



Grant Agreement No.: 604590
Instrument: Large scale integrating project (IP)
Call Identifier: FP7-2012-ICT-FI



eXperimental Infrastructures for the Future Internet

D4.4: Baseline Tools v2

Revision: v.1.0

Work package	WP 4
Task	T4.1 to T4.5
Due date	30/09/2014
Submission date	03/10/2014
Deliverable lead	ATOS
Authors	Joaquin Iranzo, Susana Gonzalez Zarzosa, Ömer Özdemir (ATOS) Paul Grace, Bassem I. Nasser (IT-IN) Kreshnik Musaraj, Cyril Dangerville (THALES), Alvaro Alonso, Jorge Valhondo (UPM), Katarzyna Di Meo, Federico M. Facca (CREATE-NET)
Reviewers	Riwal Kerherve (imaginLab), Panos Trakadas (SYNELIXIS)

Abstract	This deliverable covers the second release of tools described in T4.2 to T4.5. These tools rely on components coming from FIWARE as well as those provided in WP2 and WP3. The scope will cover all those features required by the 5 initial nodes of the XIFI federation; taking into account new infrastructures joining the federation.
Keywords	Configuration, installation, user manual, specification, Generic Enabler, FIWARE

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	17.09. 2014	First version of the integrated document.	Joaquín Iranzo (ATOS)
V0.2	25.09.2014	Integration of the components contributions.	Joaquín Iranzo (ATOS), Susana Gonzalez (ATOS)
V0.3	29.09.2014	Final versión for internal review	Joaquín Iranzo (ATOS)
V0.4	02.10.2014	Integration of comments and final version.	Joaquin Iranzo (ATOS), all partners
V1.0	10.10.2014	Final version ready for submission	

Disclaimer

This report contains material which is the copyright of certain XIFI Consortium Parties and may only be reproduced or copied with permission in accordance with the XIFI consortium agreement.

All XIFI Consortium Parties have agreed to publication of this report, the content of which is licensed under a Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License¹.

Neither the XIFI Consortium Parties nor the European Union warrant that the information contained in the report is capable of use, or that use of the information is free from risk, and accept no liability for loss or damage suffered by any person using the information.

Copyright notice

© 2013 - 2015 XIFI Consortium Parties

Project co-funded by the European Commission in the 7 th Framework Programme (2007-2013)		
Nature of the Deliverable:		P (Prototype)
Dissemination Level		
PU	Public	✓
PP	Restricted to other programme participants (including the Commission Services)	
RE	Restricted to bodies determined by the XIFI project	
CO	Confidential to XIFI project and Commission Services	

¹http://creativecommons.org/licenses/by-nc-nd/3.0/deed.en_US

EXECUTIVE SUMMARY

The present deliverable “**D4.4 Baseline Tools v2**” contains the second software version of components described in the framework of “WP4 Services & Tools” within the XIFI project. This deliverable is composed of two main parts:

- i) source code developed and implemented in XIFI, being available in the public version control system [1]
- ii) the corresponding description per available component and the user guide (integrated in this document).

Both parts are based on the specifications defined in “D4.1b- Services and tools specification” [2] and it is aligned with the rest of the components from WP2 and WP3. This document is the second version of the document “D4.2 Baseline Tools v1” [95], focusing on development and functionality updates compared to the previous version of D4.2. Some sections have not changed from the previous version, nevertheless, the document includes the whole structure per every component. The intention of this document is to be self-contained and avoid that the reader needs to change between different documents to understand the analysed component. Hence, when one component has not changed with respect to the previous version, only its reference is included, indicating where to find the component information. On the other hand, if the component has any changes (modifications, functionality improvements, API extensions, etc), the document includes all the sections related to this component.

There are some components, which have introduced changes from the previous version: the Resource Catalogue and Recommendation Tool, the SLA (Service Level Agreement) Manager, the Interoperability Tool, the Security Dashboard, the Infographics and Status Pages, the Security Monitoring Generic Enabler and the Cloud Portal. On the other hand, the FIWARE LAB APP Template component, which is also included in the previous version, has not introduced changes from the previous version and it is only referenced herein.

Besides, there are two new components in this version: the Monitoring Dashboard and the Privacy Recommendation Service. The description of these new components has the same common structure.

For all the above mentioned components, the description follows the same structure (to facilitate understanding for its readers) also part of the reference manual (constantly updated in the project wiki). Moreover, there is a summary of the components in XIFI and their dependencies. This is mainly an overview of each component, providing a motivation of its existence. Then, the user stories backlog describes the different user stories that have to be implemented. Also the State of the Art, which explains the different available solutions; also where the component is presented in relation to the XIFI architecture. Then, the release plan is also set out, with all details of all steps to follow and the planned sprints. Also the Test Case defines the list of tests that can be executed.

Finally, there is a detailed Installation Manual, describing all the steps to be configured, installed and used. It provides information on how to get started (software dependencies, hardware requirements and platforms supported, installation instructions). Finally, the User Manual describes how to use them both through the graphical interface and the APIs.

Moreover, the document presents a conclusion and future transversely collaboration works (cross-WP) and updates to be achieved in the upcoming deliverables within WP4.

TABLE OF CONTENTS

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS	4
LIST OF FIGURES	7
LIST OF TABLES	10
ABBREVIATIONS	12
1 INTRODUCTION	13
1.1 Purpose	13
1.2 Overview	13
2 RESOURCE CATALOGUE & RECOMMENDATION TOOL	16
2.1 Summary	16
2.2 Component leaders	17
2.3 Motivation	17
2.4 User stories backlog	18
2.5 State of the art	20
2.6 Architecture design	23
2.7 Release plan	26
2.8 Test case	26
2.9 Installation Manual	26
2.10 User Manual	27
3 SLA MANAGER	39
3.1 Summary	39
3.2 Component leaders	40
3.3 Motivation	40
3.4 User stories backlog	41
3.5 State of the art	42
3.6 Architecture design	44
3.7 Release plan	47
3.8 Test cases	47
3.9 Installation Manual	51
3.10 User Manual	55
4 INTEROPERABILITY TOOL	77
4.1 Summary	77
4.2 Component leaders	78
4.3 Motivation	78
4.4 User stories backlog	79

4.5	State of the art	79
4.6	Architecture design	81
4.7	Release plan	86
4.8	Test Cases	86
4.9	Installation Manual	88
4.10	User Manual.....	91
5	SECURITY DASHBOARD	103
5.1	Summary	103
5.2	Component leaders	104
5.3	Motivation.....	104
5.4	User stories backlog.....	105
5.5	State of the art	106
5.6	Architecture design	107
5.7	Release plan	108
5.8	Test cases	108
5.9	Installation Manual	110
5.10	User Manual.....	114
6	INFOGRAPHICS AND STATUS PAGES.....	126
6.1	Summary	126
6.2	Component leaders	127
6.3	Motivation.....	127
6.4	User stories backlog.....	127
6.5	State of the art	129
6.6	Architecture design	130
6.7	Release plan	131
6.8	Test cases	131
6.9	Installation Manual	133
6.10	User Manual.....	134
7	SECURITY MONITORING GE	136
7.1	Summary	136
7.2	Component leaders	137
7.3	User stories backlog.....	137
7.4	Architecture design	138
7.5	Release plan	140
7.6	Test cases	140
7.7	Installation Manual	142
7.8	User Manual.....	149

8	CLOUD PORTAL	153
8.1	Summary	153
8.2	Component leaders	154
8.3	Motivation.....	154
8.4	User stories backlog.....	154
8.5	State of the art.....	155
8.6	Architecture design.....	155
8.7	Release plan.....	156
8.8	Test cases	156
8.9	Installation Manual	159
8.10	User Manual.....	159
9	FIWARE LAB APP TEMPLATE.....	162
10	MONITORING DASHBOARD	163
10.1	Summary	163
10.2	Component leaders	164
10.3	Motivation.....	164
10.4	User stories backlog.....	164
10.5	State of the art.....	165
10.6	Architecture design.....	165
10.7	Release plan.....	166
10.8	Test cases	166
10.9	Installation Manual	166
10.10	User Manual.....	168
11	POLICY RECOMMENDATION SERVICE.....	170
11.1	Summary	170
11.2	Component leaders	171
11.3	Motivation.....	171
11.4	User stories backlog.....	171
11.5	State of the art.....	172
11.6	Architecture design.....	176
11.7	Release plan.....	178
11.8	Test cases	178
11.9	Installation Manual	178
11.10	User Manual.....	179
12	CONCLUSIONS	180
	REFERENCES	181

LIST OF FIGURES

Figure 1: Location of the Portal and Resource Catalogue in the XIFI Reference Architecture	16
Figure 2: Resource Catalogue FMC compositional structure diagram	23
<i>Figure 3: Recommendation Tool model</i>	<i>25</i>
Figure 4: Resource Catalogue IdM Authentication.....	28
Figure 5: Initial page after the user has been identified.	28
Figure 6: Resource Catalogue – Catalogue section.....	29
Figure 7: Resource Catalogue – List of generic Enablers.	30
Figure 8: Resource Catalogue – Browse by keyword.	31
Figure 9: Resource Catalogue – fixing the node Trento	31
Figure 10: Resource Catalogue – Generic Enable Detail.....	32
Figure 11: Resource Catalogue.	32
Figure 12: Resource Catalogue – selecting the chapter.	32
Figure 13: Resource Catalogue – Search with keyword.	33
Figure 14: Resource Catalogue – Search by map and node details.	33
Figure 15: Resource Catalogue – GE description.	34
Figure 16: Resource Catalogue, create new offering.	35
Figure 17: Resource Catalogue, created new offering	35
Figure 18: Description of the services that will be included in the Catalogue.....	35
Figure 19: List of services pending to be validated.....	36
Figure 20: Specific Enabler available in the Catalogue.	36
Figure 21: Resource Catalogue – Non Conventional Services	37
Figure 22: Resource Catalogue – Other Services and keyword search.....	37
Figure 23: Resource Catalogue – Detail Non Conventional Services or SE.....	38
Figure 24: Location of the SLA Manager in the XIFI Reference Architecture	39
Figure 25: WS-Agreement schema	44
Figure 26: SLA Manager FMC compositional structure diagram.....	45
<i>Figure 27: Test 10 Verification of the SLA Dashboard installation</i>	<i>50</i>
<i>Figure 28: Step1 to add a new application in the IdM.....</i>	<i>53</i>
<i>Figure 29: Step2 to add a new application in the IdM.....</i>	<i>54</i>
<i>Figure 30: Step3 to add a new application in the IdM.....</i>	<i>54</i>
<i>Figure 31: Detail of the credentials for this application.</i>	<i>55</i>
<i>Figure 32: Add a consumer in the SLA Dashboard.....</i>	<i>71</i>
<i>Figure 33: Add provider for the SLA Dashboard.....</i>	<i>72</i>
<i>Figure 34: Login in the SLA Dashboard.</i>	<i>72</i>
<i>Figure 35: View of SLA Dashboard for customers.</i>	<i>73</i>

<i>Figure 36: Status of the agreements.</i>	73
<i>Figure 37: Violation details.</i>	73
<i>Figure 38: Info of the agreement.</i>	74
<i>Figure 39: Create an agreement.</i>	74
<i>Figure 40: SLA Dashboard for providers.</i>	75
<i>Figure 41: Status of agreements for providers.</i>	75
<i>Figure 42: Info of agreements for providers.</i>	75
<i>Figure 43: List of templates for providers.</i>	76
<i>Figure 44: Add template for providers.</i>	76
<i>Figure 45: Location of the Interoperability Tool in the XIFI Reference Architecture.</i>	77
<i>Figure 46: Interoperability tool - Architecture diagram.</i>	82
<i>Figure 47: Display available patterns in the Interoperability Tool.</i>	91
<i>Figure 48: Display subscriber interoperability test pattern.</i>	92
<i>Figure 49: Result of interoperability testing and monitoring.</i>	94
<i>Figure 50: Location of the Security Dashboard in the XIFI Reference Architecture.</i>	103
<i>Figure 51: Security Dashboard Architecture diagram.</i>	107
<i>Figure 52: Security Dashboard Unit Test 1.</i>	108
<i>Figure 53: Security Dashboard Unit Test 2.</i>	109
<i>Figure 54: Security Dashboard Unit Test 3.</i>	109
<i>Figure 55: Create Security Dashboard application in the IdM.</i>	112
<i>Figure 56: Create Roles for the Security Dashboard in the IdM.</i>	113
<i>Figure 57: Login in the Security Dashboard.</i>	114
<i>Figure 58: View of Security Dashboard for Infrastructure Owners.</i>	115
<i>Figure 59: Security Dashboard - Visualization of Alarms.</i>	116
<i>Figure 60: Security Dashboard - Visualization of Tickets.</i>	116
<i>Figure 61: Security Dashboard - Visualization of Events.</i>	117
<i>Figure 62: Security Dashboard - Vulnerabilities.</i>	117
<i>Figure 63: Security Dashboard - Save Reports.</i>	118
<i>Figure 64: View of Security Dashboard for Federation Manager.</i>	118
<i>Figure 65: Security Dashboard - configure policies and actions.</i>	119
<i>Figure 66: Security Dashboard - configure network directives.</i>	120
<i>Figure 67: Security Dashboard - configure service directives.</i>	120
<i>Figure 68: Security Dashboard - configure cross correlation.</i>	121
<i>Figure 69: Security Dashboard - configure data sources.</i>	122
<i>Figure 70: Security Dashboard - show servers status.</i>	122
<i>Figure 71: Security Dashboard - show security probes status.</i>	123
<i>Figure 72: Security Dashboard - show hosts.</i>	123

<i>Figure 73: Security Dashboard - show networks</i>	124
<i>Figure 74: Security Dashboard - configure network</i>	124
<i>Figure 75: Security Dashboard - configure access groups</i>	125
<i>Figure 76: Location of the Infographics and Status Pages in the XIFI Reference Architecture</i>	126
<i>Figure 77: Rackspace status dashboard</i>	129
<i>Figure 78: Amazon service health dashboard</i>	130
<i>Figure 79: Architecture of the Infographics and Status Pages</i>	131
<i>Figure 80: Home Page – the Infographics page</i>	132
<i>Figure 81: Information region– Infographics and Status Pages</i>	135
<i>Figure 82: Example cores per node – Infographics and Status Pages</i>	135
<i>Figure 83: Location of the Security Monitoring GE in the XIFI Reference Architecture</i>	136
<i>Figure 84: Service Level SIEM Server Architecture</i>	138
<i>Figure 85: Security Probe Architecture</i>	139
<i>Figure 86: Security Monitoring - Data Source Group</i>	149
<i>Figure 87: Security Monitoring - New policy</i>	150
<i>Figure 88: Security Monitoring - Service Directives</i>	150
<i>Figure 89: Security Monitoring - Network Directives</i>	151
<i>Figure 90: Security Monitoring - Cross Correlation</i>	152
<i>Figure 91: Location of the Cloud Portal in the XIFI Reference Architecture</i>	153
<i>Figure 92: Switch Region in Cloud Portal</i>	160
<i>Figure 93: Switch Region in a Blueprint Tier in Cloud Portal</i>	161
<i>Figure 94: Home Page – FIWARE Lab App Template</i>	162
<i>Figure 95: Location of the Monitoring Dashboard in the XIFI Reference Architecture</i>	163
<i>Figure 96: Monitoring Dashboard Architecture</i>	166
<i>Figure 97: main view of NAM Dashboard</i>	168
<i>Figure 98: Type of test to perform NAM Dashboard</i>	168
<i>Figure 99: Live Request or History NAM Dashboard</i>	169
<i>Figure 100: Test options for NAM Dashboard</i>	169
<i>Figure 101: Location of the Policy Recommendation Service in the XIFI Reference Architecture</i>	170
<i>Figure 102: Policy Recommendation service</i>	177
<i>Figure 103: Policy Recommendation service components</i>	177
<i>Figure 104: Technology used in Policy Recommendation Service components</i>	178

LIST OF TABLES

Table 1: Relationships between components	15
Table 2: Summary Resource Catalogue.....	17
Table 3: List of responsible – Resource Catalogue.....	17
Table 4: Resource Catalogue - User stories backlog.....	20
Table 5: Release plan – Resource catalogue	26
Table 6: Summary SLA Manager.	40
Table 7: List of responsible – SLA Manager.	40
Table 8: Interoperability tool - User stories backlog.....	42
Table 9: Release plan – SLA Manager.....	47
Table 10: Test cases – SLA Manager	48
Table 11: Summary - Interoperability tool.....	78
Table 12: List of responsible - Interoperability tool.....	78
Table 13: Interoperability tool - User stories backlog	79
Table 14: Interoperability tool – Architecture elements	83
Table 15: Interoperability tool – Release plan	86
Table 16: Interoperability tool - User stories backlog.....	86
Table 17: Summary Security Dashboard.	103
Table 18: Component responsible - Security Dashboard.....	104
Table 19: User Stories backlog - Security Dashboard.	106
Table 20: Release Plan - Security Dashboard.....	108
Table 21: Security Dashboard – Test Cases.....	108
Table 22: Summary Infographics and Status Pages.	127
Table 23: List of responsible – Infographics and Status Pages.....	127
Table 24: Infographics and Status Pages - User stories backlog.....	129
Table 25: Release plan – Infographics and Status Pages	131
Table 26: Test cases – Infographics and Status Pages	131
Table 27: List of responsible Security Monitoring GE.....	137
Table 28: User stories backlog - Security Monitoring GE.....	138
Table 29: Release plan - Security Monitoring GE.....	140
Table 30: Test cases - Security Monitoring GE.....	140
Table 31: List of responsible Cloud Portal.	154
Table 32: User Stories backlog - Cloud Portal.....	155
Table 33: Release plan - Cloud Portal.	156
Table 34: Test cases – Cloud Portal.....	156
Table 35: Summary Monitoring Dashboard.....	163

Table 36: List of responsible – Monitoring Dashboard.164

Table 37: Monitoring Dashboard - User stories backlog165

Table 38: Monitoring Dashboard166

Table 39: Summary Policy Recommendation service.....170

Table 40: List of responsible – Policy Recommendation Service.....171

Table 41: Policy recommendation service - User stories backlog172

Table 42: Policy Recommendation Service Release plan178

ABBREVIATIONS

ALOIS	Advanced Log Data Insight System.
API	Applications Programming Interface
AMQP	Advanced Message Queue Protocol
DDS	Data Distribution Service
EPR	Endpoint reference
FI	Future Internet
FI-PPP	Future Internet Public-Private Partnership Programme
FIWARE	Future Internet Core Platform
GE	Generic Enabler
GUI	Graphical User Interface
IdM	Identity Management
KPI	Key Performance Indicators
OGF	Open Grid Forum
PaaS	Platform as a Service
P3P	Platform for Privacy Preferences
QoS	Quality of service
SaaS	Software as a Service
SBF	Service Business Framework
SLA	Service Level Agreement
SLO	Service Level Objective
SOA	Service Oriented Architecture
SSO	Single sign-on
SIEM	Security Information Event Management
UDDI	Universal Description, Discovery and Integration
USDL	Unified Service Description Language
WADL	Web Application Description Language
WSDL	Web Services Description Language
WSLA	Web Service Level Agreements
WSOL	Web Service Offerings Language

1 INTRODUCTION

This deliverable contains the second software version of the services and tools described in the framework of WP4. This document is an upgraded version of the D4.2 Baseline Tools v1 [95], and it should be self-contained. Due to this, the new version contains all the sections to avoid that the reader should change between different documents and versions in order to manage the different components. Hence, all the components have to include the complete sections and only the components which have not been changed have been referenced to the previous version and the wiki documentation. The structure of the components is composed mainly by: the general XIFI overview, the technical analysis, links to the source code, the installation guide, the test case and the user Manual.

This document is a compilation of the different components identified in WP4 and is based on the specifications defined in the “D4.1b- Services and tools specification” [2] and it is aligned with the rest of the components from WP2 and WP3. They are the basis to elaborate the showcase of WP6 and provide content for the learning of WP7.

The components of the previous version, which have been upgraded are: the Resource Catalogue and Recommender, the SLA Manager, the Interoperability tool, the Security Dashboard, the Infographics and Status page, the Security monitoring GE and the Cloud Portal. The FIWARE Lab App Template, which was also included in the previous version, has not been included the entire content again, since there are not changes from the previous version.

The upgraded version has the following modifications in the different sections: i) the section state of the art and the motivation do not include changes; ii) the sections summary, Release plan and user stories have been adjusted to reflect the new version iii) the Architecture design, Test case, Installation Manual and User Manual include the appropriate modifications to cover the second version of the components.

Besides these components, there are two new components introduced in this version: the Monitoring Dashboard and the Privacy Recommendation Service. They have followed the same structure, which has been defined, to describe the component.

1.1 Purpose

This Baseline Tools document provides a description of the software design and installation manuals of the components developed in WP4 for XIFI. The document describes the current development status (version two) and there are ten components described. The document presents instructions on using and installing these software components, based on the corresponding installation, test and user manuals.

1.2 Overview

This is the new version of “D4.2 Baseline tool v1” [95] and the document has to reflect the changes from the previous version. There were two options to upgrade the content in this version: i) to include only the changes of these sections, which avoid duplicating the content from the previous version, ii) to include all of the content for every component, duplicating part of the content. After analyzing the two options, we have decided to include all of the content again for every component, since it is better for the reader. Thus, the document is self-contained and readers can use it as a reference manual and understand without the need to change between different documents and references.

The components, which do not have changes from the previous version, are only referenced, since the reader can look up their content in the previous version or in the wiki documentation. For this version, only the FiLab App Template component is in this situation.

This document is a presentation of the different components and their reference to the source for the second version. Each section has followed the same structure to facilitate its reading, regardless of the

component. The details of the changes from the second version are included, as well:

- *summary* provides a brief description of the component and it situates it within the full architecture diagram, in order to understand better the dependencies and the implications. The new version includes the alignment with current architecture diagram and reviews the dependencies with other components;
- *component leaders* indicates the main people who are responsible for the component. The new version has updated the list of involved people;
- *motivation* section deals with the reasoning for the design and development of these components. This section has not been changed, since the motives have not changed in this short period between versions;
- *user stories* backlog identifies the list of user stories that will be implemented, in order to follow up. This new version contains the new user stories, which have been identified. This section also includes the links of the Redmine [121], which allows to map the user stories with the requirements, as described in D1.3 [5];
- *state of the art* provides the analysis of the existing solutions in order to select the components to be used, extended and adapted. The situation of the state of the art is (in the majority of the components) similar to the previous version and the section has the same content and references;
- *architecture design* presents the relations between the components and the technical detail of the different parts of the component. This section has been updated to be aligned mainly with the federation components and their relationships;
- *release plan* identifies the different versions and what are the user stories that will be released. The new version includes the new releases for the second version;
- *test case* provides in detail the tests that the developer has to execute in order to be sure that the component works properly. This section has changed mainly to be aligned with the new version of the components;
- *installation manual* explains all the steps in order to install the component. The new version includes the current status in order to install the component;
- *user manual* is the reference manual for the component's use. In this case, it can be a description of the API (focus on the developers) or the description of the GUI (more oriented to the End User). There are significant changes in this section in order to be included in the new version of the APIs and the released graphical user interface. As well, a description of the error for the defined APIs has been included, to better detail the error management.

There are eight components, which have been included in the previous version, the Resource Catalogue and Recommender, the SLA Manager, the Interoperability tool, the Security Dashboard, the Infographics and Status page, the Security monitoring GE, the Cloud Portal and the FIWARE Lab App Template. All the above described sections have been included for them, less so for the FIWARE Lab App Template, since it has been not changed from the previous version and it is only included as a reference.

In addition, there are two new components that have not been included in the previous version: the Monitoring Dashboard and the Privacy Recommendation Service. Both follow the same structure as the rest of components and they have been included to visualize the monitoring data and to complement the recommendation for the final users.

In this new version, the components have been evolved to finalize their new versions and they have also focused in the integration with the rest of the federated components. For the WP4 components, it has been a challenge to integrate at the same time as the federated components' release. This integration is reflected in the following table:

Components/Relationships	Components									Portals	
	Federation IdM	Federation Manager	Federation monitoring	DCA	Cloud Portal	Resource Catalogue	Security Monitoring GE	Access Control GE	Policy Recommendation service	FIWARE Lab	FIWARE Ops
Infographics and Status Pages	√	√	√	-	-	-	-	-	-	√	-
Marketplace/Resource Catalogue & Recommender	√	√	√	√	√	√	-	-	√	√	√
SLA Manager	√	√	√	√	-	-	-	-	-	√	√
Monitoring Dashboard (Nam)	√	-	√	-	-	-	-	-	-	-	√
Monitoring Dashboard (Vm)	√	-	√	-	√	-	-	-	-	√	-
Interoperability Tool	√	-	-	-	-	-	-	-	-	√	√
Security Dashboard	√	√	-	-	-	-	√	√	-	-	√
Cloud Portal	√	-	√	-	-	-	-	√	-	√	-
Policy Recommendation service	√	-	-	-	-	√	-	-	-	√	-
FIWARE Lab app Template	√	-	-	-	-	-	-	-	-	-	-

Table 1: Relationships between components

It is also indicated, in the table, the relation between the GUI components and the integration of the Portals. There are two possible orientations: i) FIWARE Lab [120] is a user oriented portal; ii) FIWARE Ops [119] is infrastructure Owner oriented.

The reader can observe that there are components that have both orientations; due to they have functionalities and roles for both users and infrastructure owners, as it is described following.

Type of ownership	Extension & Adaptation
Original tool	Store – Wstore [9] Repository GE [10]
Planned OS license	As the original tool.
Reference OS community	FIWARE

Table 2: Summary Resource Catalogue.

Consist of

- Resource Catalogue services interface (API specs)
- Search, Offerings, Recommender engines

Depends on

- FIWARE WStore & Store GE [9]
- FIWARE Repository GE [10]
- Federation Monitoring [91]
- FIWARE MarketPlace GE [11] (Optional)
- Deployment and Configuration Adapter – DCA [90]
- Federation Manager [94] (next releases)
- IdM GE [82]
- Policy Recommendation service [section 11]

2.2 Component leaders

Developer	Email	Company
Joaquin Iranzo	joaquin.iranzo@atos.net	ATOS
jorge Valhondo	jvr@gatv.ssr.upm.es	UPM-SSR
Ömer Özdemir	omer.ozdemir@atos.net	ATOS

Table 3: List of responsible – Resource Catalogue.

2.3 Motivation

One of the XIFI aims is to offer services and resources in a federated environment, for which it is necessary that the services provider can publish the detail of them, and on the other hand, the users can discovery the services easily and efficiently. The responsible to provide these functionalities through a unified platform is the Resource Catalogue and Recommendation tool.

It represents the catalogue where all the available information on infrastructures can be found:

- The endpoints of the GEs instances deployed (SaaS model)
- The data center capabilities for hosting a software platform
- Other capabilities, such as the presence of Sensor Network, Mobile Network and, in general, other features different from the bare data center.

For each exposed service/resource, the user can also check the monitoring and performance data when

it is available.

The component is mainly used by a developer that wants to find and access the XIFI federated resources/services and the infrastructure owner that wants to advertise the services provided by his infrastructure through this component. The user can browse/search the catalogue of the services/resources offered and can find what he needs. Beside this, he can rate and introduce comments about the services.

The section State of the art provides an overview of available approaches, which we analysed prior to selecting FIWARE GE as the starting point.

This component has to cover the repository where the services are described, as well as the management of these services through a marketplace/store. Hence, the decision has been taken as a whole. The GEs included in the Application/Services Ecosystems and Delivery Framework [12] are integrated and they cover all the current necessities and the future business requirements.

The Repository GE and WStore GE were selected mainly for the following reasons:

Use of the USDL standard that allows defining technical specification as well as business specification. It is oriented to describe all the business lifecycle of the services. Furthermore, it allows defining resources and introducing ontologies of different domains.

Linked USDL [21] provides a user interface to define the service.

- The Repository GE uses the USDL standard and it provides a REST API in order to manage the repository.
- WStore manages the resources/services lifecycle and it is integrated with the Repository GE which stores the services descriptions.
- WStore is flexible and modular and it can manage the complete business lifecycle (future business requirements about the billing, payment...)
- WStore also covers the rating and recommendation of the services.
- WStore is also integrated with other useful GE (on the Application/Services Ecosystems and Delivery Framework) as “Revenue Settlement and Sharing System” [13], “Application Mashup – Wirecloud” [14], “Marketplace –SAP RI” [11], “Business Calculator GE” [15], “Business Modeler” [16] which will allow to increase the future functionalities.
- The Repository GE and the WStore are integrated with the IdM GE [18].

This component has different competitive advantages (apart from the previous ones):

- It can operate in a federated environment, including functionalities such as the visualization of the different nodes where the same service has been deployed.
- It can be integrated with the FIWARE Catalogue.
- It can define and offer non-conventional services.
- It can provide details of the status of the services and their evolution.

2.4 User stories backlog

id	User story name	Actors	Description	Task id
1	The user is identified in the	End User	The user enters in the Marketplace, he is identified by the XIFI idM and he can enter in the Marketplace (Resource	745

	federation.		Catalogue).	
2	Navigate through the GE type.	End User	The user can navigate through the catalogue of GE type (the information is gathered from the FIWARE Catalogue) and he can also see which nodes contain the instance of the GE type.	717
3	Search GE type.	End User	The user can search a text through the catalogue of GE type (the information is gathered from the FIWARE Catalogue) and he can also see which nodes contain the instance of the GE type.	746
4	Recover the GE overview	End User	The user looks at the main description of Generic Enabler type (the information is gathered from the FIWARE Catalogue) and the user can also see more detailed information about the instances of the GE type and the status of the nodes.	747
5	Browse through the Non-conventional services.	End User	The user can navigate through the catalogue of Non-conventional services and he can see the main information and it is complemented by a map with the node which contains it.	748
6	Search Non-Conventional services.	End User	The user can search the text through the catalogue of Non-conventional services (the information is gather from the internal catalogue) and he can see the main information, complemented by a map with the nodes which contains it.	749
7	Recover the Non-Conventional service description	End User	The user looks at the description of Non-conventional services (the information is gathered from the internal catalogue and it is described by USDL) and the user can also see a map with the nodes which contains it.	750
8	Search the resources through the nodes.	End User	The user can select the different nodes in a map and see the details of the resources which are published on.	751
9	Search through advance criteria.	End Users	The user can search through the advanced criteria the XIFI resources. So, he refines the search introducing filters composed by key-words which define the description of the resources.	752
10	Recover the GE instance description	End User	After selecting the GE type, the user looks at the details of this GE instance (EndPoint, Status, monitoring data, etc)	753
11	Advertise a new resource	Infrastructure Owner	The infrastructure owner advertises the non-conventional services that he offers. He will describe the non-conventional service through a form that will be mapped to the USDL description.	754
12	Modify/delete his published resources	Infrastructure Owner	The infrastructure owner can select his resources (Non-conventional services) that have been advertised previously and he can modify and delete them.	755
13	Validate the publication of SE and Non-conventional Services	Federator	The Federator reviews and validates that the SE and Non-conventional services has been correctly defined and the Federator confirms them in order to be published in the catalogue.	1502
14	Vote and	End User	The user selects the XIFI resource and he can vote it or add	756

	comment the resources		comments about it.	
15	Recover the recommendation of resources	End User	The user selects a resource and he sees the ranking and the associated comments.	757
16	Recommend federated facilities.	End User	The user is supported by a recommendation tool which informs him about the XIFI federated facility that fits best according to application requirements. The recommendation is mainly based on Federated Monitoring data (Physical Hosts, Virtual Machines and Networking data).	758
17	Search federated facilities	End User & Infrastructure Owner	The user is able to view all federated facilities through a map where he is able to interact and discover facility details.	1044
18	Search user registered and/or deployed services	End User	The user searches through a map identifying available federated facilities and federated facilities where the user has some services. The users also identifies own services on each facility and is able to distinguishes between registered (SaaS) and deployed (PaaS) services.	1045
19	Search all deployed services on a facility.	Infrastructure Owner	The user searches through a map identifying own federated facilities and all other federated. The users also identify all available services on his/her own facility and its satatus.	1046
20	Deploy new service on a facility	Infrastructure Owner	The user is able to provide new services on his/her own facility to the Cloud Community, as well as manage them.	1047
21	Integration third party catalogues	End user & Infrastructure Owners	Allow to third party catalogues to be integrated with the XIFI environment. The users can discover them in the same portal without navigate between different environments.	1505
22	Notification of new entries in the Marketplace.	End user	The component notifies automatically the register user when the new resources, services or GEs are available in the Marketplace/Catalogue.	1506

Table 4: Resource Catalogue - User stories backlog

2.5 State of the art

Two approaches have been considered to carry out this overview. On one hand, the different options to manage a common repository which contains the resource description and its storage have been analyzed, while on the other hand we considered a way to manage them into a store/marketplace.

Service description and repository.

- There are several approaches, like the UDDI, WSDL and WSAL, which are only focusing on describing technical interfaces, omitting the description of the business and operational aspects concepts. They had been created to cover services exposed through concrete technologies as the WS or REST, and there are not general description to cover others approaches. On the other hand, there are semantic approaches like OWL-S, WSMO and SA-WSDL that allow describing service through semantic information. However, there aren't common vocabulary and taxonomies for defining business aspect like the license model.
- Dublin Core [19] defines the “core metadata” for simple and generic resource descriptions and its discovery. So, it can describe web and physical resources through a common definition to

combining metadata vocabularies, to providing interoperability for metadata vocabularies in the linked data cloud and semantic web implementations.

- Unified Service Description Language (USDL) [20] provides a unified description of business, operational and technical aspects of services. Further, it is a platform-neutral language for describing services. The kinds of services targeted for coverage by USDL include human services (e.g., consultancy), business services (e.g. purchase order requisition), software services (e.g., WSDL and RESTful services), infrastructure services (e.g., CPU and storage services), etc. As many services have a hybrid character with both, a digital and physical or manual footprint, USDL can facilitate the combination and aggregation of such services. Therefore, USDL can be considered one of the foundational technologies to set up an Internet of Services around today's core enterprise systems.
- Linked USDL [21] - In addition, Linked USDL aims to better promote and support the use of the USDL on the Web. It is a remodelled version of USDL that builds upon the Linked Data principles and the Web of Data. This effort is therefore most concerned with remodelling the existing USDL specification as an RDF(S) vocabulary that could better support machines in trading services on the Web. To maximize the potential interoperability Linked USDL adopts, where possible, existing RDF(S) vocabularies.

This description can be extended and complemented by existing domains specific vocabularies, in order to cover other scopes like Internet of Things, Security, etc. Besides that, a simple web application is available for creating Linked USDL service descriptions. Using a form based slick UI, it is easy to conduct a service description in a matter of minutes. Import and export for various formats is possible. The editor can connect directly to a service description repository.

The above list summarizes the different options to describe resources and services in a common way. However, we also need to analyze the different approaches of the repositories which manage the different actions including in the resource lifecycle (create, update, modify, delete and search) through a common API.

- Repository GE [10]: The repository provides a consistent and uniform API to USDL service descriptions and associated media files for applications targeting the FIWARE business framework. A service provider can use the Repository to publish the description of various aspects of the service according to a uniform description language. It is based on USDL which describes services on a metadata level and can refer to supplemental resources of any media type. Therefore the repository must be able to store resources in arbitrary formats. The RDF data model of USDL provides references to entities of the service description via the resource URL. USDL is used in its Linked Data version "Linked USDL" [21] which is already well prepared to allow the distribution of service descriptions across the Internet.

The Repository is a place to store service models, especially USDL descriptions but also other models required by components of the overall delivery framework (e.g. technical models for service composition and mashup). The repository provides a common location for storage (centrally or distributed and replicated), reference and/or safety. The service descriptions in a repository are typically used by several components of the platform, such as service stores or marketplaces, by extracting information needed for the specific functionality.

- Z39.50 [22] defines the protocol for information retrieval, it specifies procedures and formats for a client to search a database provided by a server, retrieve database records and perform related information retrieval functions. It facilitates interconnections of client-server for applications where clients search and retrieve information from server databases. An abstract model describes the databases which allow the systems to communicate in standard, for the purpose of searching and retrieving information from a database. It is one of the oldest and the most widely used method of searching remote library catalogues. It is currently supported by public libraries, most academic libraries and many public/private collections.

- The Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [23] is a protocol developed by the Open Archives Initiative [24]. It collects the metadata description of the records in an archive so that services can aggregate metadata from many archives. It can support representing metadata in Dublin Core, but may also support additional representations. It fosters promoting interoperability standards to facilitate the efficient dissemination of content on the Internet. It emerged as an effort to improve access to electronic publications files (eprints), ultimately, to increase the availability of scientific publications and is widely used by institutional repositories, digital libraries and digital media archives.
- Content Management Interoperability Services (CMIS) [25] defines a domain model and set of bindings that include Web Services and ReSTful AtomPub [26] that can be used by applications to work with one or more Content Management repositories/systems. The CMIS interface is designed to be layered on top of existing Content Management systems and their existing programmatic interfaces. It is intended to define a generic/universal set of capabilities provided by a CM system and a set of services for working with those capabilities.

Marketplace/Store.

The repository only manages the description of the resources; however it should be integrated in a marketplace or store in order to manage the lifecycle of the business. So, this is not directly related with the Resource Catalogue definition and it is more related to the marketplace; nevertheless it is important to analyze all the solutions as a whole.

There is a wide number of proprietary stores that have control over their portfolio, define and manage the cycle of their services like Apple, Google, Amazon, Windows, etc. Nevertheless, a marketplace is a platform where different stores can publish their offerings in order to be discovered and compare by the consumers. Hence, the store is the place where a provider can manage different offers and it can publish them in a common marketplace in order to increase their visibility. The Application/Services Ecosystems and Delivery Framework cover these functionalities through different GEs providing a fully-fledged business framework:

- WStore [9] is a Service Business Framework (SBF) and it supports managing services in the business framework across the whole service lifecycle: from creation and composition of services to monetization and revenue sharing. This framework is built for selling services to both consumers and developers of Future Internet applications and services and for end-to-end managing of offerings and sales. The WStore has full control over a specific service/app portfolio and offerings. This enable covers all the functionalities for the discovery through its own portfolio that allows the user discovering and comparing the resources which has been described by the USDL and storage in the Repository GE. On the other hand, this enabler provides more than the Resource Discovery needs, because it can manage the complete business lifecycle.
- Marketplace GE [11]: In general a marketplace is an instrument to facilitate commerce by bringing together vendors and buyers, or offers and demand, or producers and consumers. A marketplace can support a variety of mechanisms to achieve this. Any offering in context of the application and service business can be supported by marketplace functionality. A marketplace implementation can offer many different kinds of services to the participants. Currently, the marketplace provides three main components: Registry & Directory, Offering & Demand and Discovery & Matching. And it is planned to deliver two more: Recommendation and Review & Rating, in the 3 Major Release of FIWARE [26]. Therefore, the Marketplace provides functionality necessary for bringing together offering and demand for making business. These functions include basic services for registering business entities, publishing and retrieving offerings and demands, search and discover offerings according to specific consumer requirements as well as lateral functions like review, rating and recommendation. This core functionality provides a basis for extended services depending on the domain and nature of the target markets. Besides the core functions, the marketplace may offer value because of its "knowledge" about the market in terms of market intelligence services, pricing

support, advertising, information subscription and more.

2.6 Architecture design

The figure below shows the FMC compositional structure diagram highlighting the relations between the components:

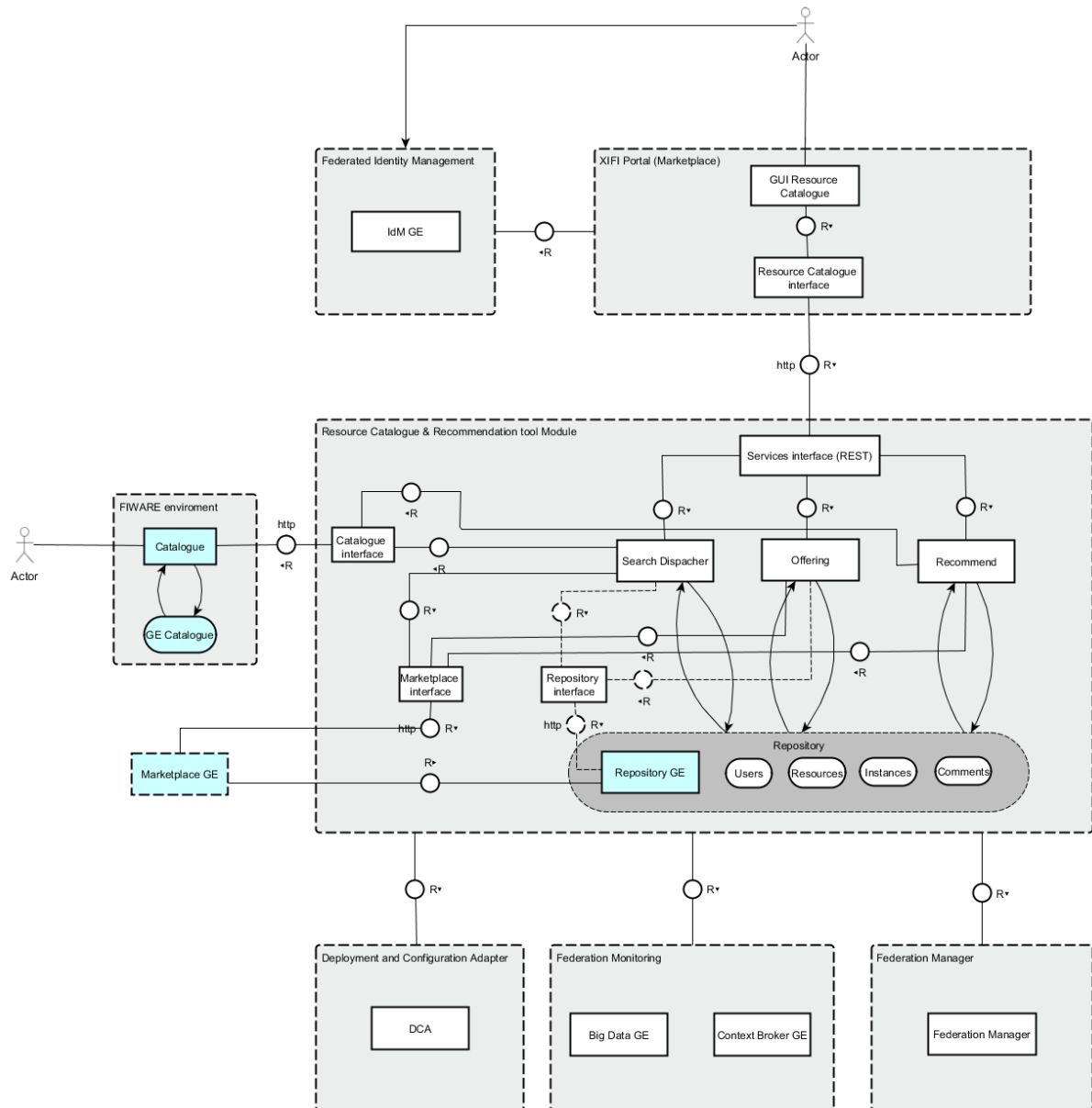


Figure 2: Resource Catalogue FMC compositional structure diagram

The tool is split into two main modules: the first one will be responsible for the visual part and the other one will manage all the business logic and the data layer.

The decision to separate it into two modules allow us to decouple the presentation layer, which is directly related to the GUI (look and feel, usability, etc) and the backend, which is responsible to provide all the functionalities through an exposed interface (in our case the RESTful API), thus allowing any module to access and interact with them. Thus, the graphical module will focus on the

“look and feel” and it will manage the necessary calls to the central module which will execute the user actions.

In this way, other systems or other visual components may interact with the central module without any structural modifications. For example, other components (existing or new) only need to invoke the exposed interface in order to include into their GUI, the list of instances of one node and their description.

Below we describe the components of each module:

Presentation Layer:

It will be integrated in the XIFI portal. The visualization will be via web (HTML). The technology is based on jQuery [88] and it is deployed over Django framework (Python) [83].

To access the business logic will use a module responsible for making calls to the remote module via REST, the Resource Catalogue Interface.

Business Layer and Data Layer (Main Module):

It is composed of different components that will allow management of the functionalities described in the D4.1b [2]. Basically, it is split in three sections: Search, Offering and Recommend. This classification will allow us to prioritize the development and to work in parallel over the different components.

- *SearchDispatcher*: it is responsible to dispatch the search depending on the type of resource that we want to manage. In order to recover the information about:
 - *The Generic Enablers types*, it will call the remote FIWARE Catalogue;
 - *The Specific Enablers types and the non-conventional services*, it will interact with the Repository GE module;
 - *The instances (of the GE and SE)*, it will access to the XIFI Repository (data model) to recover the associated data.
- *Offering*: It is responsible to expose the resources of the infrastructures and it interacts with both the Marketplace GE (to expose the Advance Capabilities and the SE) and the XIFI repository (data model) to publish the instances.
- *Recommendation Tool*: It is responsible to support users informing the XIFI federated facility that fits the best according to application requirements. The recommendation tool is mainly based on Federated Monitoring data (Physical Hosts & Network data).

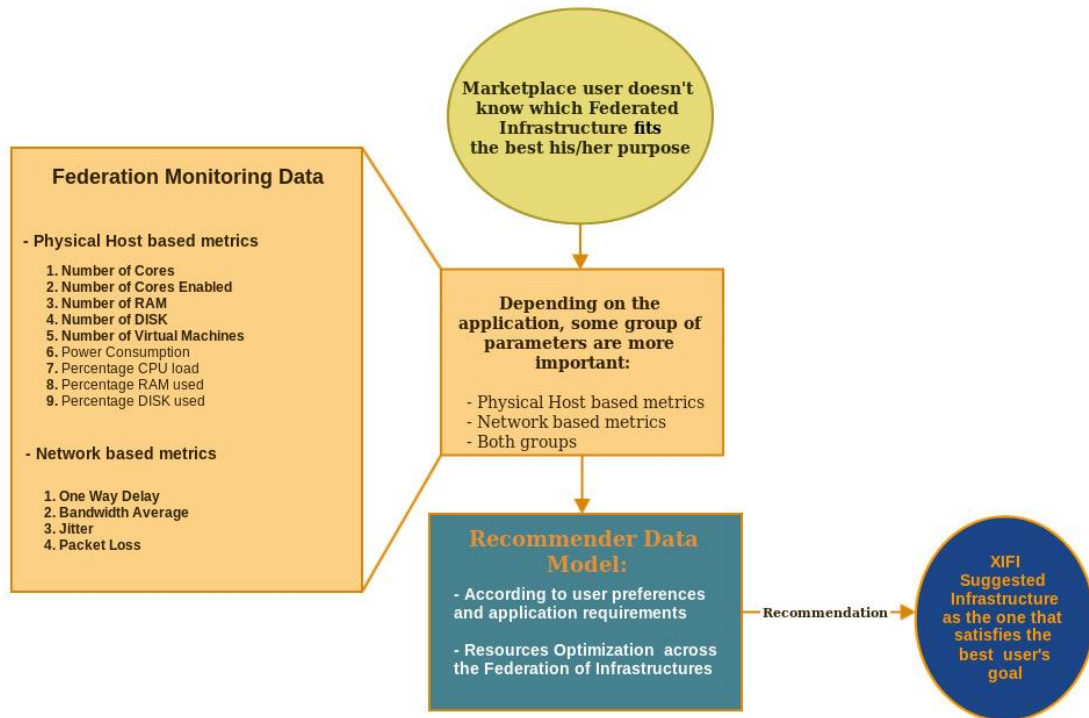
MARKETPLACE: Recommendation Tool (WP4 - T4.2)


Figure 3: Recommendation Tool model

As it can be extracted from the above image, there are two main groups of metrics that can be gathered from the Federation Monitoring API [91]: i) Physical Host based (related to physical hosts by region) and ii) Network based (related to network capabilities for transmitting data).

The Recommendation Tool supports XIFI users when a new service is going to be registered or deployed and the user does not know about the facility that fits the best for the application requirements. In this case, the user weights relevance of main parameters according to the application needs. Furthermore recommending the infrastructure that fits the best, the Recommendation Tool optimizes resources across the XIFI Federation. To sum up, the Recommendation Tool suggests the XIFI facility that satisfies the best user's goal, and also avoids wasting federation resources.

- *The repository*, it is the responsible to store and to persist all the XIFI data. We can consider two parts:
 - *Repository GE*: it is responsible to provide the metadata to describe the XIFI resources through the USDL specification; hence it is the Catalogue Metamodel. It is part of the repository, but it is an independent module, so we cannot access directly to these entities, we have to call this module through the exposed API.
 - *The XIFI Repository*, it is responsible to store all the data that are not managed by the Resource GE, which is responsible to describe the resources through the USDL description. Hence, entities like the users, instances, comments and their relationships should be managed by this XIFI Repository.

- *Marketplace GE*, It is an external module where different stores can publish their offerings in order to be discovered and compared by the end user, so it is responsible for searching, offering and rating the resources in the market. It exposes an API for access to its functionalities. The integration of this component is optional, since this module can operate without it.
- *FIWARE Catalogue*. It is the remote and centralized catalogue of FIWARE where all the Generic Enablers are described. So, the aim is to maintain only one GE catalogue. In order to access it, the system will use the Catalogue Interface to call it.

2.7 Release plan

Version Id	Milestone	User Stories
1.0	31/12/2013	2,4,6,7
1.1	31/01/2014	3,5,11
1.2	28/02/2014	8,12
1.3	31/04/2014	10,9,14,1
2.0	30/09/2014	13, 15, 16
2.1	30/11/2014	17,18,19,20

Table 5: Release plan – Resource catalogue

2.8 Test case

Please refer to WStore test cases [27].

2.9 Installation Manual

Resource Catalogue is a component built on top of WStore GE in XIFI architecture. Here are the steps for the installation and configuration of the Resource Catalogue (RC):

- Installation and configuration of WStore GE [27]
- Installation and configuration of Repository GE [10] (If you have installed a Repository GE, you don't need to install it again, just the remote repository URL is enough)
- Configuration of Resource Catalogue

Resource Catalogue is implemented as a part of the Wstore GE, both Wstore and RC source codes can be downloaded from URL [28].

Since there aren't any specific installation steps for RC, it is only needed to follow WStore Installation and Administration guide [27]

Resource Catalogue Configuration

Here are the three configuration values that must be configured for the RC module:

- CHAPTER_PAGING_NUMBER = Number of items to be displayed in the Resources page.
- REMOTE_CATALOGUE_SEARCH_URL = Fiware API for catalogue searching based on keyword

Example [29] The XX word must exist in the URL in order to parse and replace the search keyword within the Resource Catalogue module.

- `REMOTE_CATALOGUE_URL_BASE` = Fiware API for catalogue retrieval with chapter and page number

Example: [30]

- `PAGE_CACHING_TIME` = Configure the caching time for the resource catalogue pages.

(example value = 150, this value must be numeric value in terms of minutes)

2.10 User Manual

Resource Catalogue component is built on WStore GE, so please see the official WStore documentation and demo's on how to configure and use the overall system. Please refer to WStore programming guide [31] for further info.

