- Using a fixed template is not best approach since semantics are something more than a fixed structure. Thus, the option of "annotating by placeholders" can probably make the work but it might cause issues in the future/
  The best is to use actual semantic libraries so that annotations can be modelled directly from the ontology. An easier approach is to make usage of the AaaS.
- It is better to register and send observations into the playground by, at least, one sensor from each of the types .Even if the larger the tests in the playground the better since hidden problems will probably be discovered, typically it is not necessary to register ALL resources (mostly if they are a really large number).

4 BEST PRACTICES FOR EXPERIMENTS AND TESTBEDS INTEGRATION ON FIESTA-IOT PLATFORM

In this section we list best practices an experimenter needs to follow to perform successful experimentation on FIESTA-IoT metacloud.

The best practices are based on the multi-channel feedbacks, lessons leanrt and interactions with OC participants, which gives to the FIESTA-IoT technical team more insight of the platform users’ perspective. These best practices mainly span the best practices an experimenter need to follow to create the FEDSpec and write efficient queries. Additionally, the key aspects that testbed providers should observe during the integration of their testbeds are also described.

4.1 Best practices for FIESTA-IoT experimenters

4.1.1 Creating the FEDSpec

To consolidate the previous section, we encourage the experimenters to follow the best practices. These include:

- Please do not use the template/provided FEDSpec's as it is. The provided template is a generic template. We further encourage you to remove all #*# and also the properties that are not used. For the properties you are using, change #*# with the actual values you want to use. The data format of the values you provide should be same as these reflected in the examples. Further, use logical values. Please do not use `Test`, `Blah Blah`, etc, in the description and other places.
- Have userID specified in the FEDSpec. You should modify `#USERID#`. Your FEDSpec will not be saved if `#USERID#` is different from the `ACCESSTOKEN` used.
- `fed:startTime` should be less than `fed:stopTime`. Ideally, the difference between `startTime` and `stopTime` should not be 1 sec. If this is the case, this will only allow the experiment to be executed once in the lifetime. If this is a needed functionality we recommend to do so. However there is another option. Instead of scheduling the experiment on the Execution Engine, experimenters can use the Polling option available via the Management Console. To avail Polling service, experimenters will first have to deploy the FEDSpec, but are not supposed to "START" the execution of a particular FISMO. By not starting the execution, the FISMO status remains "NOT YET SCHEDULED".
  If only one execution is not the needed functionality please be realistic in the difference. Do not set `#PERIODICITY#` to 1 second for the queries. This will only overload both our as well as your system, block the network without providing any insight. As a best practice we recommend you to use periodicity of not less than `600`.
  EEE always queries the global graph. This means, from the set of time dependent observation graphs available within FIESTA-IoT federation, the most recent observation graph will be queried. Thus it is important to note that the `fed:startTime`, `fed:stopTime` and `fed:Periodicity` are accordingly defined and are within 6 hours time limit.
- If you use dynamic query parameters please use them as specified for the query, i.e., using **`%%fromDateTime%%`**, **`%%toDateTime%%`**, **`%%geoLatitude%%`**, and **`%%geoLongitude%%`**. For other cases, other parameters used in dynamic query, use **`%%DA_NAME%%`**. Note that the DA_NAME correspond to the dynamic attribute.
- Before writing the queries we strongly encourage you to have a look at the documentation of the kind of data that is currently provided and what is the semantic structure of it. We recommend you to write queries that would serve your purpose and are efficient. Please DO NOT:
  - Write queries that are irrelevant for your experiment. This will have overloading effect on your Experiment.

- Use overly complex queries for simple cases unless no other solution.
  We would be happy to help you writing efficient queries. However, as an experimenter we expect that you first write queries and we validate them. Additionally do not use “Select * Where {?s ?p ?o.}” query, this goes against our principles and shall be completely prohibited. More details on the best practices for efficient querying is provided in the next section. Further, in case of specific queries, if you have something with “PARAMETER”, we recommend you to use \”PARAMETER\”. Note the \”. This is done to correctly parse the query when invoking IoT-Registry APIs.

