Tailored IoT & BigData Sandboxes and Testbeds for Smart, Autonomous and Personalized Services in the European Finance and Insurance Services Ecosystem

Infinitech

# D4.6 – Semantics Streams Analytics Engine III

| | |
|---|---|
| **Lead Beneficiary** | NUIG |
| **Task Reference** | T4.2 |
| **Revision Number** | 3.0 |
| **Deliverable Type** | Report (R) |
| **Dissemination Level** | Public (PU) |
| **Due Date** | 2021-12-31 |
| **Delivered Date** | 2022-04-30 |
| **Internal Reviewers** | GFT, NUIG-Insight |
| **Quality Assurance** | INNOV |
| **Review Status** | Internally Reviewed and Quality Assurance Reviewed |
| **Acceptance** | WP Leader Accepted and/or Coordinator Accepted |
| **EC Project Officer** | Beatrice Plazotta |

HORIZON 2020 - ICT-11-2018

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement no 856632

# Contributing Partners

| Partner Acronym | Role | Author(s) |
|---|---|---|
| **NUIG** | Lead Beneficiary | Eoin Jordan<br>Yasar Khan<br>Martin Serrano<br>Alex Acquier |

# Revision History

| Version III | Date | Partner(s) | Description |
|---|---|---|---|
| 2.1 | 2021-09-20 | NUIG | ToC Version III reviewed |
| 2.2 | 2021-09-25 | NUIG, N OVA | ToC with Final APIs and Implementation |
| 2.3 | 2021-10-10 | NUIG | TOC contributions on Extensions |
| 2.4 | 2021-10-15 | NUIG | Docker version Integration of CI/CD Process and implementation |
| 2.5 | 2021-11-20 | NUIG | Semantic Engine Documentation<br>Final version re-organised |
| 2.6 | 2021-12-25 | NUIG | Executive Abstract Final version, final edits on Section 3, Section 5, and Section 6 |
| 2.7 | 2021-02-15 | NUIG | Section 2 and Section 4 Comments and Section 1 and Section 7 final edits |
| 2.8 | 2021-04-10 | NUIG, INNOV | Pre-Final Version for Internal Review |
| 2.9 | 2021-04-30 | NUIG, GFT | Version with Quality Assurance |
| 3.0 | 2021-04-30 | NUIG | Version for Submission |

| Version II | Date | Partner(s) | Description |
|---|---|---|---|
| 1.1 | 2021-03-10 | NUIG | ToC Version reviewed |
| 1.2 | 2021-03-15 | NUIG, N OVA | Updated ToC with Improved APIs and first Implementation |
| 1.3 | 2021-04-10 | NUIG | Updated TOC contributions |
| 1.4 | 2021-04-10 | NUIG | Integration of CI/CD Process<br>and implementation |
| 1.5 | 2021-05-15 | NUIG | Additional contributions on<br>Semantic Engine Documentation |
| 1.6 | 2021-06-20 | NUIG | Updates in Executive Abstract, Section 4, Section 5 and Section 6 |
| 1.7 | 2021-07-15 | NUIG | Contributions in Section 5 and Section 6 |
| 1.8 | 2021-07-20 | NUIG | Contributions in Section 7 |
| 1.9 | 2021-08-21 | NUIG, INNOV | Pre-Final Version for Internal Review |
| 1.95 | 2021-08-30 | NUIG, LINX | Version with Quality Assurance |
| 2.0 | 2021-08-30 | NUIG | Version for Submission |

| **Version I** | **Date** | **Partner(s)** | **Description** |
|---|---|---|---|
| 0.1 | 2020-08-08 | NUIG | ToC Version |
| 0.2 | 2020-08-08 | NUIG, N OVA | Updated ToC with requested contributions Standard Vocabularies from Stakeholders on Hold until new online events are organised |
| 0.3 | 2020-09-09 | NOVA | Updated contributions |
| 0.4 | 2020-10-10 | NUIG | Integration of contributions<br>Vocabularies form D4.1 integrated |
| 0.5 | 2020-11-11 | NUIG | Additional contributions on FIBO, FIGI and LKIF vocabularies and taxonomies |
| 0.6 | 2020-11-11 | NUIG | Updates in Section 1,<br>Section 3 and Section 4 |
| 0.7 | 2020-12-20 | NUIG | Contributions in Section 4 and Section 5 |
| 0.8 | 2020-12-23 | NUIG | Final Contributions in Section 5 |
| 0.9 | 2020-12-23 | NUIG, INNOV | First Version for Internal Review |
| 0.95 | 2020-12-23 | NUIG, LINX | Version for Quality Assurance |
| 1.0 | 2020-12-23 | NUIG | Version for Submission |

# Executive Summary

Semantics Stream Analytics Engine – SeSA-ME Version III

This deliverable is the third version of the INFINITECH Semantics Streams Analytics Engine (SeSA-ME) and the related tools for enabling semantic data exchange. In this final version of this deliverable the final design architecture and the design specifications are reviewed and described in detail and parts of the previous deliverables are referred in order to make the three doc8ments complementary. This deliverable includes the specification and implementation of the semantics engine from previous deliverables to also make this final deliverable consistent with the design and to be able to identify the improvements, the feedback from stakeholders after testing SeSA-ME semantic engine in the context of pilot use cases, all this activity is part of the Task 4.2 in the WP4.

The Semantic Engine APIs descriptions have been included in this version to clearly identify the services that each APIs is offering and also to provide the full alignment with the online tools, providing the structure of step by step procedure in how to achieve semantic data interoperability. This document is the third version that is considered as final and as such it includes all the related information that is relevant from previous versions to make this final version a self-contain document that can be used for further designs, extensions and deployments.

In this deliverable the prototype provided and documented is a reference implementation that was improved following general requirements coming from the study and purposes at INFINITECH pilot level following stakeholder's requirements and also from particular domains where the use of a semantic engine for mashup building with semantic interoperability capabilities. The deliverable includes references to previous versions where the implementation of the SeSA-ME component was provided, in this version the open source implementation, additionally to the docker version, that can be used when data sharing and data exchange is required and when there is a need to install the semantic engine tools instead of using it as a service is provided.

The prototype implementation from period I review and its extensions towards including feedback from stakeholders and pilots was done taking into consideration the progress of the project with the stakeholders from the financial sector and their data models (i.e. mainly banking data models) which reported the creation of two new ontologies that are considered necessary to be included in the graph data model and also in the set of final APIs that were initially planned to be used.

The change of some of the pilots and the recent start o tow new ones required to follow the same process to be able to introduce semantics and stream data services in the final version based on the pilot re-focus and their objectives and thus new vocabularies were updated. The introduction of these new pilots generated the need to delay the submission of the final version, however this delay did not affect the overall progression of the project because the involvement of re-focused pilots and their stakeholders yet will run for few months later and also the extensions of the project alike the pilot's stakeholders validation are yet in time. The delay in the submission of the final version of this deliverable was informed to the coordination and agreed amongst the consortium as informed in the WP4 online meetings and the general Assembly of the INFINITECH project.

Semantics Stream Analytics Engine - Version II

In the second version of this deliverable the INFINITECH Semantics Streams Analytics Engine (SeSA-ME) version II was enhanced following the new requirements exposed form the inclusion of identified needs form the pilot use cases a set of new APIS was implemented in order to provide the alternative to perform more analytics services and facilitate the visualisation of results.

In the second version of the deliverable we introduced the INFINITECH Graph Data Model, the Semantic Annotator-Middleware Pre-Processing Layer for FinTechs (SAMPLE-Fin) procedure and the Semantic validator. The improvements can be listed as follow:

(a) The inclusion of new APIs, implementation and specifications, for registering and unregistering data sources is implemented and tested, and

(b) to enable the semantic interoperability by using running query APIs and an executed running APIs to facilitate the access and manipulation of data for post processing analytics.

Semantics Stream Analytics Engine - Version I

In the first version of this deliverable the INFINITECH Semantics Streams Analytics Engine (SeSA-ME) and the related tools for enabling semantic data exchange were introduced, The SeSA-ME and its tool are based on the development of an interoperability (ontology-based) database/registry supporting linking of diverse systems and datasets based on shared semantics, as well as semantically interoperable analytics.

The Semantic Engine is an extension of the Super Stream Collider (SSC) tool, which provides a set of web-based interfaces and tools for building data mashups combining semantically annotated Linked Stream and Linked Data sources into easy-to-use data mashups for applications. The SeSA-ME system includes tools along with a visual SPARQL query editor using Swagger APIs and visualization tools for novice users while supporting full access and control over the data mashups for expert users. Tied with the development of the SeSA-ME platform is the development and deployment of the INFNITECH Graph Data Model which enables the support for both the design and deployment of stream-based web applications in a very simple and intuitive way and the analytics services using stream-based applications and services.

In the first version of the deliverable we also introduce the INFINITECH Graph Data Model as an Ontology or set of Standards Ontologies:

(a) to model and represent Finance and Insurance concepts with additional concepts in related relevant areas – e.g., from the Security Transactions domain, Security and Privacy domain – within the INFINITECH project stakeholders,

(b) to enable the semantic interoperability between Internet-connected objects for Finance and Insurance applications in diversity of applications and services settings, and

(c) to enable the application of analytics services and reasoning algorithms for seamless automated information exchange for more complex services and combined applications.

INFINITECH Graph Data Model is composed by following Core ontology standards, such as FIBO, FIGI and LKIF standards and we bootstrap the implementation and deployment of the Semantic Analytics Engine from those existing efforts towards the ontological descriptions of concepts, applications and online services, etc. relevant for the INFINITECH project pilots.

# Table of Contents

1 Introduction ........ 10

1.1 Objective of the Deliverable ........ 10

1.2 Insights from other Tasks and Deliverables ........ 11

1.3 Structure ........ 11

2 INFINITECH Graph Data Models Online Tool ........ 13

2.1 Deployment ........ 13

2.2 Deployment of New Changes ........ 13

2.3 Configuring Insight Hosted portal to INFINITECH Domain ........ 14

3 INFINITECH Data Pack ........ 16

4 Semantic Annotator-Middleware  Pre-processing Layer for FinTechs - SAMPLE-FIN ........ 20

4.1 Data Transformation Guide ........ 20

4.2 Step 1: Selecting Ontologies ........ 20

4.2.1 FIBO ........ 20

4.2.2 FIGI ........ 20

4.2.3 LKIF ........ 21

4.2.4 INFINITECH Core ........ 21

4.3 Step 2: Mapping Native Data to Selected Ontologies ........ 21

4.3.1 RML: RDF Mapping language ........ 21

4.3.2 RML Editor ........ 22

4.3.3 R2RML: RDB to RDF Mapping Language ........ 22

4.4 Step 3: Generating RDF ........ 22

4.4.1 RMLMapper ........ 23

4.4.2 Step 4: Making data queryable ........ 23

4.4.3 Step 5: Data Transformation Example ........ 23

4.4.4 MAPPINGS ........ 24

4.4.5 RDF DATA ........ 24

5 Infinitech Semantic Validator ........ 25

5.1 Deployment ........ 25

6 SeSA-ME Final Specifications and Implementation ........ 26

6.1 SeSA-ME Architecture ........ 26

6.1.1 Source Selection ........ 26

6.1.2 Query Planner ........ 28
6.1.3 Query Builder ........ 30
6.1.4 Query Executor ........ 31
6.1.5 Stream Processor ........ 33
6.1.6 Access Policy Framework ........ 34
6.2 SeSA-ME APIs ........ 36
6.2.1 Static Data APIs – Version I ........ 36
6.2.1.1 Know Your Customer (KYC) Profiler ........ 36
6.2.1.1.1 KYC Data Providers ........ 36
6.2.1.1.2 KYC Data Consumers ........ 38
6.2.1.1.3 Identity Verification ........ 46
6.2.1.1.4 Business Verification ........ 51
6.2.2 Streaming Data APIs – Version I ........ 55
6.2.2.1 Stream Registration ........ 56
6.2.2.1.1 Register for Streams API ........ 56
6.2.3 SeSA-ME New APIS Implementation – Version II ........ 57
6.2.3.1 Stream Registration II ........ 57
6.2.3.1.1 Unregister Data Source API ........ 57
6.2.3.1.2 Run Query API ........ 59
6.2.3.1.3 Run Query Plan API ........ 61

7 SeSA-ME Continuous Integration/Continuous Development ........ 68
7.1 SeSA-ME Engine Development ........ 68
7.1.1 Run SeSA-ME Engine ........ 68
7.1.2 Build SeSA-ME Engine ........ 68
7.2 Deployment using Docker ........ 68
7.3 SeSA-ME Engine APIs ........ 68
7.4 SeSA-ME Engine Deployment ........ 69
7.4.1 SeSA-ME Engine Project Structure ........ 69
7.4.2 Dependencies List ........ 69
7.4.3 Docker File ........ 71

8 Conclusions ........ 72

9 References ........ 73

# List of Figures

Figure 1. INFINITECH Graph Data Models Online Tool Files Structure ........ 15
Figure 2. INFINITECH Graph Data Pack ........ 19
Figure 3. Semantic Stream Analytics Middleware-Engine Architecture ........ 26
Figure 4. Semantic Stream Analytics Middleware-Engine API Services ........ 36