There are 3 roles in the component:

- *FI Developers*. This role has assigned to users that want to interact with the XIFI Portal to find and use the services. They have to be registered in the IdM.
- *Infrastructure Owners*: This role identifies the nodes which want to be federated. They can publish both the specific enablers and the non-conventional services. They have to be identified by the IdM as organization and validated by the federation.
- *Software providers*: This role identifies organizations that want to expose the own specific enablers, for example SME, Technological Providers, Use Case Projects.... They cannot publish non-conventional service since only the Infrastructure Owner can provide this kind of resources. They have to be identified by the IdM as organization.
- *Federator*. This role indicates the XIFI federation that is composed by all the responsible to maintain the federated platform like administrative tasks and resource publishing

There are 3 types of resources in the Resource Catalogue:

- *Generic Enablers*:
 - The web-based services offered by FIWARE [84]
 - The Fiware Catalogue is the central repository for implementations of Generic Enablers (GE) that are part of the FIWARE platform. You will find all the information, documentation and tools you need as a developer to start using a Generic Enabler Implementation. All the information of the GE is centralized in this catalogue and it is responsible to collect, validate approval and publish all the new ones [32]. The Catalogue provides an API REST interface in order to read the public information. RCM exposes information about the instances (for each generic enabler) of each node on each region.
 - Currently, it offers a complete dump of the information of each node (piece of content) as well as a list of all the nodes that can be filtered.
- *Non-Conventional Services and Specific Enablers*
 - XIFI is a federation of resources offered to the FI-PPP developer community by FI infrastructures. FI-PPP developers themselves may contribute to the community cloud by offering additional hardware capacity (under their own control or open to other PPP participants) and their own services (the so called Specific Enablers).

These services are created and advertised through WStore GE by Service Providers and Infrastructure Owners.

- User has to be authenticated via IDM to enter the Resource Catalogue.

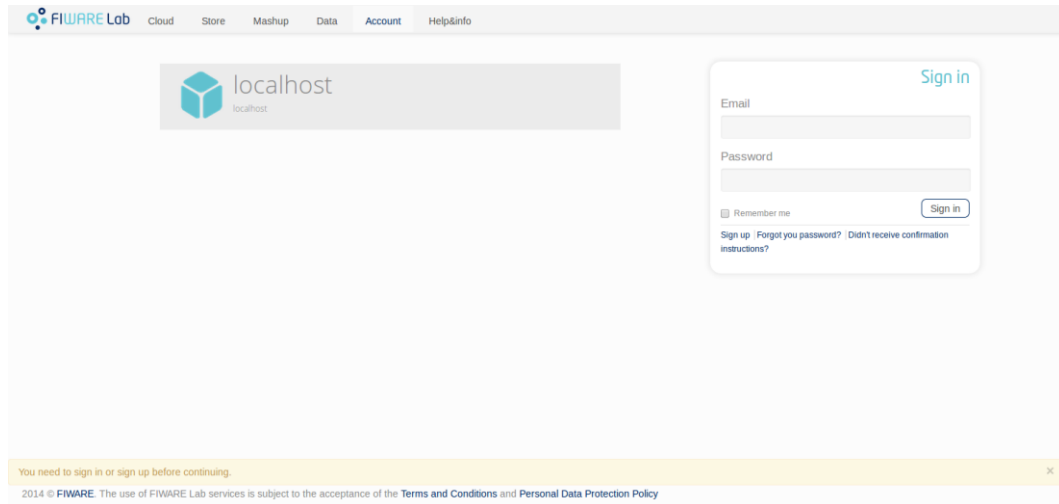


Figure 4: Resource Catalogue IdM Authentication.

- The user has been identified and he can start to use the component.

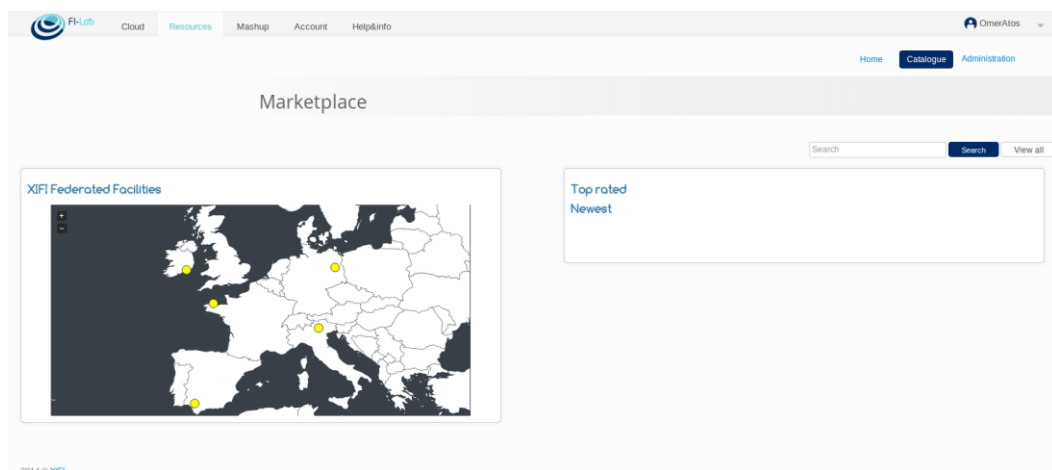


Figure 5: Initial page after the user has been identified.

- There are 5 main sections in the Resource Catalogue.

These sections are:

- Home: Landing page for
- Manage: User can advertise both SE and NCS via manage section depending on his/her role type
- Catalogue: User can browse the SE, NCS and GE in this section
- Publish: User who has Federator/admin role can only see this section and publish newly created resources
- Administration: Section for administrative tasks

Catalogue Section

This section contains 3 resources that are retrieved via Remote Catalogue API or defined/created in the Resource Catalogue.

- Generic Enablers
- Other Services (Non-Conventional Services)
- Specific Enablers

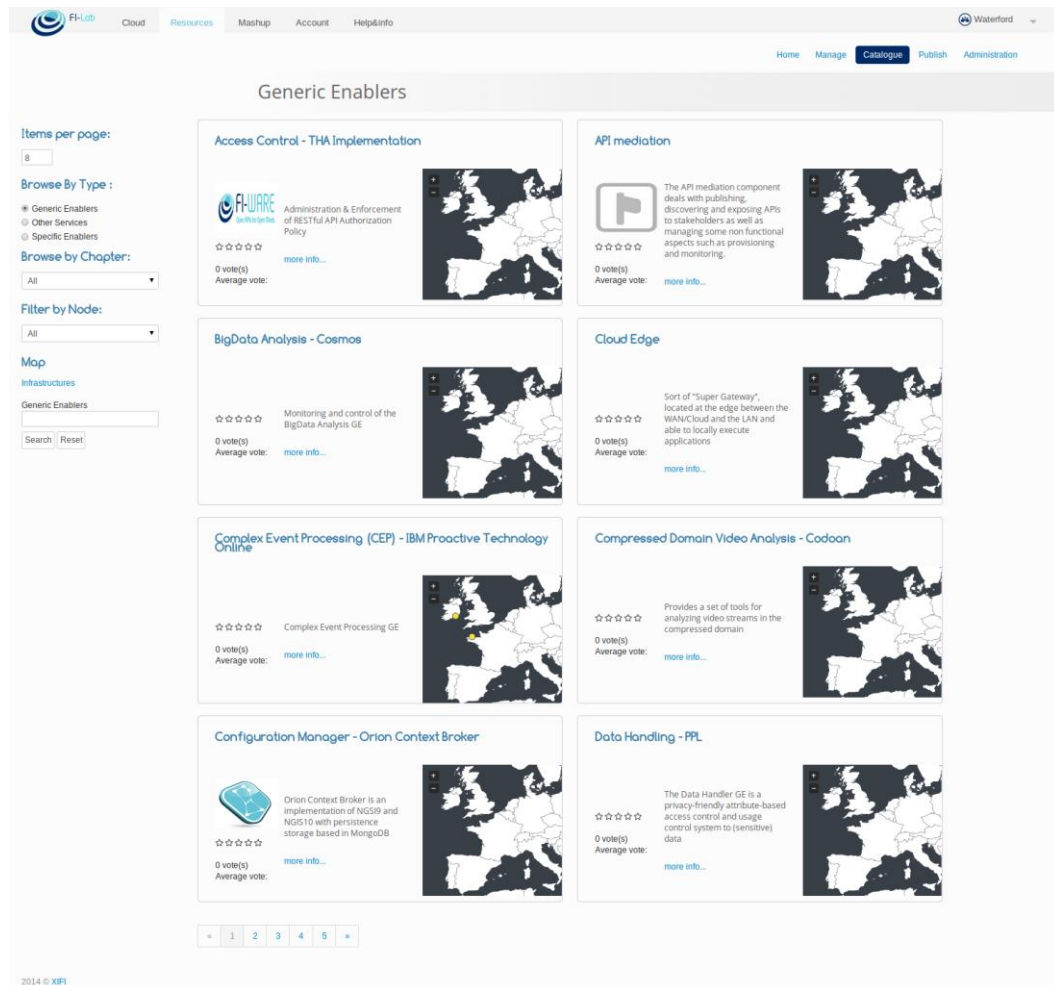


Figure 6: Resource Catalogue – Catalogue section

Generic Enablers

Generic Enablers are divided into 6 sub-chapters and listed in the combo-box menu:

- All
- Services Ecosystems and Delivery Framework
- Cloud-hosting
- Context Management
- Interface to Networks and Devices
- Internet of Things Services Enablement
- Security

- Each Generic Enabler is deployed as a different instance on the nodes (Berlin, Trento, Waterford etc.). Each instance of GE is shown on the node that is deployed and running

This map has information about where the GE has been deployed (nodes) and how many instances do exist in every node.

User can apply filters in order to view the resources; moreover can set other parameters such as:

- configuring the number of items per page;
- viewing sub-chapters (we can see that subchapter: Services Ecosystems and Delivery Framework is selected).

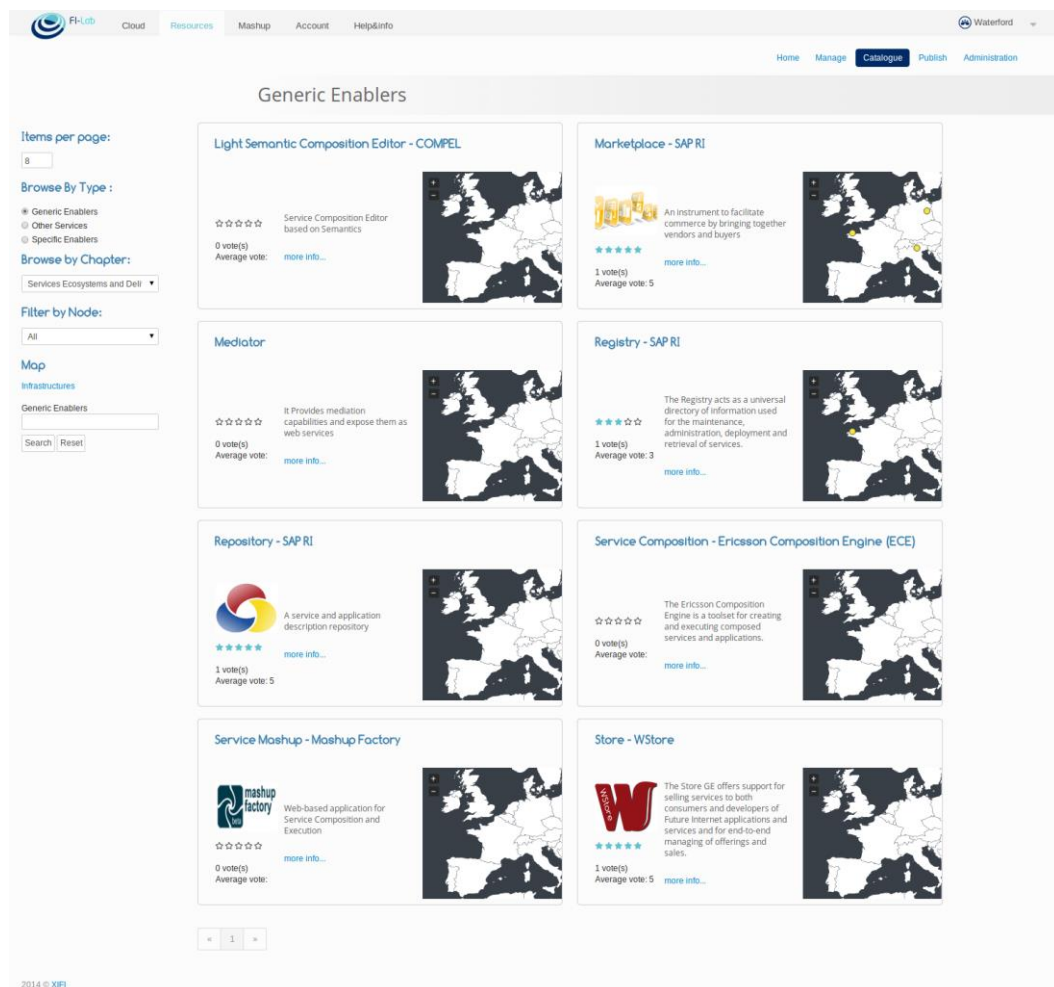


Figure 7: Resource Catalogue – List of generic Enablers.

- searching with the specified keywords. (Below an example of search with the keyword "Event", all the GE's contain that word listed after the search)

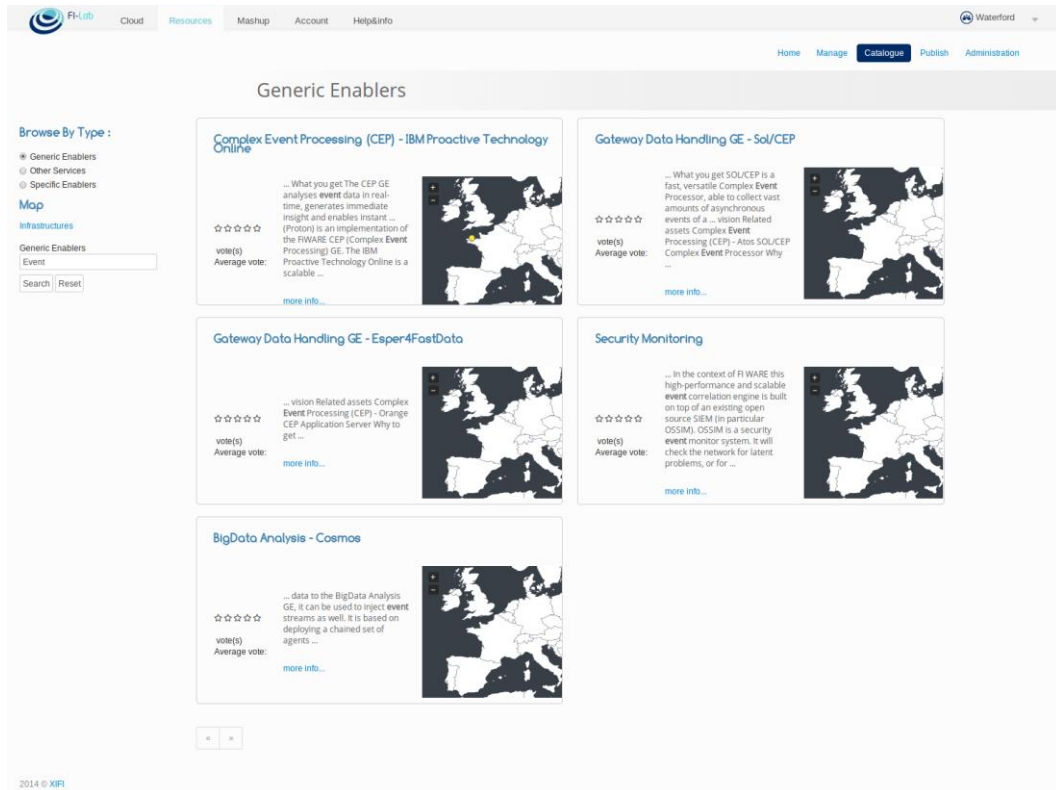


Figure 8: Resource Catalogue – Browse by keyword.

- Selecting node name in order to view the GE's that are deployed in that node (We can see that Trento node is selected in the combo-box and 2 of the GE's that are currently deployed to Trento node is listed with the number of instance).

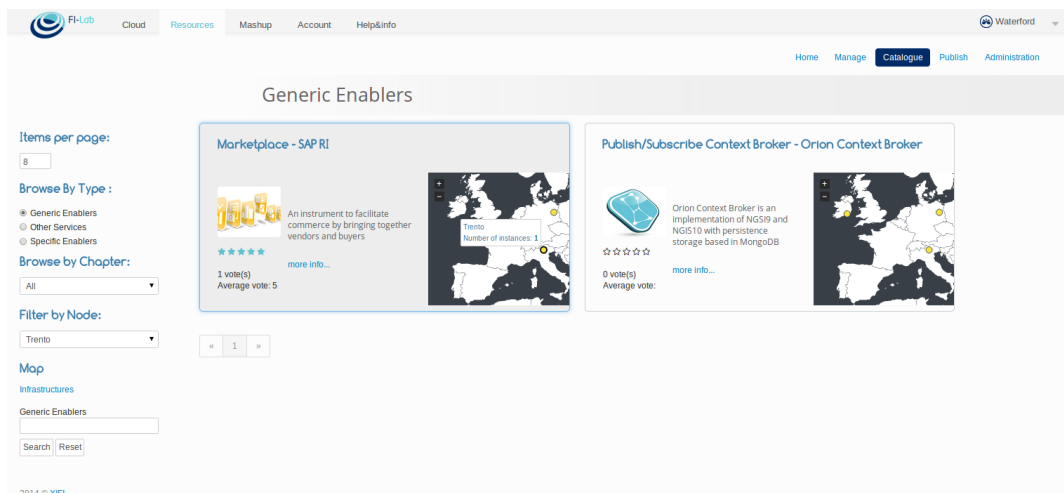


Figure 9: Resource Catalogue – fixing the node Trento

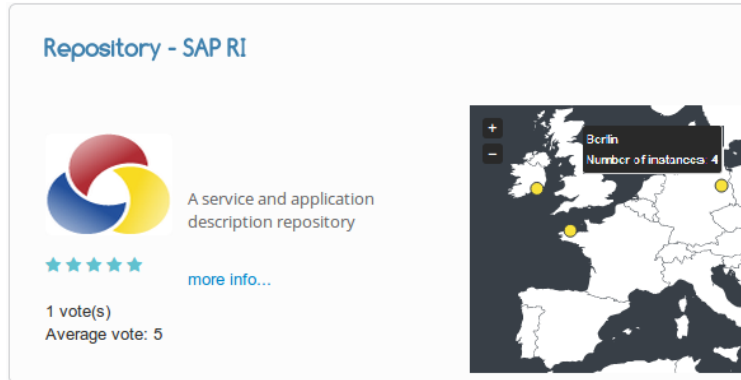


Figure 10: Resource Catalogue – Generic Enable Detail.

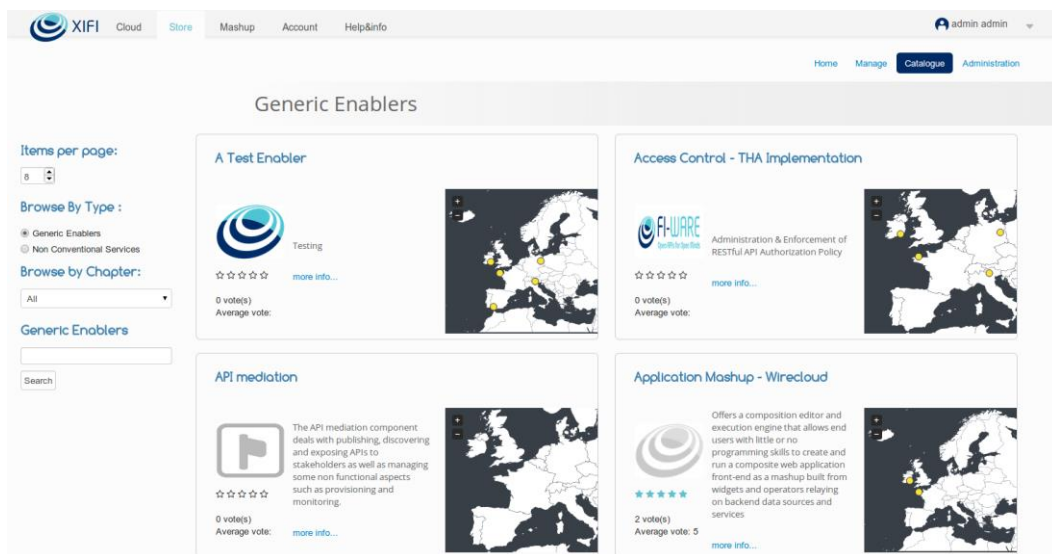


Figure 11: Resource Catalogue.

- GE's can be viewed with chapter by selecting the chapter name from the left panel:

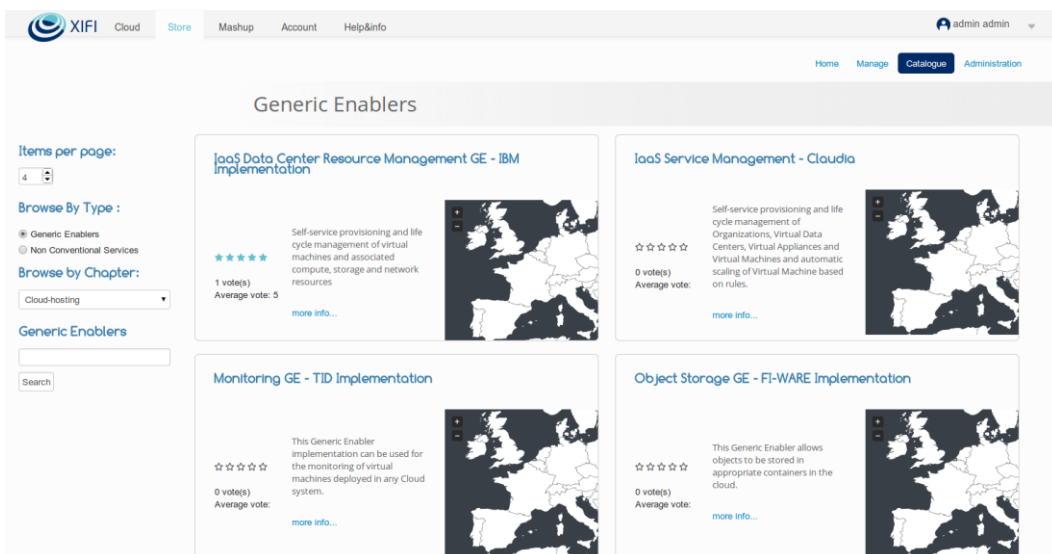


Figure 12: Resource Catalogue – selecting the chapter.

- GE's can be searched with the keyword:

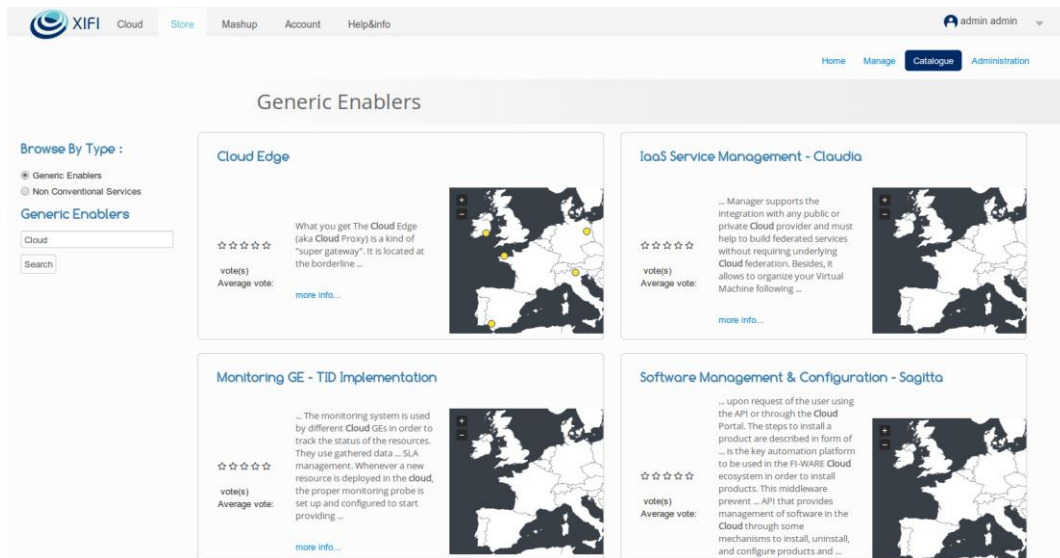


Figure 13: Resource Catalogue – Search with keyword.

- User can click the Map link on the left panel to see the details of the all nodes with the information about GE, SE and NCS's. The next figure depicts the number of the instances of SE, GE and NCS that are deployed in Trento node.

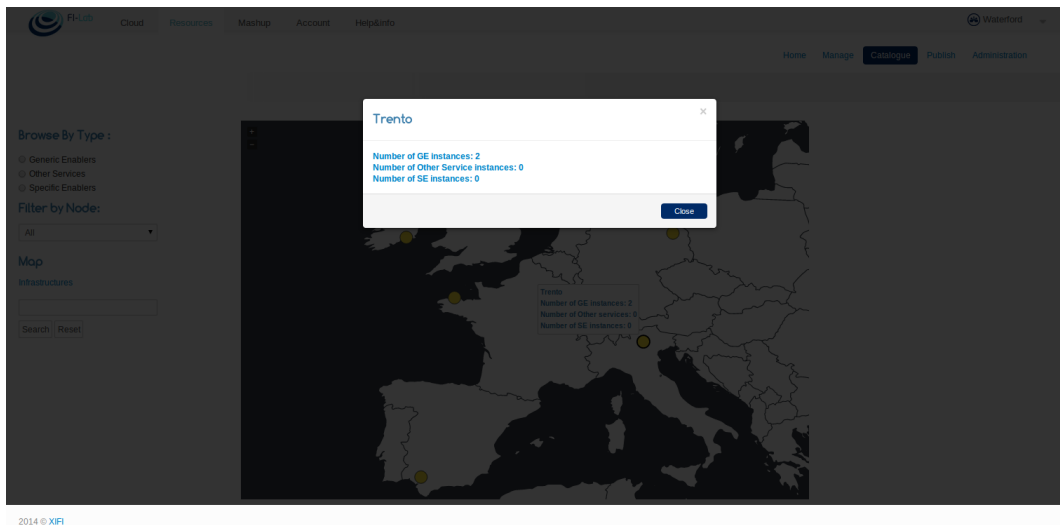


Figure 14: Resource Catalogue – Search by map and node details.

- User can click on the link to see the details of the Generic Enabler

This section is the place where user can see more information about the GE, for example:

- Logo and voting info about the GE
- In which nodes this GE is deployed
- Comments from users
- Terms and conditions

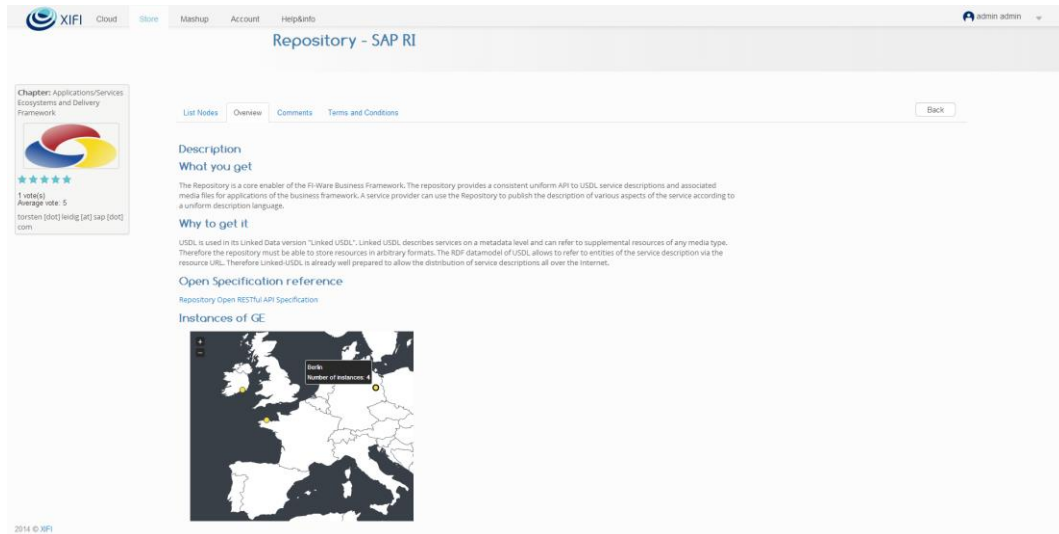


Figure 15: Resource Catalogue – GE description.

Non Conventional Services and Specific Enablers

Register Non Conventional Services and Specific Enablers:

- It is possible to register two types of services: i) Specific Enablers and ii) Advanced/Non-Conventional Services. There is an option to specify if this is a SE or NCS, if the user has the rights see this selection and selects this option then it will be created as a SE otherwise it will be a NCS.
 - Specific Enabler can be published by Software Providers and Infrastructure Owners.
 - Register the image in the Cloud Portal, to be deployed automatically for the stakeholders that want to use and monitoring.
 - Advanced/non-conventional capabilities (e.g. sensor networks) can be published by the Infrastructure Owners. These don't need to be registered in the Cloud Portal, since it is only necessary to provide their properties and characteristics. This allows obtaining the contact information of the infrastructure owner in order to ask him or negotiate with him the offer details.
- In both cases, register the service description through the Linked-USDL [21]
- User can select an USDL file and register the services also; here is an example of a service that is registered by using an USDL file. The Figure 16 and Figure 17 are related with this action.

There are some restrictions for the USDL content format as:

- *Currency value* must be in capital letters (EUR eg.)
- *hasUnitOfMeasurement tag* should contain the values like per week, per day, per month, per year..

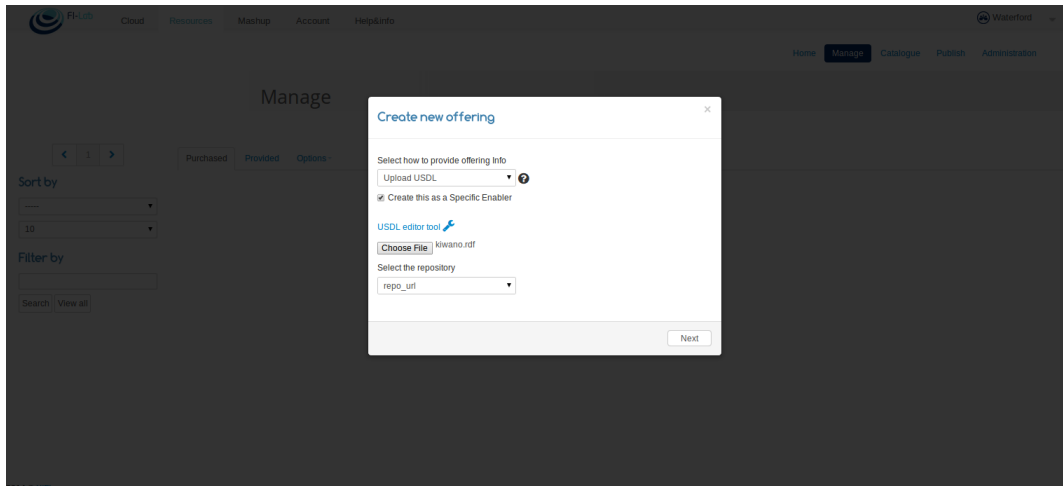


Figure 16: Resource Catalogue, create new offering.

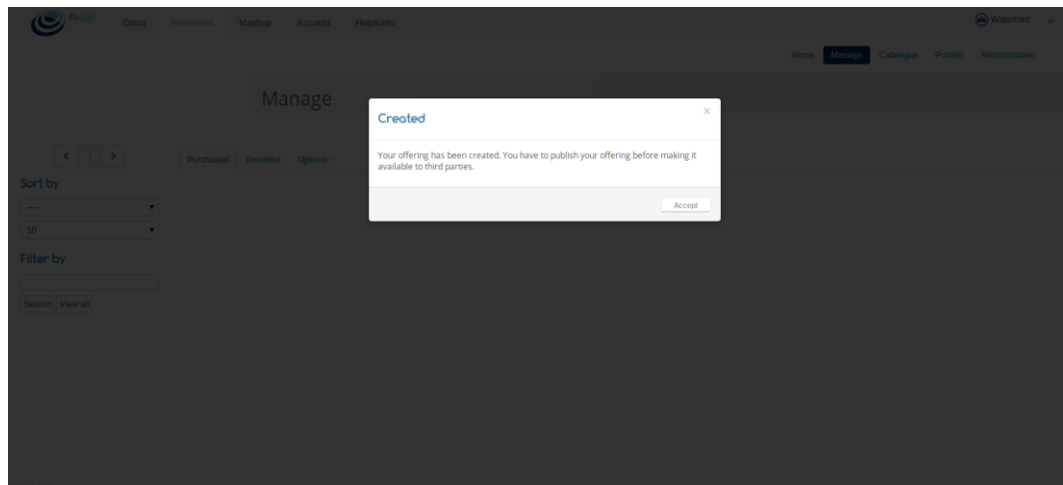


Figure 17: Resource Catalogue, created new offering

- Users are not forced to provide USDL files all the time, they can be created by clicking "Create basic USDL" option in the combo-box and provide information related with the USDL content.

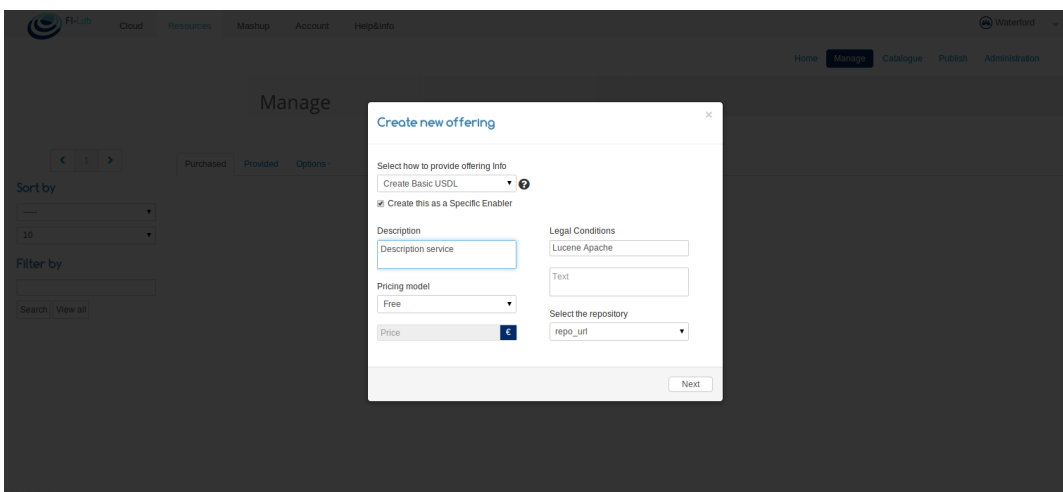
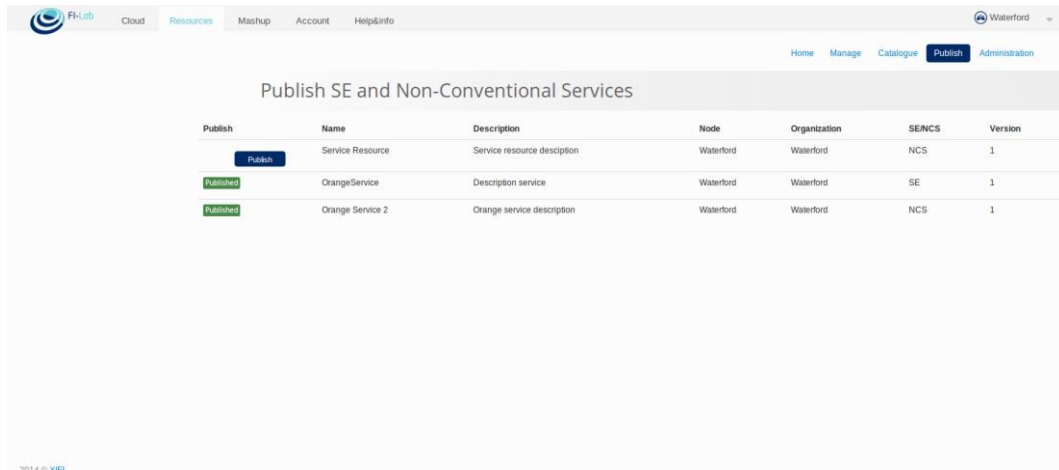


Figure 18: Description of the services that will be included in the Catalogue.

Validate the services. Resources needed to be published/validated before going live.

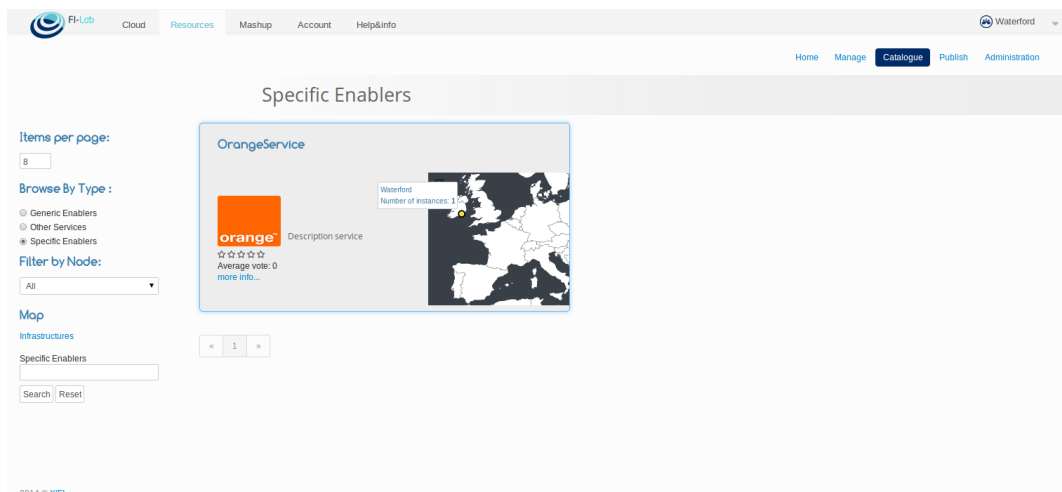
- The federator role needs to validate that all the content are correct, before to confirm the publication in the catalogue.



Publish	Name	Description	Node	Organization	SE/NC/S	Version
Publish	Service Resource	Service resource description	Waterford	Waterford	NCS	1
Published	OrangeService	Description service	Waterford	Waterford	SE	1
Published	Orange Service 2	Orange service description	Waterford	Waterford	NCS	1

Figure 19: List of services pending to be validated.

- After the Federator has approved them, the services will be available to find and see the status in the XIFI Portal.



Items per page: 8

Browse By Type:

- Generic Enablers
- Other Services
- Specific Enablers**

Filter by Node: All


Map

Infrastructures

Specific Enablers

Search | Reset

OrangeService



Description service

Waterford
Number of instances: 1

Average vote: 0
more info...

Figure 20: Specific Enabler available in the Catalogue.

Afterwards, the FI Developers can find and use these services. Hence, they need to do the following steps: i) Single Sign On in the XIFI Portal, ii) Browse Resource Catalogue; iii) Selection and deployment of resources.

List of Non-Conventional Services and Specific Enablers:

These services are stored and created in the XIFI catalogue and can be accessed by using internal APIs [31].

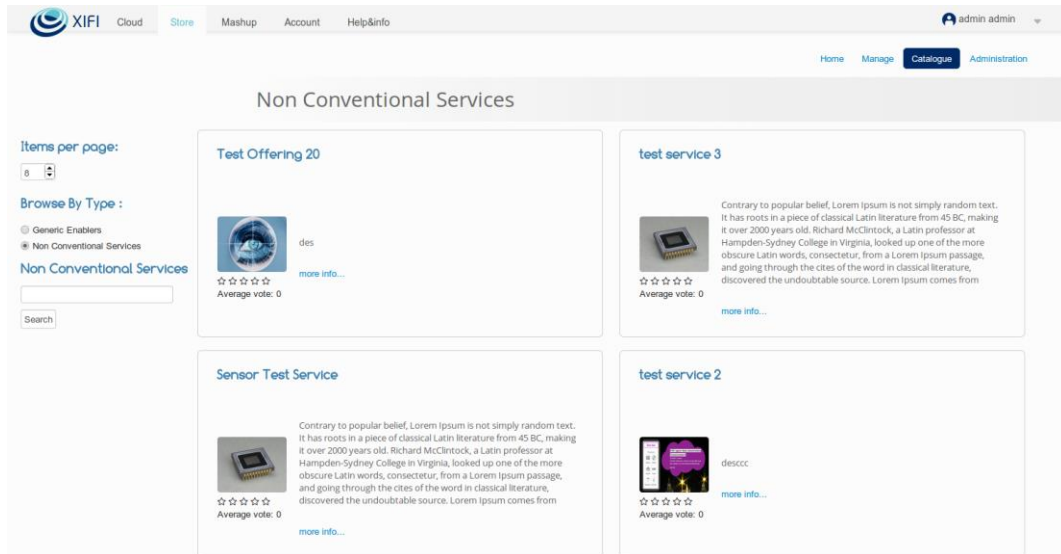


Figure 21: Resource Catalogue – Non Conventional Services

- User can search through Non-Conventional Services by a given keyword.

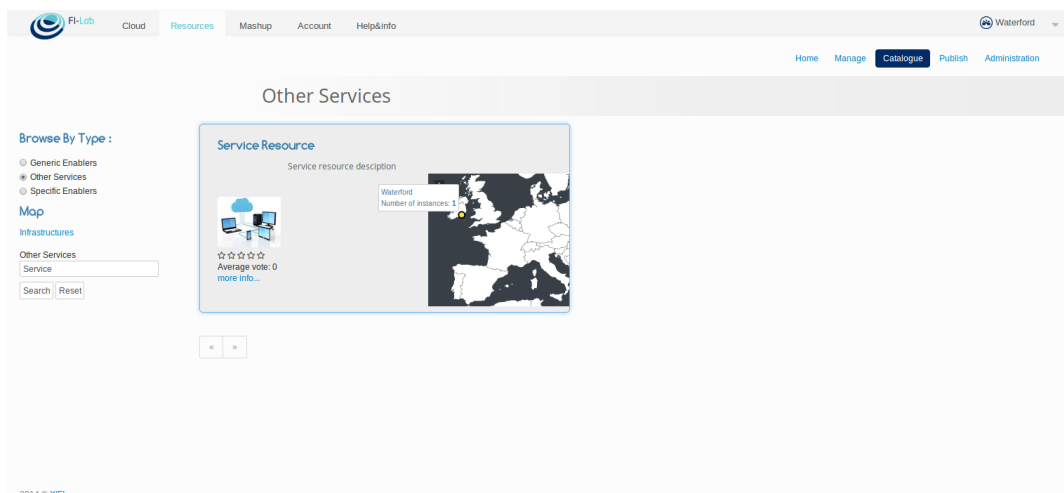


Figure 22: Resource Catalogue – Other Services and keyword search.

- User can click the “More info” link and view the details for a specific Non-Conventional Service or SE:

This tab contains information about the creation of offerings in internal WStore workflow.

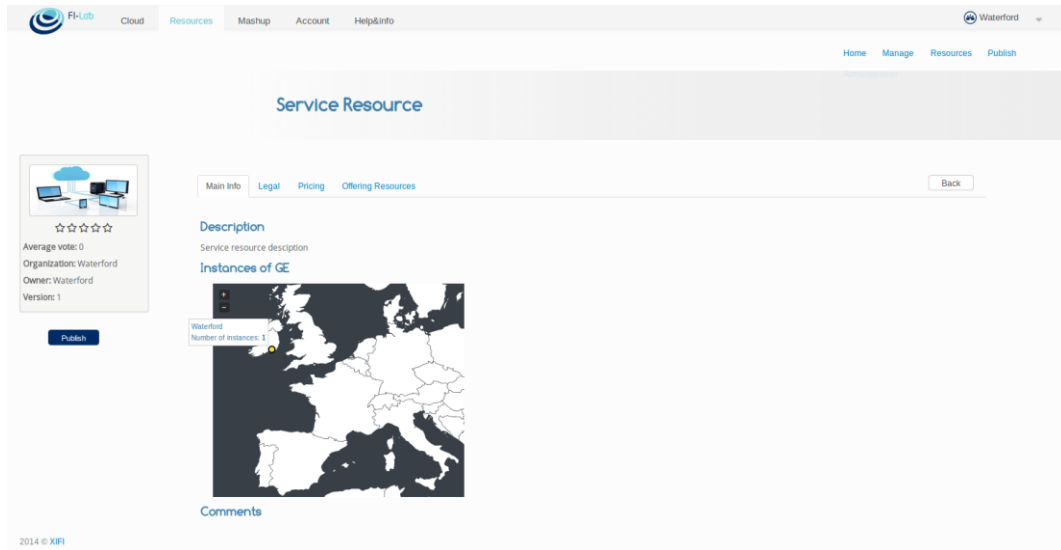


Figure 23: Resource Catalogue – Detail Non Conventional Services or SE

Planned OS license	Apache License Version 2.0.[7] As the original tool.
Reference OS community	

Table 6: Summary SLA Manager.

Consists of

- SLA Dashboard (GUI for the portal)
- SLA management (exposed an API specs, including the SLA negotiation, enforcement, decision components and the interaction with the Federation Monitoring)

Depends on

- Federation Monitoring [91]
- Federation Identity Management GE [93]
- Deployment and Configuration Adapter – DCA [90]

3.2 Component leaders

Developer	Email	Company
Roman Sosa Gonzalez	roman.sosa.external@atos.net	Atos
Joaquin Iranzo	Joaquin.iranzo@atos.net	Atos
Elena Garrido	elena.garrido@atos.net	Atos
Ömer Özdemir	omer.ozdemir@atos.net	Atos
Panos Karkazis	pkarkazis@synelixis.com	SYNELIXIS
Panos Trakadas	ptrak@synelixis.com	SYNELIXIS

Table 7: List of responsible – SLA Manager.

3.3 Motivation

The definition of the service quality in production environments is essential, both to have a mutual agreement between the participating parties and to perform actions depending on the agreed terms (violations identification, feedback for end user, mitigate problems, accounting reports, etc). However, the introduction of the SLA component is a challenge, since it is not easy neither defining the terms nor reaching agreements among the parties; also in the enterprise environments (less technical) the definition of Key Performance Indicators (KPI) and their monitoring are complex tasks.

So, it is considered important to introduce the necessary tools in order to support the SLA lifecycle, since it provides an important added value, both for the user in order to visualize the agreements/violations and for the providers in order to efficiently manage their services (taking action if it is deemed necessary).

Being aware of the challenge and complexity, it has been decided to follow a bottom-up implementation to cover different scenarios, focusing on the technical side. Mainly, we have started from the definition of the metrics, the creation of simple agreements (user desired), the detection of the violation and the integration of all the components in the federated environment.

The real negation action has been postponed for next phases, since it will involve a definition of business rules and agreements in compliance with QoS.

An overview of available specification and tools, as described in the Section State of the art [section

3.5] has been carried out. After analyzing these options, the WS-Agreement specification has been selected and the Cloud4SOA framework has been chosen as starting point. These tools have been selected for the following reasons:

- WS-Agreement:
 - It is a standard that defines an extensible protocol widely used to negotiate SLA in distributed computing.
 - It provides the definition and the mechanisms to automate the SLAs set-up, for monitoring and enforcement.
 - It is well recognized by the community. Before its introduction there was the tendency to propose new SLA languages such as WSOL, WSLA, WSML, SLang, etc. Now the community focuses on extending WS-Agreement rather than proposing new solutions.
- Cloud4SOA framework:
 - It implements the WS-Agreement specification.
 - It provides a license Apache 2.0
 - It is a light version and it has a modular and flexible design
 - It is exposed by REST API, which allows decoupling among the different layers.

This component has different competitive advantages (apart from the previous ones):

- It can be operative in a federated environment.
- It can include a GUI (dashboard) to manage the SLAs lifecycle.
- It can be isolated in order to be independent from the associated project. Moreover, it can be deployed on any environment to be reused.

3.4 User stories backlog

id	User story name	Actors	Description	Task id
1	Service Provider operations (backend)	SLA manager GUI	SLA Manager should be able to create a new provider and return a provider.	759
2	Template operations (backend)	SLA manager GUI	SLA Manager should be able to create a new template associated to a service and return a template.	760
3	One-shot negotiation (backend)	SLA manager GUI	SLA Manager should be able to accept an Agreement Offer as a new service contract	761
4	Get agreement (backend)	SLA manager GUI	SLA Manager should be able to provide the contract agreed between the two parties	762
5	Start/stop enforcement (backend)	SLA manager GUI	SLA Manager should be able to start/stop an enforcement job for a service	763
6	Enforceme	SLA	SLA Manager should periodically retrieve the metrics associated	764

	nt (backend)	Manager	with the service level objectives, and evaluate if the objectives are being fulfilled or not, generating service violations in the latter case.	
7	Get violations (backend)	SLA manager GUI	SLA Manager should be able to provide the violations generated by a service or the violations generated by a provider.	765
8	Get SLA template	End user (consumer and provider)	The SLA Manager GUI should be able to provide a service SLA.	766
9	One-shot negotiation	End user (consumer)	The user selects a service, fills a range of values of the variables being monitored by the service, and then agreement is accepted as new service agreement. The enforcement job should be started. NOTE: there is no negotiation here; the agreement is accepted as is.	767
10	Get agreement info	End user (consumer and provider)	The SLA Manager GUI should be able to provide the contract agreed between the two parties, and the violations generated by the service.	768
11	View dashboard of agreements	End User (consumer and provider)	The consumers and providers can manage their agreements, visualizing the violations and their details.	1503
12	The user is identified in the federation.	End User (consumer and provider)	The user enters in the SLA Dashboard, he is identified by the XIFI idM and he can enter in the component	1504

Table 8: Interoperability tool - User stories backlog

3.5 State of the art

Service Level Agreement (SLA) is widely used for managing quality of service in distributed environments and it has been the focus of research and development projects (either industry-driven projects or as part of research activities). In this section, two different parts regarding the SLA module are analyzed: i) the standards and languages proposed for the specification of SLA ii) the different industrial frameworks and some approaches deployed in other European funded projects that allow the specification of SLAs [35].

Service Level Agreement specifications. There are two major technologies (WSLA and WS-Agreement) and other approaches that address SLA specification, creation and monitoring:

- *WSLA (Web Service Level Agreements)* [36] was an IBM research project that published the WSLA specification in January 2003. It is still available at IBM's web pages and can be downloaded. In contrast to WS-Agreement, WSLA is less flexible with respect to domain specific languages. Indeed, the WSLA XML schema already contains the necessary definitions for the various parties, the SLA parameters and how they are computed, as well as the definitions for the parties' obligations, such as Service Level Objectives (SLOs) and Guarantees. Hence the schema always has to be adapted to be used in a specific domain. WSLA was used as technology for creating SLAs in several research projects but today is rarely used outside of IBM, mostly because missing access to the source code renders the framework inflexible.
- *WS-Agreement* [33] is the only open standard specifying a language and a protocol for creating SLAs. It is a full recommendation of the Open Grid Forum (OGF) and proposes a

protocol to define the required services and operations to create and monitor service level agreements. The protocol allows a one-step negotiation between a provider and a consumer usually based on templates with the QoS offered by the service provider.

The WS-Agreement model defines two types of services, the agreement factory service and the agreement service. The agreement factory service is responsible for creating agreements between a service consumer and provider and for instantiating the associated service with the agreed QoS. The agreement service is responsible for monitoring the compliance of agreements and of the associated services. WS-Agreement specification defines a set of interfaces in order to interact with the agreement factory service and the agreement service. These interface descriptions are provided in the form of WSDL and one specific interface is defined by one WSDL port type. The WS-Agreement protocol supports the symmetric deployment of its port types. Both services are modelled as web service resources conforming to the WSRF specification. A web service resource is a web service instance that is uniquely identified by an endpoint reference (EPR). Endpoint references are defined in the WS-Addressing specification [37].

It is used in projects of related Grid/Cloud and also in different sectors like autonomic provisioning and multimedia content negotiation. While most of them are research projects, there are also developments integrating WS-Agreement in commercial software.