- Note that the Execution Engine uses `UTC timezone`. Thus all your queries should be using `UTC timezone`. Moreover, the `fed:startTime` and `fed:stopTime` should also be specified in `UTC timezone`.
- Sync the `#PERIODICITY#` and the used time interval in the query. If not synchronized, it is possible that you get redundant data that is not useful. Generating redundant data will again block resources and will be not efficient.
- If you use dynamic query and use `fed:IntervalNowToPast`, make sure that this quantity is not more than 6 hours. As said before, while using EEE, only the latest observation graph is queried when using global endpoint, it does not make sense to ask for data that is beyond this 6 hour time frame.
- Please test the `#URLLOCATION#` before providing it. Please make sure that the URL is operational and accepts incoming data. To help experimenters, a sample code is provided that opens a URL that accepts incoming data and stores it in a file. Note that, as EEE is operating in an ASYNC mode, the result when available is sent to the experimenter. An experimenter should periodically check if the data is available on their server or not. A very simple solution to not write a periodic checker is to get the file that is most recently created. All the files that are sent are time-stamped. The naming convention followed to name the such files is

```
String filename=JOBID.replace(“-”,””)+URLLOCATION.replace(":",
"").replace("/", "_")+”_”+LONG_TIMESTAMP;
```

  Note here JOBID is a UUID that is set by EEE, URLLOCATION is the location that you provide and LONG_TIMESTAMP is milliseconds after epoch.
- In case you are executing the experiment using FEDSpec, you need to install Experiment Data receiver module on your server. This module will enable a valid `#URLLOCATION#` that can be utilized in the FEDSpec. The Experiment Data receiver installation guide is available at the following link

```
https://github.com/fiesta-
iot/experiment.data.receiver/blob/master/ExperimentServer/Readme.md
```

  Please note that the module is currently tested on Tomcat and creates a `#URLLOCATION#` that looks like `http(s)://HOST:PORT/ExperimentServer/store`. In case you are using HTTPS please make sure that you are using LetsEncrypt/JVM Already trusted Certificates/Terena SSL root certificate. In case any other certificate is used the Experiment Execution Engine will not be able to send the resultset to `#URLLOCATION#`. Self-signed certificates will not work. In case you use HTTP, there is no such restriction. The priority is given to the link that is specified. In case this URL is not reachable for some reason, the results are stored locally within

FIESTA-IoT repositories for experimenters to later download the results. Note that the experimenters have to create their own java code to download the results.
In case you decide to directly call the IoT-Registry APIs, you need not install this component. Nevertheless, you will have to deal with authenticating yourself every time you are calling IoT-Registry APIs, scheduling your experiment with in SYNC periodicity and time interval used in the query (Still you need to follow #3 above).

- In the `fed:presentationAttr value` of the `widget` attribute please follow the guidelines as said in the section above. The value should be a JSON string that is of form:

```
{
 &quot;Method&quot;: [&quot;fft&quot;, &quot;linReg&quot;],
 &quot;Parameters&quot;: [&quot; &quot;, &quot;Predict&quot;]
}
```

- Send your FEDSpec and the queries (for advanced experimenters) that you intend to execute to the support channel, get it validated before you proceed with registering the FEDSpec into the system or using the query and directly calling the IoT-Registry API. In case you want to test your queries, before providing them (in the FEDSpec and scheduling them, giving it to us to validate) you can still validate the queries by yourself. Note that here ve refer to just getting a feel of what data is available and if the syntax of the query is correct. Based on our recommendations about your experiments you need to decide if creating a FEDSpec is the best option of directly using the IoT-Registry is best for you.
  Note that there are 2 aspects that we mentioned before validating the syntax and getting a feel of the data. You can validate the syntax of the query using http://www.sparql.org/query-validator.html. This will ensure that your query is executable on the FIESTA-IoT Platform. To get a feel of what the query will return you can use any REST Client (Postman, Google Chrome's Advanced REST Client, Curl, etc.) to send your request to the FIESTA-IoT platform. There are 3 main APIs that you need to deal with:
    - https://platform.fiesta-iot.eu/openam/json/authenticate
    - https://platform.fiesta-iot.eu/iot-registry/api/queries/execute/observations
    - https://platform.fiesta-iot.eu/iot-registry/api/queries/execute/resources