# List of Tables

Table 1: FIBO Useful Links ........ 20
Table 2: FIGI Useful Links ........ 20
Table 3: LKIF Useful Links ........ 21
Table 4: INFINITECH Core Useful Links ........ 21
Table 5: RDF Mapping Language Useful Links ........ 22
Table 6: RML Editor Useful Links ........ 22
Table 7: RDB 2 RDF Mapping Language Useful Link ........ 22
Table 8: RML Mapper Useful Link ........ 23
Table 9: Triple Stores Useful Links ........ 23
Table 10: Example Customer Table ........ 23
Table 11: Data Mapping Example ........ 24
Table 12: Example RDF Data ........ 24
Table 13: Source Selection - Component Description and API Documentation ........ 26
Table 14: Query Planner – Component Description and API Documentation ........ 28
Table 15: Query Builder - Component Description and API Documentation ........ 30
Table 16: Query Executor - Component Description and API Documentation ........ 31
Table 17: Stream Processor - Component Description and API Documentation ........ 33
Table 18: Access Policy Framework - Component Description and API Documentation ........ 34
Table 19: Example Data Source Registration Information ........ 37
Table 20: Example Register Data Source Functionality and URL notation ........ 37
Table 21: Example KYC Data Consumer Method using JSON Schema ........ 37
Table 22: Example Template for Identity Verification ........ 38
Table 23: Example Template for Business Verification ........ 39
Table 24: Example Get Template Functionality and URL notation ........ 39
Table 25: Example Identity Verification method using JSON Schema ........ 39
Table 26: Example Get List of Fields Functionality and URL notation ........ 43
Table 27: Example Business Verification method using JSON Schema ........ 43
Table 28: Example Verify Customer Identity Functionality and URL notation ........ 46
Table 29: Example Verify Customer Identity method using JSON Schema ........ 46
Table 30: Example Verify Business API Functionality and URL notation ........ 51
Table 31: Example Verify Business method using JSON Schema ........ 51
Table 32: Example Register for Streams API Functionality and URL notation ........ 56
Table 33: Example Register for Streams method using JSON Schema ........ 56
Table 34: Example for Unregister Data Source API Functionality and URL notation ........ 58
Table 35: Example unregister a Data Source method using JSON Schema ........ 58
Table 36: Example for Unregister Data Source API Functionality and URL notation ........ 59
Table 37: Example Runs a SPARQL query on the specified data source ........ 59
Table 38: Example for Run Query API Functionality and URL notation ........ 62
Table 39: Example Run a query plan on both static and streaming data sources ........ 62
Table 40: The SeSA-ME component has the following structure: ........ 69
Table 41: SeSA-ME Dependecies ........ 69
Table 42: Docker File ........ 71

# Abbreviations/Acronyms

| | |
|---|---|
| AI | Artificial Intelligence |
| CQELS | Continuous Query Evaluation over Linked Streams |
| DnS | Descriptions and Solutions |
| DOI | Digital Object Identifier |
| DOLCE | Descriptive Ontology for Linguistic and Cognitive Engineering |
| DUL | DOLCE+DnS Ultralite |
| FIBO | Financial Industry Busines Ontology |
| FIGI | Financial Instrument Global Identifier |
| FOAF | Friend of a Friend |
| GIS | Geographic information system |
| GSN | Global Sensor Networks |
| HTML | HyperText Markup Language |
| HTTP | Hypertext Transfer Protocol |
| ICT | Information and Communications Technology |
| LD | Linked Data |
| LKIF | Legal Knowledge Interchange Format |
| LOD | Linking Open Data |
| MIME | Multipurpose Internet Mail Extensions |
| NOAA | National Oceanic and Atmospheric Administration |
| OGC | Open Geospatial Consortium |
| OMG | Object Management Group |
| OWL | Web Ontology Language |
| RDF | Resource Description Framework |
| RDFS | RDF Schema |
| RFS | Request for Service |
| SaS | Sensing-as-a-Service |
| SIOC | Semantically Interlinked Online Communities |
| SLA | Service Level Agreement |
| SOAP | Simple Object Access Protocol |
| SPARQL | SPARQL Protocol and RDF Query Language |
| TaS | Traceability-as-a-Service |
| TCP | Transmission Control Protocol |
| UDP | User Datagram Protocol |
| URI | Uniform Resource Identifiers |
| URN | Uniform Resource Name |
| USB | Universal Serial Bus |
| W3C | World Wide Web Consortium |
| XHTML | Extensible HyperText Markup Language |
| XML | Extensible Markup Language |

# 1 Introduction

This deliverable is the third version of the INFINITECH Semantics Streams Analytics Engine (SeSA-ME) specification, implementation and the related implemented tools for enabling semantic data exchange. In this version of this deliverable the implementation of the engine and the semantic online tools have been produced and improved after the feedback received from stakeholders regarding the specifications provided in version I and version II.

The Semantic Stream Analytics Middleware-Engine (SeSA-ME) is an extension of the Super Stream Collider (SSC) platform and tools. The SeSA-ME system includes tools along with a visual SPARQL query editor using Swagger APIs and visualization tools for novice users while supporting full access and control over the data mashups for expert users. The development of the SeSA-ME platform requires the use of an associated data model, thus the INFINITECH data model and its deployment is also an important part in this deliverable. The INFINITECH Graph Data Model enables supporting both the design and deployment of stream-based web applications in a very simple and intuitive way and the extension to analytics services using stream-based applications and services.

In this deliverable the specification are focused on the deployment of the Semantic Engine and the semantic tools, thus the previous deliverables together with this final deliverable compose the set of documents that put together not only the design and implementation efforts but also the experiences and improvements of the SeSA-ME engine and its associated semantic services.

## 1.1 Objective of the Deliverable

This deliverable reports the final set of specifications of the Semantic Stream Analytics Engine (SeSA-ME) as a framework and the online tools for interoperability and data exchange. This document is the third version of three, at the first one the design and specification were providing taking in consideration the initial requirements and specifications from the super stream collider.  In the second version of this deliverable the basic streaming data services and tools for data interoperability and their use in particular use cases addressing pilots requirements were described. In this third version of the deliverable the implementation and deployment of the online tools and the SeSA-ME engine are provided, following the feedback received and the INFINITECH way references to make the implementation as simple as possible to install. Following the INFINITEHC way there is a CI/CD process that aims to ensure the replicability and easy integration of the SeSA-ME engine within other solutions and platforms. The SeSA-ME engine implementation documented in this third deliverable is the reference implementation considered final and that was compared against the general requirements from the stakeholders in the pilots of the INFINITECH project.

The version I and version II of this document live documents aggregating information making them self-contained and with the objective to provide the evolution of the implementation, while this version is focused on complement them by focusing in describing the way to deploy and instantiate the semantic tools. The need to have the semantic tools online derivates from (a) the requirements from the use case descriptions, i.e. the involved concepts and relationships between them identified in Task 4.1,  (b) the set of related ontologies relevant to INFINITECH identified in task 4.2 as reported in their deliverables and (c) the relevance of some terms used in different domains that can be used for exchange data and ma it interoperable.

## 1.2 Insights from other Tasks and Deliverables

This deliverable is an ultimate updated report that complement the previous two deliverables this, document describes the implementation of the semantic tools that make the semantic engine functional and at the same time easy to deploy and easy to use, this deliverable also make references to other parts in previous deliverables and particularly to the online frameworks that facilitates the use of the data model principles described in the Deliverable D4.1 and D4.2 already as initial design and reviewed version of the SeSA-ME engine implementation.

This deliverable extends the functionalities and capabilities about the INFINITECH Semantic Interoperability Framework, introduced and explained in the Deliverable D4.1 and Deliverable 4.2, where basic data models are described and the implementation is based in simulated data. In this deliverable the specifications correspond to test that uses synthetic data as a more realistic approach to the final data sets that can be found in financial and insurance sectors.

## 1.3 Structure

The overall content of this document focuses on providing the final specifications that complements the implementation, deployment and testing of the SeSA-ME and the semantic online tools. The previous versions of this document provided the comprehensive analysis of the state of the art on semantic modelling and the necessary background for initiating people in the use of semantic technologies and also provided overall information around related areas to graph data modelling, stream processing and data mashups building.

Section 2 outlines the graph data model online tool, it provides the deployment process facilitating the use of the semantic models and include the new changes from version I and II, it also describes the way to configure the online portal that currently is being used in the INFINITECH project.

Section 3 reviews the data pack. This section mainly focus on providing the current implementation of the existing ontologies related to the financial and insurance sectors that are relevant for the INFINITECH project. It also focuses on adding the semantics model that complements the finance sector addressing the overview of the INFINITECH Core and other most used standards.

Section 4 provide the specification to the SAMPLE-FIN online tools for supporting the graph data pack where the Financial Industry Busines Ontology (FIBO) from the EDM council, the Financial Instrument Global Identifier (FIGI), an established global standard issued under the guidelines of the Object Management Group (OMG) and the Legal Knowledge Interchange Format (LKIF) Ontologies are included as part of the necessary online tools that SeSA-ME uses to operate.

Section 5 provide the specifications for the deployment of the semantic validator, this section focuses on describing the process to verify the model once it is built. This section do not describes how to use the model instead how to deploy the semantic validator and how it helps to refine and identify possible flows in the data model before even this become operative and used.

Section 6 includes the SeSA-ME Specification. The rationale to include the specification in this version is that designers and developers understand the use of the tools in a form of steps to achieve the semantic interoperability. The design and implementation of SeSA-ME follows the recommended best-effort practice to reuse design, implement and re-use existing, popular ontologies/vocabularies as much as possible. The implementation of SeSA-ME corresponds to the need to follow the process towards annotating data sets and be able to process it as data streams.

Section 7 is the documentation for the CI/CD process documenting the deployment and procedure to allow developers and people interested in the use of the SeSA-ME component. The process to Run and Build SeSA-ME is included and also the process to deploy it using a Docker File

Section 8 present the conclusions and present some pointers in how the INFINTECH SeSA-ME component will follow the sandboxes design in the INFINITECH project and outlines the support of pilots following the proposed methodology.

Finally Section 9 includes a list of relevant references that have been used along the three versions of the document alike the ones used across this particular document.

This deliverable refers to section 4 in the Deliverable D4.2 Semantic Models and Ontologies II, where INFINITECH Core Data Model & Semantic online tools  are included, and also the section 5 where the overall specifications are described. In this deliverable, even though sections have been taken from other deliverables, this document is organised in the way that it incorporates the most relevant design and implementation parts from the previous to deliverables as a way to make this final version of the deliverable a self-contain document but that in a global view the three documents are the overall complete set of specifications.

# 2 INFINITECH Graph Data Models Online Tool

## 2.1 Deployment

The INFINITECH Graph Data Models Online Tools portal is deployed on the below server (*vmiot15*) hosted in Insight:

```
Host:  vmiot15.datascienceinstitute.ie
Username & Password: Create a JIRA ticket to get access to this server.
```

This portal is deployed on Tomcat Application Server on the path `/var/lib/tomcat9/`.

Username and Password for Tomcat Manager are `admin` and `admin` respectively.

## 2.2 Deployment of New Changes

If you make any changes to this portal, follow the below steps to deploy these changes to the online server.

**Step 1:** Push the new changes to this code repository.

**Step 2:** Login to the Insight Server *vmiot15* and go to the *webapps* folder within Tomcat using the following command:

```
cd /var/lib/tomcat9/webapps/
```

**Step 3:** Go to the project folder within *webapps*:

```
cd /infinitech-data-models/
```

**Step 4:** Pull all the new changes using the following command:

```
git pull origin master
```

**Step 5:** Go to the following URL to view the deployed website.

http://vmiot15.datascienceinstitute.ie:8080/infinitech-data-models/

**Step 6:** (*OPTIONAL*) In case if tomcat needs restart, the following command can be used to restart tomcat:

```
systemctl restart tomcat9
```

## 2.3 Configuring Insight Hosted portal to INFINITECH Domain

The INFINITECH Graph Data Models Online Tools portal is hosted on Insight server and the host URL corresponds to Insight. To configure this portal to be served using INFINITECH domain name, we have installed NGINX server as a reverse proxy.