- *WSOL* (Web Service Offerings Language)[38] is a XML notation compatible with WSDL which enables formal specification of functional constraints, QoS constraints, simple access rights (for service differentiation), price, and relationships with other service offerings of the same Web Service.
- *SLAng*[39] is a language for defining SLAs that considers end-to-end QoS and highlights that QoS provisioning has multiple facets and requires complex agreements between compute services, network services, and storage services. The language has been developed by the Department of Computer Science at UCL, U.K..
- *RBSLA* (Rule-based Service Level Agreement language) [40] is based on RuleML [16] which describes contracts such as service level agreements or policies in a machine readable syntax such that it can be easily used by a rule engine in order to monitor existing performance and apply contractual rules as necessary.

Frameworks and existing approaches.

- *SLA@SOI* [41] aimed at defining a SLA-enabling reference architecture suitable for both new and existing service-oriented systems and Cloud infrastructures. Based on this definition the project released a suite of open-source software and components to implement SLA-aware solutions. SLA@SOI follows a vertical approach including the complete business/IT stack. The SLA@SOI SLA framework covers all levels of the service provisioning process. Although it is partly derived from WS-Agreement and early versions of WS-Agreement Negotiation, it is not standards-compliant and defines a proprietary SLA stack. License LGPL.
- *OPTIMIS* [42]. Another important aspect to be considered when building Cloud application is the trust created between service and infrastructure providers, since service providers rely on the infrastructure provider to run their services. In OPTIMIS has been developed a bidirectional Trust Framework (TF) [43]. OPTIMIS Trust framework aims to evaluate and help with the service deployment and optimization task. Enclosed inside the TREC (Trust, Risk, Eco-efficiency and Cost) parameters, the TF monitors can evaluate both the service provider(SP) and the infrastructure provider (IP) work. For both sides, the information retrieved from the monitoring system is processed using statistical and logical operators, and stored to create a historical asset used to compute the final trust. Controlling the IP and SP, the framework developed focuses on key aspects for each of the actors, controlling in the case of the SP parameters related to the performance and IP control of the service behaviour. From the IP side resource consumption or legal constraints are elements to be controlled. However, for

both sides there is a strict control of the compliance of the SLA agreement negotiated at pre-deployment time. License BSD.

- *Cloud4SOA* [44]. In a scenario where PaaS systems are used and very diverse architectures presenting dissimilar resource-level metrics are considered, the Cloud4SOA European Project provides a RESTful implementation of the WS-Agreement specification that consume monitoring data based on a unified Cloud Platform-Independent metrics approach. The framework allows Cloud application developers and Cloud users to monitor the health and performance of business-critical applications hosted on multiple Clouds environments in a complete way. This approach is based on the definition of specific metrics based on Cloud4SOA semantic modelling. The management of SLAs is a key aspect in Cloud4SOA. However, in the scope of project, SLAs do not aim at representing a contractual relationship between the customers consuming virtualized Platforms, and the PaaS vendors that provide them. The SLA framework is an adaptation to PaaS environment from the SLA@SOI developments. Licence Apache2.0
- *Contrail* [45] is a Cloud software stack of components that are designed to work together to combine a number of independent clouds into one integrated federated cloud. Users can for instance submit work to the cloud federation and let the federation decide to which resource provider it should be sent for execution. The placing is decided by means of Service Level Agreements (SLAs). The user can define the SLA for his/her work. The federation can split work, if possible, and split the associated SLAs, thus distributing the work over the resource providers that (best) meet the SLAs. License BSD
- *mOSAIC* (Open source API and Platform for multiple Clouds) [46] is an FP7-ICT project which is developing a platform that promotes an open-source Cloud application programming interface (API) and a platform targeted for developing multi-Cloud oriented applications. Its goal is to provide enough freedom both at resource and programming level such that Cloud-based services can be easily developed and deployed. SLA in mOSAIC is focusing on infrastructure-as-a-service (IaaS), mostly on the definition of parameters for Cloud resources such as compute, storage, and network resources. Initial approach to SLA definition was done using WS-Agreement, partially coupled with WS-Policy, and was later changed to SLA@SOI. Furthermore, the SLA was kept compliant with existing OCCI Infrastructure terms. License Apache 2.0.

3.6 Architecture design

SLA Management in XIFI will be based on the WS-Agreement specification [33]. The objectives of the SLA module are: i) to define a language and a protocol for advertising the capabilities of service providers; ii) to create agreements based on the templates; iii) to monitor agreement compliance at runtime. An agreement between a service consumer and a service provider specifies one or more Service Level Objectives (SLOs). These are the expressions of the requirements of the service consumer and of the assurances by the service provider about the quality of services. An agreement lifecycle includes the creation, monitoring and termination of the agreement.

The WS-Agreement specification describes an XML schema for specifying service level agreements (both applicable to SLA Templates, Agreement Offers and Agreements). SLA Templates, Agreement Offers and Agreements in XIFI will be defined and described using the *WS-Agreement schema* [37]. The main parts of the WS-Agreement schema are described below.



Figure 25: WS-Agreement schema

We have introduced two subcomponents in the architecture in order to cover the graphical user interface and the business layer. This allows to decouple the architecture in order to support modularity and flexibility.

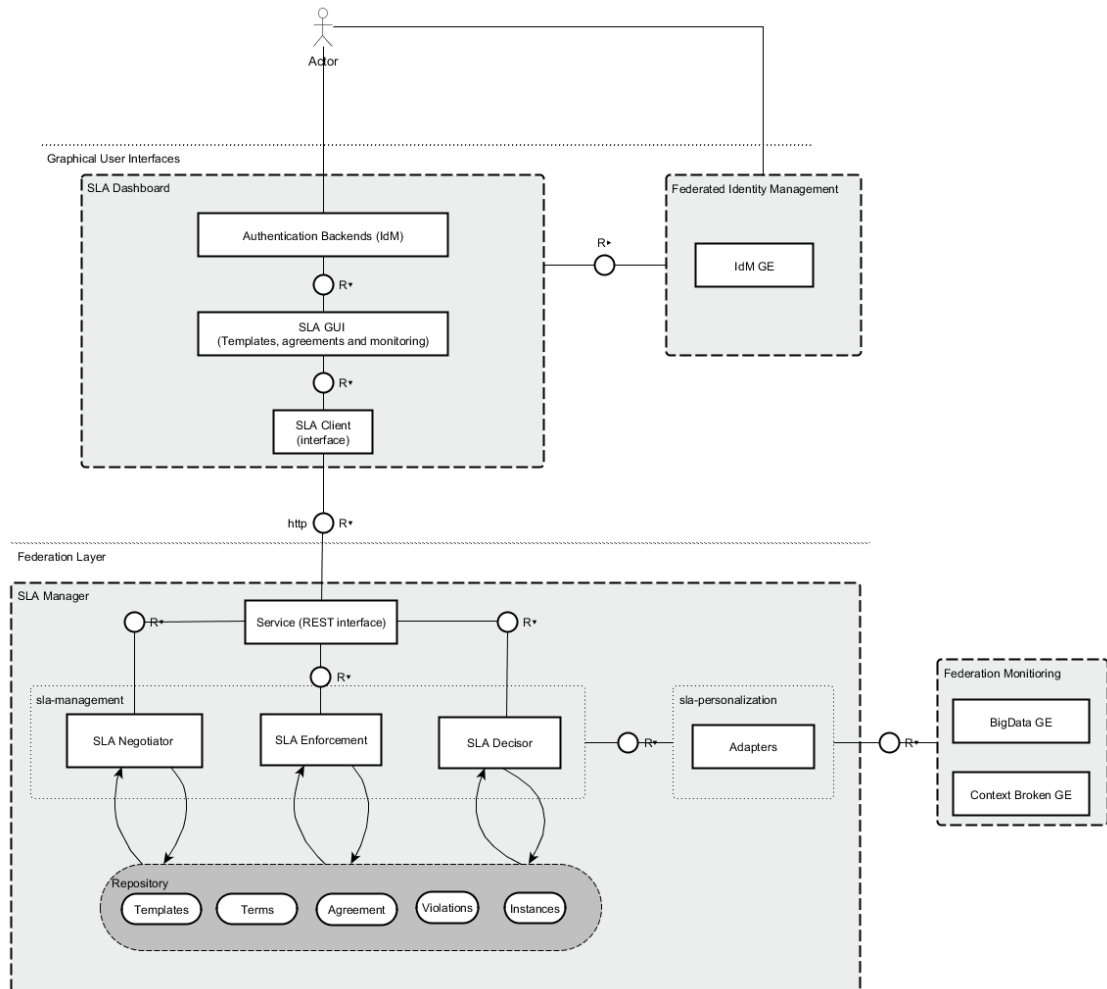


Figure 26: SLA Manager FMC compositional structure diagram.

- *SLA Dashboard* (GUI Layer): This part is responsible to show developers and providers all the functionalities through the GUI. It is also responsible to interact with the SLA Manager (backend layer) and then process the result visually.
- *SLA Management* (Federation Layer) is the module responsible to provide all the backend functionalities. It can be considered part of the Federation Layer since it is responsible both to collect the information of the Federation Monitoring and expose the SLA information to the rest of the components (SLA Dashboard, Cloud Portal, Recommenders...)

The *SLA Dashboard* subcomponent is composed by the following modules:

- *Authentication Backend (IdM)*: it is responsible to identify the users and delegate their authentication to the Federator Identity Manager. It is responsible to manage the authorization of the functionalities and the organizations;
- *SLA GUI*: it is responsible to manage all the visual SLA lifecycle. It allows managing the templates, the agreements and showing the status of the agreements through the dashboard;

- *SLA Client*: it is responsible to interact with the SLA Manager, since the SLA Dashboard doesn't contain the business logical;

The *SLA Management* subcomponent is divided in the following modules:

- *service (REST interface)*: It is responsible to expose the backend functionalities through the REST interfaces. It always use the "sla-core" to execute the actions and never access directly to the repository;
- *sla-management*: it contains the backend business code (publish, negotiation, enforcement, recovery);
- *repository*: access to the main entities (defined in the class diagram);
- *sla-personalization*: it contains all the specific adapters for every project. This allows to change the adapters easily, even if the origin of the monitoring is changed, without affecting the sla-management modules.

The SLA Management subcomponent will interact with other modules:

- The SLA Management needs to recover all the monitoring data in order to guarantee the service level agreement that has been agreed on the previous interactions. Hence, the Federation Monitoring component provides to this module all the required monitoring information.
 - *BigData GE* is responsible to provide the historical monitoring data.
 - *Context Broker GE* is responsible to provide the current monitoring data.

Metrics retrieval

It is necessary to implement an adapter in order to retrieve the metrics needed by the SLA component to perform the enforcement. In this case, the adapter has to retrieve the metrics from the Federated Monitoring component.

The adapter has to implement the following interface:

```
public interface IMetricsRetriever {
    public IMonitoringMetric getMetrics(
        String serviceId, String serviceScope, String variable,
        Date begin, Date end, int maxResults);
}
```

Metrics evaluation

To keep the module as generic as possible, the SLA module does not impose a way to evaluate the constraints. Anyway, a simple constraint definition and evaluation is implemented by the following grammar:

```
E ? id OP VALUES
OP ? "GT" | "GE" | "EQ" | "LT" | "LE" | "NE" | "BETWEEN" | "IN"
VALUES ? "[" VALUELIST "]"
VALUES ? VALUE
VALUELIST ? VALUELIST "," VALUE
VALUE ? val
```

A constraint is satisfied if a metric or a set of metrics (determined by 'id') evaluate the condition to true. Otherwise, the metric is considered as a Breach (SLA violation).

The SLA Management needs to use an adapter in order to retrieve the metrics from the Monitoring Module. If this simple constraint definition and evaluation is not sufficient, there is an interface which the adapter has to implement in order to create more complex constraints. This interface is the following:

```
public interface IMetricsValidator {
    Iterable<IMonitoringMetric> getBreaches(
        String variables, Collection<IMonitoringMetric> metrics, String constraint);
}
```

Asking for the breaches in a collection of metrics (instead of directly evaluate if a metric is a breach) allows to evaluate the metrics as a whole (e.g., to calculate an average value).

Repository

The SLA repository service provides useful capabilities to manage the persistence of SLATemplates, SLA Agreements and the relation between Services / Agreements / Templates. Moreover, it provides a place where SLA Violations and Recovery Actions (actions taken after a violation) will be stored and retrieved.

The logic of the service will be implemented in Java library; it will provide HTTP Interface / REST Interfaces. Hibernate [71] helps to automatically map our objects to the relational database.

There is not a specific requirement with regards to the DB, although MySQL has been chosen for the development.

3.7 Release plan

Version Id	Milestone	User Stories
0.1	30.11.2013	5
1.0	31.12.2014	1,2,3,4
2.0	31.03.2014	6,7,8
3.1	31.07.2014	12,11
3.0	30.09.2014	9,10

Table 9: Release plan – SLA Manager

3.8 Test cases

Test id	Test description	Test script
1	Add provider	see below
2	List providers	see below
3	Add template	see below
4	List templates	see below
5	Add agreement	see below
6	Get agreement	see below
7	Get enforcement job	see below

8	Start enforcement job	see below
9	Get violations	see below
10	Verification of the SLA Dashboard installation	see below

Table 10: Test cases – SLA Manager

All test cases are executed in *SLA_Manager\sla-core\trunk\samples*

1-Add provider

```
curl -u user:password -H "Accept: application/xml" -H "Content-type: application/xml" -d@provider-trento.xml localhost:8080/sla-service/providers -X POST
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<message code="201" message="The provider has been stored successfully in the SLA Repository Database. It has location http://localhost:8080/sla-service/providers/Trento"/>

<provider>
  <uuid>Trento</uuid>
  <name>Trento</name>
</provider>
```

2-List providers

```
$ curl -u user:password localhost:8080/sla-service/providers
<?xml version="1.0" encoding="UTF-8"?>
<collection href="/providers">
<items offset="0" total="1">

<provider>
  <uuid>Trento</uuid>
  <name>Trento</name>
</provider>
</items>
</collection>
```

3-Add template

```
$ curl -u user:password -H "Content-type: application/xml" -d@Template_vm_Trento.xml localhost:8080/sla-service/templates -X POST
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<message code="201" message="The agreement has been stored successfully in the SLA Repository Database. It has location http://localhost:8080/sla-service/templates/template_vm-Trento:193.205.211.85"/>
```

4-List templates

```
$ curl -u user:password -H "Accept: application/xml" localhost:8080/sla-service/templates
<?xml version="1.0" encoding="UTF-8"?>
<collection href="/templates">
<items offset="0" total="1">
<wsag:Template xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
  wsag:TemplateId="template_vm-Trento:193.205.211.85">
  <wsag:Name>Template for service VM 193.205.211.85 in Trento</wsag:Name>
  <wsag:Context>
    <wsag:AgreementInitiator>Trento</wsag:AgreementInitiator>
    <wsag:ServiceProvider>AgreementInitiator</wsag:ServiceProvider>
    <wsag:ExpirationTime>2015-03-07T12:00:00+0100</wsag:ExpirationTime>
    <sla:Service xmlns:sla="http://sla.atos.eu">vm-Trento:193.205.211.85</sla:Service>
```



```

</wsag:Context>
<wsag:Terms>
  <wsag:All>
    <!-- FUNCTIONAL DESCRIPTION -->

```

5-Add agreement

```

$ curl -u user:password -H "Content-type: application/xml" -d@agreement_vm_Trento.xml localhost:8080/sla-
service/agreements -X POST
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<message code="201" message="The agreement has been stored successfully in the SLA Repository Database.
It has location http://localhost:8080/sla-service/agreements/agreement-vm-Trento:193.205.211.85-
MonitoringB"/>

```

6-Get agreement

```

$ curl -u user:password -H "Accept: application/xml" localhost:8080/sla-service/agreements/agreement-vm-
Trento:193.205.211.85-MonitoringB
<?xml version="1.0" encoding="UTF-8"?>
<wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
  wsag:AgreementId="agreement-vm-Trento:193.205.211.85-MonitoringB">

  <wsag:Name>Agreement Monitoring1 for one vm of Trento</wsag:Name>
  <wsag:Context>
    <wsag:AgreementInitiator>joaquin-iranzo</wsag:AgreementInitiator>
    <wsag:AgreementResponder>Trento</wsag:AgreementResponder>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:ExpirationTime>2015-03-07T12:00:00+0100</wsag:ExpirationTime>
    <wsag:TemplateId>template_vm-Trento:193.205.211.85</wsag:TemplateId>
    <sla:Service xmlns:sla="http://sla.atos.eu">vm-Trento:193.205.211.85</sla:Service>
  </wsag:Context>
  <wsag:Terms>

```

7-Get enforment job

```

$ curl -u user:password -H "Content-type: application/xml" localhost:8080/sla-service/enforcements/agreement-
vm-Trento:193.205.211.85-MonitoringB
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<enforcement_job>
  <agreement_id>agreement-vm-Trento:193.205.211.85-MonitoringB</agreement_id>
  <enabled>>false</enabled>
  <last_executed xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</enforcement_job>

```

8-Start agreement enforcement

```

$ curl -u user:password -H "Content-type: application/xml" localhost:8080/sla-service/enforcements/agreement-
vm-Trento:193.205.211.85-MonitoringB/start -X PUT
The enforcement job with agreement-uuid agreement-vm-Trento:193.205.211.85-MonitoringB has started

```

Check enforcement job status

```
$ curl -u user:password -H "Content-type: application/xml" localhost:8080/sla-service/enforcements/agreement-vm-Trento:193.205.211.85-MonitoringB
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<enforcement_job>
  <agreement_id>agreement-vm-Trento:193.205.211.85-MonitoringB</agreement_id>
  <enabled>true</enabled>
  <last_executed xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
</enforcement_job>
```

9-Get violations

```
$ curl -u user:password localhost:8080/sla-service/violations?agreementId=agreement-vm-Trento:193.205.211.85-MonitoringB
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<violations></violations>
```

The list of violations will be empty until one of the boundaries has not fulfilled. For example the CPU in this agreement:

```
$ curl -u user:password localhost:8080/sla-service/violations?agreementId=agreement-vm-Trento:193.205.211.85-MonitoringB
```

```
<?xml version="1.0" encoding="UTF-8"?>
<collection href="/violations">
<items offset="0" total="1">
<violation>
  <uuid>ab39160f-2340-43bb-8e1f-94f41146d808</uuid>
  <contract_uuid>agreement-vm-Trento:193.205.211.85-Monitoring1</contract_uuid>
  <service_scope></service_scope>
  <metric_name>Performance</metric_name>
  <datetime>2014-09-11T11:28:22Z</datetime>
  <actual_value>80</actual_value>
</violation>
</items>
</collection>
```

10- Verification of the SLA Dashboard installation

If the SLA Dashboard has been successfully installed and configured the Federation IdM, the following picture should be shown:

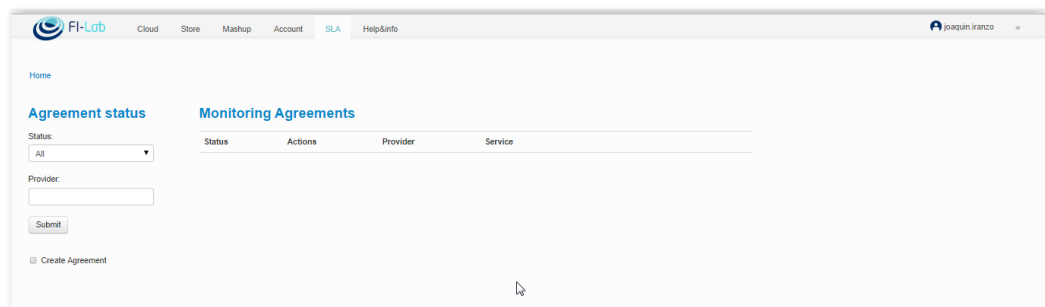


Figure 27: Test 10 Verification of the SLA Dashboard installation

3.9 Installation Manual

The two subcomponents are completely decoupled and their installation is also independent. They can be installed in the same server or in different ones and one only needs to configure them. The only restriction is that it is necessary to install first the SLA Management and afterwards the SLA Dashboard.

SLA Management

Requirements

- Oracle JDK 1.6.x - When installed, make sure you have the variable *JAVA_HOME* pointing to the Java installation folder.
- MySQL database [57] to install the database schema for the service.
- Maven tool for building applications.

Installation

- Download the source code [1]
- Create a database called 'atossla' with root user. If another database name is required some additional changes to jpa configuration will be required in order to set it.

```
create database atossla;
```

- Create a SLA user

```
mysql> create user atossla@localhost identified by 'password';
mysql> grant all privileges on atossla.* to atossla@localhost;
```

From command prompt, create needed tables:

```
mvn test exec:java -f sla-repository/pom.xml
```

Another option to create the database is execute a sql file (*sla-repository/src/main/resources/sql/atossla.sql*) from the project root directory:

```
bin/restoreDatabase.sh
```

Configuration

The project is made up of five main modules:

- SLA Repository
- SLA Management
- SLA Service
- SLA Tools
- SLA Personalization

A *_configuration.properties.sample_* that is placed in the parent directory has to be copied to *configuration.properties*. Several parameters can be configured through this file.

- *tomcat.directory* when building, war will be automatically copied to this directory,

- *db.** allows to configure the database username, password and name in case it has been changed from the proposed one in the section Creating the mysql database](#database). It can be selected if queries from hibernate must be shown or not,
- *log.** allows to configure the log files to be generated and the level of information,
- *manager.enforcement.** several parameters from the enforcement can be customized,
- *service.basicsecurity.** basic security is enabled. These parameters can be used to set the user name and password to access to the rest services.
- *parser.** different parsers can be implemented for the agreement and template. By default, wsag standard parsers have been implemented and configured in the file. Also date format can be configured.

Compiling

```
mvn install
```

If you want to skip tests:

```
mvn install -Dmaven.test.skip=true
```

The result of the command is a war in *sla-service/target*. The war is also copied to the directory pointed by *tomcat.directory* in the *configuration.properties* file.

SLA Dashboard

Before to install the SLA Dashboard, it is needed to be installed:

- the SLA Management (see previous section);
- the Federation Monitoring [91];

Requirements

- Python 2.7 [92].
- Django 1.6.4 [83]

Installation

- Download source code in the svn: https://xifisvn.res.eng.it/wp4/SLA_Manager/
- If you have not installed pip:

```
#sudo aptitude install python-pip
```

- Before install Django, you should validate that the Python version is 2.7. The component does not work with higher versions.
- Install django version 1.6.4

```
#sudo pip install Django==1.6.4
```

- Install all the dependencies

```
#sudo pip install django-extensions
#sudo pip install south
#sudo pip install django-debug-toolbar
```

```
#sudo pip install requests
#sudo pip install python-dateutil<2.0
#sudo pip install xlwt
```

- Configure the parameters of the SLA Dashboard in the file \sladashboard\setting.py:

Configuration of the SLA Management

- SLA_MANAGER_URL= Url of the sla-core service
- SLA_MANAGER_USER= username that is sent through basic requests to sla-core
- SLA_MANAGER_PASSWORD= password that is sent through requests to sla-core

Configuration of the Federation Identity Monitoring.

- TOKEN= id number provided by the IdM, when you register the application.
- SECRET= secret number provider by the IdM that it is associated to the TOKEN.
- Create the application in the Federation Identity Management.

The SLA Dashboard has to be registered in the Federation Identity Management [93].

In the Federated Identity Management, the responsible of the component has to “Add application” and introduce:

- Basic Data:
 - Name of the applications.
 - Description of the application.
 - URL of the application.
 - Callback URL is the URL where the IdM will call back after to identify the user ([http://\[Host_Name\]:\[Port_Host\]/slagui/callbackIdm](http://[Host_Name]:[Port_Host]/slagui/callbackIdm)).

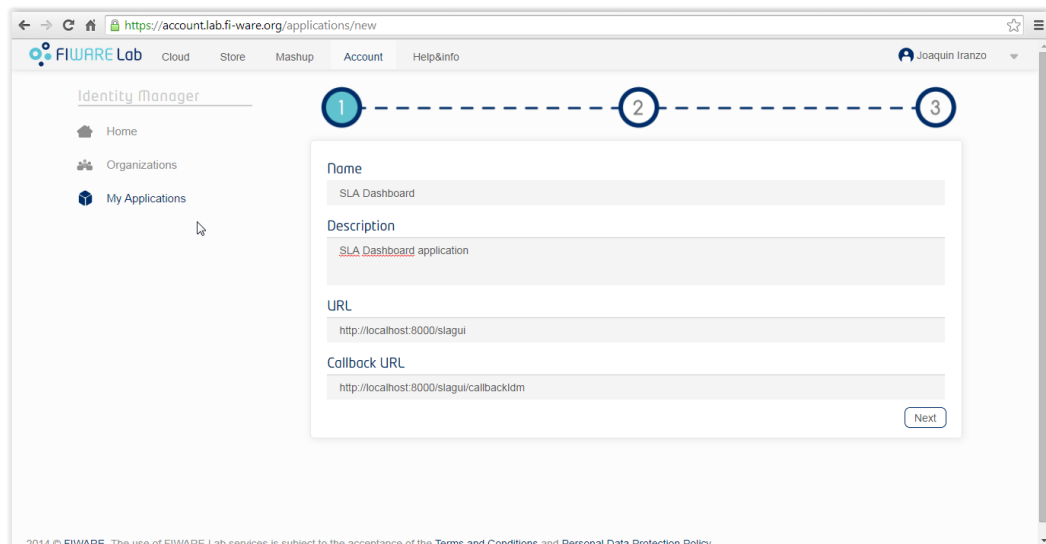


Figure 28: Step1 to add a new application in the IdM.

- Select the application's logo:

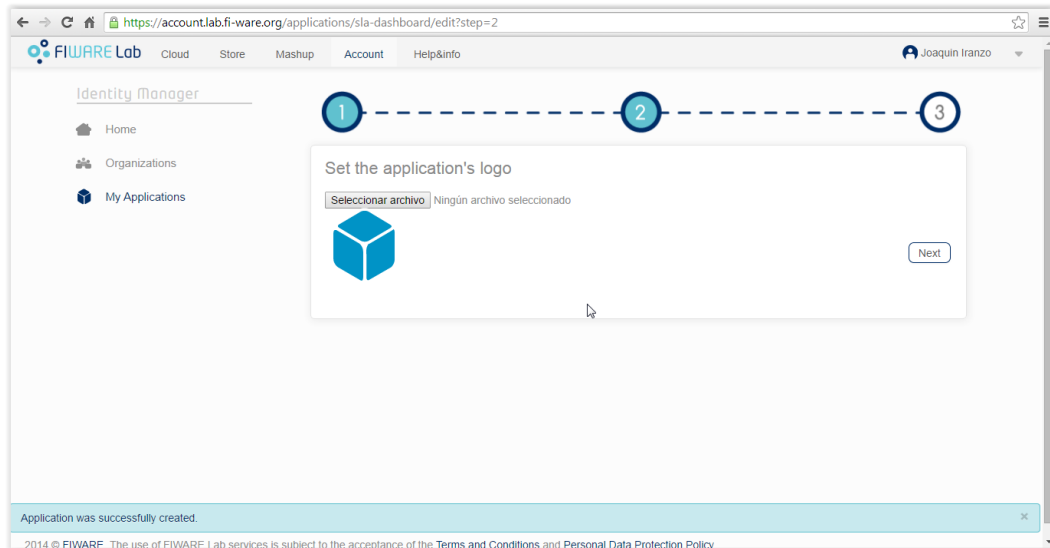


Figure 29: Step2 to add a new application in the IdM.

- Create new roles associated to the application (Customer and ProviderIO). The Provider and Purchaser roles are associated directly with the IdM.

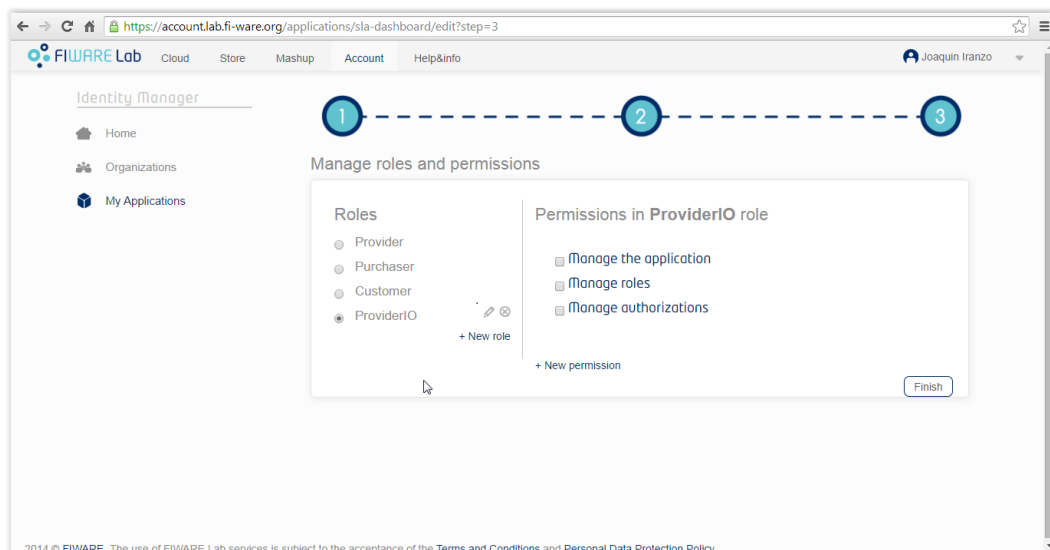


Figure 30: Step3 to add a new application in the IdM.

Afterwards, the responsible can find the credentials, in the link OAuth2 Credentials.

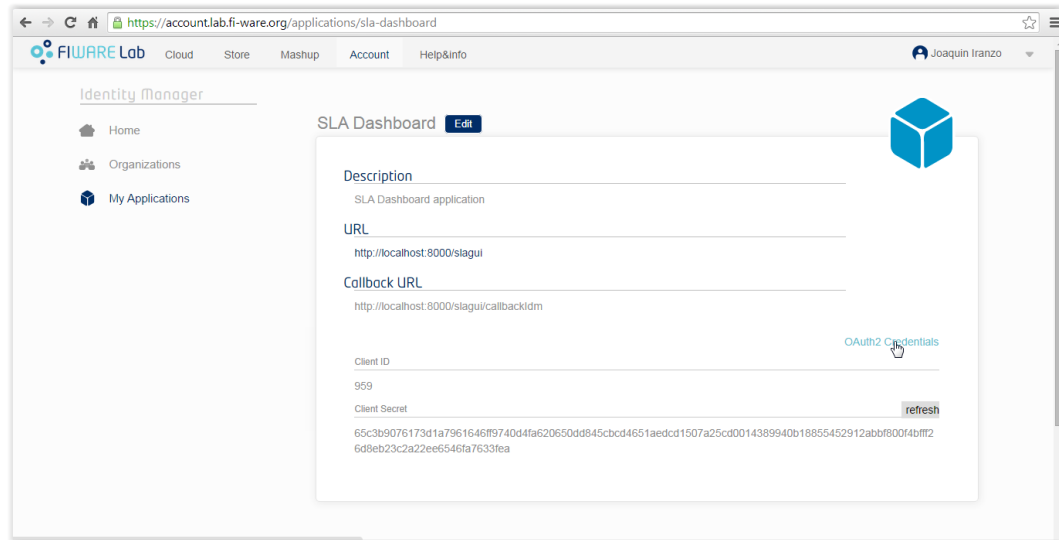


Figure 31: Detail of the credentials for this application.

These credentials have to be introduced in the Django configuration file (`\sladashboard\setting.py`).

The application can manage the users or organizations, through the IdM. It is only necessary to assign the appropriate roles (Customer or ProviderIO),

For more detail to how to use the Federation Identity Management [93] you can see the User Manual.

- The first time, it is needed to synchronize the database, in the project root folder execute:

```
#python manage.py syncdb
```

- Execute Django application, in the project root folder:

```
#python manage.py runserver
```

- After server starts, application login page can be accessed with this URL: `<IP_HOST>:Port/slagui/login`
- This redirects us to the IDM page and when a successful login is done, user is redirected to the SLA dashboard. It is important to configure correctly the application in the Federation Identity Management.
- There is not direct SLA data stored in the SLA Dashboard, so please, make sure that SLA Manager has to be installed and configured correctly, previously.

3.10 User Manual

API Specification

This API Specification describes the exposed operations for the SLA Management subcomponent, through RESTful API.

Start the `sla-service.war` in an Application Server. The module provides a REST interface to interact with the different entities.

API Introduction

The REST interface to the SLA Management (sla-core) subcomponent has the following conventions:

- Every entity is created with a POST to the collection url. The body request contains the serialized entity in the format specified in the content-type header. The location header of the response refers to the url of the new allocated resource. The return code is a 201 on success. Templates and agreements have special considerations (see the corresponding section).
- A query for an individual item is a GET to the url of the resource (collection url + external id). The format of the response is specified in the http header with the accept parameter. The return code is 200. As expected, a not found resource returns a 404.
- Any other query is usually a GET to the collection's url, using the GET parameters as the query parameters. The result is a list of entities that match the parameters, despite the actual number of entities. The return code is 200, even if the list is empty.
- Any unexpected error processing the request returns a 5xx.
- An entity (or list) is serialized in the response body by default with the format specified in the Content-type header (if specified). The request may have an Accept header, which will be used if the resource allows more than one Content-type.
- Updating an entity involves a PUT request, with the corresponding resource serialized in the body in the format specified in the content-type header. The return code is 200.

Generic operations

The generic operations of resources are shown below. Each particular resource (in following sections) shows the supported operations and any deviation from the behavior of generic operations.

The format of a resource can be modified by a project by using serializers.

GET /{resources}/{uuid}

Retrieve an entity by its uuid.

Request

```
GET /resources/{uuid} HTTP/1.1
```

Response in XML

```
HTTP/1.1 200 OK
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<resource>...</resource>
```

Response in JSON

```
HTTP/1.1 200 OK
Content-Type: application/json
{ ... }
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" http://localhost:8080/sla-service/resources/fc923960-03fe-41
curl -H "Accept: application/json" http://localhost:8080/sla-service/resources/fc923960-03fe-41
```


GET /resources{?param1=value1¶m2=value2...}

Search the resources that fulfill the params. All resources are returned if there are no parameters.

Request

```
GET /resources?param1=value1 HTTP/1.1
```

Response in XML

```
HTTP/1.1 200 OK
Content-type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<resources>
  <resource>...</resource>
  <resource>...</resource>
  <resource>...</resource>
</resources/>
```

Response in JSON

```
HTTP/1.1 200 OK
Content-type: application/json

[ {...}, {...}, {...} ]
```

Usage (for JSON and XML)

```
curl [-X GET] -H "Accept: application/xml" localhost:8080/sla-service/resources
curl [-X GET] -H "Accept: application/xml" localhost:8080/sla-service/resources?name=res-name
curl [-X GET] -H "Accept: application/json" localhost:8080/sla-service/resources
curl [-X GET] -H "Accept: application/json" localhost:8080/sla-service/resources?name=res-name
```

POST /resources

Create a new resource. The created resource will be accessed by its uuid. A message will be the usual response.

Request in XML

```
POST /resources HTTP/1.1
Content-type: application/xml

<resource>...</resource>
```

Request in JSON

```
POST /resources HTTP/1.1
Content-type: application/json

{ ... }
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" -H "Content-type: application/xml" -d@<filename> -X POST
localhost:8080/sla-service/resources
curl -H "Accept: application/json" -H "Content-type: application/json" -d@<filename> -X POST
```

```
localhost:8080/sla-service/resources
```

UPDATE /resources/{uuid}

Update an existing resource. The content in the body will overwrite the content of the resource. The uuid in the body must match the one from the url o not being informed.

Request in XML

```
PUT /resources/{uuid} HTTP/1.1
Content-type: application/xml

<resource>...</resource>
```

Request in JSON

```
PUT /resources/{uuid} HTTP/1.1
Content-type: application/xml

{...}
```

Response in XML

```
HTTP/1.1 200 OK
Content-type: application/xml

<?xml version="1.0" encoding="UTF-8"?>
<resource>
...
</resource/>
```

Response in JSON

```
HTTP/1.1 200 OK
Content-type: application/json

{...}
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" -H "Content-type: application/xml" -d@<filename> -X PUT
localhost:8080/sla-service/resources/{uuid}
curl -H "Accept: application/json" -H "Content-type: application/json" -d@<filename> -X PUT
localhost:8080/sla-service/resources/{uuid}
```

DELETE /resources/{uuid}

Delete an existing resource.

Request

```
DELETE /providers/{uuid} HTTP/1.1
<pre>

:"Response in XML and JSON"

<pre>
HTTP/1.1 200 Ok
```

```
Content-type: application/[xml | json]
```

... (Text indicating that the resource has been removed)

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" -X DELETE localhost:8080/sla-service/resources/fc923960-03fe-41
curl -H "Accept: application/json" -X DELETE localhost:8080/sla-service/resources/fc923960-03fe-41
```

Messages

Some of the above mentioned methods might return a message. Messages can be returned as XML or JSON.

Message Response in XML

```
Content-type: application/xml
```

```
<?xml version="1.0" encoding="UTF-8"?>
<message code="xxx" message="..."/>
```

Message Response in JSON

```
Content-type: application/json
```

```
{"code": "xxx", "message": ...}
```

Providers

- Provider collection URI: /providers
- Provider URI: /providers/{uuid}

A provider is serialized in XML as:

```
<provider>
  <uuid>fc923960-03fe-41eb-8a21-a56709f9370f</uuid>
  <name>provider-prueba</name>
</provider>
```

A provider is serialized in JSON as:

```
{"uuid": "fc923960-03fe-41eb-8a21-a56709f9370f",
 "name": "provider-prueba" }
```

GET /providers/{uuid}

Retrieves a specific provider identified by uuid

Error message:

- 404 is returned when the uuid doesn't exist in the database.

GET /providers

Retrieves the list of all providers

POST /providers

Create a provider. The uuid is in the file beeing send

Error message:

- 409 is returned when the uuid or name already exists in the database.

DELETE /providers/{uuid}

Removes the provider identified by uuid.

Error message:

- 404 is returned when the uuid doesn't exist in the database.
- 409 is returned when the provider code is used.

Templates

- Templates collection URI: /templates
- Template URI: /templates/{TemplateId}

The TemplateId matches the TemplateId attribute of *wsag:Template* element when the template is created. A template is serialized in XML as defined by ws-agreement.

An example of template in XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
  <wsag:Template xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:sla="http://sla.atos.eu"
  wsag:TemplateId="template012">
    <wsag:Name>ExampleTemplate</wsag:Name>
    <wsag:Context>
      <wsag:AgreementInitiator>provider02</wsag:AgreementInitiator>
      <wsag:ServiceProvider>provider01</wsag:ServiceProvider>
      <wsag:ExpirationTime>2014-03-
07T12:00:00+0100</wsag:ExpirationTime>
      <wsag:ServiceProvider>AgreementInitiator</wsag:ServiceProvider>
      <wsag:TemplateId>template01</wsag:TemplateId>
      <sla:Service xmlns:sla="http://sla.atos.eu">service3</sla:Service>
    </wsag:Context>
    <wsag:Terms>
      <wsag:All>
        <!-- functional description -->
          <wsag:ServiceDescriptionTerm wsag:Name="General"
wsag:ServiceName="Service0001">A GPS service</wsag:ServiceDescriptionTerm>
          <wsag:ServiceDescriptionTerm
wsag:Name="GetCoordsOperation" wsag:ServiceName="GPSService0001">operation to call to get the
coords</wsag:ServiceDescriptionTerm>
          <!-- domain specific reference to a service (additional or optional
to SDT) -->
          <wsag:ServiceReference wsag:Name="CoordsRequest"
wsag:ServiceName="GPSService0001">
            <wsag:EndpointReference>
              <wsag:Address>http://www.gps.com/coordsservice/getcoords</wsag:Address>
              <wsag:ServiceName>gps:CoordsRequest</wsag:ServiceName>
            </wsag:EndpointReference>
          </wsag:ServiceReference>
        </wsag:All>
      </wsag:Terms>
    </wsag:Template>
```

```

wsag:ServiceName="GPS0001">
    <wsag:ServiceProperties wsag:Name="AvailabilityProperties"
        <wsag:Variables>
            <wsag:Variable
wsag:Name="ResponseTime" wsag:Metric="metric:Duration">
                <wsag:Location>qos:ResponseTime</wsag:Location>
            </wsag:Variable>
        </wsag:Variables>
    </wsag:ServiceProperties>
    <wsag:ServiceProperties wsag:Name="UsabilityProperties"
wsag:ServiceName="GPS0001">
        <wsag:Variables>
            <wsag:Variable wsag:Name="CoordDerivation"
wsag:Metric="metric:CoordDerivationMetric">
                <wsag:Location>qos:CoordDerivation</wsag:Location>
            </wsag:Variable>
        </wsag:Variables>
    </wsag:ServiceProperties>
    <!-- statements to offered service level(s) -->
    <wsag:GuaranteeTerm wsag:Name="FastReaction"
wsag:Obligated="ServiceProvider">
        <wsag:ServiceScope wsag:ServiceName="GPS0001">
            http://www.gps.com/coordservice/getcoords
        </wsag:ServiceScope>
        <wsag:QualifyingCondition>
            applied when current time in week working hours
        </wsag:QualifyingCondition>
        <wsag:ServiceLevelObjective>
            <wsag:KPITarget>

                <wsag:KPIName>FastResponseTime</wsag:KPIName>
                <wsag:Target>

                    //Variable/@Name="ResponseTime" LOWERTHAN 1 second
                    </wsag:Target>
                </wsag:KPITarget>
            </wsag:ServiceLevelObjective>
        </wsag:GuaranteeTerm>
    </wsag:All>
    </wsag:Terms>
</wsag:Template>

```

An example of template in JSON is:

```

{
  "templateId": "template05",
  "context": {
    "agreementInitiator": "provider02",
    "agreementResponder": null,
    "serviceProvider": "AgreementInitiator",
    "templateId": "template01",
    "service": "service3",
    "expirationTime": "2014-03-07T12:00:00CET"
  },
  "name": "ExampleTemplate",
  "terms": {

```

```

    "allTerms":{
      "serviceDescriptionTerm":{
        "name":null,
        "serviceName":null
      },
      "serviceProperties":[
        {"name":null, "serviceName":null, "variableSet":null},
        {"name":null, "serviceName":null, "variableSet":null}
      ],
      "guaranteeTerms":[
        {
          "name":"FastReaction",
          "serviceScope":{
            "serviceName":"GPS0001",
            "value":"http://www.gps.com/coordsservice/getcoords"
          },
          "serviceLevelObjective":{
            "kpitarget":{
              "kpiName":"FastResponseTime",
              "customServiceLevel":null
            }
          }
        }
      ]
    }
  }
}

```

GET /templates/{TemplateId}

Retrieves a template identified by TemplateId.

Error message:

- 404 is returned when the uuid doesn't exist in the database.

GET /templates{?serviceId}

The parameter is:

- serviceId: id of service that is associated to the template

POST /templates

Create a new template. The file might include a TemplateId or not. In case of not being included, a uuid will be assigned.

Error message:

- 409 is returned when the uuid already exists in the database.
- 409 is returned when the provider uuid specified in the template doesn't exist in the database.
- 500 when incorrect data has been supplied

PUT /templates/{TemplateId}

Updates the template identified by TemplateId. The body might include a TemplateId or not. In case of including a TemplateId in the file, it must match with the one from the url.

Error message:

- 409 when the uuid from the url doesn't match with the one from the file or when the system has already an agreement associated
- 409 when template has agreements associated.
- 409 provider doesn't exist
- 500 when incorrect data has been supplied

DELETE /templates/{TemplateId}

Removes the template identified by TemplateId.

Error message:

- 409 when agreements are still associated to the template
- 404 is returned when the uuid doesn't exist in the database.

Agreements

- Agreements collection URI: /agreements
- Agreement URI: /agreement/{ AgreementId }

The AgreementId matches the AgreementId attribute of wsag:Agreement element when the agreement is created. An example of agreement in XML is:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsag:Agreement xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement"
xmlns:sla="http://sla.atos.eu">
  <wsag:Name>ExampleAgreement</wsag:Name>
  <wsag:Context>
    <wsag:ExpirationTime>2014-03-07T12:00:00+0100</wsag:ExpirationTime>
    <wsag:AgreementInitiator>RandomClient</wsag:AgreementInitiator>
    <wsag:AgreementResponder>provider03</wsag:AgreementResponder>
    <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
    <wsag:TemplateId>template04</wsag:TemplateId>
    <sla:Service>service01</sla:Service>
  </wsag:Context>
  <wsag:Terms>
    <wsag:All>
      <wsag:ServiceProperties wsag:Name="NonFunctional"
wsag:ServiceName="ServiceName">
        <wsag:Variables>
          <wsag:Variable wsag:Name="ResponseTime"
wsag:Metric="xs:double">
            <wsag:Location>qos:ResponseTime</wsag:Location>
            </wsag:Variable>
          </wsag:Variables>
        </wsag:ServiceProperties>
        <wsag:GuaranteeTerm wsag:Name="GTResponseTime">
          <wsag:ServiceScope wsag:ServiceName="ServiceName" />
          <wsag:ServiceLevelObjective>
            <wsag:KPITarget>
```

```

    <wsag:KPIName>ResponseTime</wsag:KPIName>

    <wsag:CustomServiceLevel>{"constraint" : "ResponseTime LT
100"}</wsag:CustomServiceLevel>
                                </wsag:KPITarget>
                                </wsag:ServiceLevelObjective>
                                </wsag:GuaranteeTerm>
                                </wsag:All>
                                </wsag:Terms>
</wsag:Agreement>

```

An example of agreement in JSON is:

```

{
  "agreementId":"agreement07",
  "name":"ExampleAgreement",
  "context":{
    "agreementInitiator":"client-prueba",
    "expirationTime":"2014-03-07T12:00:00+0100",
    "templateId":"template02",
    "service":"service5",
    "serviceProvider":"AgreementResponder",
    "agreementResponder":"provider03"
  },
  "terms": {
    "allTerms":{
      "serviceDescriptionTerm":null,
      "serviceProperties":[
        {
          "name":"ServiceProperties",
          "serviceName":"ServiceName",
          "variableSet":{
            "variables":[
              {
                "name":"metric1","metric":"xs:double","location":"metric1"},
              {
                "name":"metric2","metric":"xs:double","location":"metric2"},
              {
                "name":"metric3","metric":"xs:double","location":"metric3"},
              {
                "name":"metric4","metric":"xs:double","location":"metric4"}
            ]
          }
        }
      ],
      "guaranteeTerms":[
        {
          "name":"GTMetric1",
          "serviceScope":{"serviceName":"ServiceName","value":""},
          "serviceLevelObjective":{
            "kpitarget":{
              "kpiName":"metric1",
              "customServiceLevel":{"constraint\" :
\"metric1 BETWEEN (0.05, 1)\"}
            }
          }
        }
      ],{

```


POST /agreements

Create a new agreement. The body might include a AgreementId or not. In case of not being included, a uuid will be assigned. A disabled enforcement job is automatically created.

Error message:

- 409 is returned when the uuid already exists in the database
- 409 is returned when the provider uuid specified in the agreement doesn't exist in the database
- 409 is returned when the template uuid specified in the agreement doesn't exist in the database
- 500 when incorrect data has been supplied.

DELETE /agreements/{AgreementId}

Removes the agreement identified by AgreementId.

Error message:

- 404 is returned when the uuid doesn't exist in the database

GET /agreements/active

Return the list of active agreements.

GET /agreements/{AgreementId}/context

Only the context from the agreement identified by AgreementId is returned.

Error message:

- 404 is returned when the uuid doesn't exist in the database
- 500 when the data agreement was recorded incorrectly and the data cannot be supplied

Request in XML

```
GET -H "Accept: application/xml" /agreements/{agreement-id}/context HTTP/1.1
```

Request in JSON

```
GET -H "Accept: application/json" /agreements/{agreement-id}/context HTTP/1.1
```

Response in XML

```
HTTP/1.1 200 OK
```

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wsag:Context xmlns:sla="http://sla.atos.eu" xmlns:wsag="http://www.ggf.org/namespaces/ws-agreement">
  <wsag:AgreementInitiator>RandomClient</wsag:AgreementInitiator>
  <wsag:AgreementResponder>provider02</wsag:AgreementResponder>
  <wsag:ServiceProvider>AgreementResponder</wsag:ServiceProvider>
  <wsag:ExpirationTime>2014-03-07T12:00:00CET</wsag:ExpirationTime>
  <wsag:TemplateId>template02</wsag:TemplateId>
  <sla:Service>service02</sla:Service>
</wsag:Context>
```

Response in JSON

```
HTTP/1.1 200 OK
{"AgreementInitiator":"RandomClient",
"AgreementResponder":"provider02",
"ServiceProvider":"AgreementResponder",
"ExpirationTime":"2014-03-07T12:00:00CET",
"TemplateId":"template02",
"Service":"service02" }
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" http://localhost:8080/sla-service/agreements/agreement01/context
curl -H "Accept: application/json" http://localhost:8080/sla-service/agreements/agreement01/context
```

GET /agreements/{AgreementId}/guaranteestatus

Get the information of the status of the different Guarantee Terms of an agreement.

Error message:

- 404 is returned when the uuid doesn't exist in the database

Request in XML

```
GET -H "Accept: application/xml" /agreements/{agreement-id}/guaranteestatus
HTTP/1.1
```

Request in JSON

```
GET -H "Accept: application/json" /agreements/{agreement-id}/guaranteestatus
HTTP/1.1
```

Response in XML

```
HTTP/1.1 200 OK
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<guaranteestatus AgreementId="agreement02" value="FULFILLED">
  <guaranteetermstatus name="GTResponseTime" value="FULFILLED"/>
  <guaranteetermstatus name="GTPerformance" value="FULFILLED"/>
</guaranteestatus>
```

Response in JSON

```
HTTP/1.1 200 OK
{"AgreementId":"agreement02",
"guaranteestatus":"FULFILLED",
"guaranteeterms":
```

```
{
  {"name": "GTResponseTime", "status": "FULFILLED"},
  {"name": "GTPerformance", "status": "FULFILLED"}
}
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" http://localhost:8080/sla-service/agreements/agreement01/guaranteestatus
curl -H "Accept: application/json" http://localhost:8080/sla-service/agreements/agreement01/guaranteestatus
```

Enforcement Jobs

An enforcement job is the entity which starts the enforcement of the agreement guarantee terms. An agreement can be enforced only if an enforcement job, linked with it, has been previously created and started. An enforcement job is automatically created when an agreement is created, so there is no need to create one to start the enforcement.