The Authenticate API will give you an access token that you will have to use to call the other 2 APIs. Using these APIs you can execute your query and test them. Note that the APIs used should be based on which graph you want to query.

4.1.2 Writing efficient Queries

We list below the best practices to query the system as well in order to know what kind of data is currently present in FIESTA-IoT repository.

- If you run first a resource discovery, you can harness the gathered sensor information to save much time in further queries (i.e. observations based on node ids).
- If your experiment aims at short-term data (not historical values), another thing that can save time is the usage of IoT-Service endpoints instead of raw SPARQLs queries. However, please note this concept is not available for all the sensors. It can be used wherever provided. In the ontology it is represented via `iot-lite:Service`.

- In case you do not need the complete graph structure, please do not query for all the concepts and properties.
- The entire structure is divided into 2 graphs: Resource graph and observation graph. The queries must be directed to either of these graphs based on the requirements. Query these graphs is resource efficient. In case both graphs are to be queried, you must use another graph called “global”. We recommend you to not query this graphs unless it is essential.
- Observations are stored in so-called graphs as they come into the system. There is always a "current" graph which is the one to which observations are sent as they arrive. However, every hour, the "current" graph is packed and stored with a tag coming from the moment on which it was created, and another "current" graph is newly created.
  When you make a query on /iot-registry/api/queries/execute/global or /iot-registry/api/queries/execute/observations your query is resolved against the "current" graph. Thus, if you make a query just after the "current" graph has been created, probably you will have little results (or even no results matching your query), as more observations are stored in the "current" graph, subsequent queries will have more results. However, when the "hour" comes, the "current" is stored and a new empty current graph is created. The cycle starts again.
- For accessing observations in the past you have to use the "from" and "to" query parameters as it is described in the Handbook (FIESTA-IoT, Handbook for experimenters and extensions, 2018)
- By using the “from” and “to” query parameters the query is resolved against the observations that came into the system on the defined time period. While the FIESTA-IoT Platform allows a maximum of 6 hours for this time period, it is highly recommended that smaller periods are used. It is better to make several queries on consecutive periods spanning the portion of time in which the experiment is interested than making one single query on the whole interval. The aggregated response times is typically smaller.
- Learn and understand the meaning of the `FILTER, GROUP, LIMIT, BIND` etc. options, and try to use them if possible. Note that adding such keywords slows down the query execution. For example, bind significantly degrades the execution time.
- The VALUES option of the SPARQL language <REF SPARQL> helps increasing the efficiency of the query and significantly reduces the response time of your queries. Typical examples where this option can be used is to specify the phenomena and/or sensors that the query is interested in.
- It is important to optimize the query and not ask for any concepts in the WHERE clause that is not necessary and has no impact on the selected parameters.
- Try, to the extent possible, to avoid the extraction of large amounts of data at once (e.g. >5MB). In this case, split your queries into various ones; for instance, sweeping the time into small windows, etc.
- There is a possibility that the `dul:dataValue` returned is a `NaN`. This `NaN` is mainly reported currently by the SmartICS testbed. Thus, it is useful, in case you do not want to receive observations that have `NaN` value to filter such observations. In FIESTA-IoT, currently some observations have `dul:dataValue` as `NaN` while some have `dul:dataValue` as `NaN^^xsd:string`. Note the absence of data-type in the first case.
  A filter command looks like

```
FILTER (?dataValue != NaN^^xsd:string || ?dataValue != NaN)
```

The `dul:dataValue` currently can return following datatypes `xsd:int`, `xsd:double`, `xsd:dateTime`, `xsd:boolean`, `xsd:string`. Thus it is of utmost importance that experimenters look into what kind of data they need and understand the mappings between QuantityKinds, Units and datatypes.