The following steps were performed to install and configure NGINX server:

**Step 1:** Install NGINX Server on *vmiot15* server.

**Step 2:** Disable the default virtual host, that is pre-configured when Nginx is installed via Ubuntu's packet manager apt:

```
unlink /etc/nginx/sites-enabled/default
```

**Step 3:** Go to the directory `/etc/nginx/sites-available` and create a reverse proxy configuration file.

```
cd /etc/nginx/sites-available
nano reverse-proxy.conf
```

**Step 4:** Paste the following Nginx configuration in the configuration file created in the previous step. The proxy server will now redirect all incoming connections on port 80 to the Apache Tomcat Server, listening on port 8080.

```
server {
        listen 80;
        listen [::]:80;

        server_name graph-data-model.infinitech-h2020.eu;

        access_log /var/log/nginx/reverse-access.log;
        error_log /var/log/nginx/reverse-error.log;

        location / {
                    proxy_pass http://127.0.0.1:8080/infinitech-data-models/;
  }

        location /infinitech-data-models/content/ontologies/ {
                  proxy_pass http://127.0.0.1:8080/infinitech-data-
models/content/ontologies/;
}
}
```

**Step 5:** Copy the configuration from /etc/nginx/sites-available to /etc/nginx/sites-enabled. It is recommended to use a symbolic link.

```
ln -s /etc/nginx/sites-available/reverse-proxy.conf /etc/nginx/sites-
enabled/reverse-proxy.conf
```

**Step 6:** Test the Nginx configuration file.

```
nginx -t
```

which will return the below output:

```
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

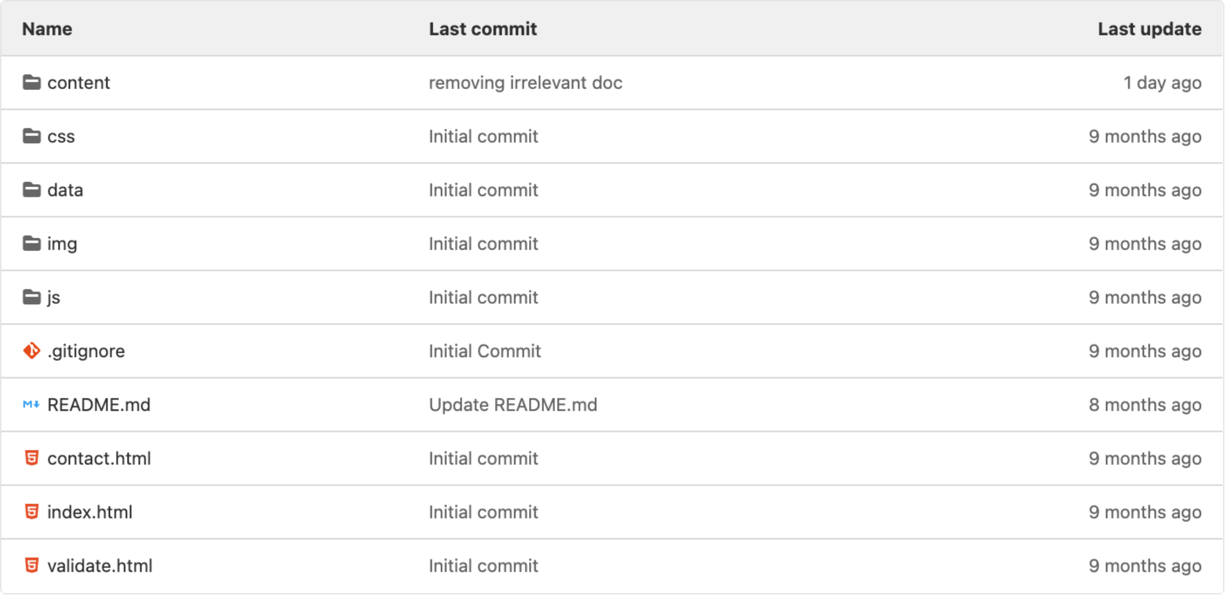
**Step 7:** Restart NGINX server:

```
service nginx restart
```

**Step 8:** Open a web browser on your local computer and paste the below url which will display your web applications homepage.

http://graph-data-model.infinitech-h2020.eu/

The figure 1 below shows the distribution of the documentation in the public repository, this is the structure that the files will have if the deployment is done following the steps as described above.



| Name | Last commit | Last update |
|---|---|---|
| content | removing irrelevant doc | 1 day ago |
| css | Initial commit | 9 months ago |
| data | Initial commit | 9 months ago |
| img | Initial commit | 9 months ago |
| js | Initial commit | 9 months ago |
| .gitignore | Initial Commit | 9 months ago |
| README.md | Update README.md | 8 months ago |
| contact.html | Initial commit | 9 months ago |
| index.html | Initial commit | 9 months ago |
| validate.html | Initial commit | 9 months ago |

Figure 1. INFINITECH Graph Data Models Online Tool Files Structure

# 3 INFINITECH Data Pack

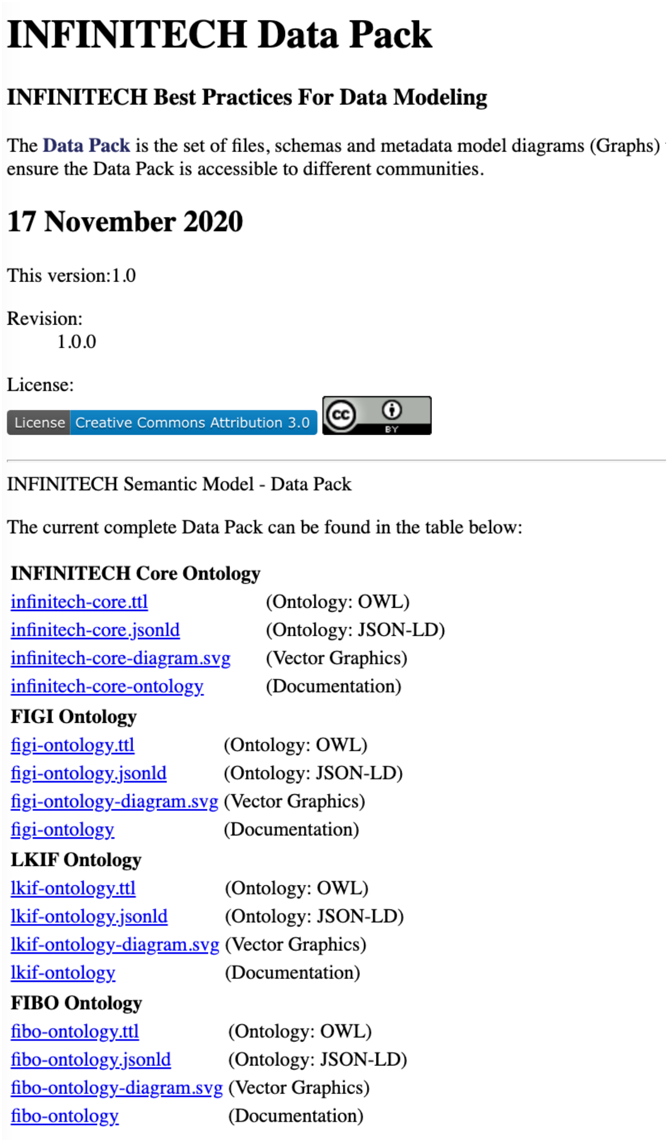
This section includes the online data pack specifications and details. The data model is crucial for the operation for the Semantic Engine, thus the use of a online machine readable accessible tool is accessible tool is considered relevant. The use of semantic technologies and particularly RDF facilitates the use of RDF-based streams. RDFbased data allows to support the operators multiple machines that uses the data model remotely. Furthermore, this interactive online tool facilitates the easy comprehension and understanding of the data model organization and the data structures. The data model online accessing process is leveraged by the online machine-readable files services which is recommend using for reducing problems when different versions of the data model is generated. INFINITECH Graph Data Model can be powered by using online semantic-enabled accessing services in a similar way SPARQL endpoints act for Jena and Virtuoso semantic repositories for example, where multiple instances can access the online data service and verify versioning and at the same time get access to the metadata.

The following is the INFINITECH data pack used to access the online services:

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
  <link rel="stylesheet" href="resources/primer.css" media="screen" />
  <link rel="stylesheet" href="resources/rec.css" media="screen" />
  <link rel="stylesheet" href="resources/extra.css" media="screen" />
  <link rel="stylesheet" href="resources/owl.css" media="screen" />
  <script src="resources/jquery.js"></script>
  <script src="resources/marked.min.js"></script>
  <script>
function loadHash() {

jQuery(".markdown").each(function(el){jQuery(this).after(marked(jQuery(this).text())).rem
ove()});
                 var hash = location.hash;
                 if($(hash).offset()!=null){
                   $('html, body').animate({scrollTop: $(hash).offset().top}, 0);
                 }
    }
    $(function(){
      loadHash();
    });
  </script>
  <title>Data Pack</title>
  <meta name="description" content="INFINITECH Data Pack" />
</head>

<body>

  <div class="container">

    <div class="head">

      <h1>INFINITECH Data Pack</h1>
      <h3>INFINITECH Best Practices For Data Modeling</h3>

      <span>
        The <strong style="color: #272a64;">Data Pack</strong> is the set of files,
schemas and metadata
        model diagrams (Graphs) that represent the way the INFINITECH data is organised
and structured, it
        contains the metadata in .ttl format and also contains the metadata in two
different formats,
        <strong style="color: #272a64;">.json-ld</strong> and <strong style="color: 
#272a64;">.owl</strong> 
        to ensure the Data Pack is accessible to different communities.
      </span>

      <h2>17 November 2020</h2>
      <dl>
        <dt>This version:1.0</dt>
        <dd></dd>
      </dl>
      <dt>Revision:</dt>
      <dd>1.0.0</dd>

<!-- <dl><dt>Author(s):</dt>
<dl><dt>Contributor(s):</dt>
<dl><dt>Download serialization:</dt> -->

<span><dd></dd>
<!-- <span><a href="ontology.json" target="_blank"><img 
src="https://img.shields.io/badge/Format-JSON_LD-blue.svg" alt="JSON-LD" /></a>
</span><span><a href="ontology.owl" target="_blank"><img 
src="https://img.shields.io/badge/Format-RDF/XML-blue.svg" alt="RDF/XML" /></a>
</span><span><a href="ontology.nt" target="_blank"><img 
src="https://img.shields.io/badge/Format-N_Triples-blue.svg" alt="N-Triples" /></a>
</span><span><a href="ontology.ttl" target="_blank"><img 
src="https://img.shields.io/badge/Format-TTL-blue.svg" alt="TTL" /></a>  -->
<dl><dt>License: </dt><a href="http://creativecommons.org/licenses/by/3.0/" 
target="_blank"><img src ="https://img.shields.io/badge/License-Creative Commons 
Attribution 3.0 -blue.svg" alt="http://creativecommons.org/licenses/by/3.0/"></img></a>
<a href="http://creativecommons.org/licenses/by/3.0/" rel="license" target="_blank"><img 
src="http://i.creativecommons.org/l/by/3.0/88x31.png" style="border-width:0" 
alt="License"></img></a>
<br/></dd></dl><dl><dt></dt>

</dl>
<hr/>
</div>
<div class="status">
<div>
<span>INFINITECH Semantic Model - Data Pack</span>
</div>
</div>
<section id="abstract">
  <p>The current complete Data Pack can be found in the table below:</p>
</section>

<div>
          <table class="table">
            <tbody>
              <tr>
                <td><strong>INFINITECH Core Ontology</strong></td>
                <td></td>
              </tr>
              <tr>
                <td><a href="../ontologies/INFINITECH-Core/infinitech-
core.ttl">infinitech-core.ttl</a></td>
                <td>(Ontology: OWL)</td>
              </tr>
              <tr>
                <td><a href="../ontologies/INFINITECH-Core/infinitech-
core.jsonld">infinitech-core.jsonld</a></td>
                <td>(Ontology: JSON-LD)</td>
              </tr>
              <tr>
                <td><a href="../ontologies/INFINITECH-Core/infinitech-
core.svg">infinitech-core-diagram.svg</a></td>
                <td>(Vector Graphics)</td>
              </tr>
              <tr>
                <td><a href="../infinitech-core-docs/">infinitech-core-ontology</a></td>
                <td>(Documentation)</td>
              </tr>
            </tbody>
          </table>

          <table class="table">
            <tbody>
              <tr>
                <td><strong>FIGI Ontology</strong></td>
                <td></td>
              </tr>
              <tr>
                <td><a href="../ontologies/FIGI/FIGI.ttl">figi-ontology.ttl</a></td>
                <td>(Ontology: OWL)</td>
              </tr>
              <tr>
                <td><a href="../ontologies/FIGI/FIGI.jsonld">figi-
ontology.jsonld</a></td>
                <td>(Ontology: JSON-LD)</td>
              </tr>
              <tr>
                <td><a href="../ontologies/FIGI/FIGI.svg">figi-ontology-
diagram.svg</a></td>
                <td>(Vector Graphics)</td>
              </tr>
              <tr>
                <td><a href="../figi-docs/">figi-ontology</a></td>
                <td>(Documentation)</td>
              </tr>
            </tbody>
          </table>

          <table class="table">
            <tbody>
              <tr>
                <td><strong>LKIF Ontology</strong></td>
                <td></td>
              </tr>
              <tr>
                <td><a href="../ontologies/LKIF/LKIF.ttl">lkif-ontology.ttl</a></td>
                <td>(Ontology: OWL)</td>
              </tr>
              <tr>
                <td><a href="../ontologies/LKIF/LKIF.jsonld">lkif-
ontology.jsonld</a></td>
                <td>(Ontology: JSON-LD)</td>
              </tr>
              <tr>
                <td><a href="../ontologies/LKIF/LKIF.svg">lkif-ontology-
diagram.svg</a></td>
                <td>(Vector Graphics)</td>
              </tr>
              <tr>
                <td><a href="../lkif-docs/index-en.html">lkif-ontology</a></td>
                <td>(Documentation)</td>
              </tr>
            </tbody>
          </table>

          <table class="table">
            <tbody>
              <tr>
                <td><strong>FIBO Ontology</strong></td>
                <td></td>
              </tr>
              <tr>
                <td><a href="#">fibo-ontology.ttl</a></td>
                <td>(Ontology: OWL)</td>
              </tr>
              <tr>
```

```
                <td><a href="#">fibo-ontology.jsonld</a></td>
                <td>(Ontology: JSON-LD)</td>
              </tr>
              <tr>
                <td><a href="#">fibo-ontology-diagram.svg</a></td>
                <td>(Vector Graphics)</td>
              </tr>
              <tr>
                <td><a href="../fibo-docs/">fibo-ontology</a></td>
                <td>(Documentation)</td>
              </tr>
            </tbody>
          </table>
</div>

</body>
</html>
```

The figure 2 below shows the distribution of the documentation in the public repository, this is the structure that the files will have if the deployment is done following the steps as described above.