- Enforcement jobs collection URI: /enforcements
- Enforcement job URI: /enforcements/{AgreementId}

An enforcement job is serialized in XML as:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<enforcement_job>
  <agreement_id>agreement02</agreement_id>
  <enabled>true</enabled>
  <last_executed>2014-08-13T10:01:01CEST</last_executed>
</enforcement_job>
```

An enforcement job is serialized in JSON as:

```
{"enabled":true,
 "agreement_id":"agreement02",
 "last_executed":"2014-08-13T10:01:01CEST"}
```

GET /enforcements/{AgreementId}

Retrieve an enforcement job identified by AgreementId.

Error message:

- 404 is returned when the uuid doesn't exist in the database

GET /enforcements

Retrieve the list of all enforcement jobs.

POST /enforcements

Creates an enforcement job. Not required anymore. The enforcement job is automatically generated when an agreement is created.

Error message:

- 409 is returned when an enforcement with that uuid already exists in the database
- 404 is returned when no agreement with uuid exists in the database

PUT /enforcements/{AgreementId}/start

Start an enforcement job.

Error message:

- 403 is returned when it was not possible to start the job

Request

```
PUT /enforcements/{agreement-id}/start HTTP/1.1
```

Response in XML and JSON

```
HTTP/1.1 200 Ok
```

```
Content-type: application/[xml | json]
```

```
The enforcement job with agreement-uuid {agreement-id} has started
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" -X PUT localhost:8080/sla-service/enforcements/fc923960-03fe-41/start
```

```
curl -H "Accept: application/json" -X PUT localhost:8080/sla-service/enforcements/fc923960-03fe-41/start
```

PUT /enforcements/{AgreementId}/stop

Stops an enforcement job

Error message:

- 403 is returned when it was not possible to start the job

Request

```
PUT /enforcements/{agreement-id}/stop HTTP/1.1
```

Response in XML and JSON

```
HTTP/1.1 200 OK
```

```
Content-type: application/[xml | json]
```

```
The enforcement job with agreement-uuid {agreement-id} has stopped
```

Usage (for JSON and XML)

```
curl -H "Accept: application/xml" -X PUT localhost:8080/sla-service/enforcements/fc923960-03fe-41/stop
```

```
curl -H "Accept: application/json" -X PUT localhost:8080/sla-service/enforcements/fc923960-03fe-41/stop
```

Violations

- Violations collection URI: /violations
- Violation URI: /violations/{uuid}

A violation is serialized in XML as:

```
<violation>
  <uuid>ce0e148f-dfac-4492-bb26-ad2e9a6965ec</uuid>
  <contract_uuid>agreement04</contract_uuid>
  <service_scope></service_scope>
  <metric_name>Performance</metric_name>
  <datetime>2014-08-13T10:01:01CEST</datetime>
  <actual_value>0.09555700123360344</actual_value>
</violation>
```

A violation is serialized in JSON as:

```
{"uuid":"e431d68b-86ac-4c72-a6db-939e949b6c1",
 "datetime":"2014-08-13T10:01:01CEST",
 "contract_uuid":"agreement07",
 "service_name":"ServiceName",
 "service_scope":"",
 "metric_name":"time",
 "actual_value":"0.021749629938806803"}
```

GET /violations/{uuid}

Retrieve information from a violation identified by the uuid.

GET /violations{?agreementId,guaranteeTerm,providerId,begin,end}

Parameters:

- *agreementId*: if specified, search the violations of the agreement with this agreementId,
- *guaranteeTerm*: if specified, search the violations of the guarantee term with this name (GuaranteeTerm[@name]),
- *providerId*: if specified, search the violations raised by this provider.
- *begin*: if specified, set a lower limit of date of violations to search,
- *end*: if specified, set an upper limit of date of violations to search.

Error message:

- 404 when erroneous data is provided in the call

Handbook

The handbook is associated the SLA Dashboard and shows how to manage the functionalities through the graphical user interfaces.

Before starting to use these functionalities, the SLA Dashboard should be started. Execute the following sentence in the Django project.

```
python manage.py runserver
```

Afterwards, you can access the component with this URL

```
<Host_Server>:<port>/slagui/login
```

Authorization for consumers and providers

Before users and providers register in the application, the SLA Dashboard has to be registered and configured in the Federated Identity Manager [93] (see the Installation guide section).

The following roles can interact with the dashboard:

- *Consumer*. This role indicates that the user, (previously identified), is the consumer in the agreement. Thus, the dashboard shows the associated agreements and their status.
- *ProviderIO*: This role indicates that the user, (previously identified), is the provider. Hence, the dashboard allows him to manage and see all the agreements for this provider.
- *Federator*: responsible to manage the application in the IdM and maintenance.

How to register a user in the SLA Dashboard.

Go to the Federated IdM and add the new user or organization and assign the appropriate role.

- If the user is a consumer assign the role Consumer.

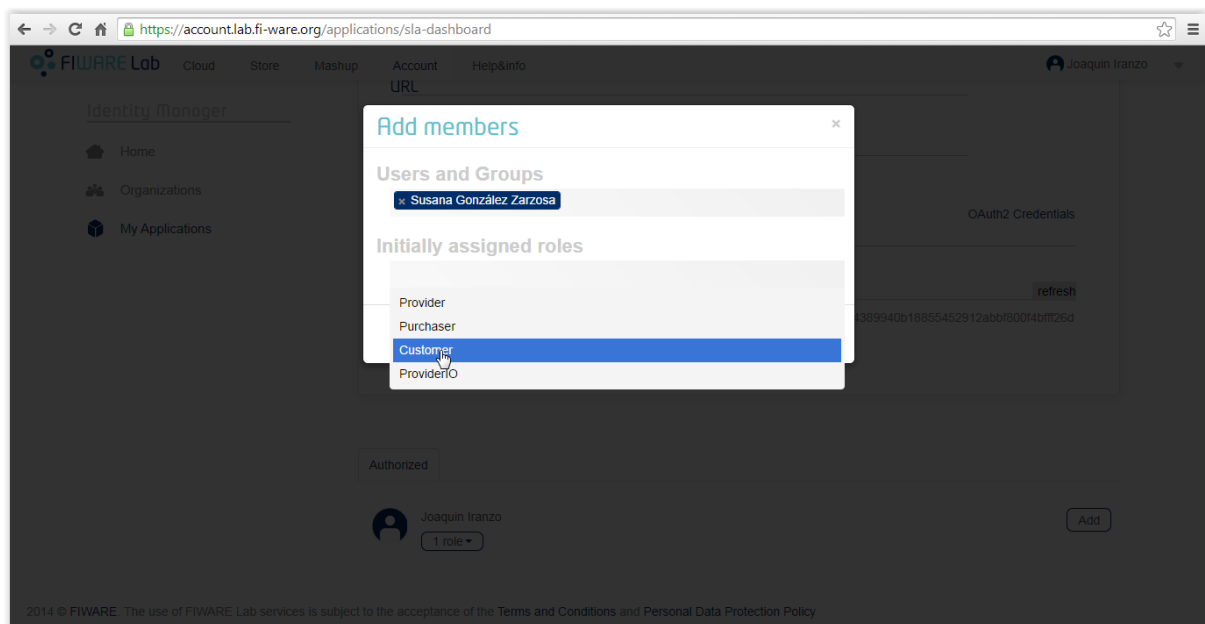


Figure 32: Add a consumer in the SLA Dashboard.

To identify an organization as provider, it should assign the roles ProviderIO and the Provider.

- ProviderIO role indicates that the organization can interact with the dashboard as a provider (manage Templates, see the agreement of his users)
- Provider role is associated with the IdM and not directly with the dashboard. The organizations, which have this role, can decide what users can enter in the dashboard on behalf of this organization.

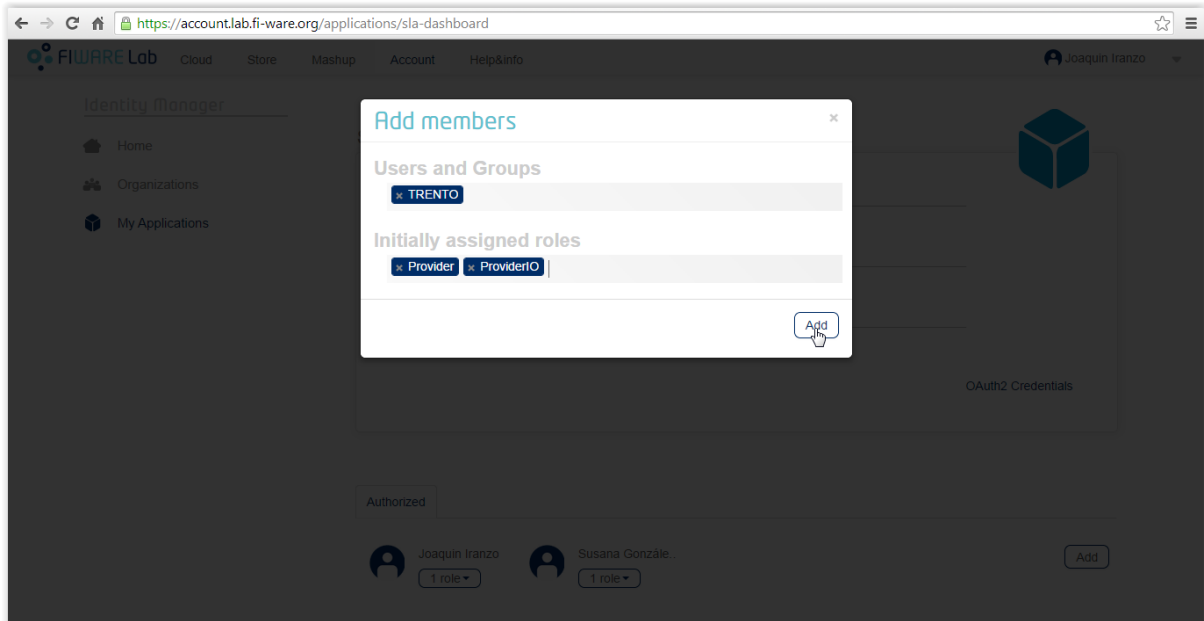


Figure 33: Add provider for the SLA Dashboard.

To see more details on how to interact with the IdM look up the Federation Identity Manager [93].

User authentication.

The user needs to be identified in the application in order to start to use it.

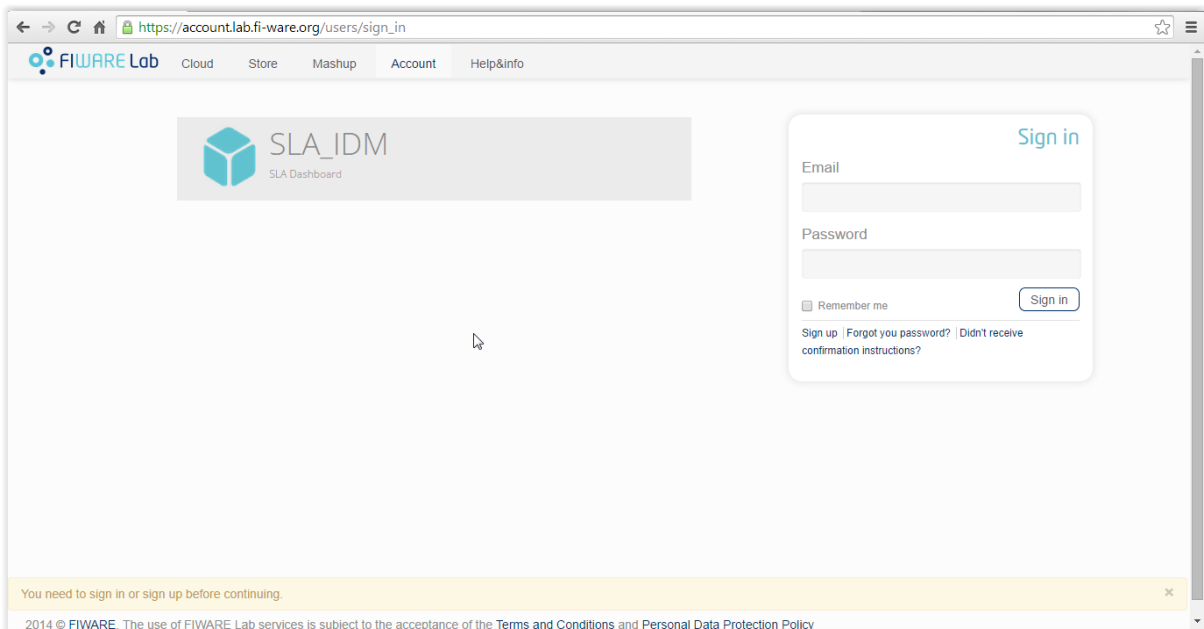


Figure 34: Login in the SLA Dashboard.

After the identification of the user, the application shows the dashboard depending of the assigned role.

Dashboard for Customers

The user can see his agreements and their status.

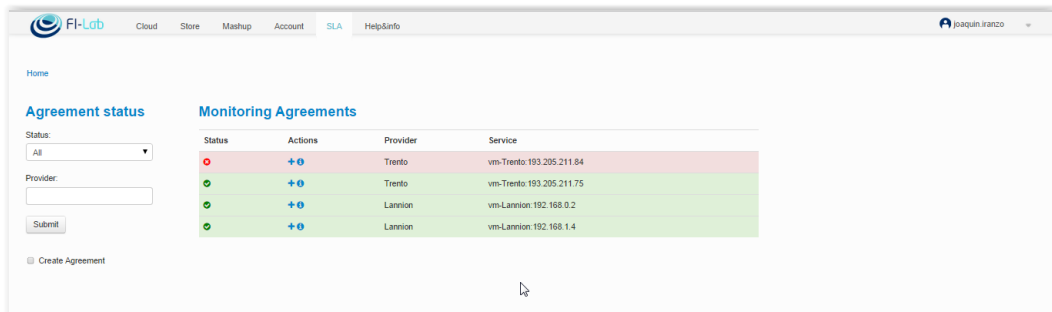


Figure 35: View of SLA Dashboard for customers.

- In the left panel, the customer can filter his agreements by the status and the providers.
- In the center panel, there are all his agreements with the summary and their status.
- In order to see the status of the agreement, the customer can click in icon

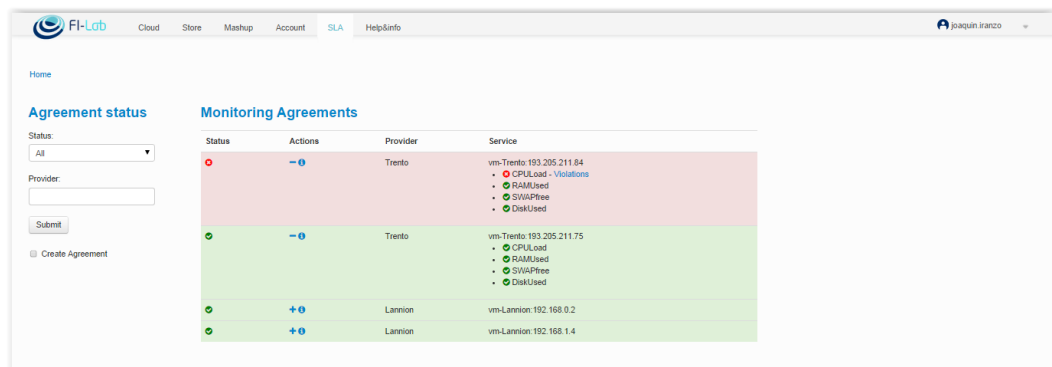


Figure 36: Status of the agreements.

- The consumer can see the associated metrics and whether these have been violated or not. If the metric has not been fulfilled, the customer can see the violations details; clicking in the "violations" link.

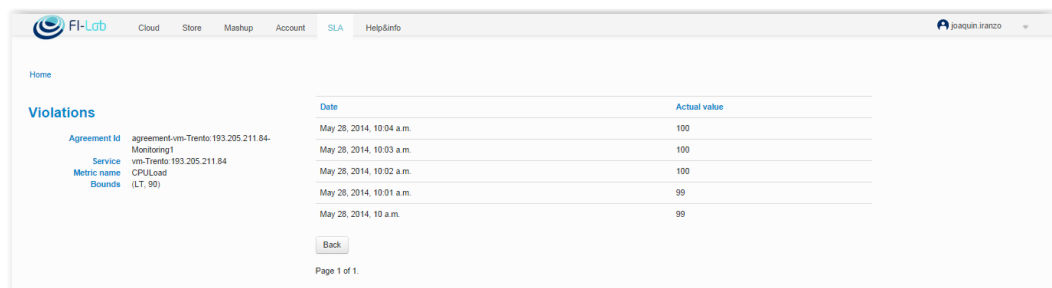


Figure 37: Violation details.

- The customer can click on the icon and see the agreement details, a summary of the guarantee terms and violations per date.

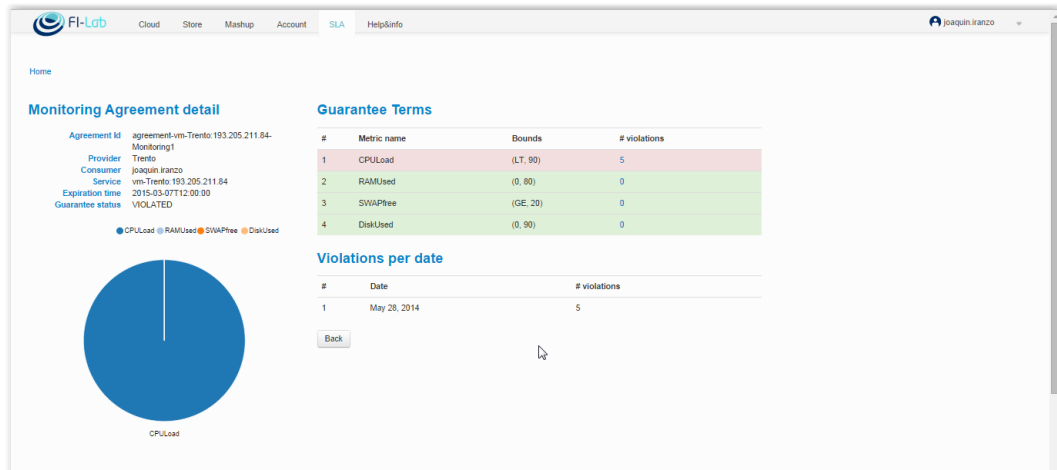
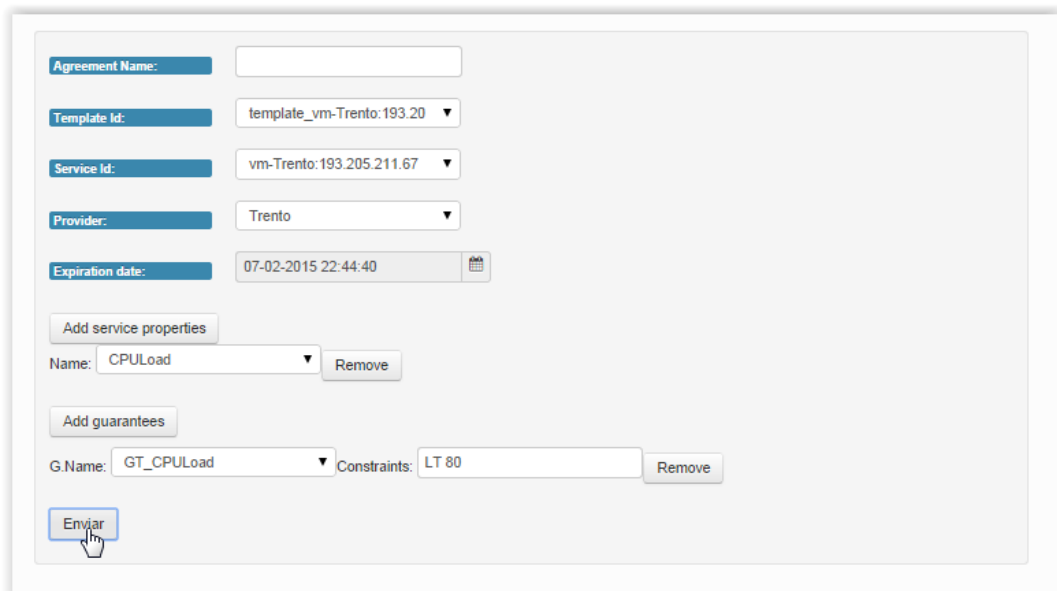


Figure 38: Info of the agreement.

- The customer can create the agreements clicking in the check box "Create Agreement":



Agreement Name:

Template Id:

Service Id:

Provider:

Expiration date:

Add service properties

Name: Remove

Add guarantees

G.Name: Constraints: Remove

Enviar

Figure 39: Create an agreement.

The customer has to fill in the following fields:

- Name of the agreement.
- Select one of the services in order to associate the new agreement.
- Select the available templates for this service.
- The assigned provider.
- Set an expiration date.
- Add the available metrics included in this template.
- Add the available Guarantee terms for this template.

Dashboard for Providers

There are two main pages in the dashboard for the providers.

- Main page (home menu)

The provider can see his agreements and the status of them.

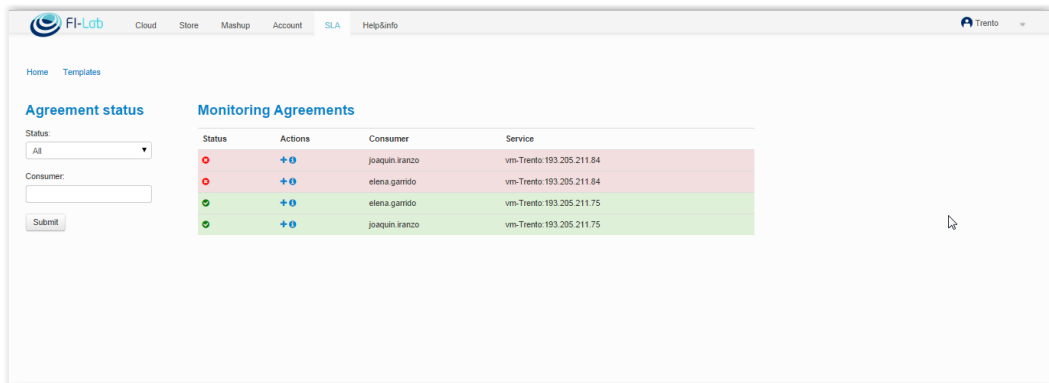


Figure 40: SLA Dashboard for providers.

- It is possible to search by status of the agreement and by associated customer.
- View the detailed summary of the agreement, clicking in the icon

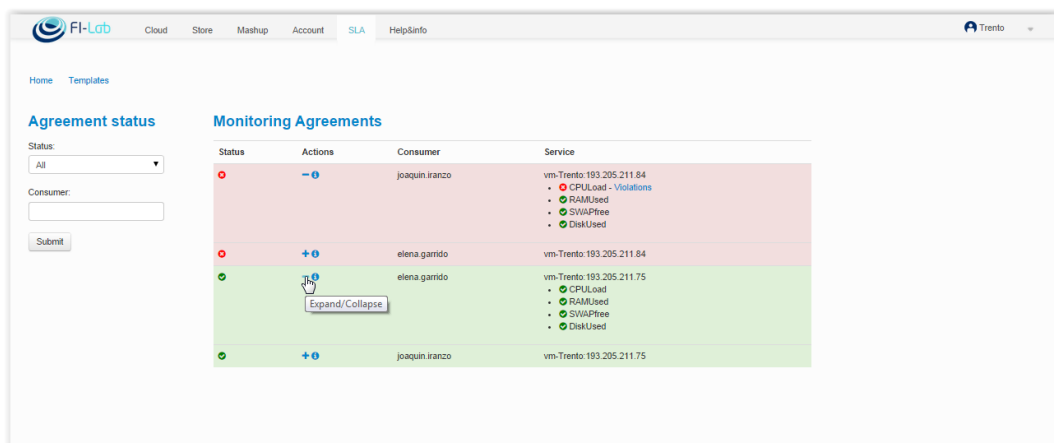


Figure 41: Status of agreements for providers.

- The provider can see the details of the violation, clicking in the “Violations” link.
- The provider can click in the and see the agreement details and a summary of the guarantee terms and the violations per date.

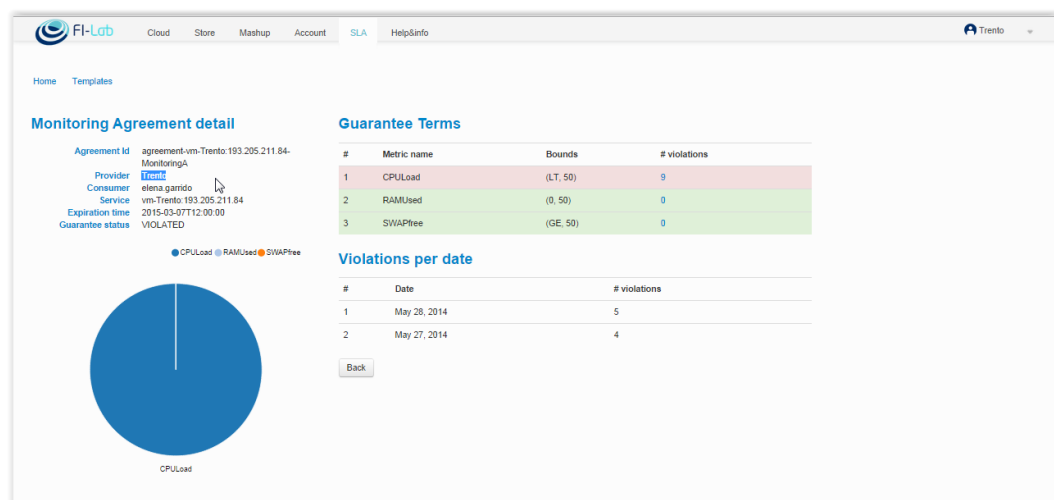
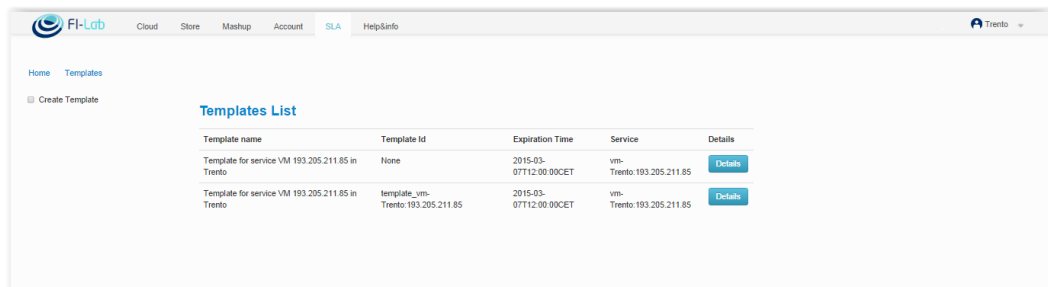


Figure 42: Info of agreements for providers.

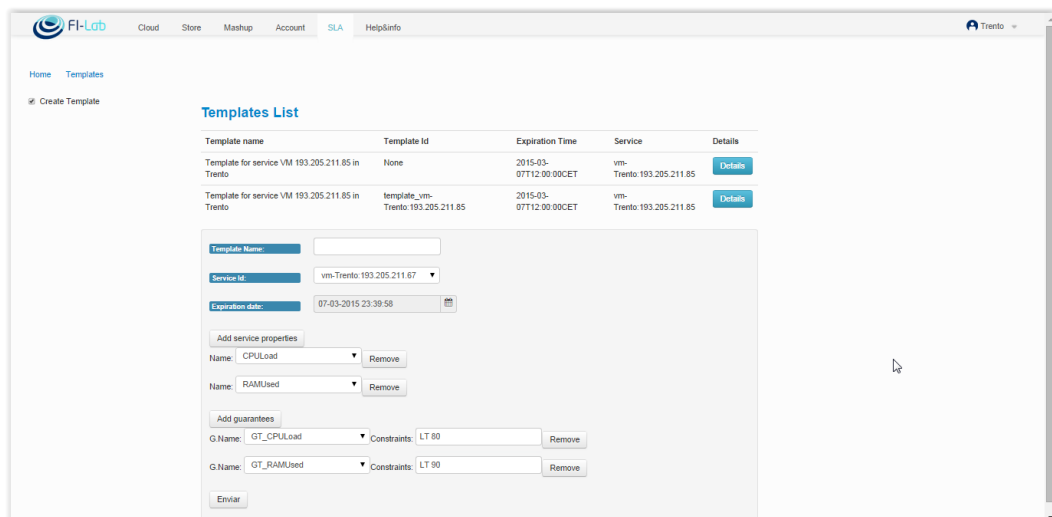
- Template page (Template menu)
The providers can manage their templates.
 - See the list of templates available for the provider and its details.



Template name	Template Id	Expiration Time	Service	Details
Template for service VM 193.205.211.85 in Trento	None	2015-03-07T12:00:00CET	vm-Trento:193.205.211.85	Details
Template for service VM 193.205.211.85 in Trento	template_vm-Trento:193.205.211.85	2015-03-07T12:00:00CET	vm-Trento:193.205.211.85	Details

Figure 43: List of templates for providers.

- Create a new template



Template Name:

Service ID:

Expiration date:

Add service properties

Name: CPUload

Name: RAMUsed

Add guarantees

G Name: GT_CPUload Constraints: LT 80

G Name: GT_RAMUsed Constraints: LT 90

Env/Var

Figure 44: Add template for providers.

The provider has to define:

- Name of the template
- The associated service
- Available metrics for this service that will be included in the template.
- Definition of the guaranties terms for these metrics.

4 INTEROPERABILITY TOOL

4.1 Summary

The Interoperability Tool is a software engineering tool that supports development and testing of FIWARE based applications and services; in particular the tool focuses on interoperability problems. The main users (i.e. most frequent) of the tool are Future Internet Developers [6]. The tool aids these stakeholders in two different ways:

- The developer can test whether their application interoperates with one or more GE instances deployed across XIFI (that is, conforms to the open specifications).
- During design and implementation of new applications and services, the developer can use the tool to observe and learn common usages of GEs; therefore, they can quickly build their applications based upon these established patterns of behaviour.

Infrastructure owners (see XIFI and stakeholders [6]) can also use the tool for specification compliance of developed and deployed services. That is, does a GE comply with a FIWARE open specification? The tool provides suites of executable patterns that can determine the extent to which a single GE can interoperate with applications and other GEs.

The following figure shows the detailed scheme of the XIFI Reference Architecture and the location of the Interoperability Tool (see yellow box). As shown, the tool has no direct dependencies and is leveraged by users (through the GUI portal) and other components via its REST interface.

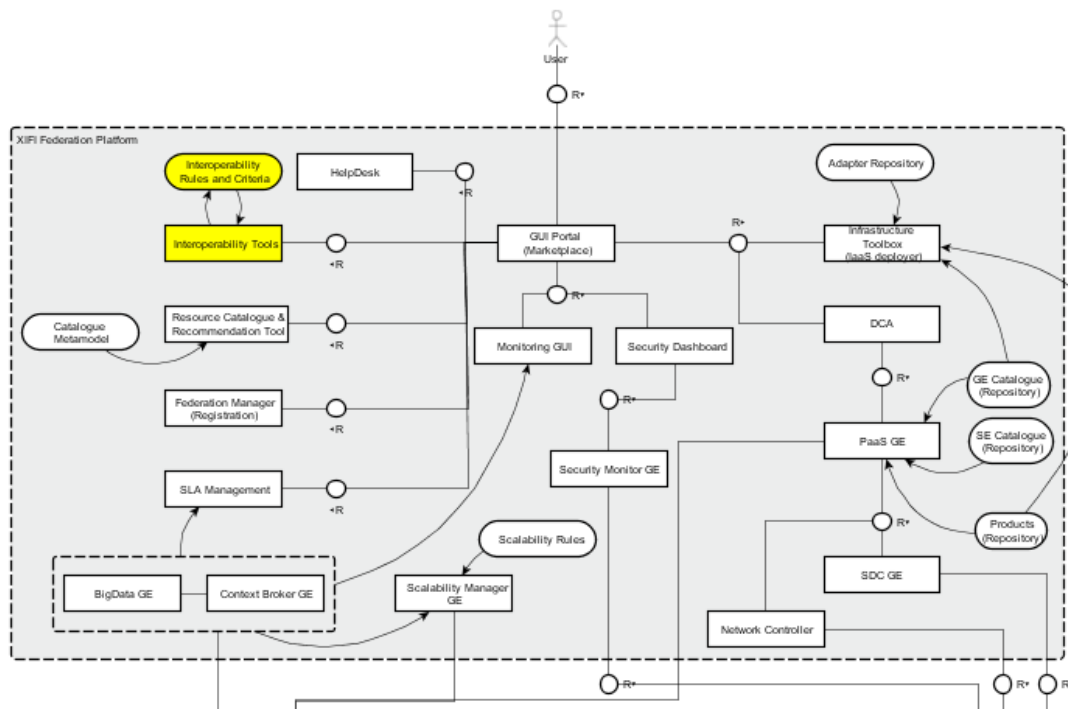


Figure 45: Location of the Interoperability Tool in the XIFI Reference Architecture

Reference Scenarios[3]	<ul style="list-style-type: none"> UC2 - Setup and use of development environment [8]
Reference Stakeholders[6]	<ul style="list-style-type: none"> Developers: those who want to test interoperability of their software. Infrastructure owners: those who want to join XIFI federation or build a private XIFI node.
Planned OS license	LGPL version 3.0[47]

Table 11: Summary - Interoperability tool.

Consists of

- *The Interoperability testing service*; based upon interoperability patterns, this monitors application and service execution and identifies interoperability successes and problems.
 - The service takes as input XML pattern descriptions of Future Internet applications and then during their execution it monitors the actual behaviour compared to the pattern specification.
 - The service generates an interoperability report describing the application behaviour trace and identification of interoperability issues (e.g. a parameter is missing, or incorrect data types and values have been used).
- *Interoperability patterns*; a set of pre defined pattern templates that highlight how developers can create their Future Internet applications. Patterns can be browsed, and a selected XML testing pattern can be leveraged to help implementation and testing.

Depends on

- No dependencies in Version 1
- Version 2 dependencies:
 - Federation IdM
 - Ensures only authentic users have access to the interoperability tool services
 - FIWARE Object Storage GE
 - Developers can save, manipulate, and re-use patterns from their own storage accounts.

4.2 Component leaders

Developer	Email	Company
Paul Grace	pjg@it-innovation.soton.ac.uk	IT-Innovation

Table 12: List of responsible - Interoperability tool.

4.3 Motivation

The XIFI federation brings together computational infrastructures (e.g. sensor networks, living labs, computational resources, networking resources, and storage resources) in order that they can be leveraged in combination by the developers of Future Internet technology. The central objective is to support industry stakeholders who require Future Internet resources to perform large-scale trials in order to evaluate and validate their technologies before they are transferred to market.

The key value proposition is that such heterogeneity and scale is not currently available for these types of trials, nor is it easily reproducible without significant costs. XIFI operates under a directive that FIWARE compliant software be used for the development of trials, where FIWARE is a collection of

software services (Generic Enablers) that provide re-usable functionality; trials may also develop their own specific services (Specific Enablers). FIWARE is a Service Oriented Architecture (SOA) and therefore each Enabler implements an open, interface specification (typically as a REST web interface). However, where such services are developed independently of one another there are significant challenges to achieve interoperability.

Developers must be able to check that the services and applications they develop interoperate with FIWARE services. Hence, the main objective of the XIFI Interoperability Tool is to support such checks via a web-based tool hosted in the XIFI portal. More specifically, this allows multiple XIFI stakeholders to perform tests regarding interoperability between enablers, and compliance of enablers with FIWARE specifications:

- *Developers*: use the Interoperability Tool during the system development and testing phase to assess how their software (e.g. a set of specific enablers) interoperates with existing patterns of Generic Enabler behaviour (e.g. cloud hosting, big data analysis, or authentication).
- *Infrastructure Owners*: use the tool to select and execute the patterns of enabler behaviour to verify that deployed GE behaviour complies with FIWARE open specifications.

4.4 User stories backlog

id	User story name	Actors	Description	Task id
1	Monitor pattern execution	Application developer	Tool presents a pattern; developer inputs their concrete enabler instances (either selected from XIFI catalogue or their own developments) and executes a sequence of behaviour. The interoperability service monitors behaviour. Hence the user can observe interoperability events.	1027
2	Interoperability test	Application developer	Tool presents a pattern; developer inputs their concrete enabler instances (either selected from XIFI catalogue or their own developments) and executes a sequence of behaviour. The Interoperability services evaluates the behaviour against interoperability rules and a report is generated and displayed to user.	1028
3	Re-use multiple patterns	Application developer	User selects from 10 available patterns to perform story ids 1 and 2. User modifies patterns to match their application behaviour.	1029
4	Compatibility test	Infrastructure Owner	Specifies enabler instance to be compared to a FIWARE specification. Tool reports where the implementation differs and is incompatible.	1030
5	Use multiple specifications	Enabler developer	User selects from 5 available specifications to perform story id 4.	1031

Table 13: Interoperability tool - User stories backlog

4.5 State of the art

Interoperability is a well-established requirement of distributed systems with a rich history of standardisation efforts (e.g. Web Services from W3C, CORBA services from the OMG, and Grid Services in the OGF and OASIS). Tanenbaum and Van Steen provide a well-accepted definition in their book Distributed Systems: Principles and Paradigms:

the extent by which two implementations of systems or components from different manufacturers can co-exist and work together by merely relying on each other's services as specified by a common standard [48].

The Interoperability problem typically cross-cuts many layers of a system architecture and this remains true of FIWARE. An important concern is that solutions typically focus only on a subset of these interoperability dimensions:

- **Data Interoperability.** Different systems choose to represent data in different ways, which manifests problems at two levels: i) syntax: where the data may employ different formats, e.g. XML versus JSON; and ii) semantics: even with the same format there is no guarantee that systems share the same understanding of data, e.g. the equivalence of data fields labelled "identity" and "id".
- **Middleware Interoperability.** Heterogeneous protocols employ a range of communication protocols for the discovery of other devices, to interact with other devices and to exchange data. The characteristics of the systems means that common standard protocols cannot be enforced, i.e. some are delay tolerant, power efficient, etc. Hence, it is not sufficient to say HTTP REST is the solution to the protocol interoperability problem in this domain.
- **Application Interoperability.** Design decisions made by developers introduce interoperability problems, e.g. differences in abstractions: publish-subscribe, streaming, request-response behaviour. The same functionality may be captured in one operation in one system versus multiple operations in another, e.g. (GetData) versus (GetTemperature & GetNoise).
- **Non-functional Interoperability.** Systems may have different security, performance and dependability requirements. Therefore, if functional interoperability is achieved the non-functional requirements may still be violated e.g. a system requiring message latency of 5ms interoperating with a system providing 10ms. FIWARE provides no support for this problem, and additional QoS and SLA solutions are required to be layered atop enabler configurations.

Where services are developed independently of one another, interoperability is a major concern. How can we ensure that services can exchange information and understand one another such that the required behaviour can be achieved? FIWARE poses new challenges for achieving interoperability between 3rd party developed software:

FIWARE is based upon the concept of open specifications. Analysis of the currently available specifications shows them to be informal REST interfaces (they use a mix of textual descriptions, architecture diagrams and sequence diagrams to describe the behaviour). Hence standard interoperability solutions based upon interface description languages may not always be feasible. FIWARE open specifications allow for pretty much any type of interface protocol: HTTP, SOAP, CORBA, DDS, AMQP; and any data format: XML, JSON, etc. Such heterogeneity is admirable and embraces the heterogeneity of the Future Internet; however, it raises significant interoperability challenges that will not be easily solved. N.b. current GE specifications are all HTTP interfaces. FIWARE lacks consistency in the machine process-able descriptions of services. Interface description Languages e.g. IDL, WSDL, and WADL, although flawed, offer a building block upon which interoperability solutions can be built. However, FIWARE does not support a single description language, nor does it direct that interfaces should be described using one. Further, description of behaviour logic sequences (i.e. how multiple operations are composed) are again left to text and/or sequence diagrams when there exist business/workflow description languages. Initial tool support seems to favour WADL (which is natural based upon REST specifications), hence it seems sensible to concentrate on web based languages (e.g. WADL) at this point.

This is clearly an important and interesting challenge for the XIFI federation for whom providing interoperability guarantees is a key service of a federation. However, it is clear that achieving the verification that systems can interoperate is beyond the resources available for the development of the interoperability tool (due to the observed complexity); therefore the tool is required to instead focus on particular interoperability problems (in this case the data, protocol and application dimensions that

match with the User Stories requirements) and help identify where systems can and do interoperate (which is a step towards allowing the future Internet developers to initially check their interoperability claims). Therefore, the tool will not prove that systems are interoperable (or not interoperable) but rather act as a guide for the development of software in the XIFI federation.

Outside of the FIWARE world there are a limited number of tools that have focused on interoperability testing in the realm of service architectures:

- *RESTAssured* is domain specific language and Java-based framework for the testing of REST implementations. It is able to check operation calls and data values (in both XML and JSON). However, it is a strictly client side testing tool and is only able to test if a client is sending and receiving the correct information. This can be used to test interoperability, but is not able to test the interoperability of co-ordinating distributed application components.
- *BPEL Testing frameworks*. There are a number of BPEL engines and frameworks that can be used to test Web Service compositions. However, these do not generally focus on identifying interoperability concerns. Furthermore, they generally target WSDL based solutions that are not appropriate for FIWARE enablers.

Hence, there is significant potential for the Interoperability tool to add to the state of the art in both the FIWARE community and in the larger field of Internet Services.

4.6 Architecture design

The Interoperability Tool follows a classic front-end, back-end architectural design. The client portion operates to collect user input regarding interoperability testing and produces the corresponding output. In order to provide a single point of access to the XIFI tools, the XIFI portal provides a common user interface (entry point and look and feel); hence, the Interoperability Tool integrates within the overall XIFI architectural design.

The back-end architecture consists of the service elements that perform interoperability testing, and store data related to pattern-based interoperability testing. These services are designed such that the architecture can be deployed and distributed across multiple hosts for scalability and reliability reasons. The deployment of a single instance of the tool may lead to performance bottlenecks (i.e. multiple pattern executions waiting to be scheduled), system outages and data inconsistency.

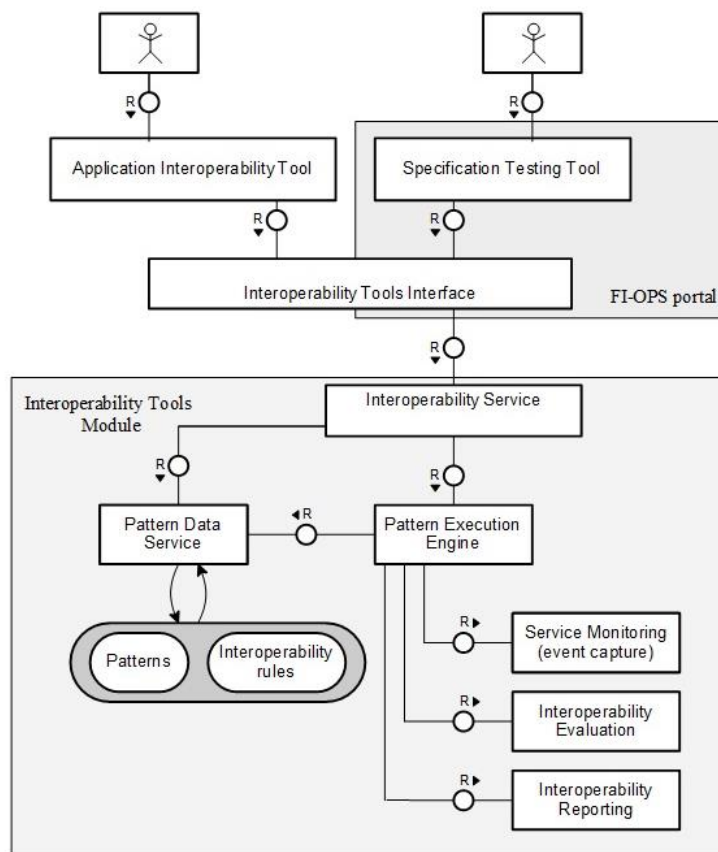


Figure 46: Interoperability tool - Architecture diagram.

The FMC compositional structure figure highlights the relationships between the individual components in the Interoperability Tool. A short functional description of each of these components is provided in the following table.

Architecture Element	Description
Application Interoperability Tool	Graphical user interface for the Future Internet developer to perform interoperability tests of their application against existing FIWARE GEs. The developer launches their own version of the interoperability tools module (using their own VM resources). The GUI can then be accessed using a web browser pointed at their deployment.
Specification Testing Tool	Graphical user interface for the Infrastructure Owner to perform compliance tests of their deployed GE behaviour. The Interoperability Tools module runs in the XIFI federation infrastructure (i.e. is deployed on the XIFI master node). A web browser GUI can then be used to execute compliance tests.
Interoperability Tools Interface	Front-end component representing the services provided by the Interoperability tool back-end implementation. Offers a single instance for the front-end to interact with i.e. it offers similar benefits to the façade pattern.
Interoperability Service	The Interoperability functions implemented as a REST service that can be interacted with using HTTP. Allows interoperability testing to be treated as a service, supporting both programmatic and interactive tools.
Pattern data service	Provides operations to: i) list current patterns, ii) view detailed information about the patterns, and iii) enter concrete information into a pattern (e.g the URL of an enabler). Also queries the resource catalogue module in order to extract

Architecture Element	Description
	information about generic and specific enablers and specific instances of these.
Patterns & Interoperability Rules	Data element describing a composition pattern and the interoperability rules that must be asserted for a valid execution of the pattern (used in the description of the pattern provided to the user).
Pattern Execution Engine	Execute an instance of a pattern of enablers; monitoring the execution to observe interoperability issues. Generates an interoperability report. Individual instances of patterns are implemented as separate components e.g. an authentication pattern. This performs the correct invocation of enablers (using correct protocols); it is the job of the execution to provide the correct input to this component and observe its outputs. Hence, new patterns can be plugged in as individual pattern implementations. For API compliance testing, the pattern injects messages to test responses of individual GEs.

Table 14: Interoperability tool – Architecture elements

Pattern Specifications

The architecture is underpinned by the concept of interoperability patterns. An interoperability pattern describes how an application must interact with other services and enablers; it is essentially a set of rules that must be followed. The Interoperability Service simply monitors a client-developed application to ensure that these rules are followed, reporting where a rule is broken. A Pattern consists of two parts:

- *The component architecture.* Which components interact with one another and via what interfaces. For example, a client interacts with a service via a REST Interface; a subscriber client interacts with the ORION context broker GE via the NSG19 REST API interface.
- *The behaviour sequence.* The sequence of valid messages that are exchanged between the prior defined interfaces. For example, the broker receives a subscription request from the client with the correct data content.

The Pattern Execution Engine understands the XML specifications of these patterns whose schema is as follows:

- **Pattern.xsd:** the pattern schema defines an element of two parts each described by their own schema.

```
<?xml version="1.0"?>
<!--Schema for an Interoperability pattern; there are two core parts: i) the
specification of an architecture, and ii) the state machine specification
of the interoperability test behaviour. -->

<xs:schema version="1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  xmlns:arch="http://PatternNamespace/archTypes"
  xmlns:behaviour="http://PatternNamespace/behaviourTypes">

  <xs:include schemaLocation="Architecture.xsd"/>
  <xs:include schemaLocation="Behaviour.xsd"/>

  <xs:element name="pattern">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="architecture"/>
        <xs:element ref="behaviour"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

</xs:sequence>
</xs:complexType>
</xs:element>

```

- **Architecture.xsd:** the schema describes the elements that compose the architecture.

```

<?xml version="1.0"?>
<!-- Schema for architecture part of the interoperability. Supports the specification of components and
interfaces that are part of the interoperability tests. -->

<xs:schema version="1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <!-- An architecture is a collection of two or more components -->
  <xs:element name="architecture">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="2" maxOccurs="unbounded" ref="component"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- description of individual architectural components -->
  <xs:element name="component">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="id"/>
        <xs:element ref="address" />
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="interface"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Unique Id of component/intf within this architecture; no other component
  can have the same identifier -->
  <xs:element name="id" type="xs:NCName"/>

  <!-- Physical address of the component. This can be either the IPv4 or
  IPv6 Address -->
  <xs:element name="address" type="xs:string"/>

  <!-- description of individual component interface -->
  <xs:element name="interface">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="id"/>
        <xs:element name="url" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- Physical address of the component. This can be either the IPv4 or
  IPv6 Address -->
  <xs:element name="url" type="xs:anyURI"/>
</xs:schema>

```

- **Behaviour.xsd:** the schema describes the elements that describe the behavior of an application.

```

<xs:schema version="1.0"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified">

  <!-- automaton describing the states and transitions of the protocol -->
  <xs:element name="behaviour">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="state"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- description of individual automaton states -->
  <xs:element name="state">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="label"/>
        <xs:element ref="type"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="transition"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- label of the start state -->
  <xs:element name="label" type="xs:NCName"/>

  <!-- label of the end state -->
  <xs:element name="type" type="xs:NCName"/>

  <xs:element name="transition">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="to"/>
        <xs:element ref="guards"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <!-- label of the start state -->
  <xs:element name="to" type="xs:NCName"/>

  <xs:element name="guards">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="equal"/>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="greater"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="equal">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="param" type="xs:string"/>
        <xs:element name="value" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

```

```

</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="greater">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="param" type="xs:string"/>
      <xs:element name="value" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

Essentially there are two parts: <architecture> and <behaviour> corresponding to the previous descriptions above. Further examples of their usage are provided in the user manual.

4.7 Release plan

Version Id	Milestone	User Stories
1.0	01/01/2014	1,2
2.0	01/10/2014	3,4,5

Table 15: Interoperability tool – Release plan

4.8 Test Cases

The following table lists a set of tests for the correct operation of a deployed instance of the Interoperability service after it has been installed.

Test id	Test description	Test script
1	Add a new pattern	see below
2	List patterns	see below
3	View pattern	see below
4	Execute simple interoperability test	see below

Table 16: Interoperability tool - User stories backlog

Each of the following is a simple command line test that interacts with the interoperability service using curl prompt commands. These tests assess the extent to which the basic operation of the service is maintained; it is not a low level software development unit testing process to test against where changes are made to the source code.

Add a new pattern

With the running tool service operating at `xxxx.xxxx.xxxx.xxxx:8192/interoperability` (replace `xxxx`), invoke a POST operation to add a pattern to the service to be stored in the pattern database.

```

$ curl --data "name=pattern1&description=Simple publish-subscribe pattern" --data-urlencode
"xml_content@src/main/resources/PublishSubscribe.xml" xxxx.xxxx.xxxx.xxxx:8192/interoperability/patterns

```

A HTTP 201 OK message will be returned if the test is successful; and the following message will be displayed:

```
> Pattern created and added to service.
```

List patterns

List the set of patterns currently installed on the running service. If a new instance with the prior test executed, a single element will be returned in the XML list.

```
$ curl xxxx.xxxx.xxxx.xxxx:8182/interoperability/patterns
```

The following content should be received:

```
<?xml version="1.0" encoding="UTF-8"?>
<patterns>
  <pattern>
    <name>pattern1</name>
    <description>A simple publish-subscribe pattern</description>
    <xml_content> .... {not fully displayed here} </xml_content>
  </pattern>
</patterns>
```

View pattern

This test will return the single XML content of the pattern:

```
$ curl xxxx.xxxx.xxxx.xxxx:8182/interoperability/patterns/pattern1
```

The following content should be received:

```
<pattern>
  <name>pattern1</name>
  <description>A simple publish-subscribe pattern</description>
  <xml_content> .... {not fully displayed here} </xml_content>
</pattern>
```

Execute simple interoperability test

```
$ curl -H "Content-type: application/xml" -d@pubsub.xml localhost:8182/interoperability/engine -X POST
```

In another terminal:

```
$ mvn unittest4
```

The following should be the output on the service terminal:

```
Initial State S1:
```

<p>Transition success -> FromIntf matches localhost -> ToIntf matches http://orion.fi-lab.eu:1026/NSGI-9 -> Message matches POST Data Rules: X-AUTH-TOKEN is present = true;</p> <p>State S2: Transition success -> FromIntf matches http://orion.fi-lab.eu:1026/NSGI-9 -> ToIntf matches localhost -> Message matches REPLY Data Rules: HTTP_code = 200;</p> <p>State S3: End state reached. Valid application interoperation</p>
--

4.9 Installation Manual

Introduction

This manual describes how to install the interoperability tool. The Interoperability tool can be deployed in two ways:

Web-based service. The tool is deployed as a remotely accessible service; a browser GUI is leveraged by the user in order to edit interoperability patterns and execute interoperability tests. This type of interaction will generally be used for early identification of interoperability problems and continued regression tests.