- Each sensor/resource has EXACTLY one QuantityKind and Unit associated to it. Please refer to Testbed documentation (link below) to understand what phenomenon is being mapped to which m3-lite concept.
    - See (FIESTA-IoT, FIESTA-IoT testbed resources, s.d.) for list of Sensors, QK and Units provided by the testbeds.
    - Some other relevant documents are available at:
        - SPARQL Query Language for RDF online documentation.
        - Material of the first training workshop for experimenters
- IoT-Service endpoints are a good deal when it comes to get the last observations carried out by the sensors. However, there is a number of points that has to be considered beforehand;
    - **FIESTA-IoT does not specify** the format of the response messages (in the current version of the platform). This means that every testbed might follow a different data set. Thus, experimenters have to manually parse them.
    - It is worth highlighting again that **these services are not mandatory for testbeds**, so they might (or might not) decide to include these endpoints as part of the resource description. Indeed, up to today (1st Aug 2017), 2 out of 4 testbeds in the federation (i.e. SmartSantander and SmartICS) do offer this possibility.
    - Even though there is a kind of de-facto agreement on the actual use of the endpoints, that is, to expose the last measurement observed by a node, this is not an official standard. Consequently, **testbed providers might use the endpoints for a different purpose**.

FIESTA-IoT testbeds are deployed[10] on different geographical areas: i.e. Spain (SmartSantander), UK (SmartICS) and South Korea (KETI). Nevertheless, SoundCity testbed does not have any specific location as this testbed uses crowdsourced information coming from uses located in different locations around the globe. Furthermore, more testbeds will gradually the federation.

4.1.3 Using added-value tools available

- Use query templates made available within the FIESTA-IoT Platform.

4.2 Best practices for integrating a testbed in FIESTA-IoT

The integration of a testbed within the FIESTA-IoT Platform can be basically structured in the following steps: 1) Testbed and FIESTA-IoT taxonomy alignment; 2) Develop your annotator and Testbed Provider Services (TPS); 3) Get FIESTA-IoT certified; 4) Register your testbed and resources; 5) Configure your resources.

[10] SoundCity testbed is actually a crowdsensing platform and is not bound to a particular physical location.

These best practices, which have been elicited from the experienced gained during the integration of the 11 testbeds that are currently federated within the FIESTA-IoT Platform, are listed following the same sequence of steps. In this sense, most of these best practices have been derived by reversing the most common doubts and mistakes made by the testbed providers coming from the Open Calls while they were integrating their testbeds. Some others have been extracted from the suggestions made by the same testbed providers in the reports that they wrote to describe the work that they performed and to evaluate their experience during the integration of their testbeds.

4.2.1 Aligning the internal data model to the FIESTA-IoT Ontology

- Consult the on-line documentation about the FIESTA-IoT Ontology and find matches for your sensors' types, phenomena that these sensors observe and their unit of measurement.
- Exploit the hierarchical nature of the FIESTA-IoT Ontology for proposing new Quantity Kinds, Sensor Types and Units of Measurement that should be defined as subclasses of already existing ones.
- Design the adaptation between the information model that the testbed uses internally for representing devices and observation, and the FIESTA-IoT Ontology. It is critical to understand how the internal concepts matches with the FIESTA-IoT Ontology classes.

4.2.2 Annotating and sending data to to the FIESTA-IoT Platform

- Making use of the Annotator as a Service your annotated documents will benefit from a service that will also fit with the latest version of the FIESTA-IoT Ontology.
- Select the appropriate mode of operation of your TPS depending on the original interfaces offered by your testbed. If your testbed exposes a synchronous interface, then the get-based mode of the TPS is better suited. If your testbed exposes an asynchronous interface, then the push-based mode would be preferable.
  If the testbed has a mixture of interfaces, then it is better to implement the two modes and tailor the integration during the configuration step.
- In the case of the get-based TPS, make sure that the same observation is not sent multiple times. This might happen if your TPS is polled for the latest observation from a sensor and that sensor has not produced any new observation since the last time it was polled. In this situation, the TPS should not send the observation again.