# INFINITECH Data Pack

### INFINITECH Best Practices For Data Modeling

The **Data Pack** is the set of files, schemas and metadata model diagrams (Graphs) ensure the Data Pack is accessible to different communities.

## 17 November 2020

This version:1.0

Revision:
1.0.0

License:

INFINITECH Semantic Model - Data Pack

The current complete Data Pack can be found in the table below:

| **INFINITECH Core Ontology** | |
|---|---|
| infinitech-core.ttl | (Ontology: OWL) |
| infinitech-core.jsonld | (Ontology: JSON-LD) |
| infinitech-core-diagram.svg | (Vector Graphics) |
| infinitech-core-ontology | (Documentation) |
| **FIGI Ontology** | |
| figi-ontology.ttl | (Ontology: OWL) |
| figi-ontology.jsonld | (Ontology: JSON-LD) |
| figi-ontology-diagram.svg | (Vector Graphics) |
| figi-ontology | (Documentation) |
| **LKIF Ontology** | |
| lkif-ontology.ttl | (Ontology: OWL) |
| lkif-ontology.jsonld | (Ontology: JSON-LD) |
| lkif-ontology-diagram.svg | (Vector Graphics) |
| lkif-ontology | (Documentation) |
| **FIBO Ontology** | |
| fibo-ontology.ttl | (Ontology: OWL) |
| fibo-ontology.jsonld | (Ontology: JSON-LD) |
| fibo-ontology-diagram.svg | (Vector Graphics) |
| fibo-ontology | (Documentation) |

Figure 2. INFINITECH Graph Data Pack

# 4 Semantic Annotator-Middleware Pre-processing Layer for FinTechs - SAMPLE-FIN

## 4.1 Data Transformation Guide

The following steps are for the purpose of guiding people to transform their data from native format to RDF format. Each step also lists a set of tools which can be used to perform a specific task.

## 4.2 Step 1: Selecting Ontologies

If you want to transform your data to RDF format, the first thing you need to do is to find an ontology which can be used to model your native data in RDF format.

In case of the INFINITECH project, there are several ontologies available. Below is the list of these ontologies.

### 4.2.1 FIBO

The Financial Industry Business Ontology (FIBO) defines the sets of things that are of interest in financial business applications and the ways that those things can relate to one another. In this way, FIBO can give meaning to any data (e.g., spreadsheets, relational databases, XML documents) that describe the business of finance.

Table 1: FIBO Useful Links

| Useful Links | | |
|---|---|---|
| | External | INFINITECH |
| **Website** | FIBO | FIBO Docs |
| **OWL Files** | FIBO OWL Files | FIBO Files |

### 4.2.2 FIGI

FIGI is a Financial Industry Global Instrument Identifiers (FIGI) Ontology.

Table 2: FIGI Useful Links

| Useful Links | | |
|---|---|---|
| | External | INFINITECH |
| **Website** | FIGI | FIGI Docs |
| **Files** | | FIGI Files |

### 4.2.3 LKIF

The Legal Knowledge Interchange Format (LKIF) is an OWL ontology of legal concepts, allowing legal knowledge bases to be represented in OWL.

Table 3: LKIF Useful Links

| Useful Links | | |
|---|---|---|
| | External | INFINITECH |
| **Website** | Project Website | LKIF Docs |
| **LKIF Files** | LKIF Github | LKIF FIles |
| **Publications** | LKIF Core Ontology | |

### 4.2.4 INFINITECH Core

INFINITECH Core defines alignment between FIBO, FIGI & LKIF in a formal way.

Table 4: INFINITECH Core Useful Links

| Useful Links | |
|---|---|
| **Website** | INFINITECH Core |
| **INFINITECH Core Files** | INFINITECH Core FIles |

## 4.3 Step 2: Mapping Native Data to Selected Ontologies

When you have selected the ontologies which can be used to model your data, the next step is to specify mapping from entities and attributes in the native data format to entities and attributes in the selected ontologies.

There are some standard mapping languages available which can be used to specify these mappings, such as RML, R2RML etc.

### 4.3.1 RML: RDF Mapping language

RML, a generic mapping language, based on and extending R2RML. The RDF Mapping language (RML) is a mapping language defined to express customized mapping rules from heterogeneous data structures and serializations to the RDF data model. RML is defined as a superset of the W3C-standardized mapping language R2RML, aiming to extend its applicability and broaden its scope, adding support for data in other structured formats. RML follows exactly the same syntax as R2RML; therefore, RML mappings are themselves RDF graphs.

Other than relational databases, currently you can define mappings from sources, such as CSV, TSV, XML and JSON to RDF. Such mappings describe how existing data can be represented using the RDF data model.

Table 5: RDF Mapping Language Useful Links

| Useful Links | |
|---|---|
| **Website** | RML |
| **Specifications** | RML: RDF Mapping Language |

## 4.3.2 RML Editor

The RMLEditor offers a Graphical User Interface (GUI) to enable data publishers, who are domain experts, to model knowledge derived from multiple, heterogeneous data sources. The RMLEditor uses RML as its underlying mapping language, offering a uniform GUI to its users to edit rules.

Table 6: RML Editor Useful Links

| Useful Links | |
|---|---|
| **Website** | RMLEditor |
| **Online Tool:** | RMLEditor Web Version |

## 4.3.3 R2RML: RDB to RDF Mapping Language

R2RML is a W3C standard to express customized mappings from relational databases to RDF datasets. Such mappings provide the ability to view existing relational data in the RDF data model, expressed in a structure and target vocabulary of the mapping author's choice. R2RML mappings are themselves RDF graphs and written down in Turtle syntax.

Table 7: RDB 2 RDF Mapping Language Useful Link

| Useful Links | |
|---|---|
| **Website** | R2RML: RDB to RDF Mapping Language |

## 4.4 Step 3: Generating RDF

When you have the mappings in place, then the next step is to generate RDF data from native data based on the mappings specified in the previous step.

The following tools can be used to transform your data to RDF:

### 4.4.1 RMLMapper

The RMLMapper executes RML rules to generate Linked Data. It is a Java library, which is available via the command line.

Table 8: RML Mapper Useful Link

| Useful Links | |
|---|---|
| **Website** | RML Mapper |

### 4.4.2 Step 4: Making data queryable

When the data is transformed to RDF successfully, the next step is to enable querying on the RDF data in order to make it easily accessible. In order to do this, you need to select a triple store and upload your data to it. The following triple stores can be used to make your data queryable.

Table 9: Triple Stores Useful Links

| Useful Links | |
|---|---|
| **Virtuoso** | Virtuoso |
| **Jena Fuseki** | Jena Fuseki |

### 4.4.3 Step 5: Data Transformation Example

This section will explain mapping example data to an ontology and then how the transformed RDF data would look like.

Below is an example database table, i.e. CUSTOMER_TABLE, which contains records of customers. To transform this table to RDF format, you need to create mappings from this table to your selected ontology.

Table 10: Example Customer Table

**CUSTOMER_TABLE**

| CUSTOMER_ID | FIRST_NAME | LAST_NAME | DATE_OF_BIRTH |
|---|---|---|---|
| 1 | John | Smith | 14-04-1985 |
| 2 | James | Oliver | 02-11-1974 |

Below is an example of mappings generated for transforming the above database table to RDF format.

## 4.4.4 MAPPINGS

Table 11: Data Mapping Example

**Example Mapping**

```
@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix fibo: <https://spec.edmcouncil.org/fibo/ontology/FND/AgentsAndPeople/People/>.

<#CustomerMap>
  rr:logicalTable [ rr:tableName "CUSTOMER_TABLE" ];
  rr:subjectMap [
    rr:template "http://data.example.com/customer/{CUSTOMER_ID}";
    rr:class ex:Person;
  ];

  rr:predicateObjectMap [
    rr:predicate ex:hasFirstName;
    rr:objectMap [ rr:column "FIRST_NAME" ];
  ];

  rr:predicateObjectMap [
    rr:predicate ex:hasSurname;
    rr:objectMap [ rr:column "LAST_NAME" ];
  ];

  rr:predicateObjectMap [
    rr:predicate ex:hasDateOfBirth;
    rr:objectMap [ rr:column "DATE_OF_BIRTH" ];
  ].
```

The example RDF data generated by transforming the database table, i.e. “CUSTOMER_TABLE” using the above mappings is shown below.

## 4.4.5 RDF DATA

Table 12: Example RDF Data

**Example RDF Data**

```
@prefix it: <http://data.example.com/customer/> .
@prefix fibo: <https://spec.edmcouncil.org/fibo/ontology/FND/AgentsAndPeople/People/>.

it:1 a fibo:Person ;
  fibo:hasFirstName "John" ;
  fibo:hasSurname "Smith" ;
  fibo:hasDateOfBirth "14-04-1985" ;

it:2 a fibo:Person ;
  fibo:hasFirstName "James" ;
  fibo:hasSurname "Oliver" ;
  fibo:hasDateOfBirth "02-11-1974" ;
```

# 5 Infinitech Semantic Validator

The building process of Semantic Data Streams and Mashups requires a basic know-how on data modelling and processing, the previous deliverables provide the stat of the art and a complete analysis. This section focuses on describing the next process when a semantic model is built, and that constitutes an important part of the modelling process. The semantic validation helps to refine and identify possible flows in the data model before even this become operative and used in the semantic engine. The semantic validator does not influence the design and implementation of the Semantics Streams Analytics Engine (SeSA-ME) architecture, but it contributes by verifying all the terms and vocabularies alike if the defined relationships are properly included and thus the design principles for high level architectures can be addressed. It is highly recommended to use the semantic validator to validate the different terms and concepts introduced but also to understand the use of the semantics in the context of the INFINITECH Graph Data model construction.

## 5.1 Deployment

The Semantic Validator is deployed on the below server (*vmiot15*) hosted in Insight:

```
Host:  vmiot15.datascienceinstitute.ie
Username & Password: Create a JIRA ticket to get access to this server.
```

The Semantic Validator is deployed on Tomcat Application Server on the path `/var/lib/tomcat9/`.

Username and Password for Tomcat Manager are `admin` and `admin` respectively.

Follow the below steps to deploy Semantic Validator to the online server.

**Step 1:** Make a `war` file of Semantic Validator.

**Step 2:** Login to the Insight Server *vmiot15* and go to the *webapps* folder within Tomcat using the following command:

```
cd /var/lib/tomcat9/webapps/
```

**Step 3:** Copy the `war` file of Semantic Validator to the *webapps* folder within Tomcat Server:

**Step 4:** Restart Tomcat Server:

```
systemctl restart tomcat9
```

**Step 5:** Go to the following URL to test the Semantic Validator.

http://graph-data-model.infinitech-h2020.eu/validate.html

The graphical interface you see when click on the above link is part of the Graph Data Model Online Tools and the interface code resides in the Graph Data Model Online Tools repository.

This code repository is for the Semantic Validator Service.

# 6 SeSA-ME Final Specifications and Implementation

INFINITECH devises a semantic interoperability solution based on a combination of concepts from FIBO, FIGI and LKIF as well as based on the selective enhancement of these ontologies with new concepts as needed by the project’s pilots use cases. SeSA-ME solution is designed in the form to be a shared semantics solution, which will take advantage of transformation of data schemas to our common INFINITECH semantics.

## 6.1 SeSA-ME Architecture

Leveraging on NUIG’s Super Stream Collider (SSC) solution, the INFINITECH project is providing the means for the deployment and provisioning of semantic reasoning and analytics capabilities in massive, distributed computing systems (i.e., large scale cloud data centres such as those hosting the INFINITECH testbeds) by implementing the Semantics Stream Analytics Middleware-Engine (SeSA-ME). In this way, INFINITECH’s SeSA-ME aims for offering capabilities for live semantic data processing and on-demand access to smart semantic analytics services. Figure 14 depicts the SeSA-ME Architecture where, it is observed the different components and how it interacts with data sources alike it provides data sharing applications.

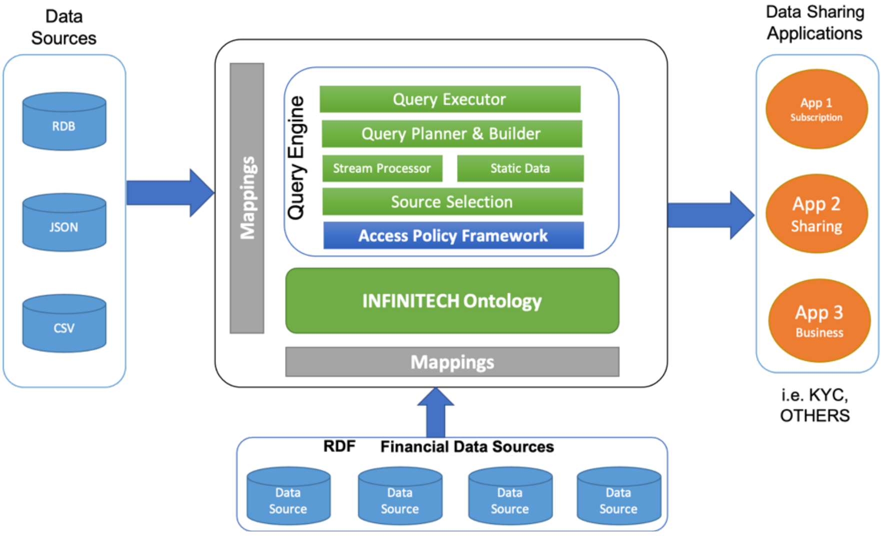


Figure 3. Semantic Stream Analytics Middleware-Engine Architecture

The high-performance semantic stream analytics functionalities of the SeSA-ME component are made available through Open APIs and will be deployed on the project’s sandboxes and testbeds as, in this section the specification for the different blocks of the SeSA-ME engine are described.