Stand alone client. The tool is individually deployed by a user, i.e. the tool use is localised for testing by the client. Typically this means that the tool will be executed either at the command line, or using an IDE. This style of interaction will largely be employed for unit and integration testing where an application is being developed to interoperate (there is a lower time overhead to run the tool).

In both cases, the installation procedure remains the same; the difference is in the execution. Hence, this document describes the common installation procedure first in the following section, and then describes the two deployment and execution procedures in the subsequent two sections.

Installation

The interoperability tool requires only three pieces of software:

- Java Development Kit (JDK 7 and later)
 - <http://www.oracle.com/technetwork/java/javase/overview/index.html>
- Maven (Version 2 and later)
 - <http://maven.apache.org/download.cgi>
- Subversion
 - <http://subversion.apache.org/>

We now describe the installation of each on Ubuntu v12.04. For other operating systems (e.g. Windows, Mac) please follow the individual software installations of each. No special configuration is required and hence the general installation instructions should work fine.

JDK 7

```
> sudo apt-get update
> sudo apt-get install openjdk-6-jre
```


Maven

```
> sudo apt-get update > sudo apt-get install maven
```

Subversion

```
> sudo apt-get update
> sudo apt-get install subversion
```

Interoperability Tool

1. Get the interoperability tool source code:

```
> svn co https://xifisvn.res.eng.it/wp4/InteroperabilityTool
```

2. Install the code

```
> mvn install
```

The interoperability tool is now ready to be executed. Nb. if there is a build failure this is down to incomplete unit tests rather than an error in the install. Carry on with next commands.

Command line execution

To run the tool with a particular pattern:

```
> mvn exec:java -Dexec.mainClass="uk.ac.soton.itinnovation.xifiinteroperability.InteroperabilityTool" -
Dexec.args="PublishSubscribe.xml"
```

Change the xml file to each new pattern.

Command line walkthrough

To run a simple test against a public GE

1. First start the interoperability tool with a pattern loaded:

```
> mvn exec:java -Dexec.mainClass="uk.ac.soton.itinnovation.xifiinteroperability.InteroperabilityTool" -
Dexec.args="PublishSubscribe.xml"
```

2. Run a java client application (that we are testing)

```
> mvn exec:java -
Dexec.mainClass="uk.ac.soton.itinnovation.xifiinteroperability.samples.NSGISubscriberOrchestrator"
```

Note that the Java code must be changed to point to the location of the Interoperability Tool proxy (this will be displayed on the screen). You may also need to change the token. This is an important step in the testing process. The client code must be aligned with the tool (and not the service GE)

```
String urlString = "http://192.9.206.71:80/NGSI10/queryContext";
String XML = "<?xml version=\"1.0\" encoding=\"UTF-8\"?>" +
"<queryContextRequest>" +
```

```

    "<entityIdList>" +
    "<entityId type=\"Room\" isPattern=\"true\">" +
    "<id>Room.*</id>" +
    "</entityId>" +
    "</entityIdList>" +
    "<attributeList>" +
    "<attribute>temperature</attribute>" +
    "</attributeList>" +
    "</queryContextRequest>";
try {
    Client client = Client.create();
    WebResource webResource = client.resource(urlString);
    ClientResponse response = webResource.type("application/xml").header("X-Auth-Token",
    "PSzRfoMrfAr5GswEKDCmg4pKSCjTbHrvru5CxklBZoRISwK58hCFOnVhwsCpF3i1suwsR1tiDnO7
    _2OMP0I1Sw").post(ClientResponse.class, XML);
    if (response.getStatus() != 200) {
        System.err.println(response.getEntity(String.class));
        throw new RuntimeException("Failed : HTTP error code : " + response.getStatus());
    }
    ... further code ...
    System.out.println("Output from Server .... \n");
    String output = response.getEntity(String.class);
    System.out.println(output);
}

```

Running as a browser service

The tool can be accessed as a web service and using a browser as the client. There are two methods of deploying the service: 1) As an OAUTH protected service - integrated with the FIWARE IDM GE i.e. only FIWARE users can use the tool. This version has user specific patterns 2) As a local non-protected service - you deploy your own web service and anyone with the link can access the tool. This version has instance specific patterns (i.e. there is no concept of multiple users).

The differences for these two are handled by editing the properties file:

```
> pico Interoperability.properties
```

There are two settings.

- If you want to run in mode 1, change the following properties as follows.
FILAB_IDM=https://account.lab.FIWARE.org:443 oauth_protected=true
- If you want to run in mode 2, simply change the oauth_protected property as follow:
oauth_protected=false

Then install and run the tool

```
> mvn install
```

```
>./run
```

Now point a browser at the following url (where xx.xx.xx.xx is the public url of your host) and the client GUI should appear as follows:

http://xx.xx.xx.xx:8192/interoperability

If you want to change the port - change the DEFAULT_PORT property in the properties file.

4.10 User Manual

User Guide

This user manual is targeted at the developers of services and applications built upon FIWARE generic enablers. At present, this is a simple guide to performing simple interoperability tests using the tool. For this, we use a running example: the testing of a simple publish-subscribe client interoperating with the FIWARE Orion Context broker GE.

The first step to launch an instance of the tool using a web browser. Simply use the URL of an instance of the tool (if the tool has been installed locally then the following URL is appropriate):

- *http://xxx.xxx.xxx.xxx/interoperability/patterns*

The following screen should be displayed:

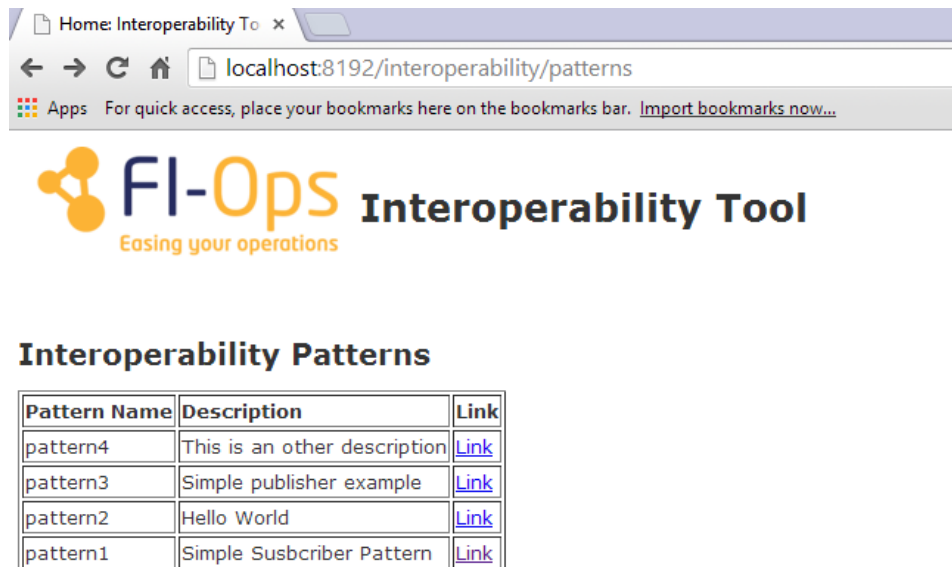
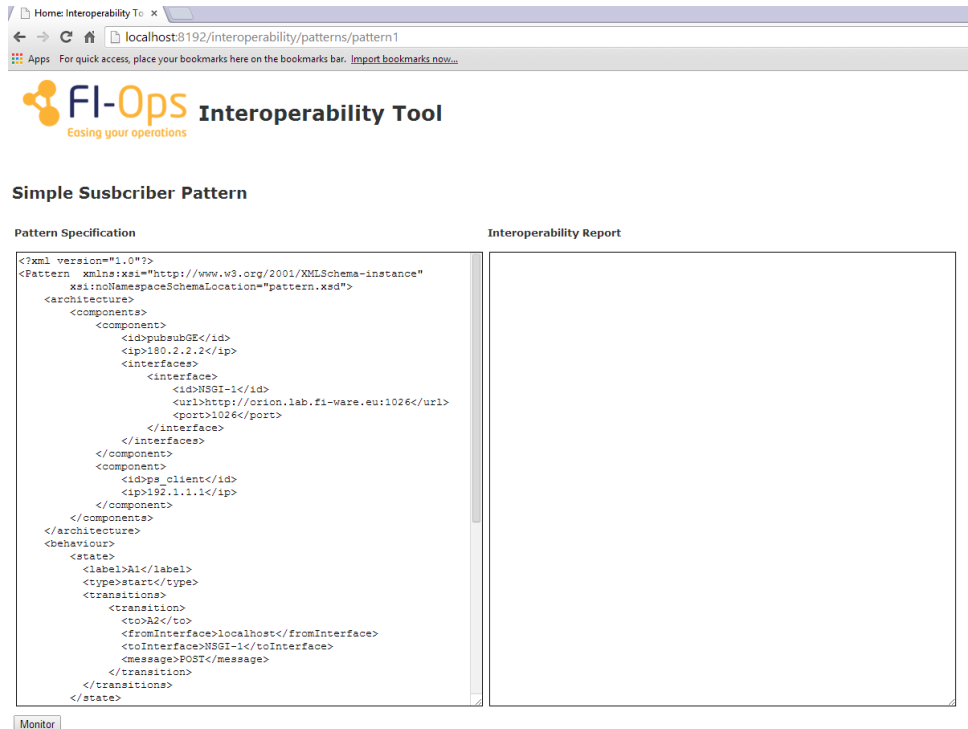


Figure 47: Display available patterns in the Interoperability Tool

Now select the publish-subscribe pattern (i.e. pattern1 link). The following screen will be displayed to you:



The screenshot shows the FI-Ops Interoperability Tool interface. The title bar indicates the browser is at `localhost:8192/interoperability/patterns/pattern1`. The main content area is titled "Simple Subscriber Pattern" and is divided into two panes:

- Pattern Specification:** Contains an XML document defining a subscriber pattern. The XML includes components for a service (`pubSubGE`) and a client (`client`), with their respective IP addresses and REST interfaces.
- Interoperability Report:** Currently empty.

Below the XML editor is a "Monitor" button.

Figure 48: Display subscriber interoperability test pattern

You can now edit the pattern via the XML document in the left text box. For example, you can edit the URL of the context broker GE. If you wish to use the FIWARE Lab instance then no changes need to be made. To test a different enabler instance, edit the URL and port tag value in the following XML content:

```
<source lang="xml">
<?xml version="1.0"?>
<Pattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="pattern.xsd">
  <architecture>
    <components>
      -- description of a service or client application
    <component>
      <id> </id> -- unique id within this pattern
      <ip> </ip> -- The IP address where the component is running
      <interfaces>
        <interface> -- The list of REST interfaces provided
          <id> </id> -- unique id within this pattern
          <url></url> -- full url of the REST interface including port number
        </interface>
      </interfaces>
    </components>
  </architecture>
  <behaviour>
    <state>
      <label>A1</label>
      <type>start</type>
      <transitions>
        <transition>
          <to>A2</to>
          <fromInterface>localhost</fromInterface>
          <toInterface>NSGI-1</toInterface>
          <message>POST</message>
        </transition>
      </transitions>
    </state>
  </behaviour>
</Pattern>
```

```

</component>
<component> -- Component with no provided interfaces i.e. a client
  <id> </id>
  <ip> </ip>
</component>
</components>
</architecture>
<behaviour>
  <state> -- State machine for the test
    <label></label>
    <type></type> -- If this is a start, normal or end node in state machine
    <transitions> -- a transition matches a REST event: request or reply
      <transition>
        <to></to> -- State to move to in machine
        <fromInterface></fromInterface> -- from rule: msg from correct host
        <toInterface></toInterface> -- to rule: to correct interface instance
        <message> </message> -- message type rule: POST, GET, etc.
        <datarules> -- set of rules checking data types and content of event
          <datarule> </datarule>
        </datarules>
      </transition>
    </transitions>
  </state>
</behaviour>
</Pattern>
</source>
<source lang="xml">
<interfaces>
  <interface>
    <id>NSGI-1</id>
    <url>http://orion.lab.FIWARE.eu</url>
    <port>1026</port>
  </interface>
</interfaces>
</source>

```

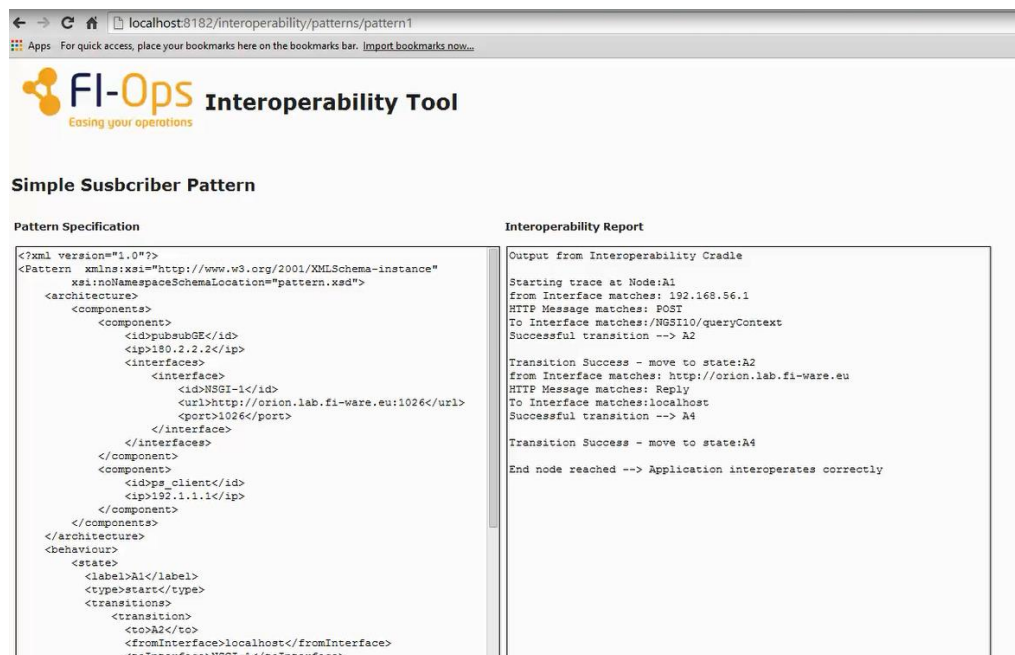
To begin the test, click the Monitor button. This will set up the tool to do interoperability monitoring. The URL of the tool will be displayed in the text box to the right. This must be used by your client, i.e., all of the client interactions must be with the tool rather than the actual enabler instance. For example, if your client calls *orion.lab.FIWARE.eu* interface directly then no interoperability testing

will be performed.

You are now ready to execute your client subscriber application (remember, this should be directed at the monitoring URL posted). There is a test client in the source distribution; this can be executed using:

```
> mvn unittest4
```

The interoperability report will be displayed as in the screenshot below.



The screenshot shows the FI-Ops Interoperability Tool interface. The title is "Simple Subscriber Pattern". It is divided into two main sections: "Pattern Specification" and "Interoperability Report".

Pattern Specification:

```
<?xml version="1.0"?>
<Pattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="pattern.xsd">
  <architecture>
    <components>
      <component>
        <id>pubsubGE</id>
        <ip>190.2.2.2</ip>
      </component>
      <interface>
        <id>NSGI-1</id>
        <url>http://orion.lab.fi-ware.eu:1026</url>
        <port>1026</port>
      </interface>
    </components>
  </architecture>
  <behaviour>
    <state>
      <label>A1</label>
      <type>start</type>
      <transitions>
        <transition>
          <to>A2</to>
          <fromInterface>localhost</fromInterface>
          <toInterface>NSGI-1</toInterface>
        </transition>
      </transitions>
    </state>
  </behaviour>
</Pattern>
```

Interoperability Report:

```
Output from Interoperability Cradle
Starting trace at Node:A1
From Interface matches: 192.168.56.1
HTTP Message matches: POST
To Interface matches:/NSGI10/queryContext
Successful transition --> A2

Transition Success - move to state:A2
From Interface matches: http://orion.lab.fi-ware.eu
HTTP Message matches: Reply
To Interface matches:localhost
Successful transition --> A4

Transition Success - move to state:A4
End node reached --> Application interoperates correctly
```

Figure 49: Result of interoperability testing and monitoring

User Guide: Writing Interoperability Patterns

The interoperability tool uses patterns of behaviour that we term Interoperability Patterns in order to carry out interoperability tests. That is, the pattern describes the expected behaviour required to demonstrate interoperability; this information is passed to the tool which monitors the behaviour of the real system and compares it against the pattern.

The purposes of these tests are twofold:

- Testing that component A interoperates with component B, C, D, etc. That is, when an application developer has created their new software (component A) they want to test whether it interacts with the other components in a compliant manner. This typically involves testing that the right messages are being exchanged and these are in the correct order.
- Testing that component Z complies with interface specification A. That is, where you are developing software that must conform to a specification e.g. a new REST service that implements the NSGI-10 interface. This typically involves firing test messages to the developed service to check the responses are compliant.

This document describes how to write an interoperability pattern for both of these cases. In section 1 we introduce the general concepts of an architectural representation of a distributed system, and then a state machine approach to describe the states within this system. In section 2, application testing

patterns are covered, and then service compliance patterns are presented in section 3.

Representing distributed systems: the state machine approach

Interoperability is about ensuring that: i) system A can exchange data with system B, and ii) that they understand one another such that the required operations are carried out. Compliance is about ensuring that an implementation of an interface or specification conforms to the required behaviour. Hence, the two are closely related, but achieving compliance in no way guarantees two or more systems will interoperate. Compliance, however, can be an important step towards achieving interoperability.

A distributed system is a set of systems that are developed independently from one another and rely on agreed interfaces to interoperate. In terms of the Interoperability Tool we are focusing on RESTful distributed systems where the individual systems are:

- Services implementing a REST interface, i.e. one that is accessed using REST HTTP messages.
- Client applications (and other services) that invoke REST operations from the previously described REST services.

Therefore, when we come to model such a distributed system we can focus on two parts:

- The architecture of the distributed system. This is the composition of the services and clients i.e. the elements that are communicating with one another. In classical architecture description these can be considered as components who either provide interfaces or require interfaces.
- The messages that are exchanged between systems. The overall behaviour of the system is driven by the REST messages that are exchanged between the components. Hence, to model the behaviour of the system we must model the effect of message exchange.

Given these two modelling requirements, we can create distributed systems models that can be leveraged to perform interoperability testing.

An interoperability pattern is an XML specification that captures both of these elements. The XML below shows the outline structure of every pattern. As it can be seen, there are two parts:

- The architecture described by the <architecture> tag
- The behaviour described by the <behaviour> tag

```
<pattern xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Pattern.xsd">
  <architecture>
  ...
  </architecture>
  <behaviour>
  ...
  </behaviour>
</pattern>
```

Architecture Description

As previously stated, we want the architecture to capture information about the components in the system. Using a simple client-server example illustrated below, we model the information about the

two components and the interface between them, i.e. the client is calling the REST API of the server. We write this information in the <architecture> tag.

```

<architecture>
  <component>
    <id>pubsubGE</id>
    <address>orion.lab.FIWARE.eu</address>
    <interface>
      <id>NSGI10</id>
      <url>http://orion.lab.FIWARE.eu:1026/NGSI10</url>
    </interface>
  </component>
  <component>
    <id>ps_client</id>
    <address>192.9.206.71</address>
  </component>
</architecture>

```

The architecture tag is always made up of 1 or more <component> elements (that correspond to a component in the distributed system). In the example there are two components a client and a server; in this case the server is a publish/subscribe broker and the client is sending a request/response query for information about a particular context.

We first describe the server. This is simply an <address> to identify the service using a public name e.g. an IP address or a domain qualified name. The <id> tag is a local identifier in the pattern (choose your own local id here). The <interface> tag (of which there can be many) describes the URL of the interface of the service used in this application; note, the <id> tag is simply an identifier local to the pattern so that it can be reused multiple times in the pattern (see later) - you can choose your own ID here.

The second <component> is the client. As this is a simple case, the only information we need to know is the identifiable location of the client so we use the <address> tag to identify the IP address of the client. Note, a <component> can be both a client and a service - hence the common <component> tag here.

Behaviour Description

We use state machines to describe the behaviour of the system. There are two overriding elements of the state machine:

- *A state.* This is a state of the overall distributed system (not a state of one of the components. It describes a point in time during the sequence of messages exchanged between components. For example, in the first state no messages have been exchanged between the components.
- *A transition.* This is the change from one state to another based upon the correct exchange of a message. For example, in a client server exchange; the correct sending of a request from the client to the server can be considered as a transition from one state to the next.

This is illustrated in the example diagram which shows two transitions (we will discuss rules later) - the first is a HTTP Post request followed by a reply message.

Therefore, we model the behaviour as a set of <state> elements in the XML pattern as follows:


```

<state>
  <label>A1</label>
  <type>start</type>
  <transition>
    <to>A2</to>
    <guards>
      <equal>
        <param>HTTP.from</param>
        <value>component.ps_client.address</value>
      </equal>
      <equal>
        <param>HTTP.to</param>
        <value>component.pubsubGE.address</value>
      </equal>
      <equal>
        <param>HTTP.toInterface</param>
        <value>component.pubsubGE.NSGI10</value>
      </equal>
      <equal>
        <param>HTTP.msg</param>
        <value>POST</value>
      </equal>
      <contains>
        <param>HTTP.Headers</param>
        <value>HTTP.X-auth-token</value>
      </contains>
    </guards>
  </transition>
</state>

```

Each state must have: A <label> tag. To name the state (this is a local name chosen by pattern writer)

A <type> tag. This must be one of five types:

- Start - the first state in the pattern. There can be one and only one. State 1 in the figure is the start state.
- End - a finishing state in the pattern. There must be one or more. State 3 is an end state in the figure.
- Normal - an intermediary state that listens for events and applies rules to the incoming events to determine which of its transitions to take. Normal nodes have <guards> based transitions. State 2 in the figure is a normal state.

- Trigger - this is a state that immediately sends a REST message out (as part of the test) i.e. it constructs a HTTP message described by the transition and executes it. It must have one and only one <message> transition.
- TriggerStart - this is for the start node that is a trigger state also. There can be one and only one. Trigger examples are provided later in Section 3.

The <transition> tag. There can be multiple transitions e.g. where a different observed event determines the trace through the state machine. A transition contains the following information A <to> tag. To identify the state where this transition should move to in this state pattern provided the guards are met. Where a transition is a trigger transition; the content is a <message> tag. This is executed instantly by converting the message data into an http message and sending it (i.e. the state machine also injects events into the distributed system to aid with testing). Trigger examples are provided later in Section 3. Where a transition is a normal transition; the content is a set of <guards> which are rules to test the content of an observed event. Only where all guards are true will the transition be executed. It can be seen that the XML guards match the rules in the figure.

Writing Rules

A rule is a test on the content of a REST event to determine if it matches expected behaviour. There are two levels of interoperability testing: Protocol testing. Does the REST message conform to the correct message structure i.e. does it have the required elements in terms of operation type (GET, PUT, etc.), does it have the correct header values, is it being sent correctly between the elements.

If we are doing higher level application interoperability testing we need to be aware if the data conforms: is it the correct type? (JSON/XML), does it have the correct content in terms of application values etc.

<guards> are composed of function tests. At present there are <equals>, <notequals> and <contains> guards. Equals tests that the field described in <param> matches the value given in the <value> tag. Not equals does the above but ensures it doesn't equal that value. Contains checks that the set of fields in <param> contains a field named by the value in <value>

Protocol testing uses HTTP specific rules:

- HTTP.from checks where the event originated. We use component.ps_client.address to resolve to the ip address listed in the <architecture> section of the pattern (although an IP address could be listed here).

```
<equal>
  <param>HTTP.from</param>
  <value>component.ps_client.address</value>
</equal>
```

- HTTP.to checks where the event was sent to. We use component.pubsubGE.NSGI10 to resolve to the ip address listed in the <architecture> section of the pattern (although an IP address could be listed here).

```
<equal>
  <param>HTTP.to</param>
  <value>component.pubsubGE.address</value>
</equal>
```

- HTTP.toInterface checks the interface that an event was sent to. We use component.pubsubGE.NSGI10 <id>to resolve to the fully qualified URL listed in the <architecture> section of the pattern (although a URL could be listed here).

```

<equal>
  <param>HTTP.toInterface</param>
  <value>component.pubsubGE.NSGI10</value>
</equal>

```

- HTTP.msg check the message type, which must be either POST, GET, DELETE or PUT here.

```

<equal>
  <param>HTTP.msg</param>
  <value>POST</value>
</equal>

```

- HTTP. Headers are the list of header fields extracted from an event e.g. host, accept, etc. We can perform a guard test on these e.g. that is contains a token field (as seen in the figure).

```

<contains>
  <param>HTTP.Headers</param>
  <value>HTTP.X-auth-token</value>
</contains>

```

Writing a pattern to test a new application

We have developed a new application that uses REST services, and we wish to test that it interoperates correctly with these services. This section describes how to write the pattern using a simple Request-Response (hello world style) example that was introduced in section 1.

The client performs a single REST request message a queryContextRequest operation on the NSGI-10 REST service interface.

- Step 1: Write the architecture description Define the REST service e.g. the ORION context broker.

```

<component>
  <id>pubsubGE</id>
  <address>orion.lab.FIWARE.eu</address>
  <interface>
    <id>NSGI10</id>
    <url>http://orion.lab.FIWARE.eu:1026/NGSI10</url>
  </interface>
</component>

```

Define the client.

```

<component>
  <id>ps_client</id>
  <address>192.9.206.71</address>
</component>

```

- Step 2: Model the messages and interoperability Rules

We are only observing events between the two real systems described above so we only need to consider <guards> based transitions.

The first transition is to test the message sent from the client to the service i.e. is it the correct HTTP message containing the queryContextRequest request.

We define two states A1 and A2. A1 has guard transitions as previously stated.

```

<state>
  <label>A1</label>
  <type>start</type>
  <transition>
    <to>A2</to>
    <guards>
      <equal>
        <param>HTTP.from</param>
        <value>component.ps_client.address</value>
      </equal>
      <equal>
        <param>HTTP.to</param>
        <value>component.pubsubGE.address</value>
      </equal>
      <equal>
        <param>HTTP.toInterface</param>
        <value>component.pubsubGE.NSGI10</value>
      </equal>
      <equal>
        <param>HTTP.msg</param>
        <value>POST</value>
      </equal>
      <equal>
        <param>HTTP.Content-type</param>
        <value>application/xml</value>
      </equal>
      <equal>
        <param>Content[name(*)]</param>
        <value>queryContextRequest</value>
      </equal>
      <equal>
        <param>Content[/queryContextRequest/attributeList/attribute]</param>
        <value>temperature</value>
      </equal>
      <contains>
        <param>HTTP.Headers</param>
        <value>HTTP.X-auth-token</value>
      </contains>
    </guards>
  </transition>
</state>

```

We write two data specific rules based upon XPATH expressions for querying the data:

```

<param>Content[name(*)]</param>
<value>queryContextRequest</value>

```

To query the data of a HTTP body we use the form:

```
Content[XPATH expression]
```

Here this checks the name of the root tag is queryContextRequest.

```

<param>Content[/queryContextRequest/attributeList/attribute]</param>
<value>temperature</value>

```

This checks that the attribute list has an element called temperature. The application is querying the temperature context and so we are checking if the data structure conforms to this.

Writing a pattern to test the compliance of a new service

We have developed a new implementation of a REST interface and we can to test that it complies with a given interface specification. Here we write the test pattern that determines interface compliance.

- Step one: write the architecture specification

This must be the services to test; this will typically be a single service; here we are testing an instance of the context broker NSGI10 API.

```
<architecture>
  <component>
    <id>pubsubGE</id>
    <address>orion.lab.FIWARE.eu</address>
    <interface>
      <id>NSGI10</id>
      <url>http://orion.lab.FIWARE.eu:1026/NGSI10/queryContext</url>
    </interface>
  </component>
</architecture>
```

- Step two: write the trigger based behaviour specification

This contains a set of trigger events - REST events constructed by the interoperability tool and sent to the service instance; then the guard transitions evaluate the responses received. Compliance is determined by a service responding correctly to events. The first node is a trigger - hence it is a triggerstart type. It describes the transition from the first state A1 to the next state A2 via a sent message in the transition.

```
<behaviour>
  <state>
    <label>A1</label>
    <type>triggerstart</type>
    <transition>
      <to>A2</to>
      <Message>
        <host>component.pubsubGE.address</host>
        <url>component.pubsubGE.NSGI10</url>
        <method>post</method>
        <type>xml</type>
        <headers>
          <header>
            <name>X-Auth-Token</name>
            <value>PSzRfoMrfAr5GswEKDCmg4pKSCjTbHrvru5CxkiBZoRlSwK58hCFOnVhwsCpF3i1suwsR1tiDnO7_2OMP0I1Sw</value>
          </header>
        </headers>
        <body>
          <queryContextRequest>
            <entityIdList>
              <entityId type="Room" isPattern="true">
                <id>Room.*</id>
              </entityId>
            </entityIdList>
            <attributeList>
```

```

        <attribute>temperature</attribute>
      </attributeList>
    </queryContextRequest>
  </body>
</Message>
</transition>
</state>

```

The <message> tag describes how to build a new rest event: Host is the location of the service URL is the interface to send the message to Method is the type of HTTP message to create (post, get, etc.) Type is the data type: XML or JSON at the moment Headers are valid HTTP headers; most common ones (e.g. Content-length) will be added automatically; if there are specific ones to the request e.g. authentication tokens then these can be added as above Body is simply the full body content as a string.

We can then test the response that is returned from the service using a normal transition type:

```

<state>
  <label>A2</label>
  <type>normal</type>
  <transition>
    <to>A4</to>
    <guards>
      <equal>
        <param>HTTP.from</param>
        <value>component.pubsubGE.address</value>
      </equal>
      <equal>
        <param>HTTP.to</param>
        <value>component.ps_client.address</value>
      </equal>
      <equal>
        <param>HTTP.msg</param>
        <value>REPLY</value>
      </equal>
      <equal>
        <param>HTTP.Code</param>
        <value>200</value>
      </equal>
    </guards>
  </transition>
</state>
<state>
  <label>A4</label>
  <type>end</type>
</state>
</behaviour>

```

5 SECURITY DASHBOARD

5.1 Summary

The Security Dashboard supports the management and visualization of security incident-related events and data. Beside core functionalities offered by such a dashboard, the Security Dashboard will also provide reporting capabilities.

The following figure shows the detailed scheme of the XIFI Reference Architecture and the location of the Security Dashboard (see yellow box).

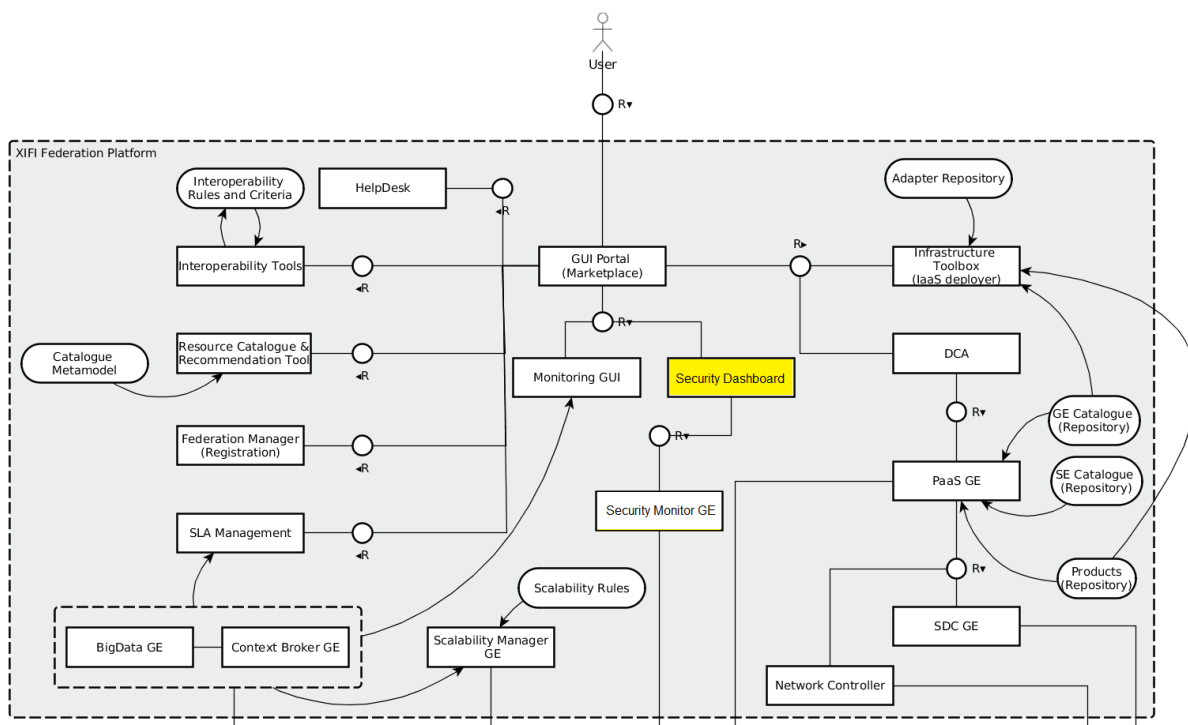


Figure 50: Location of the Security Dashboard in the XIFI Reference Architecture

Reference Scenarios [3]	<ul style="list-style-type: none"> UC5- Network and Data Centre Operations [8]
Reference Stakeholders [6]	<ul style="list-style-type: none"> Federation Manager: To monitor security incidents in the XIFI Federation Platform Infrastructure Owners: To monitor security incidents in its own Node.
Type of ownership	Adaptation and Extension
Original tool	OSSIM GUI provided with the Service Level SIEM component (FIWARE Security Monitoring GE) [50]
Planned OS license	GPL v3 license(as for the FIWARE Security Monitoring GE Terms & Conditions)
Reference OS community	FIWARE Community

Table 17: Summary Security Dashboard.

Consists of

- *OSSIM Web Engine*: it gives access to the information provided by the Service Level SIEM included in the Security Monitoring GE to be shown in the dashboard.
- *Accountability Tool*: it provides a statistic and reporting tool for accountability events generated by the Access Control GE and stored in the SIEM database.
- *XIFI Adaptation Layer*: it allows the integration of the information provided by the previous modules into the XIFI Portal and the user authentication through the XIFI Identity Management GE.

Depends on

- Federation Identity Management GE [93]
- FIWARE Security Monitoring GE [section 7]
- FIWARE Access Control GE [118]

5.2 Component leaders

Developer	Email	Company
Susana Gonzalez Zarzosa	susana.gzarzosa@atos.net	Atos
Cristo Reyes	cristo.reyes@atos.net	Atos
Pascal Bisson	pascal.bisson@thalesgroup.com	Thales
Daniel Gidoin	daniel.gidoin@thalesgroup.com	Thales
Kreshnik Musaraj	kreshnik.musaraj@thalesgroup.com	Thales
Cyril Dangerville	cyril.dangerville@thalesgroup.com	Thales

Table 18: Component responsible - Security Dashboard.

5.3 Motivation

Providing a global and functional view on security in a unified way in the XIFI Portal, that informs on the overall security situation, is not a straightforward task. Moreover, under specific circumstances, it needs to account for contradictory requirements such as information readability and information completeness. The Security Dashboard aims to satisfy the needs for a comprehensive dashboard that enables the operator to assess the on-going situation, and to take the necessary actions. A periodic polling of the dashboard enables the stakeholders a risk monitoring in the medium and the long term.

Furthermore, due to the integration of the Security Dashboard with the Federation Identity Management GE [4] it is possible to control and ensure that only those ones authorized have access to security information, which could be used by malicious actors to compromise the whole federation. In this way, only the Federator will have access to the events generated in all the federation whereas each Infrastructure Owner will only have access to the events generated in its own assigned range of IP addresses.

To get the relevant security information shown in the Security Dashboard, it will take advantage of the FIWARE Security Monitoring GE, which includes a Service Level SIEM (Security Information Event Management) component that provides real-time analysis of security events collected from the different nodes in the XIFI Platform. Section State of the art [section 3.5] provides an overview of existent open source SIEM tools in the security monitoring area and why the FIWARE Security Monitoring GE has been chosen for the Security Dashboard.

Finally, although Service Level SIEM component makes use of the dashboard provided with the OSSIM core engine included in it and the FIWARE Security Monitoring GE has its own Visualization Framework component (integrating also the security alarms generated through the SIEM), the use of a separate Security Dashboard interface in XIFI provides not only the advantages of being integrated in a federated environment through the XIFI Portal and the Federation Identity Management but also the capability of adding in a common and unified interface for the XIFI user of future expansions, such as access to advanced accountability reporting tools from events generated by the Access Control GE installed in the XIFI slaves nodes (scheduled for the second cycle of XIFI) or visual analytics techniques for the Attack Paths Engine also included in the FIWARE Security Monitoring GE.

5.4 User stories backlog

id	User story name	Actors	Description	Task id
1	Visualization of accountability logs	Federation Security Officer / Infrastructure Owner system administrator	The user wants to visualize accountability events from the logs generated by the Access Control GE installed there.	769
2	Visualization of other security events	Federation Security Officer / Infrastructure Owner system administrator	The user wants to visualize other security events generated by Security Probes installed in the infrastructure/federation.	770
3	Configure security directives in the Security Monitoring GE	Federation Security Officer / Infrastructure Owner system administrator	The user configures the correlation rules through the dashboard provided by the Service Level SIEM included in the Security Monitoring GE.	771
4	Visualize security incidents correlating events coming from one node in the Security Monitoring GE dashboard	Federation Security Officer / Infrastructure Owner system administrator	The user visualizes, through the dashboard provided by the Service Level SIEM included in the Security Monitoring GE, the events collected by the SIEM Agents and the security incidents detected.	772
5	Generate security reports through the Security Monitoring GE dashboard	Federation Security Officer / Infrastructure Owner system administrator	The user saves a record of the security information detected by the Service Level SIEM included in the Security Monitoring GE.	773
6	Main Security Dashboard	Federation Security Officer / Infrastructure Owner system administrator	The user visualizes relevant security information (events and incidents) through the Security Dashboard.	774
7	Security Dashboard integrated in the XIFI Portal	Federation Security Officer / Infrastructure Owner system administrator	The user accesses the Security Dashboard through the XIFI Portal.	775

8	Security Dashboard integrated with the IdM	Federation Security Officer / Infrastructure Owner system administrator	The user login to the Security Dashboard using the SSO provided by the IdM.	776
9	Statistics tools for accountability events	Federation Security Officer/ Infrastructure Owner system administrator	The user can use statistic tools in the Security Dashboard to visualize detailed information about accountability events.	777
10	Reports on accountability events	Federation Security Officer/ Infrastructure Owner system administrator	The user wants to visualize reports about accountability events on a weekly and monthly basis.	778
11	Visualize security incidents in the XIFI Platform	Federation Security Officer	The user visualizes through the Security Dashboard security incidents correlating events coming from different nodes in the XIFI Platform.	779
12	Security Dashboard integrated with the Federation Manager	Infrastructure Owner system administrator	The user with Infrastructure Owner role only will have access to the visualization of events collected in the range of IP addressed registered in the Federation Manager for its region.	1519

Table 19: User Stories backlog - Security Dashboard.

5.5 State of the art

Security monitoring is one of the basic requirements that need to be considered in any infrastructure and even more in a distributed system such as the XIFI Federation.

With this purpose, SIEM (Security Information Event Management) solutions provide the technology for real-time analysis of security events, collecting and normalizing them from heterogeneous sources in order to aggregate and correlate them to identify security threats or incidents and generate useful information for the risk management in a system. The result of this monitoring is usually shown in these solutions through a web interface used as security dashboard.

Some of the available open sources SIEMs that can be found on Internet are the following:

- Prelude [51]
- OSSIM [52]
- Logalyze[53]
- Cyberoam[54]
- Apache ALOIS (Advanced Log Data Insight System) [85]

Apache ALOIS is a SIEM more focused in monitoring the security of the content. So, although it has the support of Apache it is not the most suitable solution for a federation where it is important to monitor also the security from a network level more than detecting modifications in contents.

From the rest of SIEMs, the most widely used and with the biggest support community is OSSIM. Besides, this open source solution offers several advantages for the security monitoring in the XIFI Platform such as:

- It already includes a big number of plugins to normalize logs from many different data source to a common format in order to be correlated in the server, and many predefined security incidents to be detected. This is useful in a federated environment where each node can have

different type of applications and devices.

- It includes agents that can be installed in a distributed way for a more efficient collection of events and a server for the correlation of them.
- It includes a friendly and complete dashboard to visualize at a glance the main security incidents/threats, the incidents detected and go into more detail visualizing the events which generated the alarm and all the raw events collected by the agents.

However, it has some limitations, mainly related with performance and scalability which make it not suitable for (direct use by) systems such as the XIFI Federation where a huge amount of data can be generated in a short amount of time.

The Service Level SIEM component included in the FIWARE Security Monitoring GE, built on top of OSSIM tries to overcome these limitations incorporating to the features enumerated before for the OSSIM core engine, the use of a Storm cluster where Esper correlation processes are running, achieving in this way a SIEM with a high performance and scalable correlation engine (with a guaranteed message processing and robust process management thanks to the use of Storm and Zookeeper) and the capability to define security directives or correlation rules in a simple language (like-SQL) rich event conditions or patterns and time windows dealing with high frequency time-based event data. More information can be found in the FIWARE Security Monitoring GE Open Specifications [55].

5.6 Architecture design

The Security Dashboard includes a graphical user interface integrated in the XIFI Portal for the management and visualization of security and accountability events and incidents in the XIFI Platform. It is connected to the Federation Identity Management GE for the authentication of users and to the Federation Manager to recover the range of IP addresses associated to the nodes.

The information shown in the dashboard is provided by the FIWARE Security Monitoring GE (see more details about its architecture in section 7.4). It is recovered from the SIEM database through the OSSIM Web Engine and the Accountability Tool (focused on the events generated by the Access Control GE).

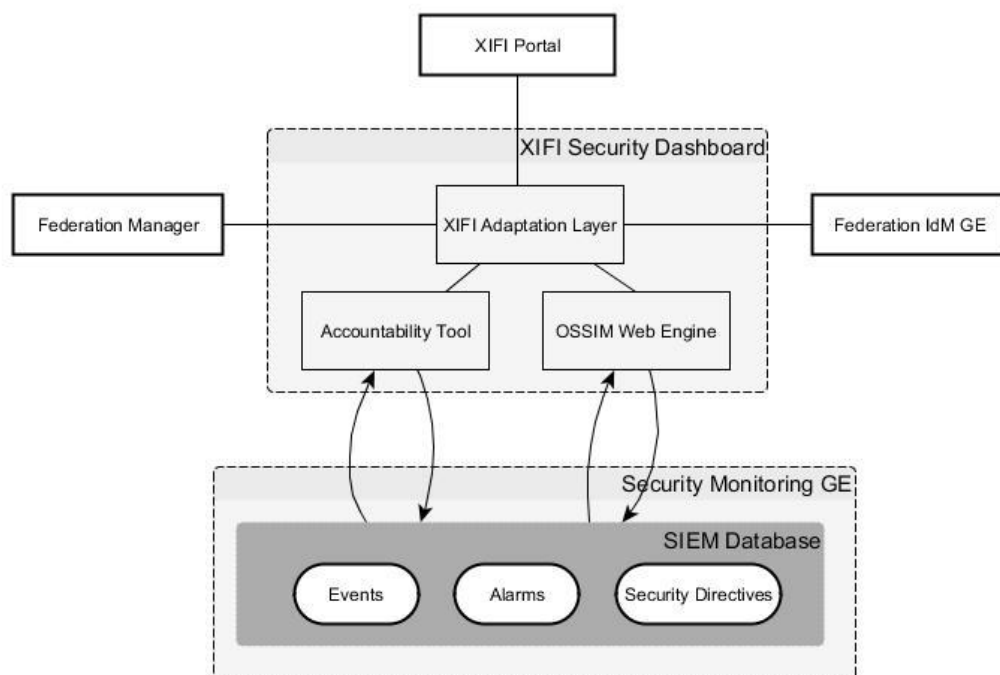


Figure 51: Security Dashboard Architecture diagram.

5.7 Release plan

Version Id	Milestone	User Stories
0.1	30.12.2013	1,2,3
0.2	31.01.2014	4,5
1.0	31.03.2014	6,7
1.1	31.06.2014	8
1.2	31.08.2014	12
2.0	31.09.2014	9, 10, 11

Table 20: Release Plan - Security Dashboard.

5.8 Test cases

Test id	Test description	Test script
1	Verification of the integration with the IdM	see below
2	Verification of the Security Dashboard installation	see below
3	Verification of the Accountability Tool installation	See below

Table 21: Security Dashboard – Test Cases

Unit Test 1– Verification of the integration with the IdM

If the Security Dashboard has been successfully integrated with the IdM, the first time the user invoke the endpoint <https://securitydashboard.fi-xifi.eu> (as the hostname is not yet in the DNS, the 193.205.211.69 IP should be included in the localhost definition), the IdM page is shown to introduce the user credentials:

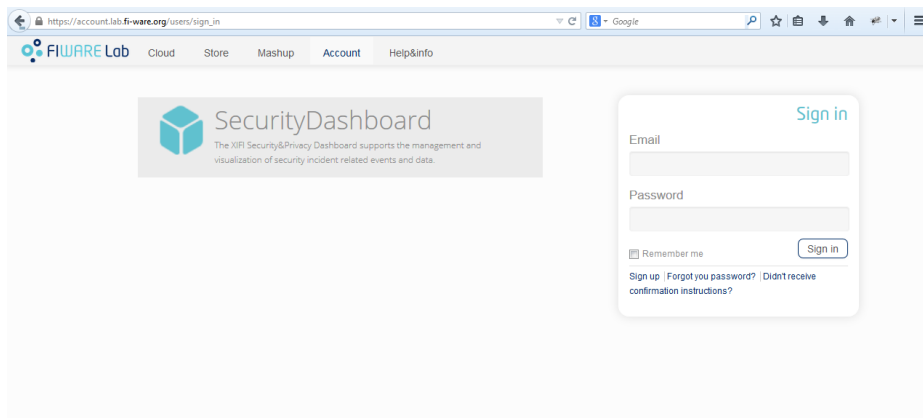


Figure 52: Security Dashboard Unit Test 1

Unit Test 2 – Verification of the Security Dashboard installation

If the Security Dashboard has been successfully installed, the following dashboard should be shown (once logged into the Identity Management GE) when the endpoint <https://securitydashboard.fi-xifi.eu> is invoked:

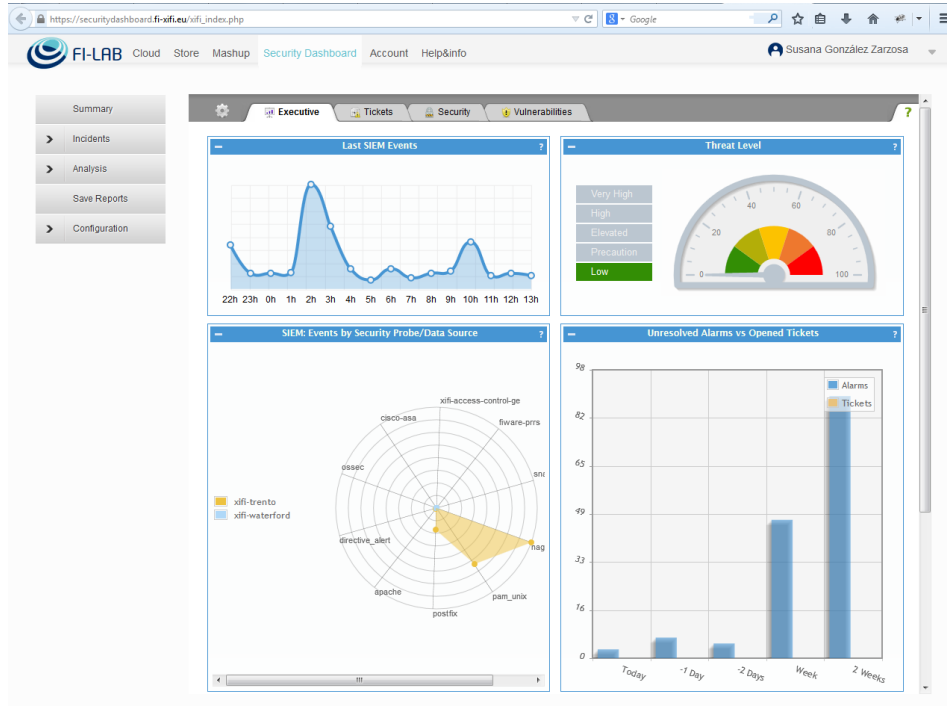


Figure 53: Security Dashboard Unit Test 2

Unit Test 3 – Verification of the Accountability Tool installation

If the Accountability Tool has been successfully installed, the following graphics should be shown (on the Security Dashboard):

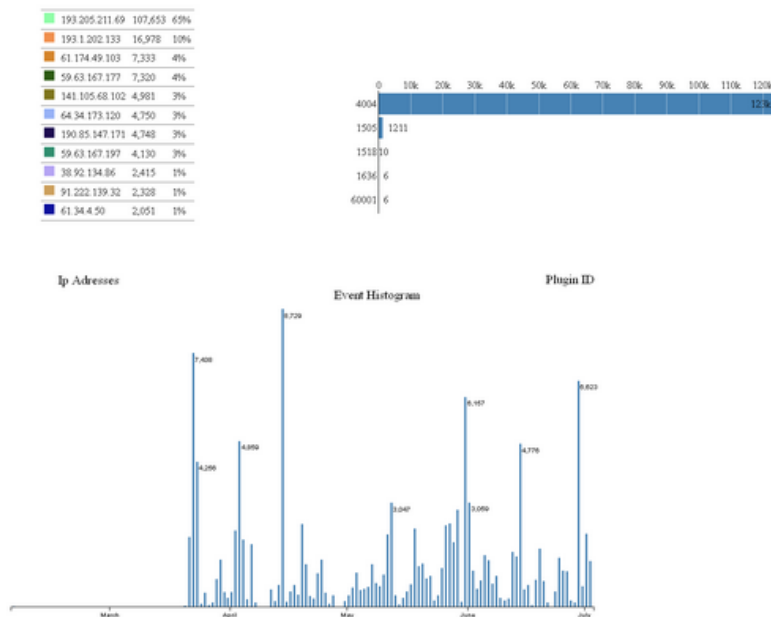


Figure 54: Security Dashboard Unit Test 3

5.9 Installation Manual

The XIFI Security Dashboard is provided as a web application to put it available in the Apache2. It is built as an extension on top of the OSSIM (Open Source Security Information Management) graphical interface and distributed as a patch to be applied once it has been installed.