4.2.3 Getting FIESTA-IoT Certified

- Make use of the tools (i.e. ontology validator and interoperability testing) at the Certification Portal as many times as necessary. These tools will validate the annotated documents and TPS interfaces and provide you with valuable feedback during the development process.

4.2.4 Registering your testbed and resources

- When generating the annotated resource descriptions it is important to provide as much information as possible from your devices as it will be useful to experimenters that discover them

- For all your devices it is compulsory to include the corresponding information about the testbed to which they belond, its location, and the phenomenon that they observe together with the unit of measurement that it uses.
- While the FIESTA-IoT Platform Portal provides a web form that makes the registration of devices straightforward, for large testbeds the best option is to use the AaaS to generate a document with all your devices descriptions and upload this document directly through the corresponding option in the FIESTA-IoT Platform Portal.
- After registering your devices, follow the steps described in the FIESTA-IoT Testbed Providers Handbook to confirm that the process have been completed successfully and that your devices are discoverable.

4.2.5 Configure your TPS for sending observation to FIESTA-IoT Platform

- Cluster your sensors in different schedules in such a way that helps you handling the operation afterwards. This means cluster them by phenomenon observed, or by mimicking the internal hierarchy within your testbed instead of doing it randomly.
  The larger your testbed is the more important this recommendation is.
- For get-based TPS:
  - Make the frequency at which you configure FIESTA-IoT Platform to poll for observations to match the frequency at which your sensors produce observations. It does not make sense to have a polling period which is smaller than the observation generation interval.
  - Unless required due to the nature of the phenomenon observed, a 15 minutes polling frequency is recommended.
  - Split your sensors in different schedules in case the number of sensors is higher than 100. This is not a hard limit but the observation storage process at the FIESTA-IoT Platform backend degrades with the amount of observations that are stored in the same transaction. 100 is a rough estimate that makes the system works in optimal conditions.
- For push-based TPS:
  - Take into account the pace at which your TPS might be pushing observations to the FIESTA-IoT Platform and avoid excessively frequent push of data towards the Platform. Two situations have to be particularly observed:
    Large testbeds: Even when the data generation frequency of your devices, individually, is small, the fact that there are many different sensors cause that your TPS is constantly pushing observations.
    Bursts of observations: When sensors generate bursts of observations with high frequency during limited periods of time.
    In this case, your TPS should internally digest observations and instead of pushing each individual observation independently, only push observations towards the FIESTA-IoT Platform after a number of them have been collected (100 is a recommended number) or after a fixed amount of time has passed since the last push (5 minutes is a recommended period).

5 CONCLUSION

This deliverable summarizes the effort that the FIESTA-IoT consortium have made to help experimenters to efficiently get started with the semantic platform and design, deploy and execute their innovative experiments upon the platform. A rich handbook is created for this purpose with plenty of examples to complete the development guide.

Feedbacks and lessons learnt from executed experiments are gathered from different interactive channels to be presented all in one single document. Best practices are created from these feedbacks and lessons learnt by the FIESTA-IoT technical team from the perspective of the external platform users in order to help future experiments to increase their efficiency. These informations aim to point out a clear way to a successful usage of the platform to the users at different stages of knowledge of the platform (novice, intermediate and advanced). These informations are not only useful for the use of FIESTA-IoT platform, but also for semantic data discovery and retrieve in general, for security mechanism for an IoT platform in general.

6 BIBLIOGRAPHY

FIESTA-IoT. (2018). *Handbook for experimenters and extensions.* Retrieved from http://moodle.fiesta-iot.eu/pluginfile.php/711/mod_resource/content/5/FIESTA-IoT_Handbook4ThirdParties_v4.0.pdf