### 6.1.1 Source Selection

Table 13: Source Selection - Component Description and API Documentation

| Attribute | Documentation & Example |
|---|---|
| Component ID | INF-DSM-130-S |

| | |
|---|---|
| Component Name | Source Selection |
| Description | This component is responsible for selecting data sources which could potentially return results for a given request. Usually there are many data sources available to get data from but all of them might not be relevant for the request. Hence sending requests to all of them would incur extra load on the data sources and the SeSA-ME engine and would cause delay in response to the request. So, identifying relevant data sources for a request is important to avoid any delays and unnecessary requests to data sources. The source selection process is performed based on the availability of pre-processed information, e.g., meta data, from data sources or availability of mechanisms to inquire about information from data sources at run-time. The identification of selecting relevant sources for a request will also contribute to building requests for each individual data source. |
| Icon | N/A |
| IP Owner & Partner in Charge | NUIG |
| INFINITECH Component Category | Data Semantics |
| IRA - BDVA Layer | Data Processing |
| Input (Required by the Component) | Request in JSON format, list of data sources and meta-data of data sources. |
| Output (Produced by the Component) | List of relevant data sources against data requested. |
| Technology or Platform to be used | Java |
| Part of INFINITECH Core | Yes |

| | |
|---|---|
| MARKETPLACE | Yes if it will be part of the Marketplace |
| Microservice | Yes if it is a dockerized microservice component |
| Endpoint/REST API | To be defined |
| License | To be defined |
| Other Information / Remarks | N/A |
| Detailed Documentation | N/A |

### 6.1.2 Query Planner

Table 14: Query Planner – Component Description and API Documentation

| Attribute | Documentation & Example |
|---|---|
| Component ID | INF-DSM-131-S |
| Component Name | Query Planner |
| Description | The query planner identifies the order in which the queries will be executed on the relevant data sources. This step is performed after the source selection step. The inputs from the source selection component is utilised to plan the queries, their order and the data source on which each query will be executed. |
| Icon | N/A |

| | |
|---|---|
| IP Owner & Partner in Charge | NUIG |
| INFINITECH Component Category | Data Semantics |
| IRA - BDVA Layer | Data Processing |
| Input (Required by the Component) | List of queries and list of data sources on which the query will be executed. |
| Output (Produced by the Component) | Query plan |
| Technology or Platform to be used | Java |
| Part of INFINITECH Core | Yes |
| MARKETPLACE | Yes, it will be part of the Marketplace |
| Microservice | Yes, it is a dockerized microservice component |
| Endpoint/REST API | To be defined |
| License | To be defined |
| Other Information / Remarks | N/A |
| Detailed Documentation | N/A |

### 6.1.3 Query Builder

Table 15: Query Builder - Component Description and API Documentation

| Attribute | Documentation & Example |
|---|---|
| Component ID | INF-DSM-134-S |
| Component Name | Query Builder |
| Description | As its name suggests, the query builder will build the actual queries, as identified in the query planning step, from existing query templates. For example, customer profile building queries. |
| Icon | N/A |
| IP Owner & Partner in Charge | NUIG |
| INFINITECH Component Category | Data Semantics |
| IRA - BDVA Layer | Data Processing |
| Input (Required by the Component) | Query Plan |
| Output (Produced by the Component) | SPARQL query or CQELS or C-SPARQL query |
| Technology or Platform to be used | Java |
| Part of INFINITECH Core | Yes |

| | |
|---|---|
| MARKETPLACE | Yes, it will be part of the Marketplace |
| Microservice | Yes, it is a dockerized microservice component |
| Endpoint/REST API | To be defined |
| License | To be defined |
| Other Information / Remarks | N/A |
| Detailed Documentation | N/A |

### 6.1.4 Query Executor

Table 16: Query Executor - Component Description and API Documentation

| Attribute | Documentation & Example |
|---|---|
| Component ID | INF-DSM-132-S |
| Component Name | Query Executor |
| Description | The query executor component will be responsible for executing the queries generated based on the API templates on the desired data source. For example, executing SPARQL query using Jena ARQ library on a data source, e.g. triple store. |
| Icon | N/A |
| IP Owner & Partner in Charge | NUIG |

| | |
|---|---|
| INFINITECH Component Category | Data Semantics |
| IRA - BDVA Layer | Data Processing |
| Input (Required by the Component) | SPARQL query and data source on which the query will be executed. |
| Output (Produced by the Component) | Result set in JSON format. |
| Technology or Platform to be used | Java |
| Part of INFINITECH Core | Yes |
| MARKETPLACE | Yes, it will be part of the Marketplace |
| Microservice | Yes, it is a dockerized microservice component |
| Endpoint/REST API | To be defined |
| License | To be defined |
| Other Information / Remarks | N/A |
| Detailed Documentation | N/A |

### 6.1.5 Stream Processor

Table 17: Stream Processor - Component Description and API Documentation

| Attribute | Documentation & Example |
|---|---|
| Component ID | INF-DSM-133-S |
| Component Name | Stream Processor |
| Description | This component will be responsible for managing RDF streams of data coming from streaming data sources. It will execute queries on streaming data and provide streams of output data to the requesting entity, based on the frequency and time frame specified in the query. For example, executing a CQEL or C-SPARQL query using a stream processing engine. |
| Icon | N/A |
| IP Owner & Partner in Charge | NUIG |
| INFINITECH Component Category | Data Semantics |
| IRA - BDVA Layer | Data Processing |
| Input (Required by the Component) | C-SPARQL or CQELS query and data source on which the query will be executed. |
| Output (Produced by the Component) | Data streams in JSON format |
| Technology or Platform to be used | Java. |

| | |
|---|---|
| Part of INFINITECH Core | Yes |
| MARKETPLACE | Yes, it will be part of the Marketplace |
| Microservice | Yes, it is a dockerized microservice component |
| Endpoint/REST API | To be defined |
| License | To be defined |
| Other Information / Remarks | N/A |
| Detailed Documentation | N/A |

### 6.1.6 Access Policy Framework

Table 18: Access Policy Framework - Component Description and API Documentation

| Attribute | Documentation & Example |
|---|---|
| Component ID | INF-DSM-135-S |
| Component Name | Access Policy Framework |
| Description | The access policy framework will be used to perform authorization of users based on the access policy rules defined. This component works after the authentication step which is not part of this. This component is composed of user profiles and access policies. User profiles should be stored and access policies on the underlying data based on the user profiles must be defined, initialised and stored. |
| Icon | N/A |

| | |
|---|---|
| IP Owner & Partner in Charge | NUIG |
| INFINITECH Component Category | Data Semantics |
| IRA  - BDVA Layer | Data Processing |
| Input (Required by the Component) | SPARQL or CQELS query, data source and user information. |
| Output (Produced by the Component) | Boolean flag which will represent whether access is granted or denied. |
| Technology or Platform to be used | N/A |
| Part of INFINITECH Core | Yes |
| MARKETPLACE | Yes, it will be part of the Marketplace |
| Microservice | Yes, it is a dockerized microservice component |
| Endpoint/REST API | To be confirmed |
| License | To be confirmed |
| Other Information / Remarks | N/A |
| Detailed Documentation | N/A |

## 6.2 SeSA-ME APIs

The high-performance semantic analytics functionalities of the project are made available through Open APIs and will be deployed on the project's sandboxes and testbeds as described in the following sections. The APIs provided by SeSA-ME Engine are divided into two categories, namely Static Data APIs and Streaming Data APIs.

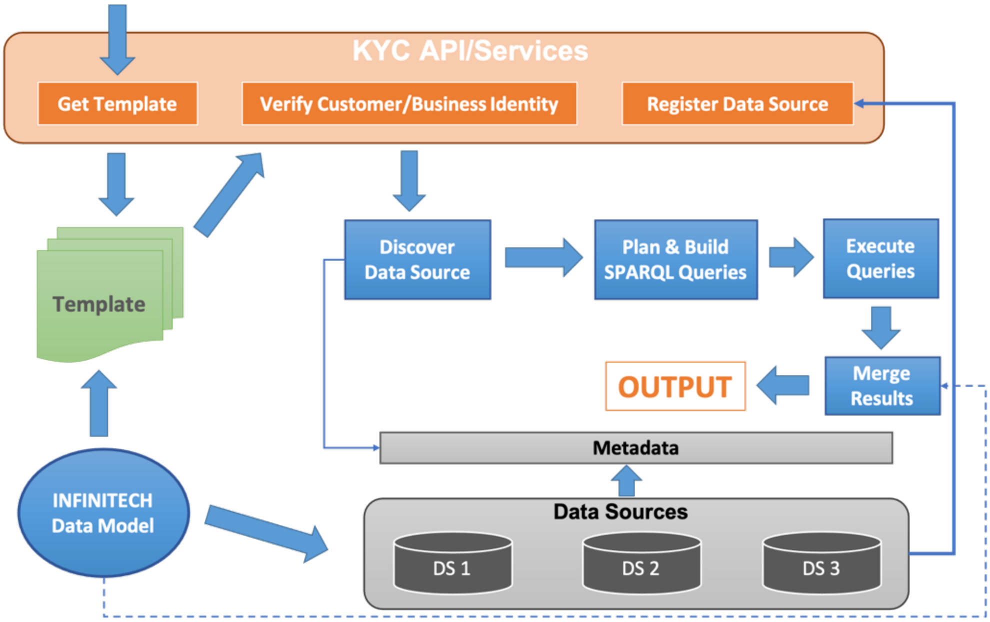


Figure 4. Semantic Stream Analytics Middleware-Engine API Services

### 6.2.1 Static Data APIs – Version I

#### 6.2.1.1 Know Your Customer (KYC) Profiler

Know Your Customer (KYC) is the process where businesses can verify the identity of their customer to ascertain the legitimacy and credibility. The KYC process is mostly used by financial institutions, such as banks, insurance companies etc. to verify their customers. This section describes RESTful APIs provided by SeSA-ME Engine for the KYC use case.

There are two perspectives of KYC, one is KYC Data Consumer, the consumer's perspective of KYC services and the other is KYC Data Provider, the data provider's perspective of KYC services. In the former, financial institutions consume the KYC services provided to verify the identity of their customers and in the later data providers provide their data to be used as a source for verifying the identity of customers.

##### 6.2.1.1.1 KYC Data Providers

This section describes the KYC APIs provided for the data providers, whose data can be used to verify the identity of customers. To be able to become a data provider for KYC services, the data source must get registered with SeSA-ME Engine.

###### *6.2.1.1.1.1* Data Source Registration API

The data source registration API is used to register a data source with SeSA-ME engine for the purpose of providing their data to be used for customer identity verification. The data source must supply the required information to this API. This information includes attributes, such as "name", "type" and "params" for this data source. The "type" attribute specifies the type of data source, e.g., SPARQL Endpoint, Data World Endpoint, Graph DB endpoint etc. The "params" attribute specifies the parameters needed to access the data source, e.g., access URL, username, passwords etc.

Table 19: Example Data Source Registration Information

| Attributes | Values |
|---|---|
| name | Bank of Ireland |
| type | SPARQL_ENDPOINT |
| accessURL | http://localhost:8890/sparql |

The details needed to use the data source registration API are listed in the table below. This table lists the example input and output in the form of JSON along with their JSON schemas.