Requirements

- OSSIM v4.1.0
- Apache2 with SSL access enabled
- PHP OAuth Client Library

Besides, the following XIFI components are used by the Security Dashboard and consequently need to be available from the host where the Security Dashboard is installed:

- Security Monitoring GE (in particular the SIEM database)
- Federation Identity Management [93]
- Federation Manager [94]

Software repository

The source code can be downloaded from here:

- Security Dashboard:
<https://xifisvn.res.eng.it/wp4/SecurityDashboard/trunk/bin>
- Accountability Tool:
<https://xifisvn.res.eng.it/wp4/AccountabilityTool>

Installation

- The Security Dashboard is an extension and adaptation of the *OSSIM graphical interface v4.1.0* and it is provided as a patch to be applied to that version. If it is not already installed (it is provided with the Security Monitoring GE (see more information in section 7), download the file `ossim-www-4.1.0.tar.gz` from the XIFI software repository and install it with the commands below:

```
# mkdir -p /usr/share/ossim
# cd /usr/share/ossim
# tar -zxvf ossim-www-4.1.0.tar.gz
```

- Download and install the *PHP OAuth Client Library* (available in GitHub [96]).
- Download the Security Dashboard source code from the XIFI software repository and unzip the files.
- Install the Security Dashboard patch in the same folder where the OSSIM graphical interface has been installed (by default `/usr/share/ossim`) and update the SIEM database with the changes required for the Security Dashboard:

```
# cd /usr/share/ossim
# patch -p1 < XIFI-securitydashboard-www-2.0.patch
# mysql -uroot -p < XIFI-securitydashboard-database-2.0.patch.sql
```

- Put the Security Dashboard available in the Apache2 configuration using SSL. The following file `/etc/apache2/sites-enabled/securitydashboard` could be used:

```

<VirtualHost *:443>
    ServerName securitydashboard.fi-xifi.eu
    Alias /securitydashboard /usr/share/ossim/www
    DocumentRoot /usr/share/ossim/www
    DirectoryIndex index.php

    Alias /ossim "/usr/share/ossim/www"

    <Directory /usr/share/ossim/www>
        <Files ~ "\.(in|am|txt|pl|local|old|conf|ini|sql|cnf)$">
            Order Allow,Deny
            Allow from 127.0.0.1
        </Files>
        <Files ~ "Makefile">
            Order Allow,Deny
            Allow from 127.0.0.1
        </Files>
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog ${APACHE_LOG_DIR}/ssl_access.log combined
    ErrorLog ${APACHE_LOG_DIR}/ssl_error.log

    ErrorDocument 401 /ossim/404.php
    ErrorDocument 403 /ossim/404.php
    # SSL Engine Switch:
    SSLEngine on

    # A self-signed (snakeoil) certificate can be created by installing
    # the ssl-cert package. See
    # /usr/share/doc/apache2.2-common/README.Debian.gz for more info.
    # If both key and certificate are stored in the same file, only the
    # SSLCertificateFile directive is needed.
    SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem

```

```

SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key

# Disable Weak Ciphers
SSLProtocol -All +SSLv3 +TLSv1
SSLCipherSuite HIGH:!SSLv2:!ADH:!aNULL:!eNULL:!NULL

<FilesMatch "\.(cgi|shtml|phtml|php)$">
    SSLOptions +StdEnvVars
</FilesMatch>

BrowserMatch "MSIE [2-6]" \
    nokeepalive ssl-unclean-shutdown \
    downgrade-1.0 force-response-1.0
# MSIE 7 and newer should be able to use keepalive
BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
</VirtualHost>

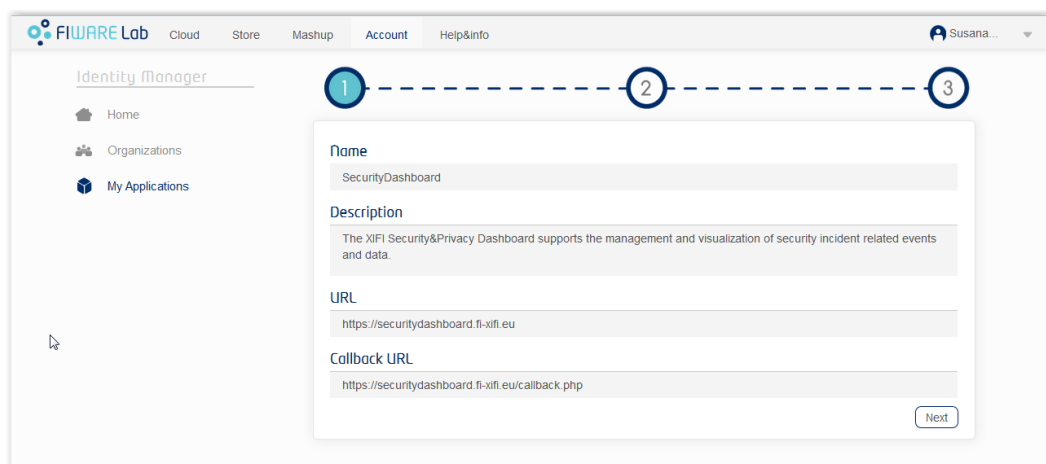
```

- Create the application in the Federation Identity Management.

The Security Dashboard has to be registered in the Federation Identity Manager [93]. In the Federated Identity Management, the responsible of the component has to go to *My Applications* -> *Add Application* and introduce:

1. Basic data:

- Name of the application
- Description of the application
- URL of the application
- Callback URL is the URL where the IdM will call back after to identify the user



The screenshot shows the FIWARE Lab Identity Manager interface. The user is logged in as 'Susana...'. The 'My Applications' section is active, and a form is displayed for adding a new application. The form fields are:

- Name:** SecurityDashboard
- Description:** The XIFI Security&Privacy Dashboard supports the management and visualization of security incident related events and data.
- URL:** https://securitydashboard.fi-xifi.eu
- Callback URL:** https://securitydashboard.fi-xifi.eu/callback.php

A 'Next' button is visible at the bottom right of the form. A dashed line with three numbered circles (1, 2, 3) is overlaid on the form, indicating the steps for creating the application.

Figure 55: Create Security Dashboard application in the IdM.

2. Select the application's logo (optional)
3. Create the new roles associated to the application and the permissions assigned to each: *FederationManager* and *IO*. The *Provider* and *Purchaser* roles are associated directly with the IdM.

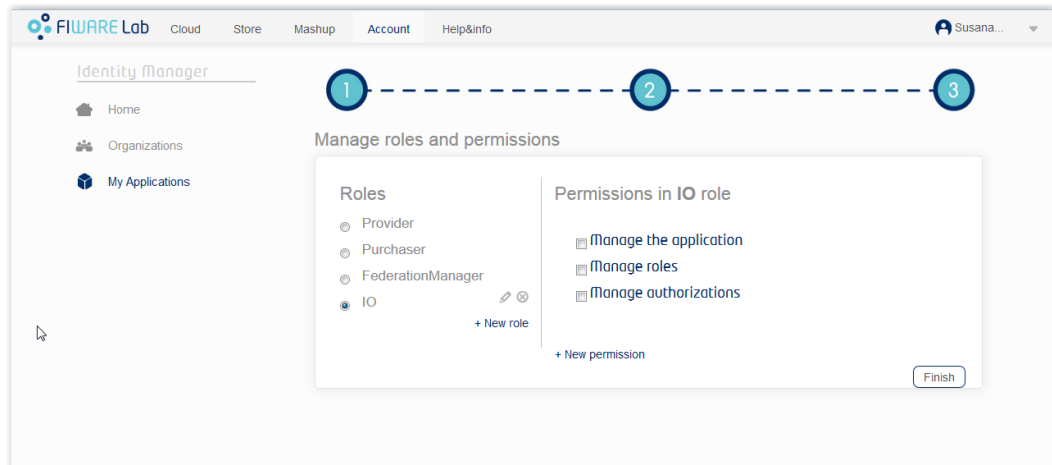


Figure 56: Create Roles for the Security Dashboard in the IdM.

Afterwards, the responsible can find the IdM credentials for the application, in the link *OAuth2 Credentials*.

These credentials (*ClientID* and *ClientSecret*) have to be introduced in the Security Dashboard configuration file (*/securitydashboard/config.ini*).

The application can manage the users or organizations, through the IdM. It is only necessary to assign the appropriate roles (FederationManager or IO).

For more detail about how to use the Federation Identity Management you can see the User Manual [93].

- Configure the parameters of the Security Dashboard in the file */securitydashboard/config.ini*:

Configuration of the Security Dashboard:

- `ossim_endpoint` = url where the OSSIM service is available (by default: <https://securitydashboard.fi-xifi.eu/ossim>)
- `securitydashboard_endpoint` = url where the Security Dashboard is available in Apache (by default: <https://securitydashboard.fi-xifi.eu>)

Configuration of the Federation Identity Monitoring:

- `idm_endpoint` = url where the Federation Identity Management is available (by default: <https://account.lab.FIWARE.org>)
- `ClientID` = id number provided by the IdM, when you register the application.
- `ClientSecret` = secret number provider by the IdM that it is associated to the ClientID.

Configuration of the Federation Manager [94]:

- `federationmanager_endpoint` = url where the Federation Manager is available
- `federationmanager_auth` = authorization to be included in the REST query header (by default: "Authorization: Bearer developer")

- Download the Accountability Tool from the XIFI software repository, copy the files under the folder "accountability" of the Security Dashboard (by default */usr/share/ossim/www/accountability*) and configure it:

For each `data_graph.php` file:

- `$host` : address of the MySQL server
- `$namesbase` : name of the event database

- \$username : login of the account fetching the events data
- \$password: password of the account fetching the events data
- Finally, restart the Apache2 service to include the changes:

```
# sudo service apache2 restart
```

If there is no problem, the XIFI Security Dashboard will be available at <https://securitydashboard.fi-xifi.eu>

5.10 User Manual

This manual shows how to access and use the functionalities provided by the Security Dashboard graphical user interface. You can access to the dashboard with this URL: <https://securitydashboard.fi-xifi.eu> (as the hostname is not yet in the DNS, the 193.205.211.69 IP should be included in the localhost definition).

User authentication

The user needs to be identified in the Security Dashboard application before starting to use it. This is done through the Federation Identity Manager (see the Security Dashboard Installation Manual section 5).

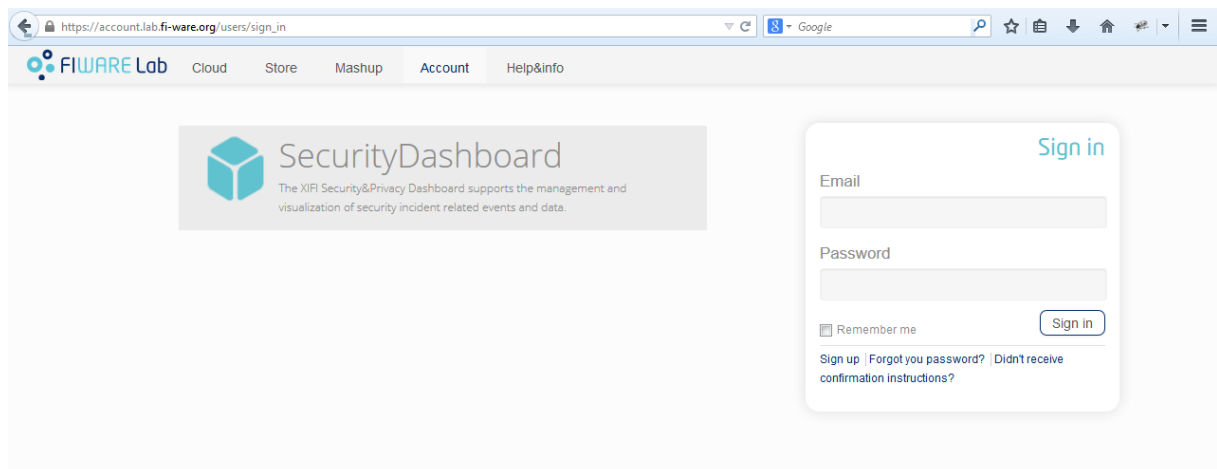


Figure 57: Login in the Security Dashboard.

The following roles can interact with the Security Dashboard:

- *FederationManager*: This role is responsible for managing the application, configure the security directives and policies to be applied in the federation and visualize the status of all the nodes.
- *IO*: This role indicates that the user, who has been identified, belongs to an Infrastructure Owner and has security administration permissions for its node. Hence, the dashboard allows him to visualize the security events and incidents taking place in this node.

To register a user in the Security Dashboard, go to the Federation IdM and add the new user or organization and assign the appropriate role. To see more details about how to interact with the IdM look up the Federated Identity Management [93].

After the identification of the user, the application shows the respective dashboard, depending on the assigned role.

Security Dashboard for Infrastructure Owners

- *Summary*

The Infrastructure Owner user will access first a set of diagrams which gives him at a glance an idea of what it is happening in his node including:

1. Executive Summary:
 - Number of SIEM Events in the last hours.
 - Threat Level (based on the risk value of the alarms detected).
 - Events collected and grouped by Security Probes and Data Sources
 - Unresolved alarms vs opened tickets
 - Top 10 SIEM events by product type (DHCP, web server, mail server, etc) and categories (authentication, application...)
2. Tickets: the status of the tickets (closed/open), the resolution time, their classes and the tickets opened by the user.
3. Security: it summarizes the security events trends during the last week or day, the most common events and alarms and the top of the attacking hosts.
4. Vulnerabilities: a summary of the vulnerability scan jobs executed.

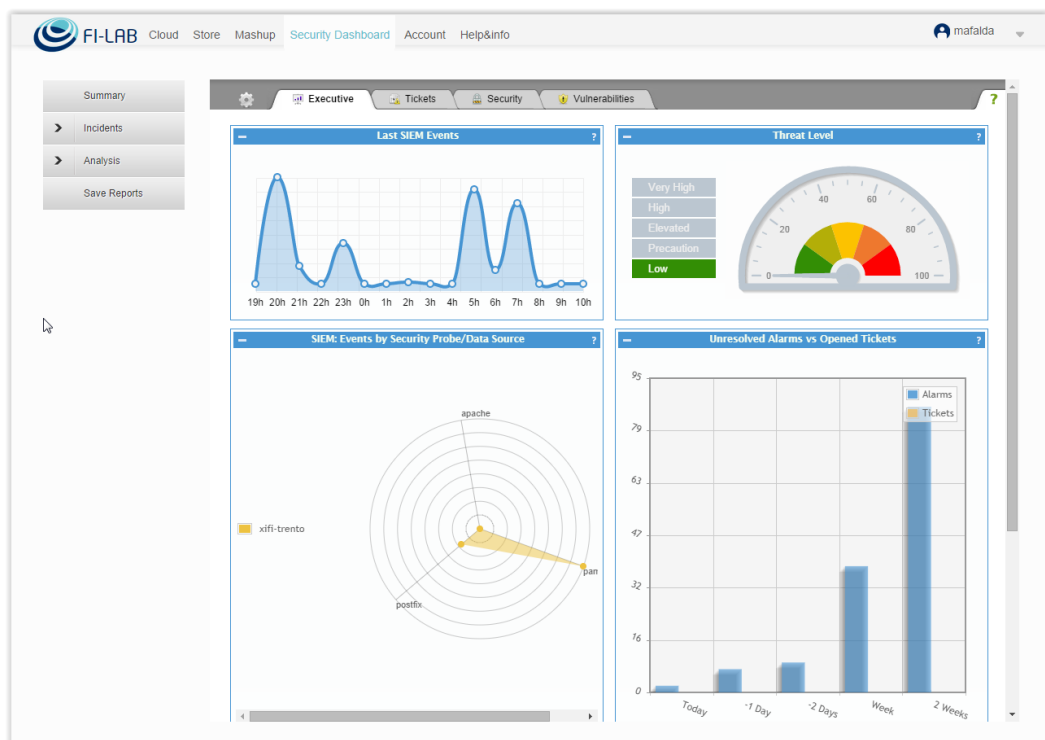


Figure 58: View of Security Dashboard for Infrastructure Owners.

- *Incidents*

- Alarms

From here the user can see the alarms detected in its node and access to a more detailed information about them, including a short description, the number of events involved in the detection and the time between the first event and the detection of the incident, the risk value calculated and its status. It is important remark that the user IO only will have access to alarms generated with events collected in its node.

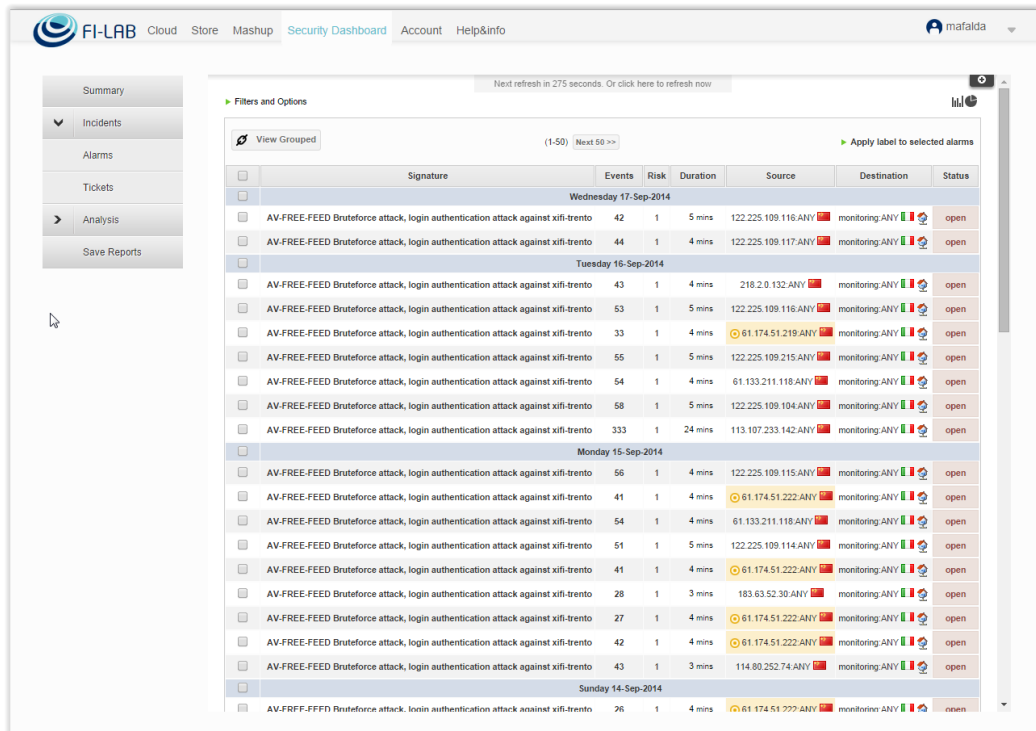


Figure 59: Security Dashboard - Visualization of Alarms.

- Tickets

Here, it is shown a detailed list of the tickets assigned to the user with its priority, when they were created and its status.

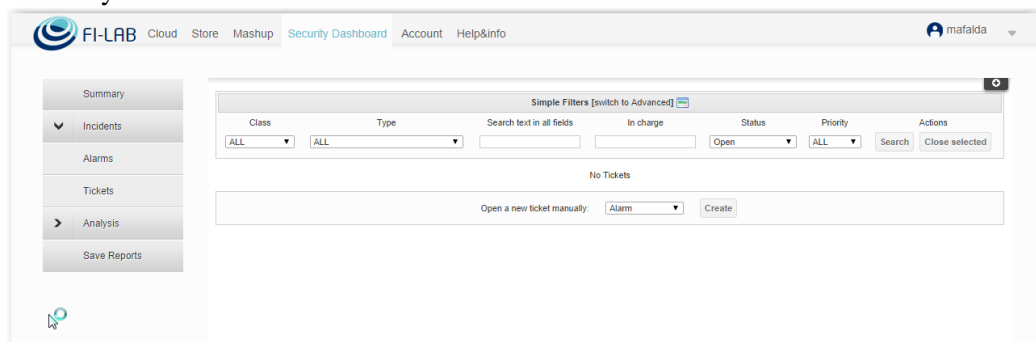


Figure 60: Security Dashboard - Visualization of Tickets.

- Analysis

- Security Events

This section shows a detailed list of the last SIEM events detected, their types, date, source, destination, sensor (=security probe) where they were collected and risk value. From here, the user can search by different criteria and see the events with more detail (for example to check the information included in each userdata or the priority and reliability of the event). Similar to the alarms, the IO user will only visualize the events collected in its node.

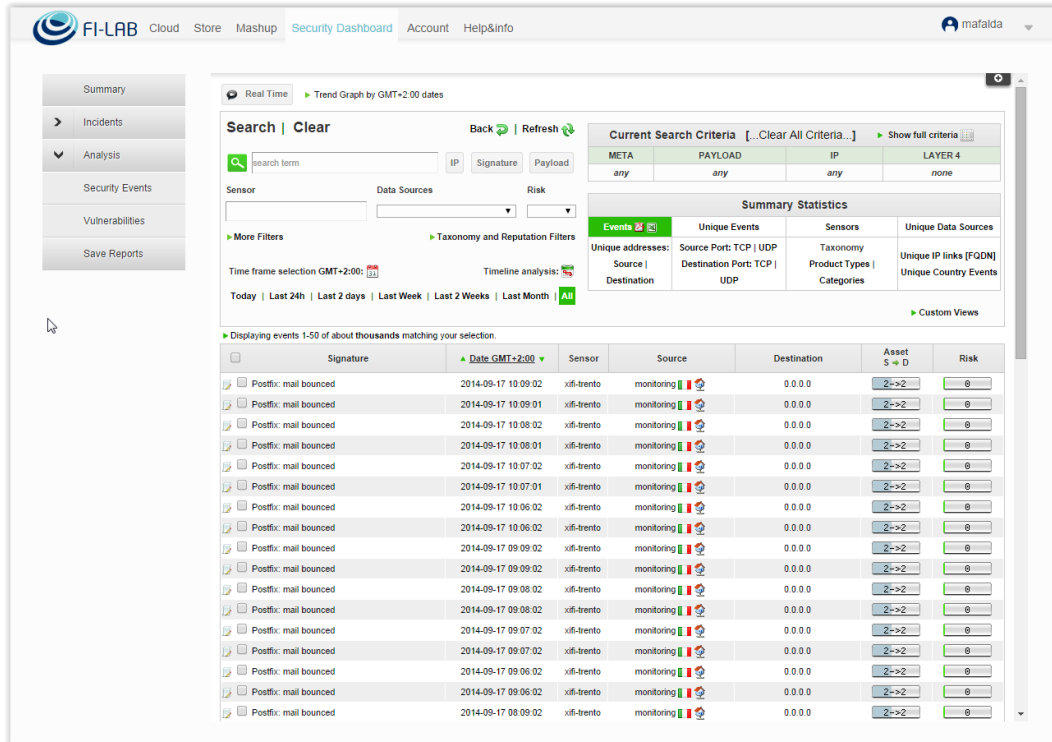


Figure 61: Security Dashboard - Visualization of Events.

- Vulnerabilities

This section allows checking vulnerabilities detected or start a new vulnerability scan job. This option is only available in case a scanner tool, such as Nessus, has been installed.

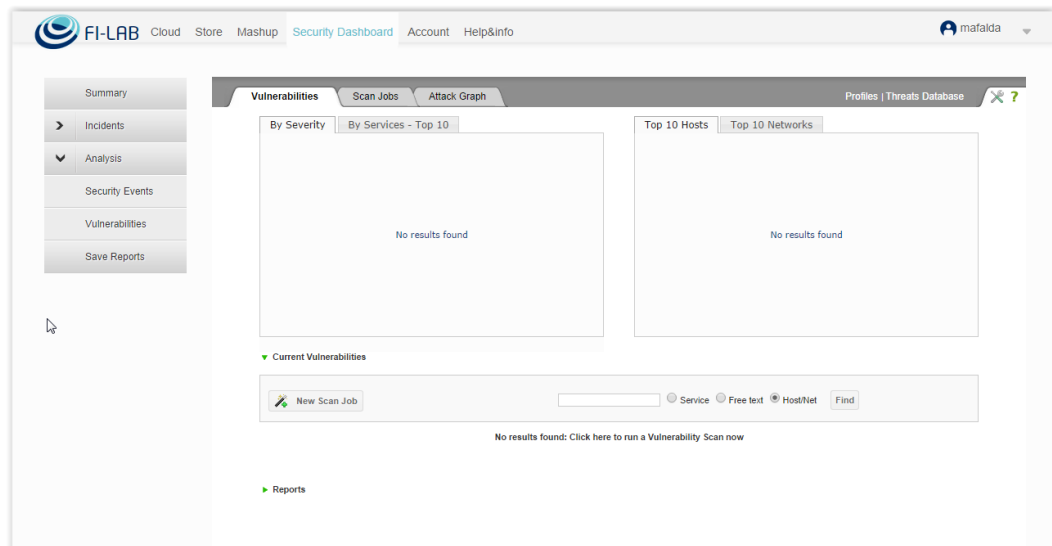


Figure 62: Security Dashboard - Vulnerabilities.

- Save Reports

This section allows the generation of different reports in PDF files (e.g. top attackers, alarms, tickets, different metrics, etc).

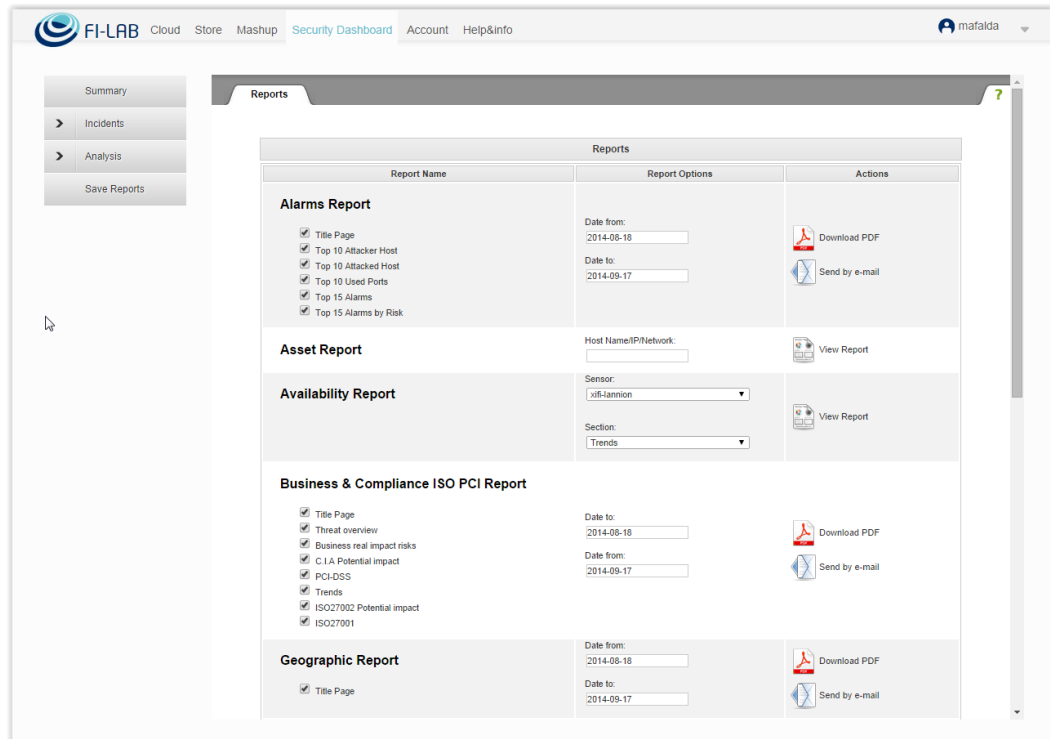


Figure 63: Security Dashboard - Save Reports.

Security Dashboard for Federation Managers

The Federation Manager has access to all the functionality provided to the Infrastructure Owners (see details above for the IO role) but considering not only a specific node but all the nodes where the Security Probes are running.

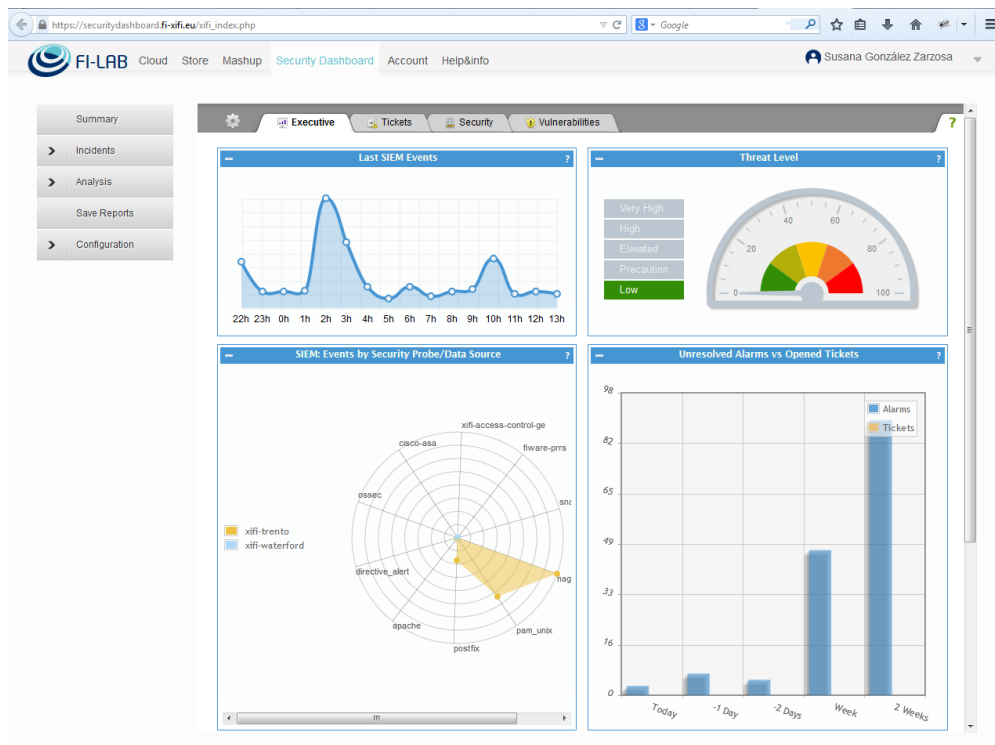


Figure 64: View of Security Dashboard for Federation Manager.

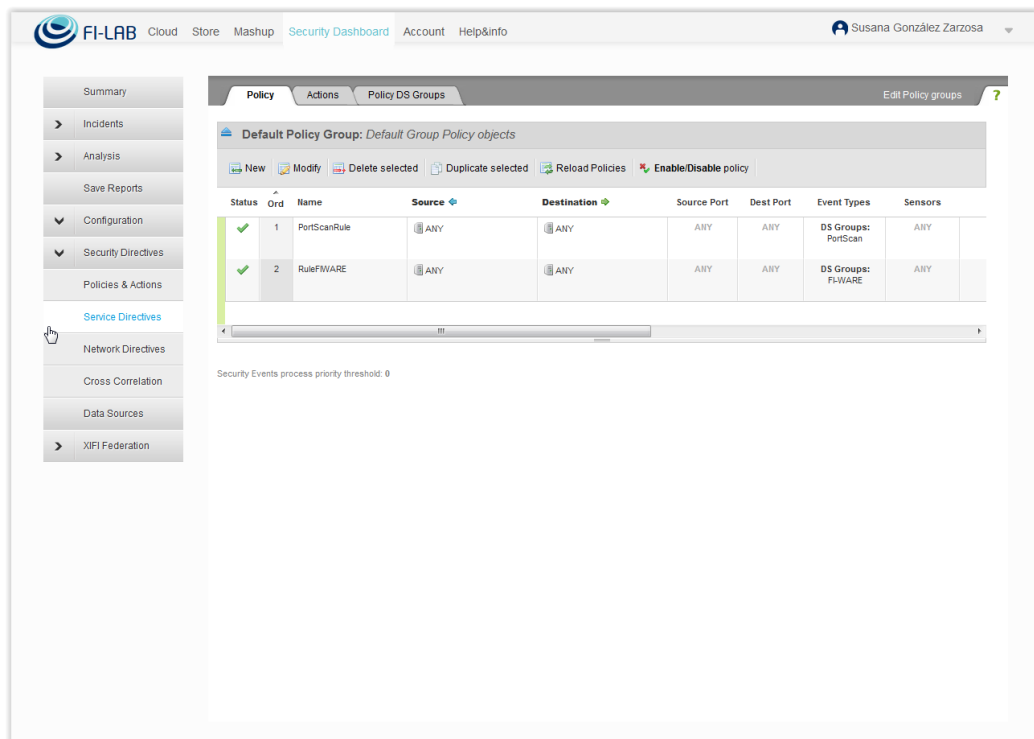
Furthermore, the Federation Manager has access to the *Configuration* menu of the Security Dashboard application which includes:

- *Security Directives*

Through this screen, the Federation Manager can configure and establish the security monitoring to be performed in the federation. In particular:

- Policies and Actions:

The Federation Manager can specify a set of filtering conditions (such as the source/destination IP, port, time/date range or the type of event) for the incoming events collected from the Security Probes installed in the slave nodes and define how the security monitoring system will react to them (open a ticket, send an email or execute a console command).



Status	Ord	Name	Source	Destination	Source Port	Dest Port	Event Types	Sensors
✓	1	PortScanRule	ANY	ANY	ANY	ANY	DS Groups: PortScan	ANY
✓	2	RuleFWARE	ANY	ANY	ANY	ANY	DS Groups: FW-WARE	ANY

Figure 65: Security Dashboard - configure policies and actions.

- Network Directives:

Through this menu, the Federation Manager can define directives that help to detect common network attacks and latent problems.

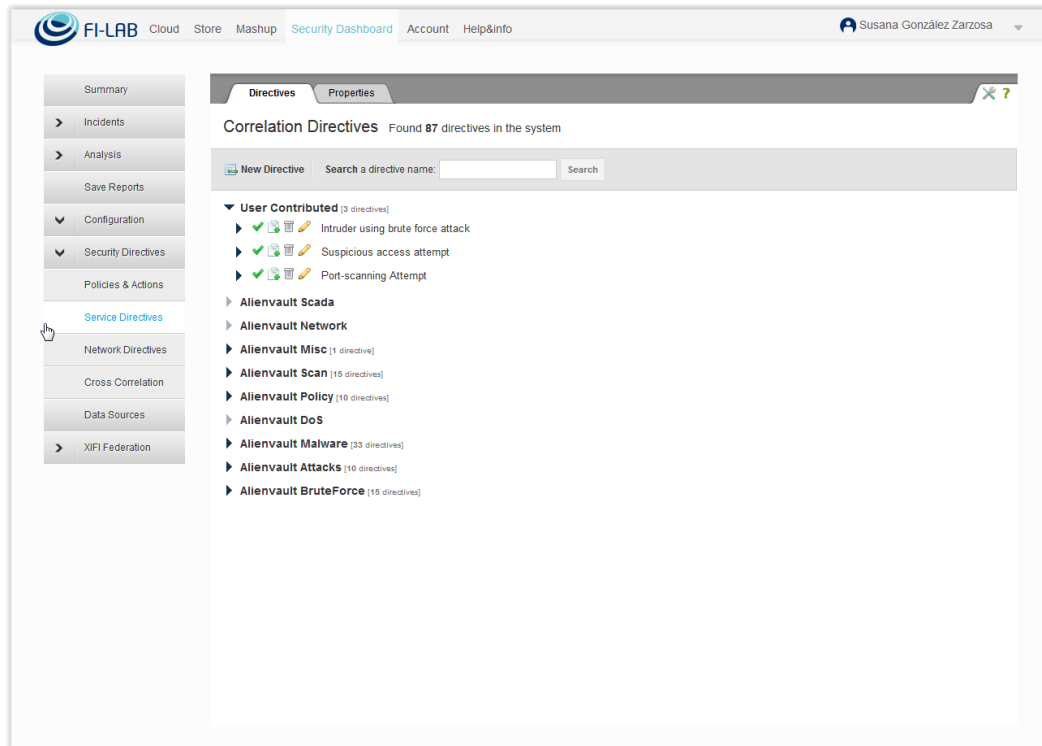


Figure 66: Security Dashboard - configure network directives

- **Service Directives:**

It allows the Federation Manager to define more complex rules or directives to be applied by the correlation engine included in the security monitoring to detect attacks or abnormal behaviours in the federation.

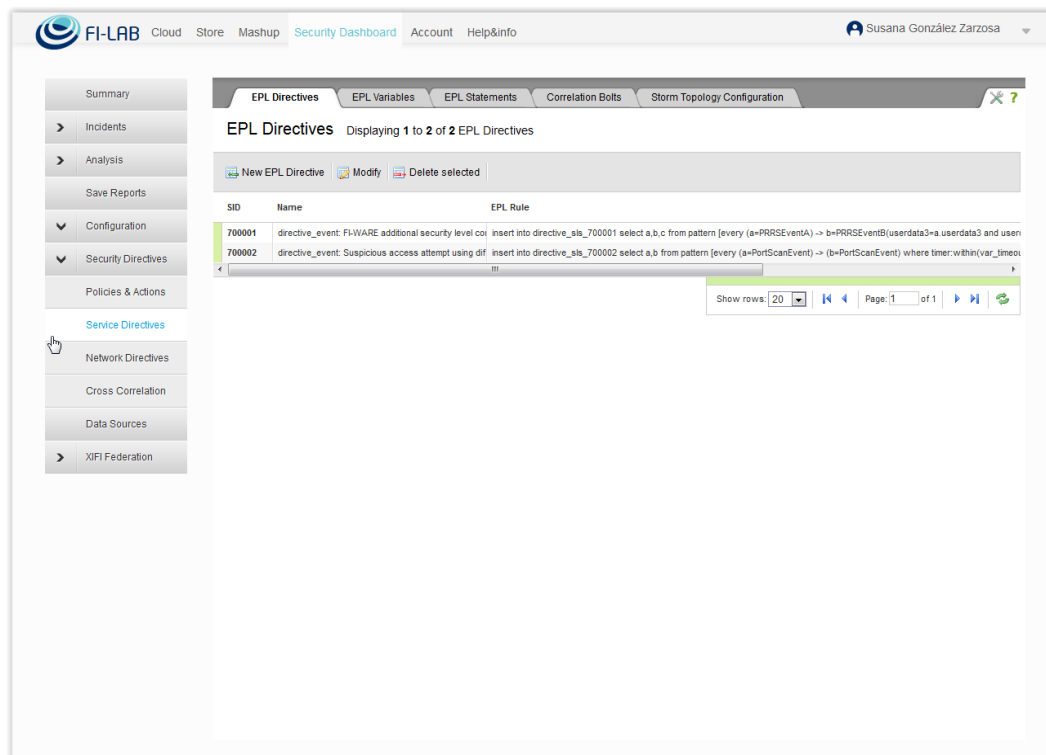


Figure 67: Security Dashboard - configure service directives

- Cross Correlation:

Finally, it is possible to define cross correlation rules between incoming events and vulnerabilities detected or between alarms generated by the service directives.

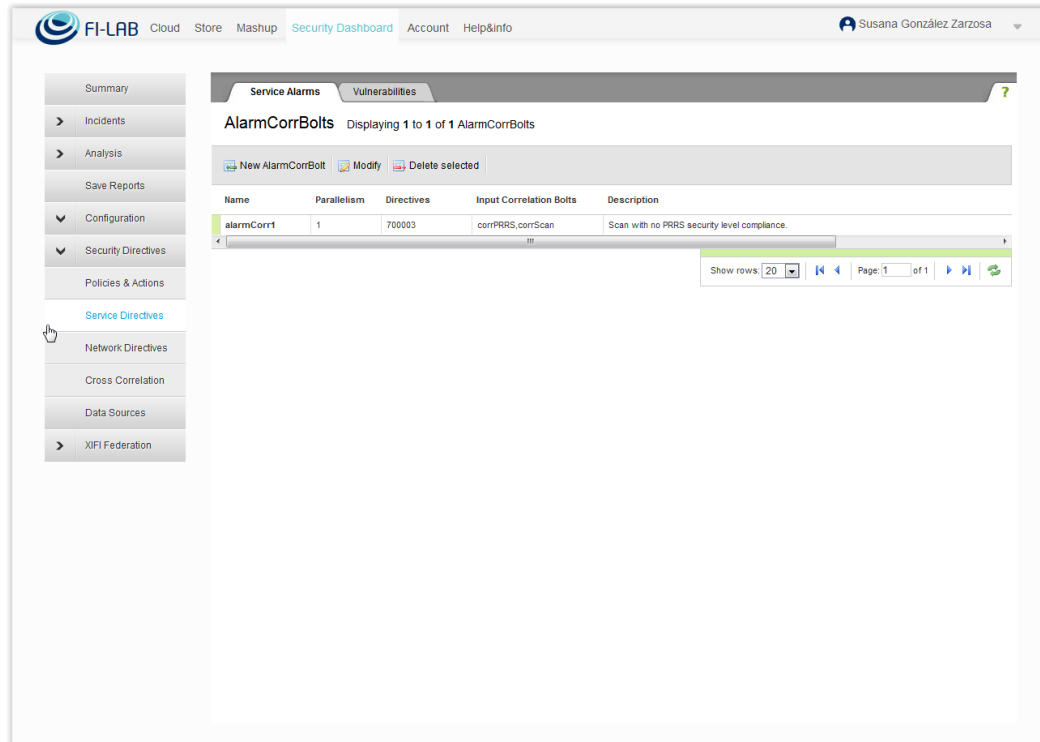


Figure 68: Security Dashboard - configure cross correlation

More information can be found in the FIWARE Security Monitoring / Service Level SIEM User and Programmers Guide [61].

- *Data Sources*

Through this screen, the Federation Manager can visualize and manage the data sources considered in the security monitoring.

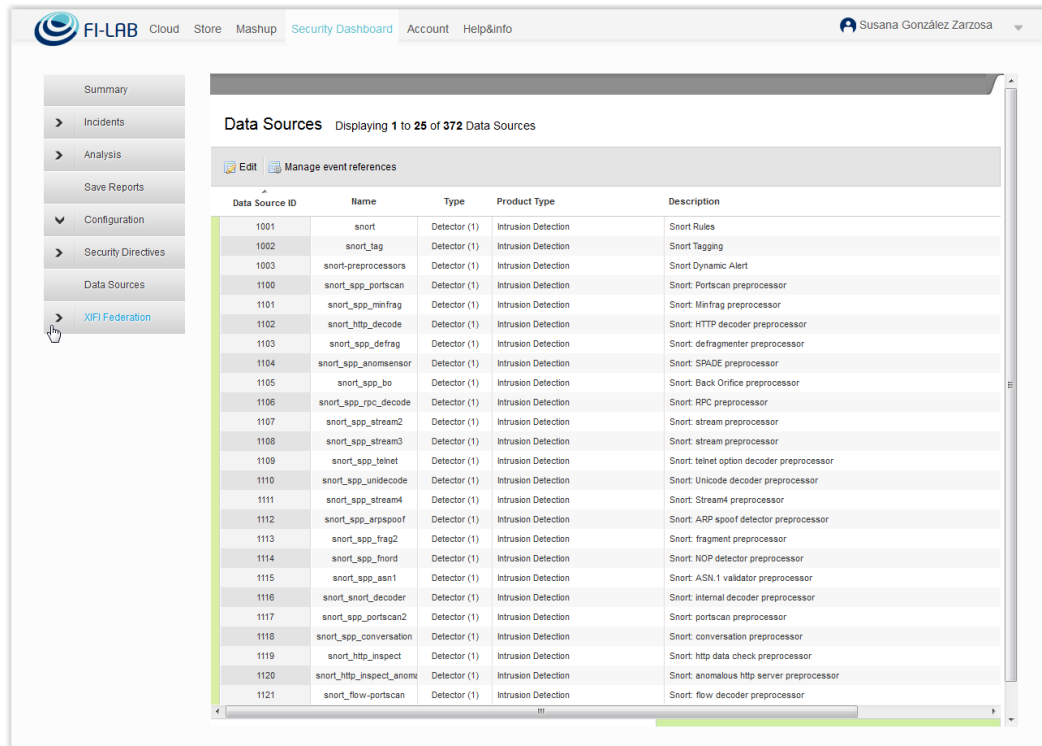


Figure 69: Security Dashboard - configure data sources.

- **XIFI Federation**
 - **XIFI Masters:** this screen shows the SIEM server which is active in the XIFI Federation and its status.

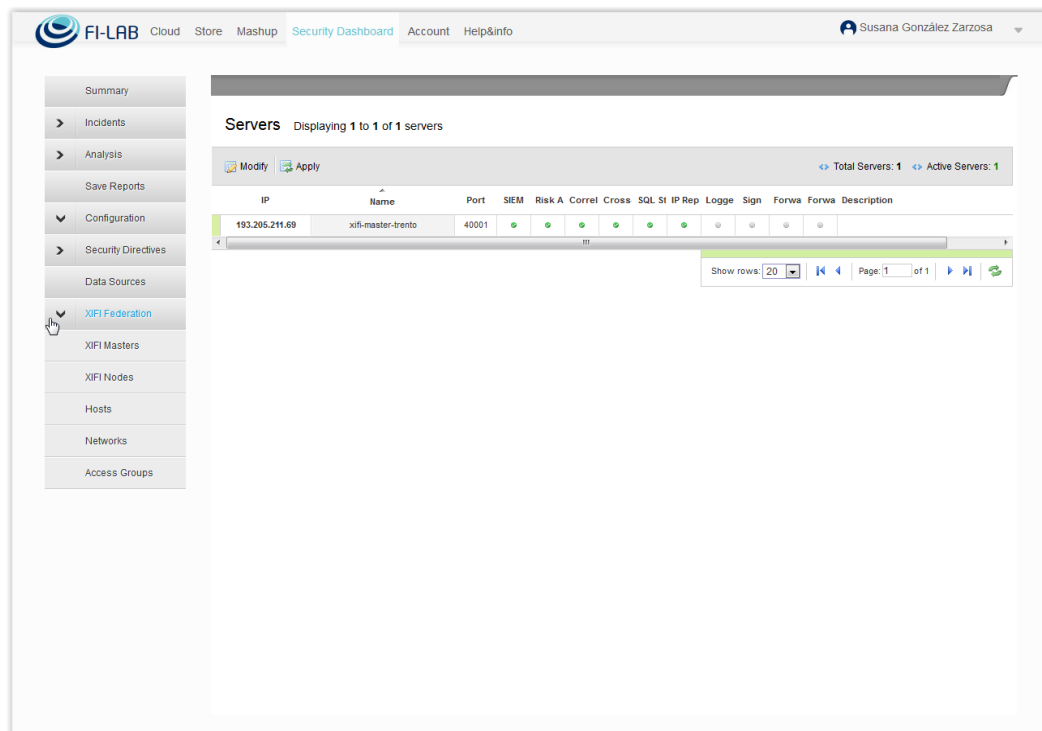


Figure 70: Security Dashboard - show servers status.

- **XIFI Nodes:** this screen shows the Security Probes which are active in the XIFI Federation and their status (if they are connected to the server).

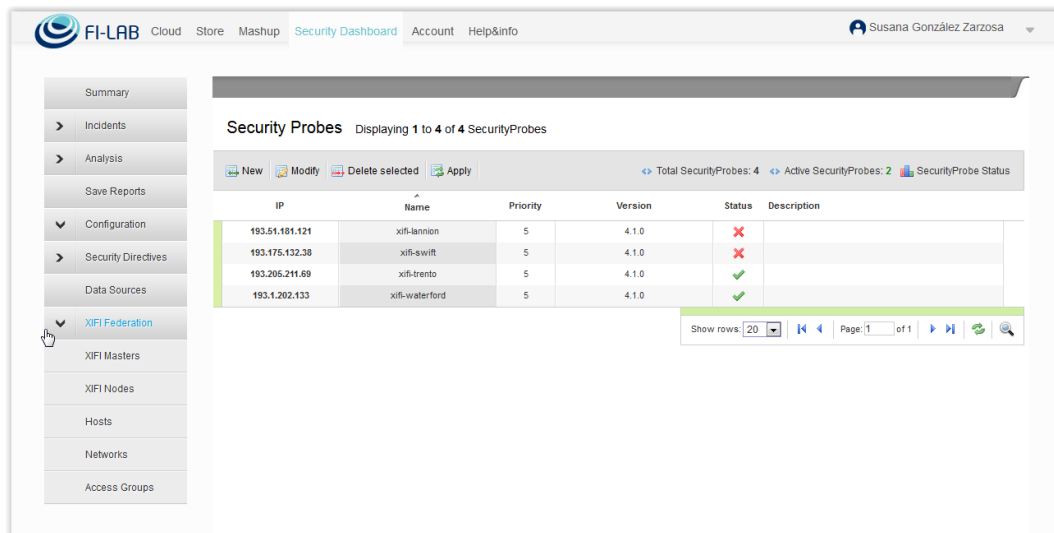


Figure 71: Security Dashboard - show security probes status.

- **Hosts:** this screen allows defining hosts in the XIFI Federation

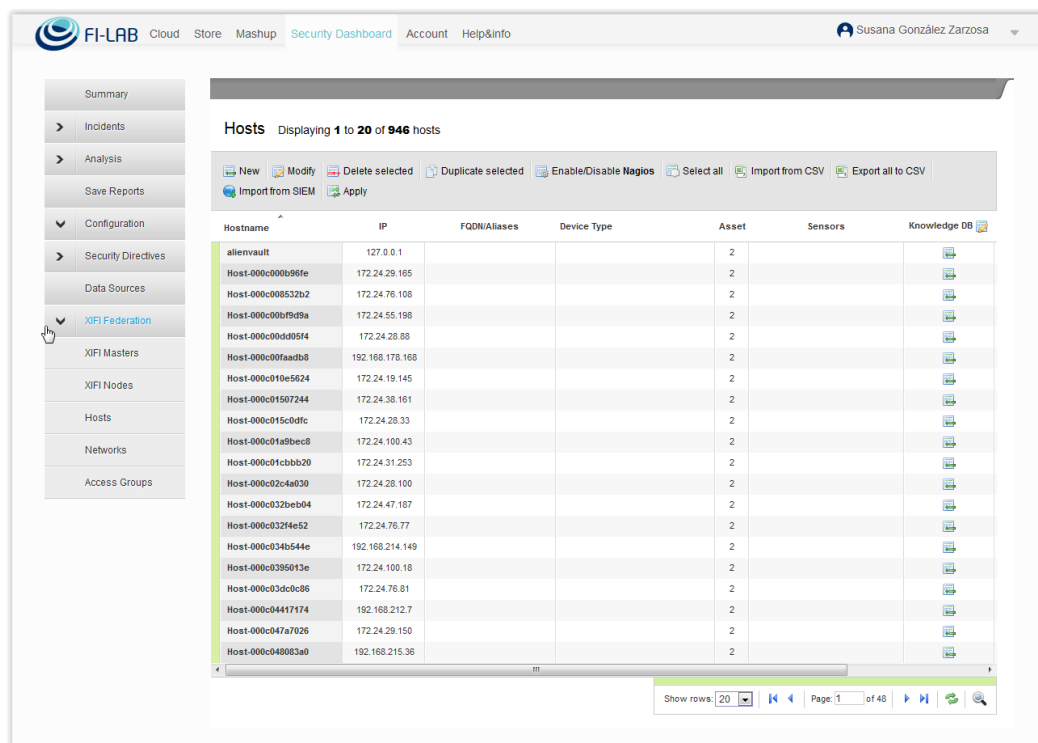
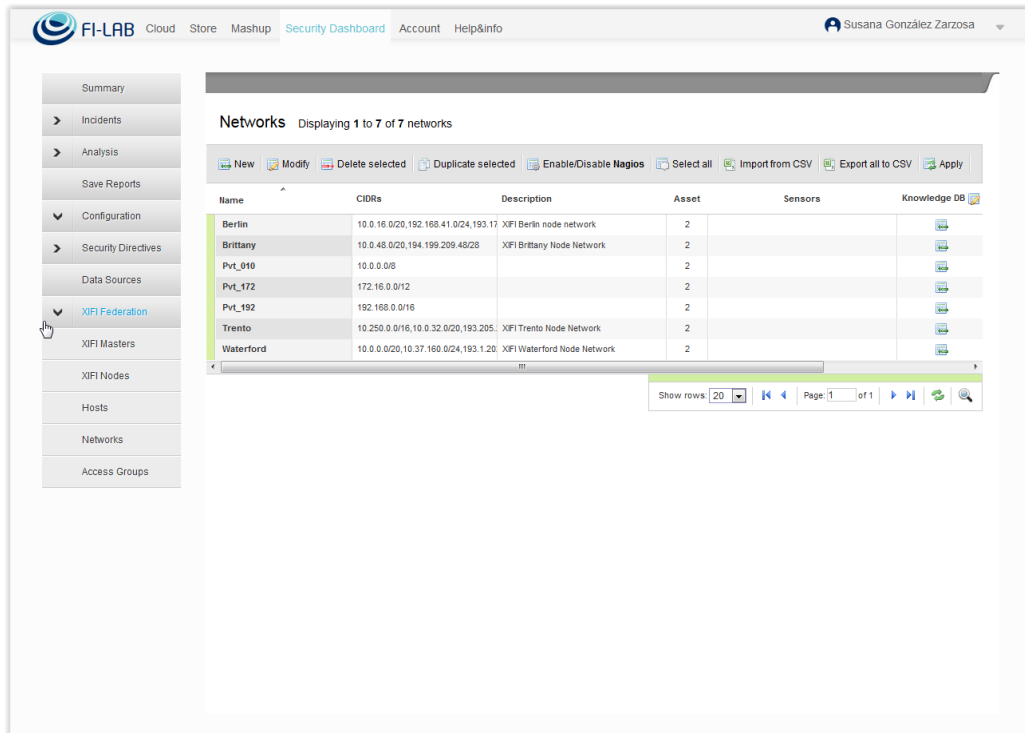


Figure 72: Security Dashboard - show hosts.