Table 20: Example Register Data Source Functionality and URL notation

| | |
|---|---|
| **Functionality:** | **Register a Data Source** |
| **URL:** | **/registerDatasource** |
| **Method:** | **POST** |
| Registers a data source whose data can be used by KYC Data Consumers for verifying the identity of their customer. | |

Table 21: Example KYC Data Consumer Method using JSON Schema

| | | |
|---|---|---|
| Input | JSON example | {<br> "name": "DS-1",<br> "type": "SPARQL_ENDPOINT",<br> "params": {<br> "accessURL": "http://localhost:8890/sparql"<br> }<br>} |
| | JSON schema | {<br> "type": "object",<br> "properties": {<br> "name": {<br> "type": "string"<br> },<br> "type": {<br> "type": "string"<br> }, |

```
  "params": {
   "type": "object",
   "properties": {
    "accessURL": {
     "type": "string"
    }
   },
   "required": [
    "accessURL"
   ]
  }
 },
 "required": [
  "name",
  "type",
  "params"
 ]
}
```

Output

JSON example

```
{
 "message": "Data source is registered successfully."
}
```

JSON schema

```
{
 "type": "object",
 "properties": {
  "message": {
   "type": "string"
  }
 },
 "required": [
  "message"
 ]
}
```

#### 6.2.1.1.2 KYC Data Consumers

This section describes the KYC APIs provided for KYC data consumers, who can use these APIs to verify the identity of a customer. We have identified two scenarios in the KYC use case, i.e. Identity verification and Business Verification, described in the next sections. Templates for KYC Consumers

Table 22: Example Template for Identity Verification

| Attributes | Values |
|---|---|
| identifier | ABC-12345 |
| firstName | Martin |
| middleName | Serrano |
| surname | Orozco |
| dateOfBirth | 12-01-1975 |
| gender | Male |

| | |
|---|---|
| addressline1 | House No. 111 |
| addressline2 | Lower Dangan |
| addressline3 | Newcastle |
| city | Galway |
| postalCode | SE06 |

Table 23: Example Template for Business Verification

| Attributes | Values |
|---|---|
| registrationNumber | HDBAKSOWI12839HGD4747 |
| businessName | XYZ Inc. |
| dateOfIncorporation | 12-12-2012 |
| addressline1 | Building No. 13 |
| addressline2 | IDA Business Park |
| addressline3 | Newcastle |
| city | Galway |
| postalCode | SE06 |

###### *6.2.1.1.2.1* Get Template API (Identity Verification)

Table 24: Example Get Template Functionality and URL notation

| **Functionality:** | **Get Templates** |
|---|---|
| **URL:** | **/getTemplate** |
| **Method:** | **POST** |
| Get the template that should be provided for verification of an identity or any other purpose. | |

Table 25: Example Identity Verification method using JSON Schema

| | | |
|---|---|---|
| Input | JSON example | {<br>"fieldsFor": "Identity Verification"<br>} |

JSON schema

```
{
 "title": "ListFields",
 "type": "object",
 "properties": {
  "fieldsFor": {
   "title": "fieldsFor"
   "type": "string",
   "description": "The purpose for which fields are requested"
  }
 }
}
```

Output JSON example

```
{
 "dataFields": {
  "person": {
   "type": "object",
   "attributes": {
    "identifier": {
     "type": "string"
    },
    "firstName": {
     "type": "string"
    },
    "middleName": {
     "type": "string"
    },
    "surname": {
     "type": "string"
    },
    "maidenName": {
     "type": "string"
    },
    "dateOfBirth": {
     "type": "date"
    },
    "gender": {
     "type": "string"
    },
    "physicalAddress": {
     "type": "object",
     "attributes": {
      "addressline1": {
       "type": "string"
      },
      "addressline2": {
       "type": "string"
      },
      "addressline3": {
       "type": "string"
      },
      "city": {
       "type": "string"
      },
      "postalCode": {
       "type": "string"
      }
     }
    }
   }
  }
 }
}
```

JSON schema

```
{
 "title": "DataFields",
 "type": "object",
 "properties": {
  "dataFields": {
   "type": "object",
   "properties": {
    "person": {
     "type": "object",
     "properties": {
      "type": {
       "type": "string"
      },
      "attributes": {
       "type": "object",
       "properties": {
        "identifier": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "firstName": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "middleName": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "surname": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "maidenName": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "dateOfBirth": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
      "gender": {
        "type": "object",
        "properties": {
          "type": {
            "type": "string"
          }
        }
      },
      "physicalAddress": {
        "type": "object",
        "properties": {
          "type": {
            "type": "string"
          },
          "attributes": {
            "type": "object",
            "properties": {
              "addressline1": {
                "type": "object",
                "properties": {
                  "type": {
                    "type": "string"
                  }
                }
              },
              "addressline2": {
                "type": "object",
                "properties": {
                  "type": {
                    "type": "string"
                  }
                }
              },
              "addressline3": {
                "type": "object",
                "properties": {
                  "type": {
                    "type": "string"
                  }
                }
              },
              "city": {
                "type": "object",
                "properties": {
                  "type": {
                    "type": "string"
                  }
                }
              },
              "postalCode": {
                "type": "object",
                "properties": {
                  "type": {
                    "type": "string"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

```
    }
   }
  }
 }
}
```

#### 6.2.1.1.2.2 Get Templates API (Business Verification)

Table 26: Example Get List of Fields Functionality and URL notation

| **Functionality:** | **Get List of Fields** |
|---|---|
| **URL:** | **/listFields** |
| **Method:** | **POST** |
| Get the list of fields that should be provided for verification of a business or any other purpose. | |

Table 27: Example Business Verification method using JSON Schema

Input — JSON example

```
{
 "fieldsFor": "Business Verification"
}
```

JSON schema

```
{
 "type": "object",
 "properties": {
  "fieldsFor": {
   "title": "fieldsFor"
   "type": "string",
   "description": "The purpose for which fields are requested"
  }
 }
}
```

Output — JSON example

```
{
 "dataFields": {
  "business": {
   "type": "object",
   "attributes": {
    "registrationNumber": {
     "type": "string"
    },
    "businessName": {
     "type": "string"
    },
    "dateOfIncorporation": {
     "type": "date"
    },
    "physicalAddress": {
     "type": "object",
     "attributes": {
      "addressline1": {
       "type": "string"
      },
      "addressline2": {
       "type": "string"
      },
      "addressline3": {
       "type": "string"
      },
      "city": {
       "type": "string"
      },
      "postalCode": {
       "type": "string"
      }
     }
    }
   }
  }
 }
}
```

JSON schema

```
{
 "type": "object",
 "properties": {
  "dataFields": {
   "type": "object",
   "properties": {
    "business": {
     "type": "object",
     "properties": {
      "type": {
       "type": "string"
      },
      "attributes": {
       "type": "object",
       "properties": {
        "registrationNumber": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "businessName": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "dateOfIncorporation": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          }
         }
        },
        "physicalAddress": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          },
          "attributes": {
           "type": "object",
           "properties": {
            "addressline1": {
             "type": "object",
             "properties": {
              "type": {
               "type": "string"
              }
             }
            },
            "addressline2": {
             "type": "object",
             "properties": {
              "type": {
               "type": "string"
              }
             }
            },
            "addressline3": {
             "type": "object",
             "properties": {
              "type": {
               "type": "string"
              }
             }
            },
            "city": {
             "type": "object",
             "properties": {
              "type": {
               "type": "string"
              }
             }
            },
            "postalCode": {
             "type": "object",
             "properties": {
              "type": {
               "type": "string"
              }
             }
            }
           }
          }
         }
        }
       }
      }
     }
    }
   }
  }
 }
}
```

### 6.2.1.1.3 Identity Verification

#### *6.2.1.1.3.1* Verify Identity API

Table 28: Example Verify Customer Identity Functionality and URL notation

| **Functionality:** | **Verify Customer Identity** |
|---|---|
| **URL:** | **/verifyIdentity** |
| **Method:** | **POST** |

Verify the identity of a customer based on the customer information provided to the API.

Table 29: Example Verify Customer Identity method using JSON Schema

Input

JSON example

```
{
  "dataFields": {
    "person": {
      "identifier": "ABC 12345",
      "firstName": "Martin",
      "middleName": "Serrano",
      "surname": "Orozco",
      "dateOfBirth": "12-01-1975",
      "gender": "Male",
      "physicalAddress": {
        "addressline1": "House No. 111",
        "addressline2": "Lower Dangan",
        "addressline3": "Newcastle",
        "city": "Galway",
        "postalCode": "SE06"
      }
    }
  }
}
```

JSON schema

```
{
  "type": "object",
  "properties": {
    "dataFields": {
      "type": "object",
      "properties": {
        "person": {
          "type": "object",
          "properties": {
            "identifier": {
              "type": "string"
            },
            "firstName": {
              "type": "string"
            },
            "middleName": {
              "type": "string"
            },
            "surname": {
              "type": "string"
            },
            "maidenName": {
              "type": "string"
            },
            "dateOfBirth": {
              "type": "string"
            },
            "gender": {
              "type": "string"
            },
            "physicalAddress": {
              "type": "object",
              "properties": {
                "addressline1": {
                  "type": "string"
                },
                "addressline2": {
                  "type": "string"
                },
                "addressline3": {
                  "type": "string"
                },
                "city": {
                  "type": "string"
                },
                "postalCode": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Output JSON example

```
{
  "verificationId": "XYZ-22222-5555-DDD",
  "verificationDate": "2020-12-01T11:50:23",
  "verification": {
    "verificationStatus": "verified",
    "verificationResults": [
      {
        "verifiedFrom": "BOI",
        "verifiedAttributes": [
          {
            "attribute": "identifier",
            "status": "verified"
          },
          {
            "attribute": "firstName",
            "status": "verified"
          },
          {
            "attribute": "middleName",
            "status": "verified"
          },
          {
            "attribute": "surname",
            "status": "verified"
          },
          {
            "attribute": "dateOfBirth",
      "status": "verified"
     },
     {
      "attribute": "gender",
      "status": "verified"
     },
     {
      "attribute": "addressline1",
      "status": "verified"
     },
     {
      "attribute": "addressline2",
      "status": "verified"
     },
     {
      "attribute": "addressline3",
      "status": "verified"
     },
     {
      "attribute": "city",
      "status": "verified"
     },
     {
      "attribute": "postalCode",
      "status": "verified"
     }
    ]
   }
  ],
  "errors": [],
  "rule": {
   "ruleName": "",
   "ruleDescription": ""
  }
 }
}
```

JSON schema

```
{
 "type": "object",
 "properties": {
  "verificationId": {
   "type": "string"
  },
  "verificationDate": {
   "type": "string"
  },
  "verification": {
   "type": "object",
   "properties": {
    "verificationStatus": {
     "type": "string"
    },
    "verificationResults": {
     "type": "array",
     "items": [
      {
       "type": "object",
       "properties": {
        "verifiedFrom": {
         "type": "string"
        },
        "verifiedAttributes": {
         "type": "array",
         "items": [
{
 "type": "object",
 "properties": {
  "attribute": {
   "type": "string"
  },
  "status": {
   "type": "string"
  }
 }
},
{
 "type": "object",
 "properties": {
  "attribute": {
   "type": "string"
  },
  "status": {
   "type": "string"
  }
 }
},
{
 "type": "object",
 "properties": {
  "attribute": {
   "type": "string"
  },
  "status": {
   "type": "string"
  }
 }
},
{
 "type": "object",
 "properties": {
  "attribute": {
   "type": "string"
  },
  "status": {
   "type": "string"
  }
 }
},
{
 "type": "object",
 "properties": {
  "attribute": {
   "type": "string"
  },
  "status": {
   "type": "string"
  }
 }
},
{
 "type": "object",
 "properties": {
  "attribute": {
   "type": "string"
  },
  "status": {
   "type": "string"
  }
```

```
       }
      },
      {
       "type": "object",
       "properties": {
        "attribute": {
         "type": "string"
        },
        "status": {
         "type": "string"
        }
       }
      },
      {
       "type": "object",
       "properties": {
        "attribute": {
         "type": "string"
        },
        "status": {
         "type": "string"
        }
       }
      },
      {
       "type": "object",
       "properties": {
        "attribute": {
         "type": "string"
        },
        "status": {
         "type": "string"
        }
       }
      },
      {
       "type": "object",
       "properties": {
        "attribute": {
         "type": "string"
        },
        "status": {
         "type": "string"
        }
       }
      },
      {
       "type": "object",
       "properties": {
        "attribute": {
         "type": "string"
        },
        "status": {
         "type": "string"
        }
       }
      }
     ]
    }
   }
  }
 ]
},
"errors": {
     "type": "array",
     "items": {}
    },
    "rule": {
     "type": "object",
     "properties": {
      "ruleName": {
       "type": "string"
      },
      "ruleDescription": {
       "type": "string"
      }
     }
    }
   }
  }
 }
}
```

#### 6.2.1.1.4 Business Verification

##### *6.2.1.1.4.1* Verify Business API

Table 30: Example Verify Business API Functionality and URL notation

| | |
|---|---|
| **Functionality:** | **Verify Business** |
| **URL:** | **/verifyBusiness** |
| **Method:** | **POST** |

Verify a business based on the business information provided to the API.

Table 31: Example Verify Business method using JSON Schema

| | | |
|---|---|---|
| Input | JSON example | |

```
{
 "dataFields": {
  "business": {
   "registrationNumber": "HDBAKSOWI12839HGD4747",
   "businessName": "XYZ Inc.",
   "dateOfIncorporation": "12-12-2012",
   "physicalAddress": {
    "addressline1": "Building No. 13",
    "addressline2": "IDA Business Park",
    "addressline3": "Newcastle",
    "city": "Galway",
    "postalCode": "SE06"
   }
  }
 }
}
```

JSON schema

```
{
  "type": "object",
  "properties": {
    "dataFields": {
      "type": "object",
      "properties": {
        "business": {
          "type": "object",
          "properties": {
            "registrationNumber": {
              "type": "string"
            },
            "businessName": {
              "type": "string"
            },
            "dateOfIncorporation": {
              "type": "date"
            },
            "physicalAddress": {
              "type": "object",
              "properties": {
                "addressline1": {
                  "type": "string"
                },
                "addressline2": {
                  "type": "string"
                },
                "addressline3": {
                  "type": "string"
                },
                "city": {
                  "type": "string"
                },
                "postalCode": {
                  "type": "string"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Output JSON example

```
{
  "verificationId": "ASD-1133-333-456",
  "verificationDate": "2020-12-01T11:50:23",
  "verification": {
    "verificationStatus": "verified",
    "verificationResults": [
      {
        "verifiedFrom": "BOI",
        "verifiedAttributes": [
          {
            "attribute": "registrationNumber",
            "status": "verified"
          },
          {
            "attribute": "businessName",
            "status": "verified"
          },
     {
      "attribute": "dateOfIncorporation",
      "status": "verified"
     },
     {
      "attribute": "addressline1",
      "status": "verified"
     },
     {
      "attribute": "addressline2",
      "status": "verified"
     },
     {
      "attribute": "addressline3",
      "status": "verified"
     },
     {
      "attribute": "city",
      "status": "verified"
     },
     {
      "attribute": "postalCode",
      "status": "verified"
     }
    ]
   }
  ],
  "errors": [],
  "rule": {
   "ruleName": "",
   "ruleDescription": ""
  }
 }
}
```

JSON schema