- **Networks:** this screen shows the networks defined in the XIFI federation.

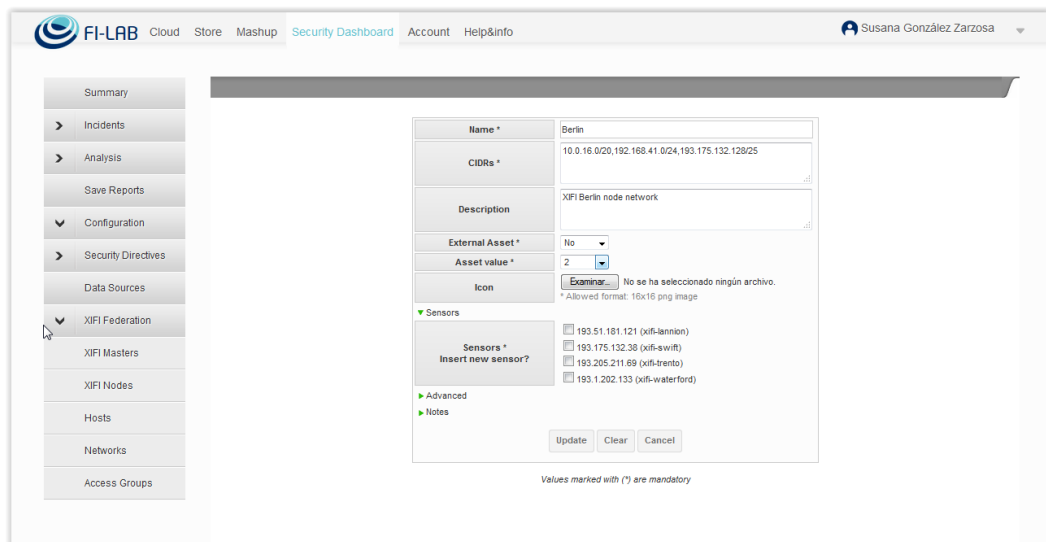


The screenshot shows the Security Dashboard interface. On the left is a navigation menu with options like Summary, Incidents, Analysis, Save Reports, Configuration, Security Directives, Data Sources, XIFI Federation (highlighted), XIFI Masters, XIFI Nodes, Hosts, Networks, and Access Groups. The main content area displays a table of networks with columns for Name, CIDRs, Description, Asset, Sensors, and Knowledge DB. The table lists seven networks: Berlin, Brittany, Pvt_010, Pvt_172, Pvt_192, Trento, and Waterford. Below the table is a pagination control showing 'Page: 1 of 1' and 'Show rows: 20'.

Name	CIDRs	Description	Asset	Sensors	Knowledge DB
Berlin	10.0.16.0/20,192.168.41.0/24,193.17	XIFI Berlin node network	2		
Brittany	10.0.48.0/20,194.199.209.48/28	XIFI Brittany Node Network	2		
Pvt_010	10.0.0.0/8		2		
Pvt_172	172.16.0.0/12		2		
Pvt_192	192.168.0.0/16		2		
Trento	10.250.0.0/16,10.0.32.0/20,193.205.	XIFI Trento Node Network	2		
Waterford	10.0.0.0/20,10.37.160.0/24,193.1.20.	XIFI Waterford Node Network	2		

Figure 73: Security Dashboard - show networks.

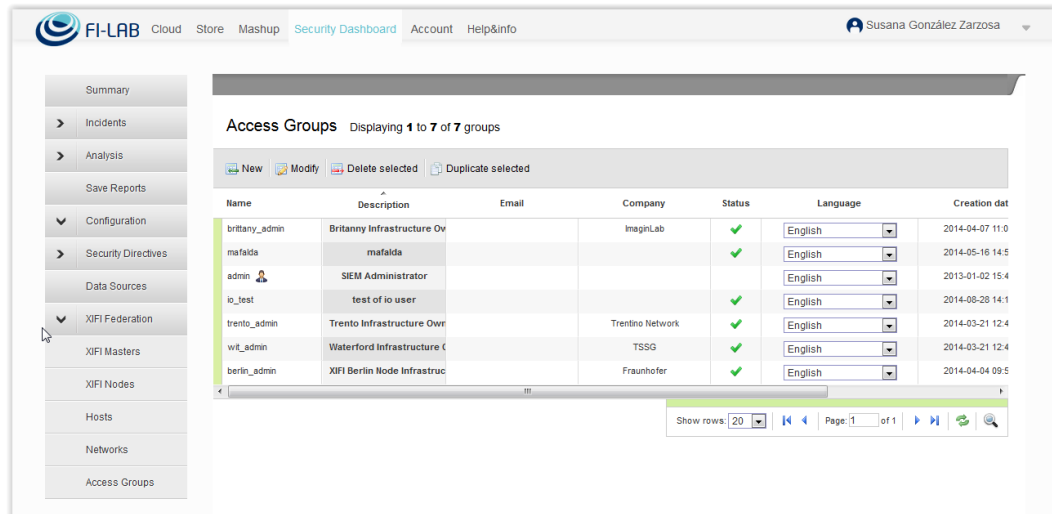
By default, the first time a user belonging to an organization (Infrastructure Owner) registered in the IdM for the application with role *IO* login into the dashboard, the ranges of assigned IP addresses for that organization are automatically recovered from the Federation Manager and a new network for that organization is created in the Security Dashboard. From this option in the menu, the Federation Manager can modify the information associated to these networks or define new ones to be considered in the security monitoring.



The screenshot shows the 'Configure network' form in the Security Dashboard. The form fields are: Name (Berlin), CIDRs (10.0.16.0/20,192.168.41.0/24,193.175.132.128/25), Description (XIFI Berlin node network), External Asset (No), Asset value (2), and Icon (Examinar). There is a section for Sensors with a list of IP addresses and checkboxes: 193.51.181.121 (xifi-lannion), 193.175.132.38 (xifi-swift), 193.205.211.89 (xifi-trento), and 193.1.202.133 (xifi-waterford). There are also sections for Advanced and Notes. At the bottom are 'Update', 'Clear', and 'Cancel' buttons. A note at the bottom states 'Values marked with (*) are mandatory'.

Figure 74: Security Dashboard - configure network.

- **Access Groups:** this screen shows the groups of users (one by each Infrastructure Owner organization apart from the administrator user) registered in the IdM for the Security Dashboard application.



Summary

- Incidents
- Analysis
- Save Reports
- Configuration
- Security Directives
- Data Sources
- XIFI Federation
- XIFI Masters
- XIFI Nodes
- Hosts
- Networks
- Access Groups

Access Groups Displaying 1 to 7 of 7 groups

New Modify Delete selected Duplicate selected

Name	Description	Email	Company	Status	Language	Creation date
brittany_admin	Brittany Infrastructure Ow		ImaginLab	✓	English	2014-04-07 11:0
mafalda	mafalda			✓	English	2014-05-16 14:5
admin	SIEM Administrator				English	2013-01-02 15:4
io_test	test of io user			✓	English	2014-08-28 14:1
trento_admin	Trento Infrastructure Ow		Trentino Network	✓	English	2014-03-21 12:4
wit_admin	Waterford Infrastructure (TSSG	✓	English	2014-03-21 12:4
berlin_admin	XIFI Berlin Node Infrastruc		Fraunhofer	✓	English	2014-04-04 09:5

Show rows: 20 Page: 1 of 1

Figure 75: Security Dashboard - configure access groups.

By default, the first time a user belonging to an organization (Infrastructure Owner) registered in the IdM for the Security Dashboard application with role *IO* login into the dashboard, a new access group is created for that organization. The Security Dashboard associates to that access group the network assigned to that organization (according to the range of IP addresses recovered from the Federation Manager API). In that way, it is ensured that each user (except the Federation Manager) only will have access to the information generated in its own node.

From this menu option, the Federation Manager can edit, modify or delete the information stored in the Security Dashboard for these organizations.

6 INFOGRAPHICS AND STATUS PAGES

6.1 Summary

The Infographics and Status Pages component provides information on the infrastructure capacities and status of FIWARE Lab infrastructure services. The service is mainly intended for Developers and Federation Manager. The Infographics page provides via an infographic the available infrastructure capacities in FIWARE Lab. The Status page provides information on the status of the FIWARE Lab infrastructure services (e.g. Nova, Neutron, Cinder, Glance and others for a given node) and offers Jira support both for a specific node and for general portal issues. While the Status page is a service normally offered by cloud providers and other services providers, the Infographics presents an innovative and intuitive way to publish high-level information on the infrastructure.

The Infographics and Status Pages component is part of the Monitoring GUI architectural component and is integrated into the portal.

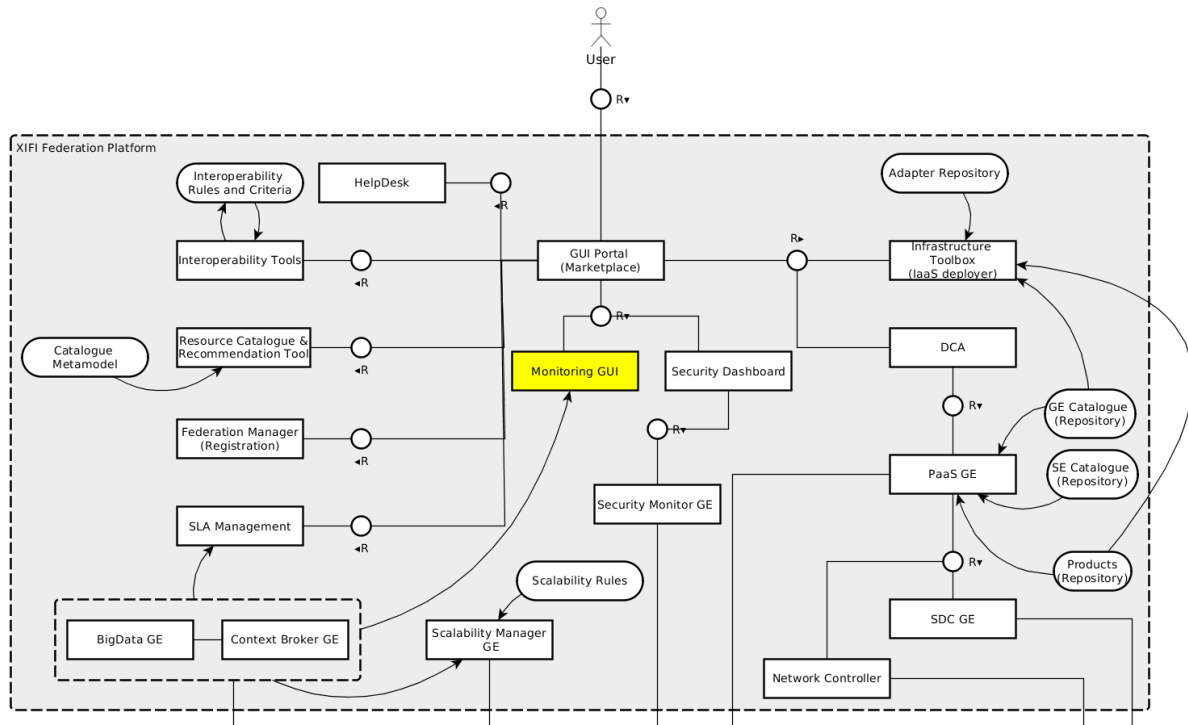


Figure 76: Location of the Infographics and Status Pages in the XIFI Reference Architecture

Reference Scenarios [3]	<ul style="list-style-type: none"> UC-5 -Network and Data Centre operations [73]
Reference Stakeholders [6]	<ul style="list-style-type: none"> Developers: that wants to know about capacities of FIWARE Lab infrastructure and status of FIWARE Lab infrastructure services. Federator: that needs to verify the status of FIWARE Lab infrastructure and status of FIWARE Lab infrastructure services.
Type of ownership	New Development
Original tool	N/A

Planned OS license	Apache License Version 2.0.[7]
Reference OS community	FIWARE / OpenStack

Table 22: Summary Infographics and Status Pages.

Consist of

- FIWARE Lab App Template (section 9)

Depends on

- Federation Monitoring [91]
- Federation Manager [94] (next releases)
- IdM GE [82] (next releases)

6.2 Component leaders

Developer	Email	Company
Katarzyna Di Meo	katarzyna.dimeo@create-net.org	CREATE-NET

Table 23: List of responsible – Infographics and Status Pages.

6.3 Motivation

The Infographics and Status Pages component provides information on the capacities and status of FIWARE Lab infrastructure services. This is a simple but important service to allow users to:

- know in an intuitive way about the infrastructure capacities made available by FIWARE Lab infrastructure;
- monitor current status of infrastructure services and know about any issue in any node of FIWARE Lab infrastructure.

While the information on infrastructure capacities is more related to marketing, the one of services status is extremely important to support Developers and Federation Managers operations. To our best knowledge there is not such service available as opensource for OpenStack related infrastructures, thus this required a new development

6.4 User stories backlog

id	User story name	Actors	Description	Task id
1	Infographics	Developer/ Federation Manager	The user should be able to access an infographic FIWARE Lab capacities accessing a web page with his browser	721
2	# Users registered	Developer/ Federation Manager	The user should be able to know how many users are registered in FIWARE Lab accessing a web page with his browser	722
3	# Regions registered	Developer/ Federation Manager	The user should be able to know how many regions/nodes are available in FIWARE Lab accessing a web page with his browser	723
4	# Cores available	Developer/ Federation Manager	The user should be able to know how many cores are available in FIWARE Lab accessing a web page with his browser	724

id	User story name	Actors	Description	Task id
5	# RAM available	Developer/ Federation Manager	The user should be able to know how much RAM is available in FIWARE Lab accessing a web page with his browser	725
6	# HD available	Developer/ Federation Manager	The user should be able to know how many users are registered in FIWARE Lab accessing a web page with his browser	726
7	# Virtual Machines registered	Developer/ Federation Manager	The user should be able to know how many Virtual Machines registered are registered in FIWARE Lab accessing a web page with his browser	727
8	Infrastructure services status	Developer/ Federation Manager	The user should be able to know the status of FIWARE Lab Infrastructure services accessing a web page with his browser	728
9	Infrastructure status messages	Developer/ Federation Manager	The user should be able to access messages regarding the the status of FIWARE Lab Infrastructure services accessing a web page with his browser	729
10	Infrastructure status messages publication	Federation Manager/ Infrastructure owner	The user should be able to publish a new message regarding the the status of FIWARE Lab Infrastructure services accessing a web page with his browser	730
11	Nova status	Developer/ Federation Manager	The user should be able to know the status of nova services at each node	731
12	Neutron status	Developer/ Federation Manager	The user should be able to know the status of neutron services at each node	732
13	Cinder status	Developer/ Federation Manager	The user should be able to know the status of cinder services at each node	733
14	Glance status	Developer/ Federation Manager	The user should be able to know the status of glance services at each node	734
15	Monitoring status	Developer/ Federation Manager	The user should be able to know the status of monitoring services at each node	735
16	IdM status	Developer/ Federation Manager	The user should be able to know the status of IdM services at each node	736
17	Infographics history	Developer/ Federation Manager	The user should be able to navigate the evolution over time of the infographics	737
18	Status history	Developer/ Federation Manager	The user should be able to navigate the evolution over time of the status	738
19	GE installed	Developer/ Federation Manager	The user should be able to know how many SaaS GEs are installed	739

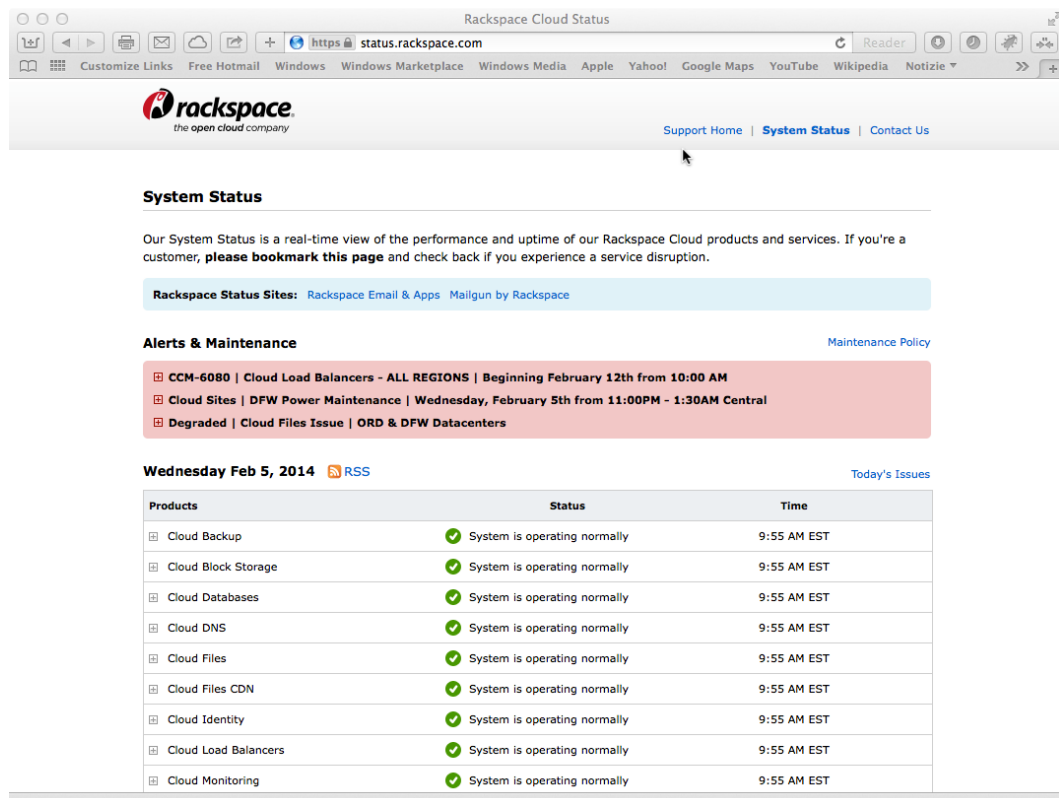
id	User story name	Actors	Description	Task id
20	GE status	Developer/ Federation Manager	The user should be able to know the status of SaaS GEs at each node	740
21	OAuth protocol	Developer/ Federation Manager	The user should be able to access all data coming from Federation Monitoring APIs as before, also if they are protected by OAuth protocol	1064
22	Jira Support	Developer/ Federation Manager	The user should be able to create Jira issues both for a specific node and for general portal issues	1461

Table 24: Infographics and Status Pages - User stories backlog

6.5 State of the art.

The component is quite specific to FIWARE Lab, nevertheless a number of similar solutions are deployed by cloud and services providers such as Rackspace [74] and Amazon [75].

The status board by Rackspace and AWS are quite similar, as showed by the attached screenshots.



System Status

Our System Status is a real-time view of the performance and uptime of our Rackspace Cloud products and services. If you're a customer, **please bookmark this page** and check back if you experience a service disruption.

Rackspace Status Sites: [Rackspace Email & Apps](#) [Mailgun by Rackspace](#)

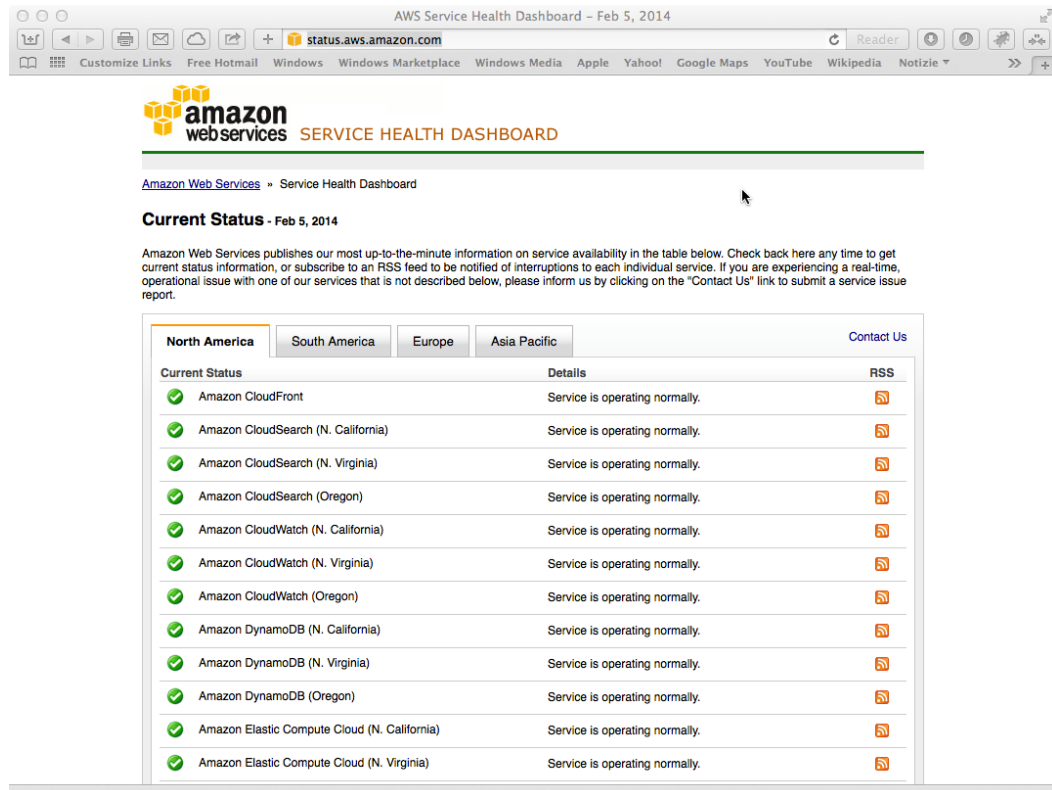
Alerts & Maintenance [Maintenance Policy](#)

- CCM-6080 | Cloud Load Balancers - ALL REGIONS | Beginning February 12th from 10:00 AM
- Cloud Sites | DFW Power Maintenance | Wednesday, February 5th from 11:00PM - 1:30AM Central
- Degraded | Cloud Files Issue | ORD & DFW Datacenters

Wednesday Feb 5, 2014 [RSS](#) [Today's Issues](#)

Products	Status	Time
Cloud Backup	System is operating normally	9:55 AM EST
Cloud Block Storage	System is operating normally	9:55 AM EST
Cloud Databases	System is operating normally	9:55 AM EST
Cloud DNS	System is operating normally	9:55 AM EST
Cloud Files	System is operating normally	9:55 AM EST
Cloud Files CDN	System is operating normally	9:55 AM EST
Cloud Identity	System is operating normally	9:55 AM EST
Cloud Load Balancers	System is operating normally	9:55 AM EST
Cloud Monitoring	System is operating normally	9:55 AM EST

Figure 77: Rackspace status dashboard



AWS Service Health Dashboard – Feb 5, 2014

status.aws.amazon.com

amazon web services SERVICE HEALTH DASHBOARD

Amazon Web Services » Service Health Dashboard

Current Status - Feb 5, 2014

Amazon Web Services publishes our most up-to-the-minute information on service availability in the table below. Check back here any time to get current status information, or subscribe to an RSS feed to be notified of interruptions to each individual service. If you are experiencing a real-time, operational issue with one of our services that is not described below, please inform us by clicking on the "Contact Us" link to submit a service issue report.













Current Status	Details	RSS
✓ Amazon CloudFront	Service is operating normally.	
✓ Amazon CloudSearch (N. California)	Service is operating normally.	
✓ Amazon CloudSearch (N. Virginia)	Service is operating normally.	
✓ Amazon CloudSearch (Oregon)	Service is operating normally.	
✓ Amazon CloudWatch (N. California)	Service is operating normally.	
✓ Amazon CloudWatch (N. Virginia)	Service is operating normally.	
✓ Amazon CloudWatch (Oregon)	Service is operating normally.	
✓ Amazon DynamoDB (N. California)	Service is operating normally.	
✓ Amazon DynamoDB (N. Virginia)	Service is operating normally.	
✓ Amazon DynamoDB (Oregon)	Service is operating normally.	
✓ Amazon Elastic Compute Cloud (N. California)	Service is operating normally.	
✓ Amazon Elastic Compute Cloud (N. Virginia)	Service is operating normally.	

Figure 78: Amazon service health dashboard

The open source solutions include:

- Statshboard [76]
- Overseer [77]

Both solutions are quite similar in term of functionalities. They allow to define a list of services and status. Events define the status of a service. The events are pushed through APIs CRUD operations. These solutions are general purpose and, as such, have some limitations for our case of interest:

- Do not support Regions/Nodes as concept aggregator for services
- Require additional integration with Monitoring data from FIWARE Lab infrastructure

More as mentioned in the motivation chapter, no of the existing solutions provide live information about infrastructure capacity via infographic representation

6.6 Architecture design

The Infographics and Status Pages component, as mentioned in the summary is part of the Portal. Its interaction with other components is represented in the figure below.

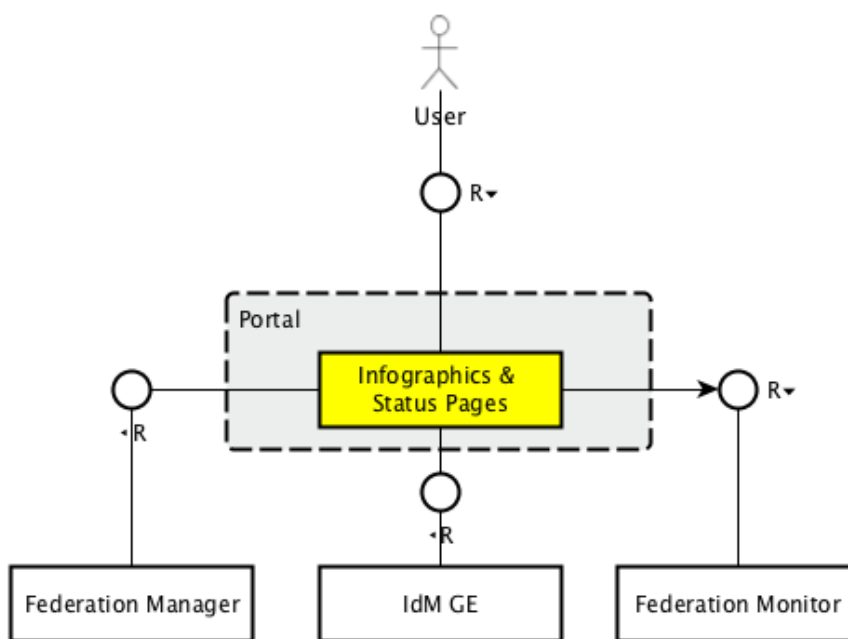


Figure 79: Architecture of the Infographics and Status Pages

The Figure 79 do not represent component of the portal beyond the Infographics and Status Pages component. The component retrieves existing infrastructure services and regions from the Federation Manager. This component provides as well the status information of infrastructure services. The IdM GE provides authentication mechanism to support message posting by the Federation Manager and the Infrastructure Owners. The Federation Monitor component provides data on the capacity and status of FIWARE Lab infrastructure.

6.7 Release plan

Version Id	Milestone	User Stories
1.0	31.12.2013	1,2,3,4,5,6,7
1.1	31.03.2014	8,9,10,11,12,13,14,15,16
1.2	30.06.2014	19,20
1.3	30.09.2014	17,18

Table 25: Release plan – Infographics and Status Pages

6.8 Test cases

Test id	Test description	Test script
1	Test the infographics graphical interface	see description below

Table 26: Test cases – Infographics and Status Pages

Unit Test 1 – Test the infographics graphical interface

Description

Use the dummy federation monitor to test the infographics graphical interface.

Procedure

Configure the end point of the federation monitor in `config/initializers/0infographics.rb` to your local server:

```
require_dependency 'fi_lab_infographics'

FiLabInfographics.setup do |config|
  # Node.js proxy for monitoring
  config.nodejs = 'http://localhost:1339'
end
```

Launch the dummy federation monitor

```
ubuntu@ubuntu:~/fi-lab-infographics$ cd test/dummy_federation_monitor
ubuntu@ubuntu:~/fi-lab-infographics/test/dummy_federation_monitor$ nodejs dummyAPI.js
```

Launch the application

```
ubuntu@ubuntu:~/fi-lab-infographics$ rails server
```

Open your browser at `http://localhost:3000/`

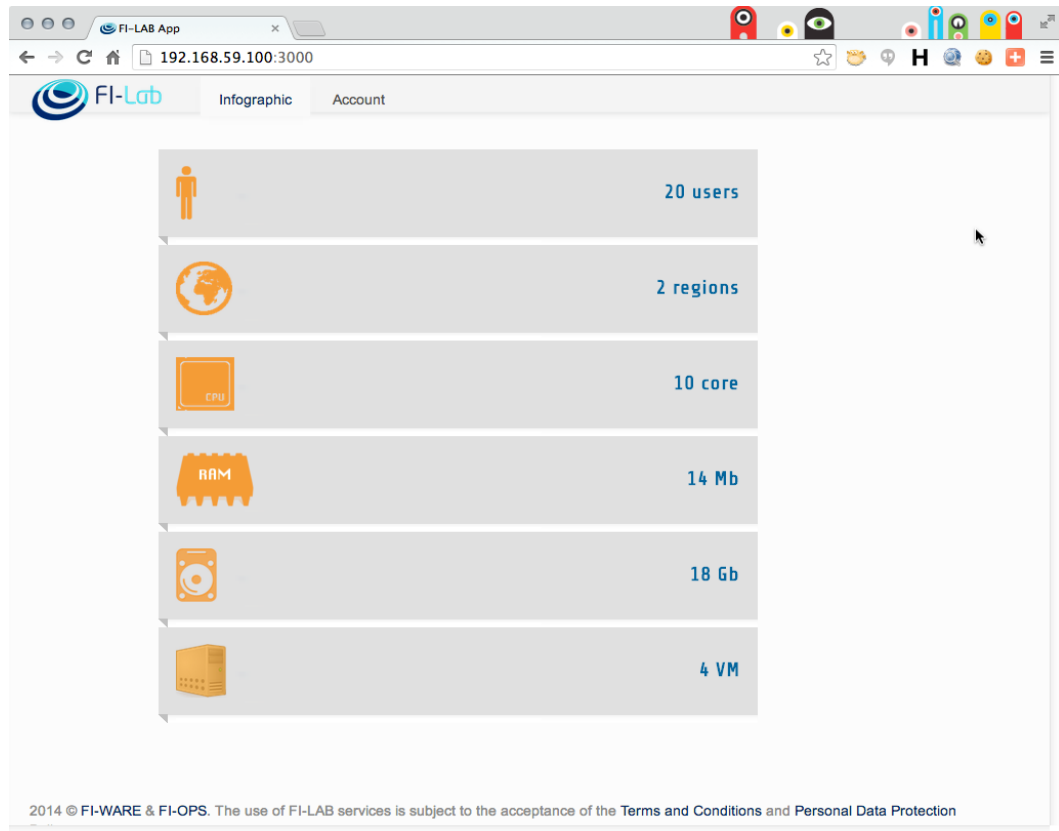


Figure 80: Home Page – the Infographics page.

You should see the above screen.

6.9 Installation Manual

Requirements

- Ubuntu 12.04 as operating system ²

Software repository

The source code can be downloaded from the following repository: <https://github.com/SmartInfrastructures/fi-lab-infographics> [78].

The releases of the Infographics and Status Pages component are available here: <https://github.com/SmartInfrastructures/fi-lab-infographics/releases/> [79].

Setup Guidelines

Install Ruby on Rails

The first step required to install the FIWARE Lab App Template is the installation of Ruby on Rails v2.0.0-p247. The procedure for Ubuntu 12.04 OS is described in the following procedure:

- install developer libraries

```
sudo apt-get install --yes build-essential curl git
```

- install Ruby on Rails v2.0.0-p247

```
ubuntu@ubuntu:~$ curl -L get.rvm.io | bash -s stable --auto
ubuntu@ubuntu:~$ . ~/.bash_profile
ubuntu@ubuntu:~$ rvm requirements
ubuntu@ubuntu:~$ sudo apt-get install build-essential openssl libreadline6 libreadline6-dev \
git-core zlib1g zlib1g-dev libssl-dev libyaml-dev libsqlite3-dev sqlite3 \
libxml2-dev libxslt-dev autoconf libc6-dev ncurses-dev automake libtool bison \
subversion pkg-config
ubuntu@ubuntu:~$ rvm install 2.0.0
ubuntu@ubuntu:~$ rvm use 2.0.0
ubuntu@ubuntu:~$ rvm install ruby-2.0.0-p247
ubuntu@ubuntu:~$ rvm --default use 2.0.0-p247
ubuntu@ubuntu:~$ gem install rails -v 4.0.0
```

- install a javascript runtime (nodejs)

```
ubuntu@ubuntu:~$ sudo apt-get install --yes python-software-properties python g++ make
ubuntu@ubuntu:~$ sudo add-apt-repository --yes ppa:chris-lea/node.js
ubuntu@ubuntu:~$ sudo apt-get update
ubuntu@ubuntu:~$ sudo apt-get install --yes nodejs
```

Install Infographics and Status Pages

Once Ruby on Rails is installed, we can proceed with the download of the code on our system, configure and run the Infographics and Status Pages application.

² it works also on other OS, here we provide only installation instructions for the reference OS in FIWARE Lab servers.

- Download the code

```
ubuntu@ubuntu:~$ git clone git@github.com:SmartInfrastructures/fi-lab-infographics.git
```

- Build the application

```
ubuntu@ubuntu:~$ cd fi-lab-infographics
ubuntu@ubuntu:~/filab-style-app$ bundle install
```

- Configure the end point of the federation monitor in config/initializers/0infographics.rb:

```
require_dependency 'fi_lab_infographics'

FiLabInfographics.setup do |config|
  # Node.js proxy for monitoring
  config.nodejs = 'http://monitoring.fi-xifi.eu:1339'
end
```

- Launch the application

```
rails server
```

For production environments we advice to use Phusion Passenger on Apache [80]

6.10 User Manual

The Infographics and Status Pages component is self-explanatory. The current version includes only the Infographics page and allows the users to browse the following information:

- Users registered in FIWARE Lab
- Regions registered in FIWARE Lab
- Number of cores registered in FIWARE Lab
- Amount of RAM registered in FIWARE Lab
- Amount of HD registered in FIWARE Lab
- Number of Virtual Machines registered in FIWARE Lab

The user can access the detailed information by clicking on each panel, as showed in the figure below for the regions/nodes.

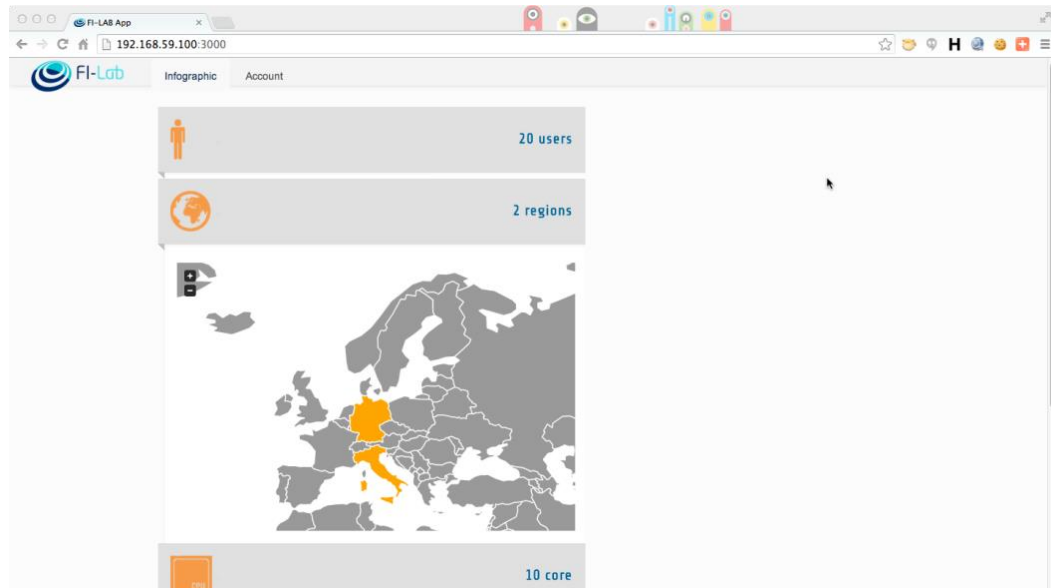


Figure 81: Information region– Infographics and Status Pages.

The other sections contain the detailed information distributed per regions/nodes. For example, expanding the core section, it is possible to see how many cores are available in Berlin.

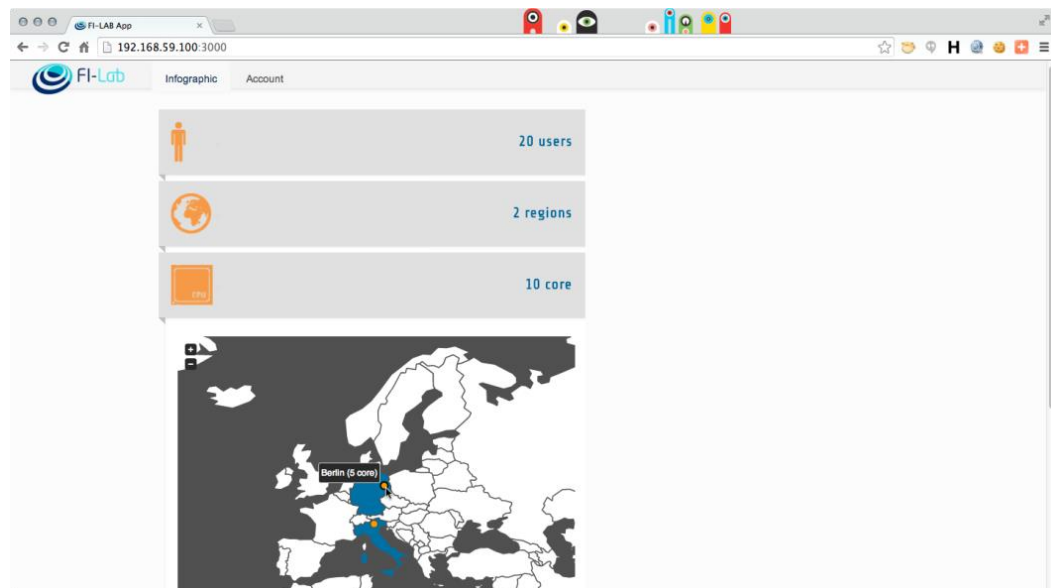


Figure 82: Example cores per node – Infographics and Status Pages.

APIs

This version relies entirely on APIs provided by the Federation Monitor. Next release may include APIs for infrastructure status messaging, in which case they will be documented.

7 SECURITY MONITORING GE

7.1 Summary

According to D1.3 [5] the FIWARE Security Monitoring GE (which includes the component Service Level SIEM) is, working together with Security Probes installed in the slave nodes, responsible for the collection and correlation of security monitoring data. The following figure shows the detailed scheme of the XIFI Reference Architecture and the location of the Security Monitoring GE (see yellow box).

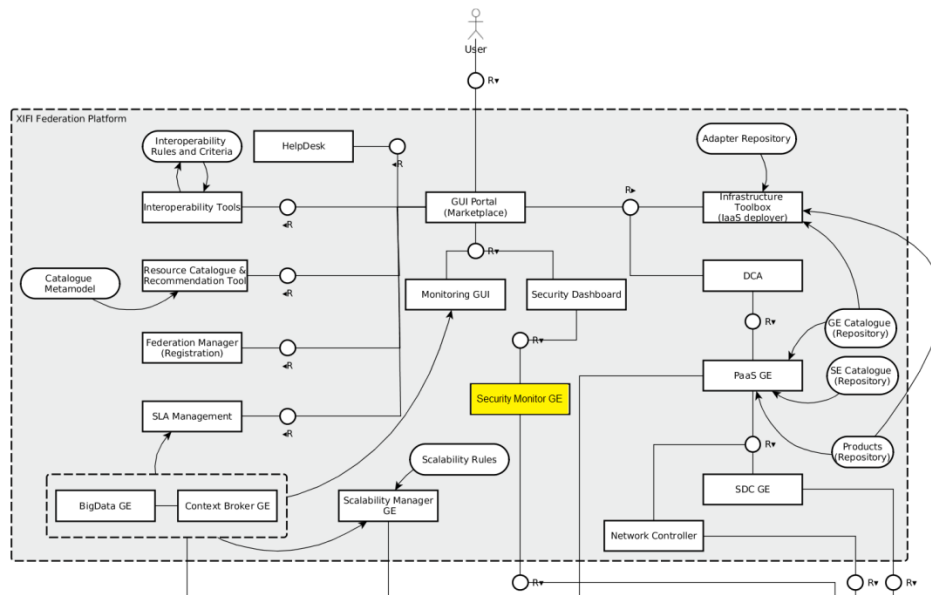


Figure 83: Location of the Security Monitoring GE in the XIFI Reference Architecture

Reference Scenarios [3]	<ul style="list-style-type: none"> • UC-5 -Network and Data Centre operations [73]
Reference Stakeholders [6]	<ul style="list-style-type: none"> • Federation Manager: To monitor security incidents in the XIFI Federation Platform • Infrastructure Owners: To monitor security incidents in its own Node.
Type of ownership	Adaptation & Extension
Original tool	FIWARE Security Monitoring GE [97]
Planned OS license	GPL v3 license (as for the FIWARE Security Monitoring GE Terms & Conditions [98])
Reference OS community	FIWARE Community

Consists of

- Service Level SIEM (Security Information and Event Management): component of the Security Monitoring GE installed in the XIFI Federation Platform
- Security Probes: SIEM agents installed in the Nodes to collect security events and send them to the Service Level SIEM server.

Depends on

The Service Level SIEM component depends on the following open source products:

- OSSIM Open Source SIEM [52]
- Storm [56]
- MySQL [57]

7.2 Component leaders

Developer	Email	Company
Susana González Zarzosa	susana.gzarzosa@atos.net	ATOS
Cristo Reyes	cristo.reyes@atos.net	ATOS

Table 27: List of responsible Security Monitoring GE.

7.3 User stories backlog

The Service Level SIEM component included in the Security Monitoring GE is an extension of the one developed in FIWARE. In here we document only stories related to XIFI development.

id	User story name	Actors	Description	Task id
1	Collection of accountability logs.	Federation Manager / Infrastructure Owner	The user wants to visualize accountability events from the logs generated by the Access Control GE installed there. An additional plugin needs to be implemented to preprocess those logs and adapt the Security Probe to collect and normalize the new data source.	648
2	Install and configure Security Probes with recipes	Infrastructure Owner	Security Probes can be installed and configured using Chef Server recipes.	649
3	Collection of other security events generated in the XIFI Nodes.	Federation Manager / Infrastructure Owner	The user wants to visualize security events generated by other Security Probes installed in the platform. Security Probes can be required to be adapted to collect events from specific data sources installed in the infrastructure.	650
4	Detection of accountability incidents from one node	Infrastructure Owner	The user wants to visualize suspicious access attempts detected with the correlation of events coming from the Access Control GE. The correlation engine needs to be able of processing these events.	651
5	Integration with the Security Dashboard	Federation Manager / Infrastructure Owner	The user visualizes relevant security information (events and incidents) through the Security Dashboard instead of using the Service Level SIEM dashboard.	652
6	User management released to XIFI	Federation Manager / Infrastructure Owner	The user login directly to the Service Level SIEM once logged to the Security Dashboard. An access group will be created automatically for each role IO associated to a different node with the information recovered from the XIFI IdM and Federation Manager APIs.	653

id	User story name	Actors	Description	Task id
7	Detection of security incidents in the XIFI Platform	Federation Manager	The user visualizes security incidents detected correlating events coming from different nodes and data sources in the XIFI Platform.	654
8	Integration with statistics/reporting tools for accountability events	Federation Manager/Infrastructure Owner	The statistics/reporting tools included in the Security Dashboard access to the Service Level SIEM repositories to get information about the accountability events.	655
9	Secure transference of events from slave nodes to master node.	Infrastructure Owner	The user wants the events collected in its infrastructure are sent in a secured way to the SIEM server installed in the master node.. The SIEM agent included in the Security Probe needs to be adapted to have capabilities for using SSL (Security Socket Layer) protocol in the communication.	1520

Table 28: User stories backlog - Security Monitoring GE

7.4 Architecture design

In the architecture we can differentiate between the Security Probes installed in the Infrastructures (slave nodes) and the Service Level SIEM component of the Security Monitoring GE installed in the XIFI Federation Platform (master node). See the overall XIFI architecture diagram in the deliverable D1.3 [5].

The *Service Level SIEM server* is the component which receives the events already normalized coming from the different slave Nodes in the XIFI Platform, stores them in a MySQL database and performs its correlation to show the user through the Security Dashboard interface the relevant security events and incidents detected. In the master node it will be also installed a Security Probe (SIEM Agent and plugins) to monitor it. More detailed information about the Service Level SIEM architecture can be found in the FIWARE Service Level SIEM Open Specifications [58].

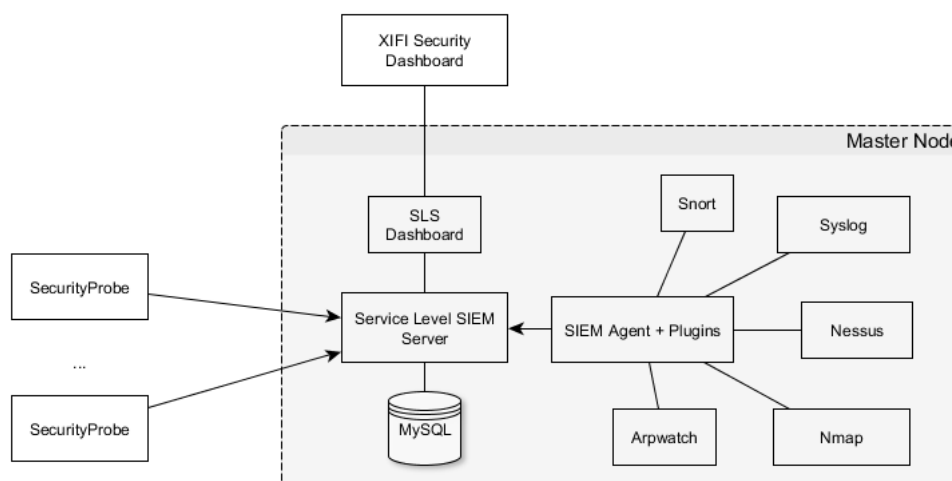


Figure 84: Service Level SIEM Server Architecture

The *Security Probes* can be installed in each slave Node:

- At the level of the entire node in a specific server, to collect accountability events generated by the Access Control GE and events generated by network devices such as firewalls installed in the infrastructure. This server will have installed a syslog server so the events can be sent from the different sources. Each VM deployed will be preconfigured to send all the logs through syslog to this Security Probe in order to be monitored;
- Inside each virtualization server in the node, to collect events from different data sources depending on the plugins activated (e.g. syslog, nagios, snort...).

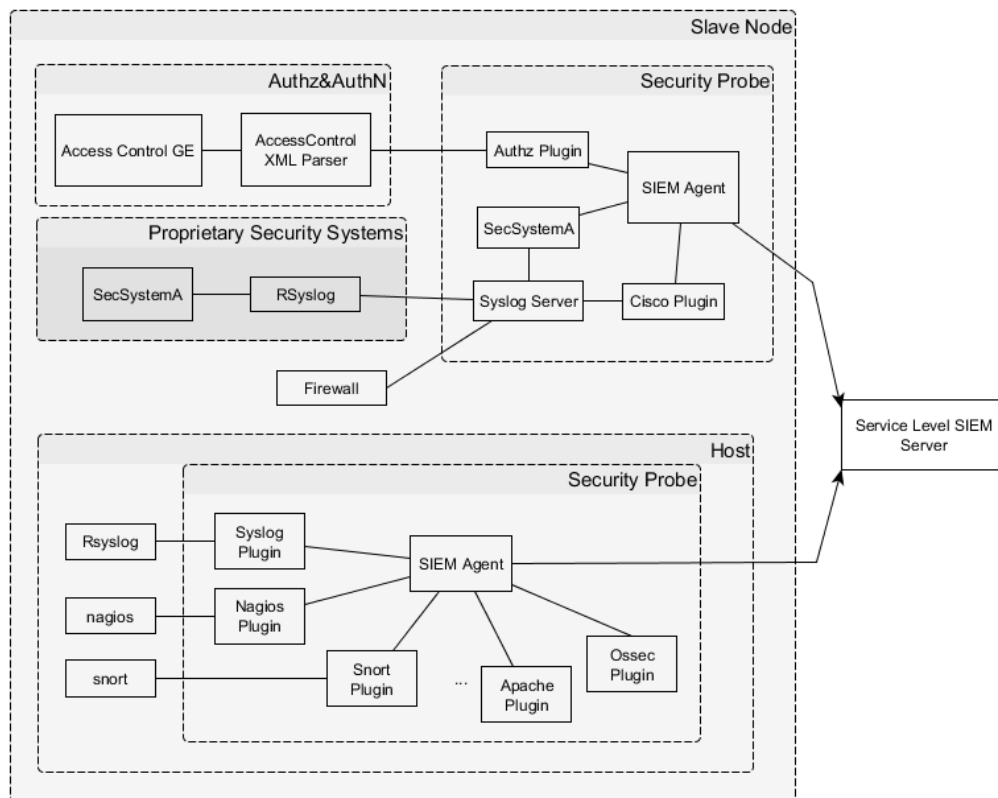


Figure 85: Security Probe Architecture

NOTE: In case of data sources generating logs in a format not compatible with the plugins included in the Security Probes, it is needed to include an intermediary process to do the parsing from the original log format to a compatible log. For example, this is the case of the logs generated by the Access Control GE because they include XMLs with the XACML requests/response which cannot be processed and normalized directly by the Security Probes.

7.5 Release plan

Version Id	Milestone	User Stories
0.1	30.12.2013	1
0.2	31.01.2014	4
0.3	28.02.2014	9
1.0	31.03.2014	5
1.1	30.08.2014	6
2.0	31.09.2014	7
0.1	30.12.2013	1

Table 29: Release plan - Security Monitoring GE

7.6 Test cases

Test id	Test description	Test script
1	Verification of the Security Probe installation	see below
2	Verification of AccessControl parser installation	see below
3	Verification of Service Level SIEM installation	see below
4	Verification of connectivity between SecurityProbes and Service Level SIEM server	see below

Table 30: Test cases - Security Monitoring GE

Unit Test 1 – Verification of the Security Probe installation

Description

The Security Probes are responsible for collecting data from the nodes and normalize them to security events into ossim format. In this test, we assume that the Security Probe has been installed and it is running in a host and we are going to check the status of the ossim-agent service.