```
{
 "type": "object",
 "properties": {
  "verificationId": {
   "type": "string"
  },
  "verificationDate": {
   "type": "string"
  },
  "verification": {
   "type": "object",
   "properties": {
    "verificationStatus": {
     "type": "string"
    },
    "verificationResults": {
     "type": "array",
     "items": [
      {
       "type": "object",
       "properties": {
        "verifiedFrom": {
         "type": "string"
        },
        "verifiedAttributes": {
         "type": "array",
         "items": [
          {
           "type": "object",
  "properties": {
    "attribute": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  }
},
{
  "type": "object",
  "properties": {
    "attribute": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  }
},
{
  "type": "object",
  "properties": {
    "attribute": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  }
},
{
  "type": "object",
  "properties": {
    "attribute": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  }
},
{
  "type": "object",
  "properties": {
    "attribute": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  }
},
{
  "type": "object",
  "properties": {
    "attribute": {
      "type": "string"
    },
    "status": {
      "type": "string"
    }
  }
},
                    {
                      "type": "object",
                      "properties": {
                        "attribute": {
                          "type": "string"
                        },
                        "status": {
                          "type": "string"
                        }
                      }
                    },
                    {
                      "type": "object",
                      "properties": {
                        "attribute": {
                          "type": "string"
                        },
                        "status": {
                          "type": "string"
                        }
                      }
                    }
                  ]
                }
              }
            }
          ]
        },
        "errors": {
          "type": "array",
          "items": {}
        },
        "rule": {
          "type": "object",
          "properties": {
            "ruleName": {
              "type": "string"
            },
            "ruleDescription": {
              "type": "string"
            }
          }
        }
      }
    }
  }
}
```

## 6.2.2 Streaming Data APIs – Version I

The second component provided by SeSA-ME Engine is the stream processor, which is responsible for processing multiple available linked data streams and providing the results to the consumers of data streams. This section describes the SeSA-ME APIs for streaming data.

### 6.2.2.1 Stream Registration

SeSA-ME Engine can process multiple streams available and to consume the available streams of data, the consumer first needs to register for these streams. The consumer needs to have the stream Ids and also callback URL for receiving back the stream. The callback URL should be a RESTful API and the streaming data should be received at this API. The technical details are provided in the next section.

#### 6.2.2.1.1 Register for Streams API

This API is used for registering for linked streams. The example JSON inputs and outputs along with their JSON schemas are provided below.

Table 32: Example Register for Streams API Functionality and URL notation

| | |
|---|---|
| **Functionality:** | **Register for Streams** |
| **URL:** | **/registerForStream** |
| **Method:** | **POST** |
| Registers for a stream or a list of streams. | |

Table 33: Example Register for Streams method using JSON Schema

JSON Example

```
{
 "callbackURL": "http://localhost/sesame-client/getRDFStream",
 "streams": [
  {
   "streamId": "http://infinitech.eu/rdf/stream-2",
  },
  {
   "streamId": "http://infinitech.eu/rdf/stream-4",
  }
 ]
}
```

```
{
 "type": "object",
 "properties": {
  "callbackURL": {
   "type": "string"
  },
  "streams": {
   "type": "array",
   "items": [
    {
     "type": "object",
     "properties": {
      "streamId": {
       "type": "string"
      }
     },
     "required": [
      "streamId"
```

```
          ]
        },
        {
          "type": "object",
          "properties": {
            "streamId": {
              "type": "string"
            }
          },
          "required": [
            "streamId"
          ]
        }
      ]
    }
  },
  "required": [
    "callbackURL",
    "streams"
  ]
}
```

```
{
  "message": "You have successfully registered for the requested streams."
}
```

```
{
  "type": "object",
  "properties": {
    "message": {
      "type": "string"
    }
  },
  "required": [
    "message"
  ]
}
```

## 6.2.3 SeSA-ME New APIS Implementation – Version II

The second version implementation component provided by SeSA-ME Engine is the stream processor, which is responsible for processing multiple available linked data streams and providing the results to the consumers of data streams. This section describes the SeSA-ME APIs for streaming data.

### 6.2.3.1 Stream Registration II

SeSA-ME Engine can process multiple streams available and to consume the available streams of data, the consumer first needs to register for these streams. The consumer needs to have the stream Ids and also callback URL for receiving back the stream. The callback URL should be a RESTful API and the streaming data should be received at this API. The technical details are provided in the next section.

#### 6.2.3.1.1 Unregister Data Source API

The details needed to use the un-register data source API are listed in the table below. This table lists the example input and output in the form of JSON along with their JSON schemas.

Table 34: Example for Unregister Data Source API Functionality and URL notation

| | |
|---|---|
| **Functionality:** | **Un-register a Data Source** |
| **URL:** | **/unregisterDatasource** |
| **Method:** | **POST** |

The method to unregister a data source already registered with SeSA-ME and what data can be used.

Table 35: Example unregister a Data Source method using JSON Schema

| | | |
|---|---|---|
| Input | JSON example | {<br>"datasource": "http://localhost:8890/sparql"<br>} |
| | JSON schema | {<br>"type": "object",<br>"properties": {<br>"datasource": {<br>"type": "string"<br>}<br>},<br>"required": [<br>"datasource"<br>]<br>} |
| Output | JSON example | {<br>"message": "Data source is un-registered successfully."<br>} |
| | JSON schema | {<br>"type": "object",<br>"properties": {<br>"message": {<br>"type": "string"<br>}<br>},<br>"required": [<br>"message"<br>]<br>} |

#### 6.2.3.1.2 Run Query API

The details needed to use the run query API are listed in the table below. This table lists the example input and output in the form of JSON along with their JSON schemas.

Table 36: Example for Unregister Data Source API Functionality and URL notation

| | |
|---|---|
| **Functionality:** | **Run Query on a Data Source** |
| **URL:** | **/runQuery** |
| **Method:** | **POST** |

Runs a SPARQL query on the specified data source and returns back the obtained results.

Table 37: Example Runs a SPARQL query on the specified data source

| | | |
|---|---|---|
| Input | JSON example | {<br> "datasource": "http://localhost:8890/sparql",<br> "query": "SELECT * WHERE { ?s ?p ?o . } LIMIT 1"<br>} |
| | JSON schema | {<br> "type": "object",<br> "properties": {<br>  "datasource": {<br>   "type": "string"<br>  },<br>  "query": {<br>   "type": "string"<br>  }<br> },<br> "required": [<br>  "datasource",<br>  "query"<br> ]<br>} |
| Output | JSON example | {<br> "head": {<br>  "vars": [<br>   "s",<br>   "p",<br>   "o"<br>  ]<br> },<br> "results": {<br>  "bindings": [<br>   {<br>    "s": { |

```
          "type": "uri",
          "value": "http://infinitechproject.eu/data/joabos"
        },
        "p": {
          "type": "uri",
          "value": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
        },
        "o": {
          "type": "uri",
          "value": "https://spec.edmcouncil.org/fibo/ontology/FND/AgentsAndPeople/People/Person"
        }
      }
    ]
  }
}
```

JSON schema

```
{
  "type": "object",
  "properties": {
    "head": {
      "type": "object",
      "properties": {
        "vars": {
          "type": "array",
          "items": [
            {
              "type": "string"
            },
            {
              "type": "string"
            },
            {
              "type": "string"
            }
          ]
        }
      },
      "required": [
        "vars"
      ]
    },
    "results": {
      "type": "object",
      "properties": {
        "bindings": {
          "type": "array",
          "items": [
            {
              "type": "object",
              "properties": {
                "s": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                },
                "p": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                },
                "o": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                }
              }
            }
          ]
        }
      },
      "required": [
        "bindings"
      ]
    }
  },
  "required": [
    "head",
    "results"
  ]
}
```

#### 6.2.3.1.3 Run Query Plan API

The details needed to use the run query plan API are listed in the table below. This table lists the example input and output in the form of JSON along with their JSON schemas.

Table 38: Example for Run Query API Functionality and URL notation

| | |
|---|---|
| **Functionality:** | **Run a query plan** |
| **URL:** | **/runQueryPlan** |
| **Method:** | **POST** |

Table 39: Example Run a query plan on both static and streaming data sources

Input

JSON example

```
{
 "staticRequest": [
  {
   "datasource": "http://localhost:8890/sparql",
   "query": "SELECT * FROM <http://infinitechproject.eu/graph> WHERE { ?s ?p ?o . } LIMIT 1"
  }
 ],
 "streamRequest": [
  {
   "callbackURL": "http://10.196.2.55:8082/sesame-client/getRDFStream",
   "streams": [
    {
     "streamId": "http://infinitech.eu/rdf/stream-1"
    }
   ]
  }
 ]
}
```

JSON schema

```
{
 "type": "object",
 "properties": {
  "staticRequest": {
   "type": "array",
   "items": [
    {
     "type": "object",
     "properties": {
      "datasource": {
       "type": "string"
      },
      "query": {
       "type": "string"
      }
     },
     "required": [
      "datasource",
      "query"
     ]
    }
   ]
  },
  "streamRequest": {
   "type": "array",
   "items": [
    {
     "type": "object",
     "properties": {
      "callbackURL": {
       "type": "string"
      },
      "streams": {
       "type": "array",
       "items": [
        {
         "type": "object",
         "properties": {
          "streamId": {
           "type": "string"
          }
         },
         "required": [
          "streamId"
         ]
        }
       ]
      }
     },
     "required": [
      "callbackURL",
      "streams"
     ]
    }
   ]
  }
 },
 "required": [
  "staticRequest",
  "streamRequest"
 ]
}
```

Output for static request

JSON example

```
{
 "head": {
  "vars": [
   "s",
   "p",
   "o"
  ]
 },
 "results": {
  "bindings": [
   {
    "s": {
     "type": "uri",
     "value": "http://infinitechproject.eu/data/joabos"
    },
    "p": {
     "type": "uri",
     "value": "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
    },
    "o": {
     "type": "uri",
     "value":
"https://spec.edmcouncil.org/fibo/ontology/FND/AgentsAndPeople/People/Person"
    }
   }
  ]
 }
}
```

JSON schema

```
{
  "type": "object",
  "properties": {
    "head": {
      "type": "object",
      "properties": {
        "vars": {
          "type": "array",
          "items": [
            {
              "type": "string"
            },
            {
              "type": "string"
            },
            {
              "type": "string"
            }
          ]
        }
      },
      "required": [
        "vars"
      ]
    },
    "results": {
      "type": "object",
      "properties": {
        "bindings": {
          "type": "array",
          "items": [
            {
              "type": "object",
              "properties": {
                "s": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                },
                "p": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                },
        "o": {
         "type": "object",
         "properties": {
          "type": {
           "type": "string"
          },
          "value": {
           "type": "string"
          }
         },
         "required": [
          "type",
          "value"
         ]
        }
       }
      }
     ]
    }
   },
   "required": [
    "bindings"
   ]
  }
 },
 "required": [
  "head",
  "results"
 ]
}
```

Output for stream request

JSON example

```
{
 "head": {
  "vars": [
   "s",
   "p",
   "o"
  ]
 },
 "results": {
  "bindings": [
   {
    "s": {
     "type": "uri",
     "value": "http://infinitech.eu/rdf/customer-41"
    },
    "p": {
     "type": "uri",
     "value":
"https://spec.edmcouncil.org/fibo/ontology/FND/AgentsAndPeople/People/hasLastName"
    },
    "o": {
     "datatype": "http://www.w3.org/2001/XMLSchema#string",
     "type": "typed-literal",
     "value": "Last Name 41"
    }
   }
  ]
 }
}
```

JSON schema

```
{
  "type": "object",
  "properties": {
    "head": {
      "type": "object",
      "properties": {
        "vars": {
          "type": "array",
          "items": [
            {
              "type": "string"
            },
            {
              "type": "string"
            },
            {
              "type": "string"
            }
          ]
        }
      },
      "required": [
        "vars"
      ]
    },
    "results": {
      "type": "object",
      "properties": {
        "bindings": {
          "type": "array",
          "items": [
            {
              "type": "object",
              "properties": {
                "s": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                },
                "p": {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "type",
                    "value"
                  ]
                },
                "o": {
                  "type": "object",
                  "properties": {
                    "datatype": {
                      "type": "string"
                    },
                    "type": {
                      "type": "string"
                    },
                    "value": {
                      "type": "string"
                    }
                  },
                  "required": [
                    "datatype",
                    "type",
                    "value"
                  ]
                }
              }
            }
          ]
        }
      },
      "required": [
        "bindings"
      ]
    }
  },
  "required": [
    "head",
    "results"
  ]
}
```

# 7 SeSA-ME Continuous Integration/Continuous Development

The following CI/CD documentation is considering you have an instance of the SeSA-ME components running in your machine and all the file and ports are accessible/configured as local host, to run SeSA-ME Engine use the below commands:

## 7.1 SeSA-ME Engine Development

### 7.1.1 Run SeSA-ME Engine

To run SeSA-ME Engine use the below command:

```
mvnw spring-boot:run
```

Then Navigate to the below URL to check SeSA-ME Engine.

http://localhost:8080/sesame-engine/swagger-ui.html

### 7.1.2 Build SeSA-ME Engine

To build SeSA-ME Engine run the below command:

```
mvnw package
```

## 7.2 Deployment using Docker

**Step 1:** Use the following command to build a local docker image of SeSA-ME Engine.

```
docker build -t sesame-engine-image-local .
```

**Step 2:** Use the following command to deploy the docker image of SeSA-ME Engine built in the previous step.

```
docker run --name sesame-container -d -p 8080:8080 sesame-engine-image-local
```

**Step 3:** Navigate to the below URL to check SeSA-ME Engine.

http://localhost:8080/sesame-engine/swagger-ui.html

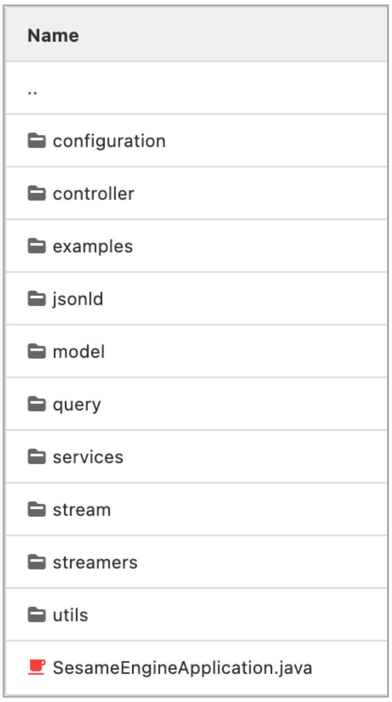
## 7.3 SeSA-ME Engine APIs

The description of SeSA-ME Engine APIs can be found in the section 4 in this deliverable

## 7.4 SeSA-ME Engine Deployment

### 7.4.1 SeSA-ME Engine Project Structure

Table 40: The SeSA-ME component has the following structure:



| Name |
|---|
| .. |
| configuration |
| controller |
| examples |
| jsonld |
| model |
| query |
| services |
| stream |
| streamers |
| utils |
| SesameEngineApplication.java |

### 7.4.2 Dependencies List

Table 41: SeSA-ME Dependecies

**Example RDF Data**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
        <modelVersion>4.0.0</modelVersion>

        <groupId>eu.infinitech.nuig</groupId>
        <artifactId>sesame-engine</artifactId>
        <version>0.0.1-SNAPSHOT</version>
        <packaging>jar</packaging>

        <name>sesame-engine</name>
        <description>Sesame Engine</description>

        <parent>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-starter-parent</artifactId>
                <version>1.5.4.RELEASE</version>
                <relativePath /> <!-- lookup parent from repository -->
        </parent>

        <properties>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
                <java.version>1.8</java.version>
        </properties>

        <dependencies>
                <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-swagger2</artifactId>
                    <version>2.6.1</version>
                </dependency>

                <dependency>
                    <groupId>io.springfox</groupId>
                    <artifactId>springfox-swagger-ui</artifactId>
                    <version>2.6.1</version>
                </dependency>

                <!-- uncomment below dependency if want to build a war -->
                <!-- <dependency>
                  <groupId>org.springframework.boot</groupId>
                  <artifactId>spring-boot-starter-tomcat</artifactId>
                  <scope>provided</scope>
                </dependency> -->

                <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-actuator</artifactId>
                </dependency>
                <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-data-rest</artifactId>
                </dependency>
                <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-web</artifactId>
                </dependency>

                <dependency>
                    <groupId>org.springframework.boot</groupId>
                    <artifactId>spring-boot-starter-test</artifactId>
                    <scope>test</scope>
                </dependency>

                <dependency>
                    <groupId>eu.larkc.csparql</groupId>
                    <artifactId>csparql-core</artifactId>
                    <version>0.9.6</version>
                </dependency>

                <dependency>
                  <groupId>com.squareup.okhttp</groupId>
                  <artifactId>okhttp</artifactId>
                  <version>2.7.5</version>
                </dependency>

                <dependency>
                  <groupId>com.apicatalog</groupId>
                  <artifactId>titanium-json-ld</artifactId>
                  <version>1.0.0</version>
                </dependency>

                <dependency>
              <groupId>org.glassfish</groupId>
              <artifactId>jakarta.json</artifactId>
              <version>2.0.0</version>
```

```
     </dependency>

    </dependencies>

    <repositories>
        <repository>
            <id>maven Repo1</id>
            <name>maven Repo1</name>
            <url>http://repo1.maven.org/maven2</url>
        </repository>
        <repository>
            <id>Sonatype repository</id>
            <name>Sonatype's Maven repository</name>
            <url>http://oss.sonatype.org/content/groups/public</url>
        </repository>
        <repository>
            <id>streamreasoning_repository</id>
            <name>streamreasoning repository</name>
            <url>http://streamreasoning.org/maven/</url>
            <layout>default</layout>
        </repository>

    </repositories>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>


</project>
```

## 7.4.3 Docker File

Table 42: Docker File

| Example RDF Data |
|---|
| FROM openjdk:8-jdk-alpine<br>RUN addgroup -S spring && adduser -S spring -G spring<br>USER spring:spring<br>ARG JAR_FILE=target/sesame-engine-0.0.1-SNAPSHOT.jar<br>ADD ${JAR_FILE} app.jar<br>ENTRYPOINT ["java","-jar","/app.jar"] |

# 8 Conclusions

The Semantic Stream Analytics Engine (SeSA-ME) has been designed and implemented in the context of the INFINITECH project, it is an extension of the Super Stream Collider (SSC) framework, which provides a set of web-based interfaces and tools for building data mashups combining semantically annotated Linked Stream and Linked Data sources into easy-to-use data mashups for applications. The SeSA-ME engine make uses of online tools to includes static data and dynamic data (streams) along with a visual SPARQL query editor using Swagger APIs and visualization tools for novice users while supporting full access and control over the data mashups for expert users. The SeSA-ME engine has been provided as a Docker file to facilitate the use and integration of the tool.

The INFINITECH graph data model is made accessible online using machine readable files and also for human understanding and manipulation. The development and deployment of the INFINITECH Graph Data Model enables the support for both the design and deployment of stream-based web applications in a very simple and intuitive way and the analytics services using stream-based applications and services is tied with the development of the SeSA-ME platform.

The main ontologies used as baselines are FIBO, FIGI and LKIF, because they focused on both financial sector and financial operations containing the baseline for the metadata that represent, cross-domain and intra domain, financial transactions, and operations with an attached effort towards standardisation. Additional ontologies used as extensions towards particular pilots can be developed and further integrated withing the data pack following the documentation provided in this deliverable. The INFINTECH Core ontology is an extension generated in the project that describes cross-domain vocabularies that are used in multi-domains within the INFINITECH project domain areas, it is meant to be complemented by other domain specific vocabularies.

The INFINITECH project have facilitated the design and implemented of the Semantic Stream Analytics Middleware-Engine (SeSA-ME) and provided a thorough analysis about the already existing ontologies that are related to the finance and insurance sectors that can be reused for our purposes in the INFINITECH project. The INFINITECH graph data model facilitates a semantic interoperability aspects that are necessary to process, exchange and share data across different components, systems and platforms. The SeSA-ME engine enables a semantic layer approach that constitutes also the first step of the INFINITECH pipeline, i.e., gathering semantically annotated data from provided and/or available datasets or data streams.

In this deliverable, we have described how INFINITECH project would benefit from semantic technologies like Linked Data and ontologies as the best practices in the semantic interoperability building process. The implementation provided and documented in the three deliverables i.e. D4.4, D4.5 and D4.6 is a reference implementation that was improved following general requirements coming from the study and purposes at INFINITECH pilots and following stakeholder’s requirements from particular domains demonstrating that the use of semantic technologies is possible and that the benefits of semantic technologies and the use of a semantic engine benefits the construction and operation of data mashups and that a semantic engine can cope with resolving some of the semantic interoperability requirements.

# 9 References

[Boots 2017] Botts, M., Percivall, G., Reed, C. and Davidson. J. *OGC Sensor Web Enablement: Overview and High Level Architecture*. Technical report, OGC, December 2007.

[Compton et al 2012] Compton, M., Barnaghi, P., Bermudez, L., Castro, R. G., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K. Kelsey, W. D., Phuoc, D. L., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheath, A. and Taylor, K. *The SSN Ontology of the Semantic Sensor Networks Incubator Group*. Journal of Web Semantics: Science, Services and Agents on the World Wide Web, ISSN 1570-8268, Elsevier, 2012.

[DOI] Document Object Identifier
Accessible here: http://www.doi.org/handbook_2000/DOIHandbook-v4-4.1.pdf

[EPCGlobal-RPS 2006] EPCglobal: Reader Protocol Standard, Version 1.1, 3 Ratified Standard, 4 June 21, 2006

[EPCGlobal-ALE 2009] EPCglobal: The Application Level Events (ALE) Specification, Version 1.1.1 Part I: Core Specification, EPCglobal Ratified Standard, 13 March 2009

[EPCGlobal-A 2007] EPCglobal: The EPCglobal Architecture Framework, EPCglobal Final Version 1.2 Approved 10 September 2007

[EPCGlobal-EPC 2007] EPCglobal: EPC Information Services (EPCIS) Version 1.0.1 Specification Approved September 21, 2007

[EPCGlobal-RMS 2007] EPCglobal: Reader Management Standard 1.0.1, 3 May 31, 2007

[GENE-Ontology] GENE Ontology - bioinformatics initiative
Accessible here: http://www.geneontology.org

[Heitman 2009] Heitmann, B., Kinsella, S., Hayes, C. and Decker, S. *Implementing Semantic Web Applications: Reference Architecture and Challenges*. In International Workshop on Semantic Web enabled Software Engineering, collocated with the 8th International Semantic Web Conference (ISWC2009), 2009.

[Henson 2009] Henson, C. A., Pschorr, J. K., Sheth, A. P. and Thirunarayan, K. *SemSOS: Semantic sensor observation service*. Collaborative Technologies and Systems, International Symposium on, 0:44–53, 2009.

[Jacobs 2004] Jacobs, I. and Walsh, N. *Architecture of the World Wide Web*, Volume One, World Wide Web Consortium, Recommendation REC-webarch-20041215, 2004.

[Le-Phuoc et al. 2011a] Le-Phuoc, D., Dao-Tran, M. Parreira, J. X. and Hauswirth, M. *A Native and Adaptive Approach for Unified Processing of Linked Streams and Linked Data*. Proceedings of the 10th International Conference on The Semantic Web (ISWC'11), Springer, 2011

[Le-Phuoc et al. 2011b] Le-Phuoc, D., Nguyen Mau, H., Parreira, J. X. and Hauswirth, M.. *The Linked Sensor Middleware – Connecting the Real World and the Semantic Web*. Proceedings of the 10th International Conference on The Semantic Web (ISWC'11), Springer, 2011

[Le-Phuoc et al. 2009] Le-Phuoc, D. and Hauswirth, M. *Linked open data in sensor data mashups*. Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN09) in conjunction with ISWC 2009

[LOD-Project] World Wide Web Consortium - Linked Open Data, Accessible here: http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData

[Priest 2007] Priest, A. Na, M., Niedzwiadek, H. and Davidson, J. Sensor observation service. Technical Report OGC 06-009r6, October 2007.

[Ruta 2007] Ruta, M. , Noia, T. Di, Scioscia, F., Di Sciascio E. Semantic-enhanced EPCglobal Radio-Frequency IDentification. SWAP 2007

[Salehi 2007] Salehi, A., Aberer, K. «GSN, Quick and Simple Sensor Network Deployment», European conference on Wireless Sensor Networks (EWSN), Netherlands, 2007

[Scherp 2009] Scherp, A., Franz, T. Saatho, S. Staab. *F–a Model of Events Based on the Foundational Ontology DOLCE+DnS Ultralight*. In: International Conference on Knowledge Capturing (K-CAP), Redondo Beach, CA, USA., 2009.

[Sheth 2008] Sheth, A. Henson, C., Sahoo. S. *Semantic Sensor Web*. IEEE Internet Computing 12 (4), 2008.

[Tsiatsis 2010] Tsiatsis, V., Gluhak, A., Bauge, T., Montagut, F., Bernat, J., Bauer, M., Villalonga, C., Barnaghi, P.M., Krco, S. The SENSEI Real World Internet Architecture. Future Internet Assembly, IOS Press, 2010.

[UMLS] Unified Medical Language System
Accessible here: http://www.nlm.nih.gov/ research/umls/index.html

[IETF-RFC2141] IETF - Uniform Resource Names
Accessible here: http://tools.ietf.org/html/rfc2141

[W3C-RDF] World Wide Web Consortium - Resource Description Framework,
Accessible here: http://www.w3.org/TR/rdf-syntax-grammar/

[W3C-RDFSchema] World Wide Web Consortium - Resource Description Framework Schema
Accessible here: http:// www.w3.org/ TR/ rdf-schema

[W3C-Turtle] World Wide Web Consortium - Turtle Serialisation Specification
Accessible here: http://www.w3.org/TeamSubmission/turtle/

[W3C-N-Triples] World Wide Web Consortium - N-Triples format specification
Accessible here: http://www.w3.org/TR/rdf-testcases/#ntriples

W3C-OWL] World Wide Web Consortium – Ontology Web language
Accessible here: http:// www.w3.org/ TR/ owl-ref

[W3C RDFa] World Wide Web Consortium - Resource Description Framework in Attributes
Accessible here: http://www.w3.org/TR/xhtml-rdfa-primer/

[W3C-SPARQL] SPARQL Query Language for RDF Implementation
Accessible here: http://www.w3.org/TR/rdf-sparql-query/