Procedure

1. Check the script python called ossim-agent is running as a daemon:

```
root@ubuntu12:/home/xifi# ps -ef|grep ossim-agent
root 14574 1 7 18:11 ? 00:05:20 /usr/bin/python -OOt /usr/bin/ossim-agent -d
```

If there is no response, this means that the SIEM agent is not running and you must start the service manually by typing:

```
root@ubuntu12:/home/xifi# service ossim-agent start
```

2. Check the data sources to generate the events to be collected by the Security Probes are running. They depend on the node and the security monitoring to be performed. For example:

```
root@ubuntu12:/home/xifi# service rsyslog status
rsyslog start/running, process 1513
root@ubuntu12:/home/xifi# service snort status
* Status of snort daemon(s) [ OK ]
```

Unit Test 2 – Verification of the AccessControl parser installation

Description

The AccessControl parser needs to be running in the host where the Access Control GE is installed in order to generate a log compatible with the ossim-agent.

Procedure

1. Check a java process *AccessControlXMLParser.jar* is running:

```
root@ubuntu12:/home/xifi# ps -ef|grep AccessControlXMLParser.jar
root  16159  1 0 12:57 pts/0    00:00:00 sudo /usr/bin/java -jar
AccessControlXMLParser.jar /home/xifi/SP/pluginAccessControl
/authzforce_access.log /var/log/authzforce_access.log
root  16160 16159 0 12:57 pts/0    00:00:30 /usr/bin/java -jar
AccessControlXMLParser.jar /home/xifi/SP/pluginAccessControl
/authzforce_access.log /var/log/authzforce_access.log
```

2. Check the output file (e.g. */var/log/authzforce_access.log*) is being updated:

```
root@ubuntu12:/home/xifi# tail -f /var/log/authzforce_access.log
2014-01-15 12:13:16,396|7|10.0.2.15|28789263-ed04-4d08-
bb42-35ea4950fc79|POST|domains/71903fd2-2b6f-11e3-91a6-d337f03a31f7
/pdp|200|bjensen|null|null|null|jsmithshare/containers
/jsmithshare|PUT|null|Permit
...
```

Unit Test 3 – Verification of the Service Level SIEM installation

Description

The Service Level SIEM server must be installed at the XIFI Federation level to receive the events coming from the different Security Probes distributed through the Nodes.

Procedure

You can follow the Sanity Check Procedures [59] included in the FIWARE Service Level SIEM Installation and Administration Guide to verify the server installation is ready.

Unit Test 4 – Verification of the connectivity between SecurityProbes and Service Level SIEM server.

Description

The ossim-agent process installed with the Security Probes is the responsible of connecting to the ossim-server in order to send the events collected and normalized in the Nodes.

Procedure

1. Check the connections with the Service Level SIEM server ip (defined in the /etc/ossim/agent/config.cfg). By default, the connection between the ossim-agent and the ossim-server is done through the port 40001/tcp..

```

root@ubuntu12:/home/xifi# netstat -an |grep 192.168.215.118
tcp    0    0 10.0.2.15:50395    192.168.215.118:40002 ESTABLISHED
tcp    0    0 10.0.2.15:36285    192.168.215.118:40001 ESTABLISHED
tcp    0    0 10.0.2.15:36285    192.168.215.118:41000 ESTABLISHED
root@ubuntu12:/home/xifi#

```

7.7 Installation Manual

Introduction

The installation and administration of the Service Level SIEM component included in the Security Monitoring GE is described in FIWARE Security Monitoring Installation and Administration Guide [60].

Note that it is available in the FIWARE Cloud Portal [99] a blueprint template called *SecMon-SLS* to automatize and facilitate the Service Level SIEM component installation procedure.

In this manual we will focus on the installation and configuration of the Security Probes in each of the Nodes from the scratch and how to configure the Service Level SIEM server to work in the XIFI Federation Platform.

Software repository

- Security Probes:

The source code files to be used to install the Security Probes can be downloaded from:

<https://xifisvn.res.eng.it/wp4/SecurityDashboard/trunk/SecurityProbes>

The parser to make the logs generated by the Access Control GE compatible with the Security Probes is available from:

<https://xifisvn.res.eng.it/wp4/SecurityDashboard/trunk/SecurityProbes/accesscontrol-sls-parser/bin>

- Service Level SIEM:

All the required files used for the installation of the Service Level SIEM are included in the ServiceLevelSIEM-3.3.3.tar.gz file that can be downloaded from:

<https://xifisvn.res.eng.it/wp4/SecurityDashboard/trunk/ServiceLevelSIEM/bin>

Security Probes Installation

In the Nodes, only the Security Probes must be installed and configured. They must be installed in each of the host/VM/device that needs to be monitored in the infrastructure.

The Security Probes are composed by:

- A **SIEM Agent** responsible for collecting all the data generated by the different data sources and sends it to the server in a standardized way. Usually only one agent per machine is enough, but several could be installed if necessary each of them collecting from different data sources and sending to different servers. This agent is a modified version of the ossim-agent included in OSSIM.

- **Plugins** in charge of reading from the logs generated by the data sources considered in each host/VM/device monitored and standardize them so that the Agent can send them to the server. A set of these plugins is provided with the Security Probe package including the most common data sources, such as syslog, snare, cisco, apache, snort, nagios, nessus, etc. More information about the creation of new plugins can be found in the Security Monitoring - Service Level SIEM User and Programmers Guide [61].

Prerequisites

We are assuming that the Security Probes are deployed on the nodes with Ubuntu Server 12.04 LTS operating system installed. The following commands must be executed to ensure all the dependencies are installed:

```
# apt-get install python-dev geoip-bin geoip-database
# wget http://geolite.maxmind.com/download/geoip/api/c/GeoIP-1.4.7.tar.gz
# tar -xvzf GeoIP-1.4.7.tar.gz
# cd GeoIP-1.4.7
# ./configure --prefix=/usr
# make
# make install
# cd ..
# wget http://geolite.maxmind.com/download/geoip/api/python/GeoIP-Python-1.2.7.tar.gz
# tar -zxvf GeoIP-Python-1.2.7.tar.gz
# cd GeoIP-Python-1.2.7
# python setup.py build
# python setup.py install
# cd ..
# wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
-O /tmp/GeoLiteCity.dat.gz
# mkdir -p /usr/share/geoip
# gunzip -c /tmp/GeoLiteCity.dat.gz > /usr/share/geoip/GeoLiteCity.dat
# apt-get install python-pyinotify python-setuptools
# easy_install pytz
# apt-get install python-nmap nmap python-ldap python-libpcap python-adodb
# apt-get install openjdk-6-jdk
# apt-get install monit
```

Installation steps:

To install the Security Probe in a host it is necessary:

1. Install the SIEM agent and plugins:

Download the Security Probe source code and execute:

```
# python setup.py install --prefix=/usr
```

NOTE: The option `--help-commands` can be used to get all the available options.

2. Configure the SIEM Agent:

Once installed, the file `/etc/ossim/agent/config.cfg` must be edited to:

- check the `[control-framework]` section to define the ID for the agent (to be identified from the Framework the source of the events) and the ip where the control framework is the XIFI Federation Platform.

```
[control-framework]
enable=True
id=<node_id>
ip=<server_ip>
port=40003
```

- check the *[output-server]* section: it enables to send the events to the SIEM server installed on the master node in the XIFI Federation Platform, and its ip and port must be configured..

```
[output-server]
enable=True
ip=<server_ip>
port=40001
ssl_port=50001
send_events=True
ssl=True
cert=/etc/ossim/agent/slsserver.cert
```

NOTE: The SSL certificate file must be provided by the master node

- check the *[output-storm]* section: it enables to send the events to the Storm cluster installed with the Service Level SIEM on the master node for their correlation in the XIFI Federation Platform, and its ip and port must be configured.

```
[output-storm]
enable=True
ip=<server_ip>
port=41000
send_events=True
ssl=True
cert=/etc/ossim/agent/slsserver.cert
```

- check the *[plugins]* section: it defines the plugins to be considered by the agent. It is necessary to include a line by each plugin with the format *<plugin_name>=<plugin:location>*. The plugins required for the security directives applied in XIFI are already included by default.

```
[plugins]
nmap-monitor=/etc/ossim/agent/plugins/nmap-monitor.cfg
ntop-monitor=/etc/ossim/agent/plugins/ntop-monitor.cfg
ossim-monitor=/etc/ossim/agent/plugins/ossim-monitor.cfg
ping-monitor=/etc/ossim/agent/plugins/ping-monitor.cfg
snare=/etc/ossim/agent/plugins/snare.cfg
whois-monitor=/etc/ossim/agent/plugins/whois-monitor.cfg
wmi-monitor=/etc/ossim/agent/plugins/wmi-monitor.cfg
fiware=/etc/ossim/agent/plugins/fiware.cfg
authz=/etc/ossim/agent/plugins/authz.cfg
cisco-asa=/etc/ossim/agent/plugins/cisco-asa.cfg
snort_syslog=/etc/ossim/agent/plugins/snort_syslog.cfg
```

Finally, the *[plugins-default]* section can be used to override the default values and indicate e.g. a specific ip or interface for the sensor (instead of the one recovered by default with the eth0).

```
[plugin-defaults]
interface=eth1
override_sensor=True
sensor=<sensor_ip>
tzone=Europe/Dublin
```


3. Configure the data sources and plugins:

Each data source to be monitored requires a plugin to collect and normalize its logs. These plugins are installed under the folder `/etc/ossim/agent/plugins`. So, if for example a data source generates the logs in a location different from the default value, the file `<plugin>.cfg` associated to it should be edited and the parameter location changed with the right location of the log.

The activation/deactivation of the plugins is done through the section `[plugins]` included in the `/etc/ossim/agent/config.cfg` file.

Some of the data sources that could be installed with the Security Probes in order to generate information relevant for the security monitoring of a node are:

- pads - Passive Asset Detection System
- tcptrack - Monitor TCP connections on the network
- snort - Open source network intrusion detection system
- openVAS-Client - the client part of the OpenVAS Security Scanner
- nagios3 - Network/systems status monitoring daemon
- ntop – Network usage monitor
- arpwatc - Monitor ethernet/ip address pairings
- p0f - Identify remote systems passively

They can be installed with this command:

```
apt-get install snort rsyslog openvas-client nagios3 tcptrack ntop pads arpwatc p0f
```

4. Start the SIEM Agent:

The agent is installed as a service in the host so you can execute:

```
# /etc/init.d/ossim-agent start
```

5. Add the ossim-agent daemon to the services in the machine so it can be automatically started:

```
# update-rc.d ossim-agent start 35 3 4 5 .
```

To ensure the `ossim-agent` service is always running, add it to the services monitored with the `monit` tool. To do this, first check the following lines are included in the file `/etc/monit/monitrc`:

```
set daemon 180 with start delay 300
set logfile /var/log/ossim/monit.log
...
include /etc/monit/ossim/*
```

Then, create the folder `/etc/monit/ossim` with a file called `xifisp.monitrc` including:

```
#####
# SIEM Agent and security probe processes
#####
check process ossim-agent with pidfile /var/run/ossim-agent.pid
group agent
start program = "/etc/init.d/ossim-agent start"
```

```
stop program = "/etc/init.d/ossim-agent stop"
if totalmem > 90% for 2 cycles then restart
if 20 restart within 20 cycles then alert
```

Finally, restart the monit service

```
# service monit restart
```

6. Start additional parsers:

It is important to remark that when the data sources generate logs with a format not compatible to be processed by the plugins included in the Security Probes, it is needed to add an intermediary process which reads the original log and generates a compatible log.

One example is the case of the Access Control GE logs which add complete XMLs with the XACML requests/response inside. A process called *AccessControlXMLParser* has been developed with this purpose. To install it, it is required to copy in each Node the file *AccessControlXMLParser.jar* in the same host where the Access Control GE is running and to export the variable `JAVA_HOME`.

A script `startAccessControlXMLParser.sh` is provided in the test folder to run it:

```
#export JAVA_HOME=/usr
#./startAccessControlXMLParser.sh <origin_file> <destination_file>
```

Where `<origin_file>` is the log generated by the Access Control GE (e.g. `/opt/glassfish-3.1.2.2/glassfish/domains/domain1/logs/authzforce_error.log`) and `<destination_file>` is the file read by the Authz plugin (by default: `/var/log/authzforce_access.log`). The parameter `<outputServer>` is optional and should be used when the Access Control GE and the Security Probe are installed in different hosts. In that case, `<outputServer>` is the IP address of the host where the Security Probe is and the log will be sent to the port where the syslog service is listening there (by default 514/udp).

For the connection between Security Probes and the Service Level SIEM server installed in the XIFI Federation Platform, the following ports must be open:

- 50001/tcp and 41000/tcp: ports where the SIEM server is listening for incoming events sent using SSL protocol
- 40001/tcp, 40002/tcp and 40003/tcp: ports used by the OSSIM server

More details about the event collection performed by the Service Level SIEM can be found in the Open Specifications [58].

Service Level SIEM Installation

Prerequisites

The installation of the Service Level SIEM in the XIFI Platform requires

- Operating System and OSSIM

OSSIM is distributed as a standalone Debian based Operating System. It must be previously installed and configured in the machine where the Service Level SIEM is going to be running. Detailed instructions about the installation can be found in the OSSIM official website (<http://www.ossim.net>).

- Java VM

The Service Level SIEM requires Java release 1.6 or greater installed and the `JAVA_HOME` variable set in the user profile. In particular, the `jdk1.6.0_37` has been used in the tests.

- Database

The Service Level SIEM requires a MySQL database. In particular, the v5.5 is provided with the OSSIM distribution.

- Storm Cluster

Storm 0.9.0-wip16 has been used in the Service Level SIEM. It is a free and open source project licensed under the Eclipse Public License (EPL) that can be downloaded from <http://storm-project.net/downloads.html>

Storm has the following requirements

- Apache ZooKeeper Server Package v3.4.5 (<http://zookeeper.apache.org/>)
- ZeroMQ v2.1.7 (<http://www.zeromq.org/>)

If you want to have a supervisory process that manages each of your ZooKeeper/Storm processes, it is required a supervisory process such as Daemontools (<http://cr.yp.to/daemontools.html>).

Installation steps

- The first step is to install the Service Level SIEM component of the Security Monitoring GE in the master node of the XIFI Federation following the instructions included in the FIWARE wiki [62].
- Once installed, it is required to perform some configuration changes in the default OSSIM and Storm configuration files to adapt them to the security monitoring of the XIFI Federation Platform. This configuration should be performed by the Federation Manager because he/she is the one with administration rights and full access to the system. The following configuration files need to be checked:

- `/etc/ossim/ossim_setup.conf`

The main blocks to be checked here are: [database] (with the information to access the MySQL database), [ssl] (with the location of the SSL certificate generated in the server), [sensor] and [server] (with the plugins to be considered in the server). Note that the SSL certificate used in the SIEM server and defined in this file needs to be shared to the Security Probes and configured in the `/etc/ossim/agent/config.cfg`.

- `/etc/ossim/framework/ossim.conf`

This file includes the location of the OSSIM web interface and the data to access the OSSIM database and the Snort database (if installed snort).

- `/etc/ossim/idm/config.xml` and `/etc/ossim/server/config.xml`

These files include the access to the databases

- `/opt/storm/conf/ServiceLevelSIEM.conf`

Mainly, it is necessary to check the database parameters and the SSL configuration are correct.

- To start the required services, check the following variables have been added to the user profile.

```
export JAVA_HOME=<path>/jdk1.6.0_37
export STORM_HOME=/opt/storm
export PATH=$PATH:$STORM_HOME/bin:$JAVA_HOME/bin:/opt/zookeeper
```

And execute:

```
# sudo service ossim-server start
# sudo service ossim-socat start
# sudo service ossim-framework start
# sudo service alienvault-idm start
# sudo service zookeeper start
# sudo service storm-nimbus start
# sudo service storm-supervisor start
# cd /opt/storm/admin
# ./startSLSTopology.sh
```

Install the Security Dashboard in order to manage the configuration of the security monitoring through a web graphical interface. Follow the instructions in the Security Dashboard Installation Manual section 5.9.

Finally, the Federation Manager can configure through the Security Dashboard (see more details in the Security Dashboard section 5):

- Configuration of data sources

The Federation Manager can add, modify or delete the data sources to be considered in the security monitoring and the event types of each one.

- Configuration of security probes

The security events will be collected by Security Probes installed on the different slave nodes. When a new Security Probe is started, the server will detect it and will show a warning in the Security Dashboard (under Configuration -> XIFI Federation -> XIFI Nodes). The Federation Manager will be in charge of accepting the new Security Probe in order to add it to the SIEM server for its monitoring.

- Configuration of assets

The Federation Manager can define and modify information about hosts, hosts groups, networks, network groups, ports or port groups considered in the security monitoring. This information includes the called Asset value that is used when the risk value of an event or alarm is calculated. The formula used is:

$$\text{Risk} = (\text{priority} * \text{reliability} * \text{asset_value}) / 25$$

- Configuration of access groups

In XIFI, the user authentication is delegated to the Federated Identity Management and based on the roles defined for the Security Dashboard application. By default, a new access group will be automatically created for each new Infrastructure Owner organization. The name of the access group will be the name of the organization (as registered in the IdM for the role IO of the Security Dashboard). That access group will have a network associated with the range of IP addresses assigned to that organization and recovered from the Federation Manager API.

The Federation Manager can edit, modify or delete the information stored in the Security Dashboard for these Infrastructure Owners (e.g. email, department, language...).

- Configuration of security directives

The Federation Manager can define different security filtering policies and rules to be applied in the SIEM for the detection of attacks and security incidents. See some examples in the User Manual section.

7.8 User Manual

API Specification

The specification for the collection of security events to be processed by the Service Level SIEM is the one defined in the FIWARE Open API Specification [66].

Handbook

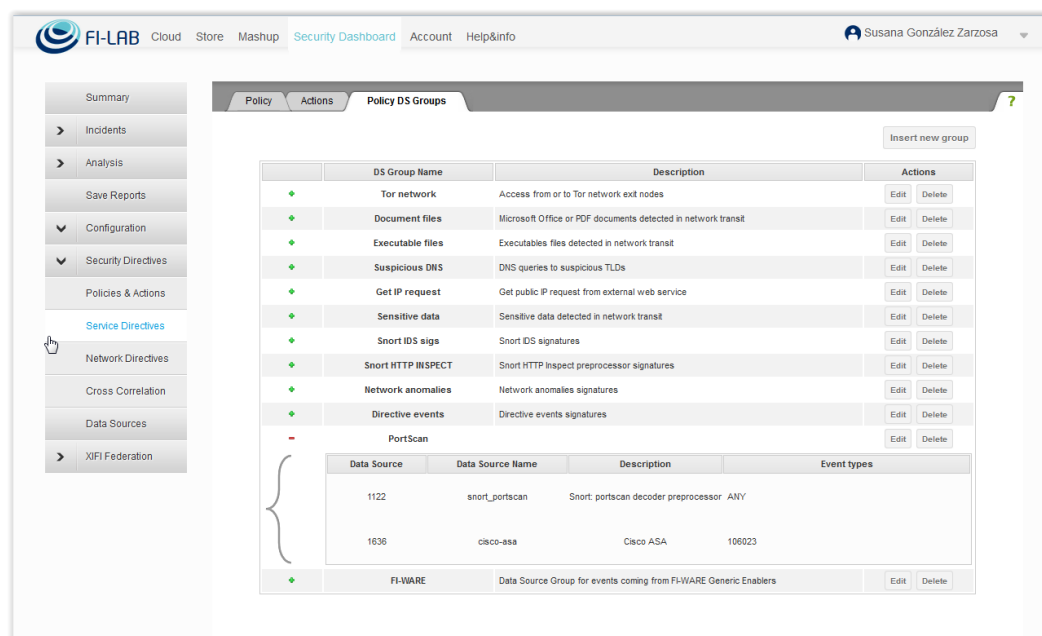
The handbook is associated to the Security Dashboard. In the Security Dashboard User Manual (section 5.10) it is shown how to manage the configuration of the security monitoring through the graphical user interface integrated in the XIFI Portal.

The user guide for the Service Level SIEM component included in the Security Monitoring GE is described in detail in Service Level SIEM User and Programmers Guide [61]. In this guide it is explained and illustrated with examples the different types of security directives that the Federation Manager can apply in the federation using the SIEM server, which include:

- **How to define policies and actions.**

This option is located in the Security Dashboard graphical interface under *Configuration -> Security Directives -> Policies&Actions*.

In the example included in the Service Level SIEM User Guide, it is first defined a data source group with events related to port scanning attempts.



DS Group Name	Description	Actions
Tor network	Access from or to Tor network ext nodes	Edit Delete
Document files	Microsoft Office or PDF documents detected in network transit	Edit Delete
Executable files	Executables files detected in network transit	Edit Delete
Suspicious DNS	DNS queries to suspicious TLDs	Edit Delete
Get IP request	Get public IP request from external web service	Edit Delete
Sensitive data	Sensitive data detected in network transit	Edit Delete
Snort IDS sigs	Snort IDS signatures	Edit Delete
Snort HTTP INSPECT	Snort HTTP inspect preprocessor signatures	Edit Delete
Network anomalies	Network anomalies signatures	Edit Delete
Directive events	Directive events signatures	Edit Delete
PortScan		Edit Delete

Data Source	Data Source Name	Description	Event types
1122	snort_portscan	Snort: portscan decoder preprocessor ANY	
1636	cisco-asa	Cisco ASA	106023

Figure 86: Security Monitoring - Data Source Group

After that, a filtering rule is created so that when it is detected an event in that data source group, a ticket with the incident is opened and assigned to the administrator user.

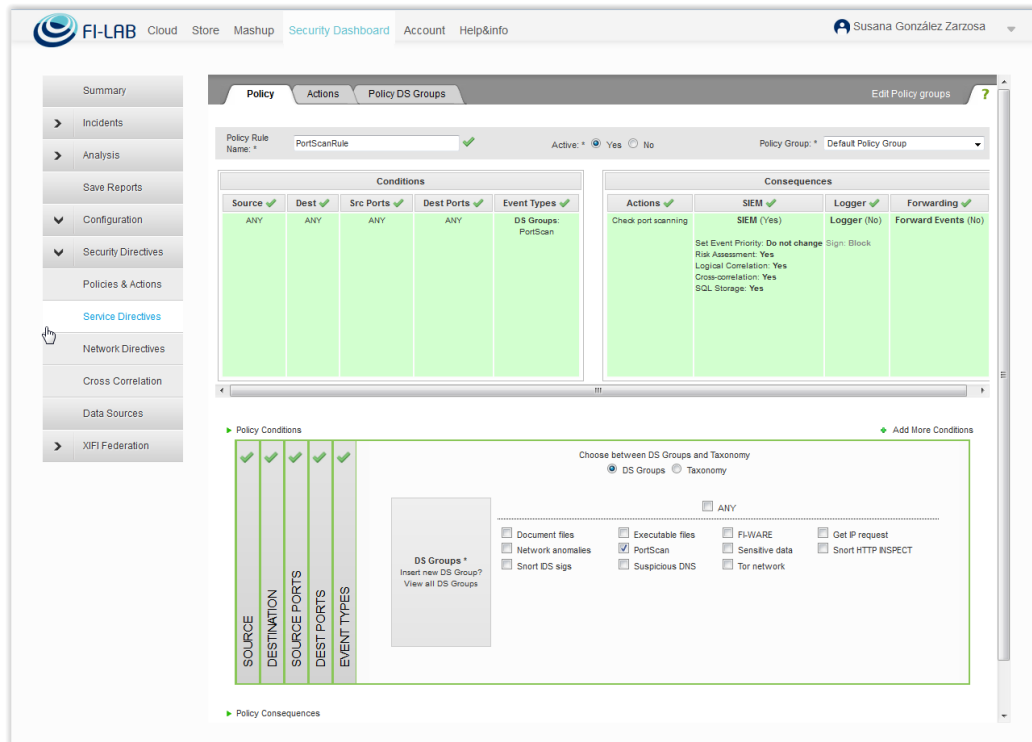


Figure 87: Security Monitoring - New policy

- **How to configure service directives.**

This is the most flexible and complex set of directives that can be defined in the SIEM to generate alarms also from a service perspective. These directives are defined in Event Processing Language (EPL), which is a SQL-like language that allows for example the detection of patterns, the definition of data windows or the aggregation and filtering of incoming events into more complex events.

This option is located in the Security Dashboard graphical interface under *Configuration -> Security Directives -> Service Directives*.

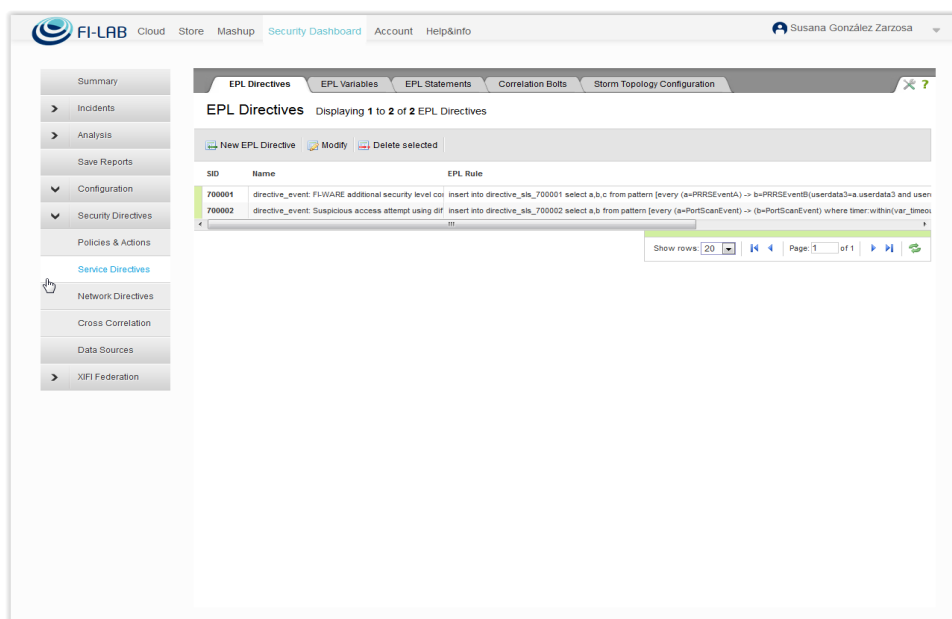


Figure 88: Security Monitoring - Service Directives

- **How to configure network directives.**

In particular, it is explained how to create the XML directive that could be used by the OSSIM core engine for the detection of one of the most common security attacks that could be detected in an infrastructure: a brute force login attack. Or in other words, how to detect an intruder who wants to get the access to a server trying consecutively with different passwords.

This option is located in the Security Dashboard graphical interface under *Configuration -> Security Directives -> Network Directives*.

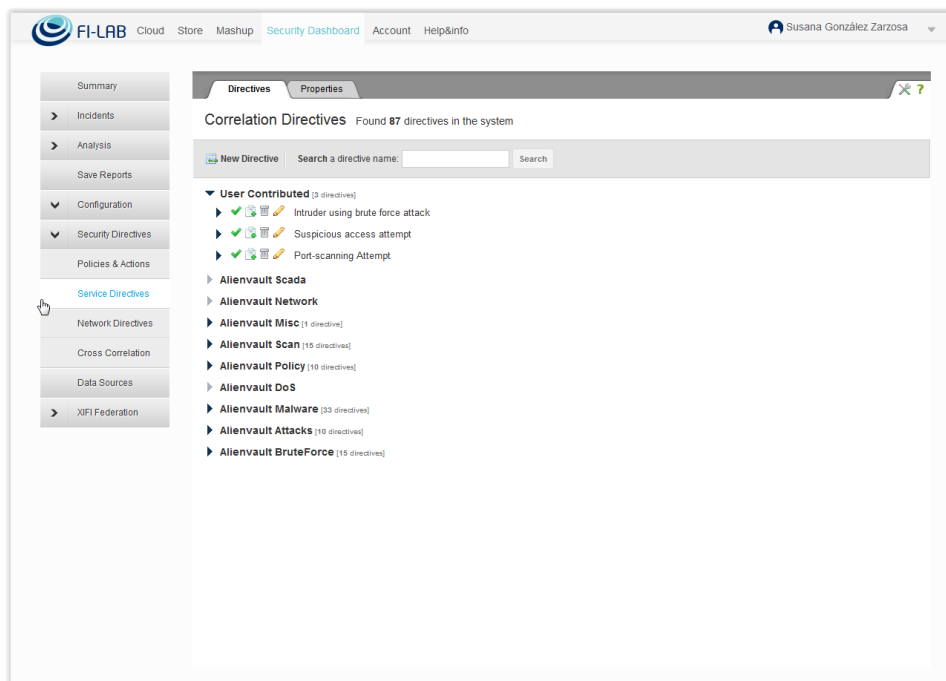


Figure 89: Security Monitoring - Network Directives

- **How to use the cross correlation.**

The cross correlation allows the definition of rules or directives using alarms generated by the service directives instead of the incoming events collected from the Security Probes. This cross correlation is also used to modify the reliability of an event increasing the risk value when the IP destination address of the event has some detected vulnerability stored in the database.

This option is located in the Security Dashboard graphical interface under *Configuration -> Security Directives -> Cross Correlation*.

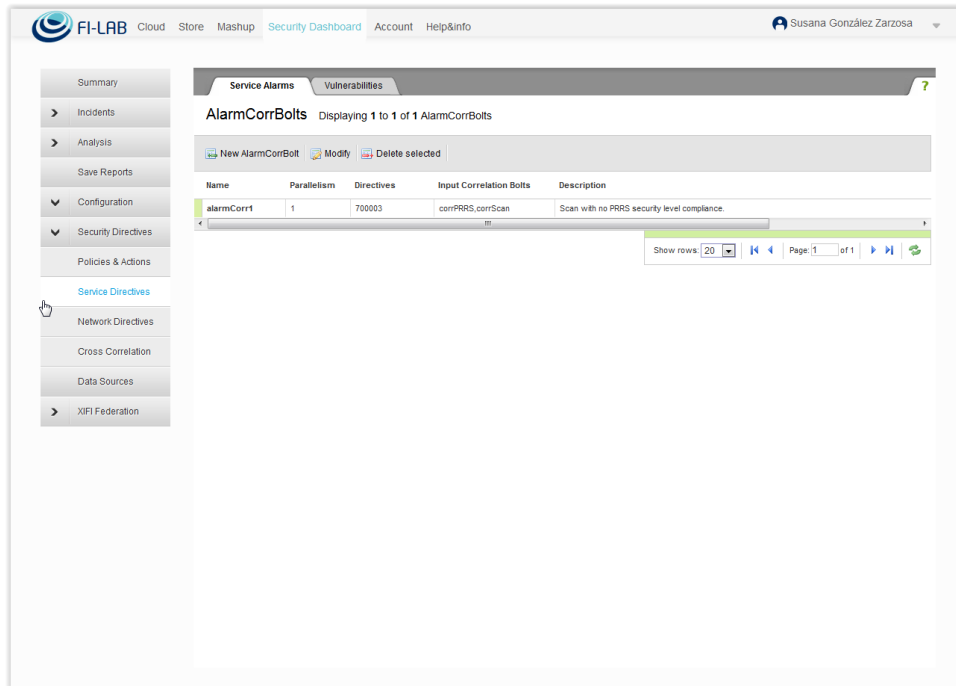


Figure 90: Security Monitoring - Cross Correlation

8 CLOUD PORTAL

8.1 Summary

The Cloud Portal provides a support for the users of the XIFI cloud infrastructure and platform to manage their services and resources deployed in cloud. It is implemented in a form of a Web GUI following the example of the portals that today's common cloud infrastructure managers like Amazon EC2, Eucalyptus, Cloud Sigma, Rackspace, etc. have. In concrete it bases its design principles on the *OpenStack Horizon Dashboard* [86].

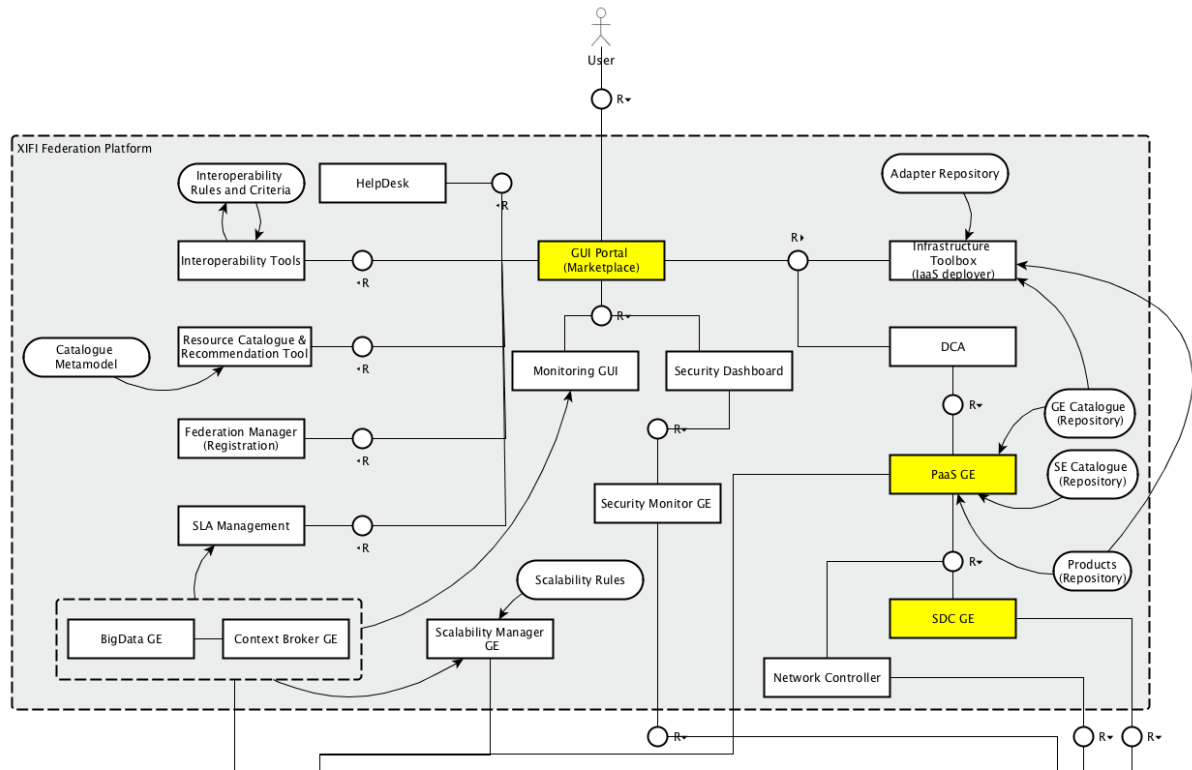


Figure 91: Location of the Cloud Portal in the XIFI Reference Architecture.

Reference Scenarios [3]	<ul style="list-style-type: none"> • UC-5 -Network and Data Centre operations [73]
Reference Stakeholders [6]	<ul style="list-style-type: none"> • Developers: that wants to manage their cloud resources. • Infrastructure Owners: that wants to manage their cloud resources.
Type of ownership	New Development
Original tool	N/A
Planned OS license	MIT License [122]
Reference OS community	FIWARE / OpenStack

Consists of

The basic objective of the Cloud Portal Portal is to facilitate the user of the cloud to perform operations over the underlying infrastructure. This includes perform actions such as: launch instances on a base of images, create images in the image repository, retrieve flavors from the resource, etc. All

these actions can be done in any of the XIFI regions registered in the system. The portal also allows the user to switch between those regions. Moreover the portal facilitates management of PaaS resources such as Blueprints and Tiers.

Depends on

- Federation Identity Management GE [93]
- FIWARE Access Control GE [100]
- Federation Monitoring [91]

8.2 Component leaders

Developer	Email	Company
Alvaro Alonso	aalonsog@dit.upm.es	UPM-DIT

Table 31: List of responsible Cloud Portal.

8.3 Motivation

XIFI nodes provide cloud resources to the users, such as virtual machines, images, networks, etc. In order to do this each node has an installation of the OpenStack [101] services (Nova, Glance, Neutron...). Openstack also provides a Dashboard [102] to help the users managing their resources.

However, in XIFI there are resources and advanced features that are not included in OpenStack:

- Platform as a Service management
- Public and private Region support.
- Integration with the other XIFI portals.
- OAuth2 support to authenticate the users using their XIFI accounts.
- High availability mode between regions.

Therefore, the goal of the Cloud Portal component is to provide a web interface that allows the users to manage all these Cloud resources and tools in an easy and intuitive way and be authenticated using their federated XIFI accounts.

8.4 User stories backlog

The cloud portal is an extension of the cloud portal component developed in FIWARE. Here we document only stories related to XIFI development.

id	User story name	Actors	Description	Task id
1	Support region selection	Developer	A user entering the cloud portal should be able to change the "cloud" region to which he is connected.	794
2	Support multiple region per project	Developer	A user should be able to deploy, within a project, virtual machines in multiple-regions and able to visualize the summary of deployed VM including information on the hosting region	795
3	Support all openstack services	Developer	A user should be able to manage flavors, images, security, snapshots, storage and network in all the regions)	796

id	User story name	Actors	Description	Task id
	multi region			
4	Multi region blueprint support - tiers	Developer	A user should be able to create blueprint templates with tiers in different regions	797
5	Multi region blueprint support - software and networks	Developer	When creating a tier the user should be able to install software and to deploy networks only available in the corresponding region	798

Table 32: User Stories backlog - Cloud Portal.

8.5 State of the art

This component is based on Openstack Dashboard [102] portal that provides a user interface to manage the Cloud resources of Openstack. But, as explained before, in the scope of XIFI there are some extra functionalities that have to be added to the the user interface. So the XIFI Cloud Portal starts from the FIWARE Self-Service Interfaces GE [107] adding the specific requirements of XIFI.

Besides Openstack, the most relevant Cloud Providers are:

- Amazon Web Services [103]: the most central and well-known of them are Amazon EC2 (compute) and Amazon S3 (storage)
- Heroku [104]: offers PaaS services.
- Joyent [105]: offers IaaS and PaaS services.
- Google Cloud Platform [106]: offers compute, storage, big data and also its PaaS service App Engine.

8.6 Architecture design

You can find the description of the architecture in the Cloud Portal FIWARE [68].

In order to use the multi-region modality you have to register the service endpoints of each one in the Keystone Service Catalogue. This is an example of the Nova service definition for two different regions:

```
ServiceCatalog = [
  {"endpoints":
    [
      {"adminURL": "http://novaHost:8774/v2/tenant_id",
        "region": "RegionOne",
        "internalURL": "http://novaHost:8774/v2/tenant_id",
        "publicURL": "http://novaHost:8774/v2/tenant_id"}
    ],
    [
      {"adminURL": "http://novaHost:8774/v2/tenant_id",
        "region": "RegionTwo",
        "internalURL": "http://novaHost:8774/v2/tenant_id",
        "publicURL": "http://novaHost:8774/v2/tenant_id"}
    ]
  },
  "endpoints_links": [],
  "type": "compute",
```

```

    "name": "nova"
  }
]

```

8.7 Release plan

Version Id	Milestone	User Stories
1.0	30.11.2013	1,2,3
1.1	30.12.2013	4
1.2	30.01.2014	5

Table 33: Release plan - Cloud Portal.

8.8 Test cases

Test id	Test description	Test script
1	Verification the operation of essential storage container manipulation functions.	see below
2	Verification of the operation of essential portal functions.	see below

Table 34: Test cases – Cloud Portal

Common Information about tests

These Unit Tests are designed to test the functionality of the Web modules that implement the features of the Cloud Portal. They have been created by the developers of the Cloud Portal and are available at <https://github.com/ging/fi-ware-cloud-portal/tree/master/test>. These tests are JavaScript based, and involve interaction with the generated web page; using Openstack API based Mock-ups instead of actual OpenStack components and other Cloud GEs.

The tests use three test libraries:

- Mocha: is a node.js framework for making unit tests. <http://visionmedia.github.io/mocha/>
- Should.js: is an assertion library used normally in Mocha tests for performing assertions. <https://github.com/visionmedia/should.js/>
- Zombie.js: is testing library for testing web applications. It emulates a web browser in order to perform navigation task that a user do when using the platform. <http://zombie.labnotes.org/>

The test can be run using the next command:

```
npm test
```

It executes *mocha test/testZombie.js* file performing the *tests*. Unit tests are divided into two parts: i) "library" (Unit Test 1) that are the command line scripts to manage Cloud infrastructure; ii) "portal" (Unit Test 2) that are the HTML views the user will use to manage in a more interactive way.

Unit Test 1

This Unit Test verifies the operation of essential storage container manipulation functions. It is composed of the following individual tests, all of which should pass:

- it should authenticate user: tests the authentication of users in the Cloud
- it should list instance: tests the read of instances from the API
- it should create instance: tests the creation of new instances in the Cloud
- it should get the instance detail: tests the read of instance details from the Cloud
- it should delete instance: tests the removal of instances through the command-line
- it should stop instance: tests to stop an instance through the command line
- it should start instance: tests the start of instances through command line
- it should resize instance: tests the resize operation of an instance in the Cloud
- it should reboot instance: tests the reboot of instances through command line
- it should create image: tests the creation of images through command line
- it should list images: tests the read of all instances in the Cloud
- it should get the image details in the portal: tests the read of image details in the Cloud
- it should delete image: tests the removal of images from the Cloud
- it should list flavors: tests the read of flavor lists through command line
- it should get the flavor details: tests the read of flavor details through operations in the command line
- it should delete flavor: tests the removal of a flavor through command line

The unit test creates some test structures before running the above tests, and then removes all temporary test structures upon completion. To perform the tests execute the 'npm test' command.

Results expected:

Library

- ✓ should authenticate user
- ✓ should list instance
- ✓ should create instance
- ✓ should get the instance detail
- ✓ should delete instance
- ✓ should stop instance
- ✓ should start instance
- ✓ should resize instance
- ✓ should reboot instance
- ✓ should create image
- ✓ should list images
- ✓ should get the image details in the portal
- ✓ should delete image

- ✓ should list flavors
- ✓ should get the flavor details
- ✓ should delete flavor

Remarks: The time elapsed could be different depending on the machine on which the tests are executed. The tests must return without Failures, Errors and/or Skipped results.

Unit Test 2

This Unit Test verifies the operation of essential portal functions.

The unit test creates some test structures before running the above tests, and then removes all temporary test structures upon completion. To perform the tests execute the 'npm test' command.

Results expected:

Portal

- ✓ should show the login page of portal
- ✓ should show the overview page in the portal for RegionOne (3006ms)
- ✓ should show the terminal of the instance for RegionOne
- ✓ should list instance in the portal for RegionOne
- ✓ should create instance in the portal for RegionOne
- ✓ should get the instance detail in the portal for RegionOne
- ✓ should delete instance in the portal for RegionOne
- ✓ should stop instance in the portal for RegionOne
- ✓ should start instance in the portal for RegionOne
- ✓ should resize instance in the portal for RegionOne
- ✓ should reboot instance in the portal for RegionOne
- ✓ should list images in the portal for RegionOne
- ✓ should get the image details in the portal for RegionOne
- ✓ should delete image in the portal for RegionOne
- ✓ should list flavors in the portal for RegionOne
- ✓ should get the flavor details in the portal for RegionOne
- ✓ should delete flavor for RegionOne
- ✓ should list projects for RegionOne
- ✓ should switch to Region Two
- ✓ should show the overview page in the portal for RegionTwo
- ✓ should show the terminal of the instance for RegionTwo
- ✓ should list instance in the portal for RegionTwo
- ✓ should create instance in the portal for RegionTwo
- ✓ should get the instance detail in the portal for RegionTwo
- ✓ should delete instance in the portal for RegionTwo

- ✓ should stop instance in the portal for RegionTwo
- ✓ should start instance in the portal for RegionTwo
- ✓ should resize instance in the portal for RegionTwo
- ✓ should reboot instance in the portal for RegionTwo
- ✓ should list images in the portal for RegionTwo
- ✓ should get the image details in the portal for RegionTwo
- ✓ should delete image in the portal for RegionTwo
- ✓ should list flavors in the portal for RegionTwo
- ✓ should get the flavor details in the portal for RegionTwo
- ✓ should delete flavor for RegionTwo
- ✓ should list projects for RegionTwo
- ✓ should log out

35 tests complete (15 seconds)

Remarks: The time elapsed could be different depending on the machine on which the tests are executed. The tests must return without Failures, Errors and/or Skipped results.

8.9 Installation Manual

For the installation documentation you can refer to FIWARE documentation [69].

In order to use the multi-region modality you have to register the service endpoints of each one in the Keystone Service Catalogue. This is an example of the Nova service definition for two different regions:

```
ServiceCatalog = [
  {
    "endpoints":
      [
        {
          "adminURL": "http://novaHost:8774/v2/tenant_id",
          "region": "RegionOne",
          "internalURL": "http://novaHost:8774/v2/tenant_id",
          "publicURL": "http://novaHost:8774/v2/tenant_id"
        }
      ],
      [
        {
          "adminURL": "http://novaHost:8774/v2/tenant_id",
          "region": "RegionTwo",
          "internalURL": "http://novaHost:8774/v2/tenant_id",
          "publicURL": "http://novaHost:8774/v2/tenant_id"
        }
      ]
    "endpoints_links": [],
    "type": "compute",
    "name": "nova"
  }
]
```

8.10 User Manual

Please refer to the original FIWARE manuals [70].

In addition to the information included there, related with the XIFI region support:

Select different Regions

A user using the cloud portal is able to change the "Cloud" region to which he is connected.

Figure 92 shows the Region selector at the left sidebar, where the user can choose the region he wants to work in. When a user changes the region the Cloud portal automatically changes the endpoints of the services to the ones that correspond to that region (as described in the Keystone Service Catalog [section 8.9]).

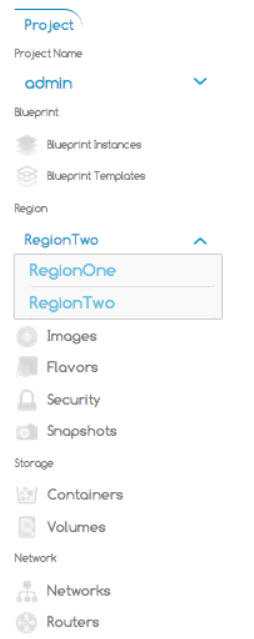


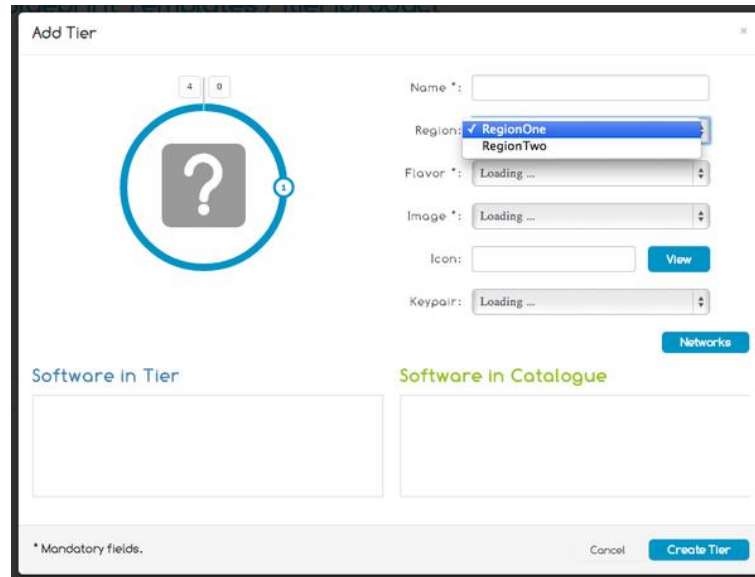
Figure 92: Switch Region in Cloud Portal

Deploy Blueprints in multiple Regions

A user using the Cloud Portal is able to deploy Blueprint Instances with Tiers in different Regions. A Blueprint Instance is the deployment of a set of Tiers previously defined in a Blueprint Template. A Tier is a description of a virtual machine from an image and with the definition of its flavor, keypair and installed software. Thus, a user can define a Blueprint template with a set of Tiers and when instantiate it, a determined number of virtual machines of each Tier will be deployed (this number is also defined in a Tier definition).

In Xifi multi-region mode, a user can specify the region in which each Tier will be deployed (i.e. the virtual machines deployed when instantiate each Tier). This is the form to define a Tier with region support:

Figure 93 shows the Region selector within the form when editing or creating a Tier in a Blueprint Template.



The screenshot shows a web form titled "Add Tier". On the left, there is a circular icon with a question mark and a small "1" in a circle next to it. The form fields are as follows:

- Name *:
- Region: (dropdown menu with "RegionOne" and "RegionTwo" options)
- Flavor *:
- Image *:
- Icon:
- Keypair:

Below the fields, there are two sections: "Software in Tier" and "Software in Catalogue", both with empty rectangular boxes. A "Networks" button is located below the Keypair field. At the bottom of the form, there is a note: "* Mandatory fields." and two buttons: "Cancel" and "Create Tier".

Figure 93: Switch Region in a Blueprint Tier in Cloud Portal

9 FIWARE LAB APP TEMPLATE

This component has only introduced the logo of the new branch FIWARE Lab instance of Fi-Lab in the style template. Hence, there aren't any changes from the previous version and it only updates the logo and the associated pictures as you can see in the following picture.

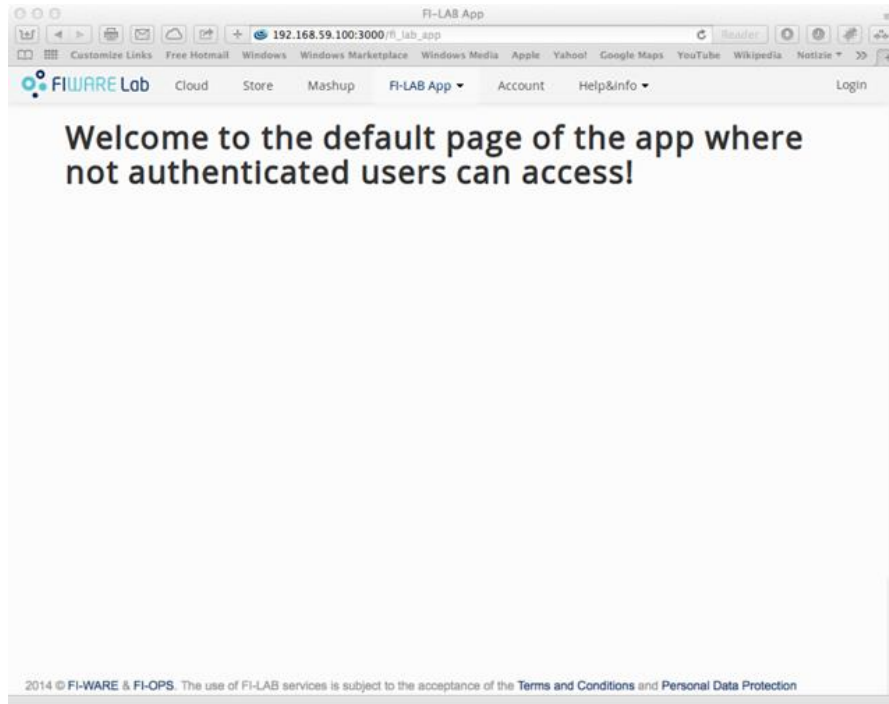


Figure 94: Home Page – FIWARE Lab App Template

You can find the description of the component in the public wiki page [108] or in the previous version of the document D4.2 Baseline tools v1 [95].

10 MONITORING DASHBOARD

10.1 Summary

The Monitoring Dashboard is a mashup of different monitoring tools for some of the XIFI components currently deployed in the Federation. It provides Graphical User Interfaces for show to users and Infrastructures Owners live and historical data about the status of the Federation environment resources in XIFI. In order to do that it retrieves the data from XIFI monitoring data such us Federation Monitoring and Network Active Monitoring.

The Monitoring Dashboard component is part of the Monitoring GUI architectural component and is integrated into the portal.

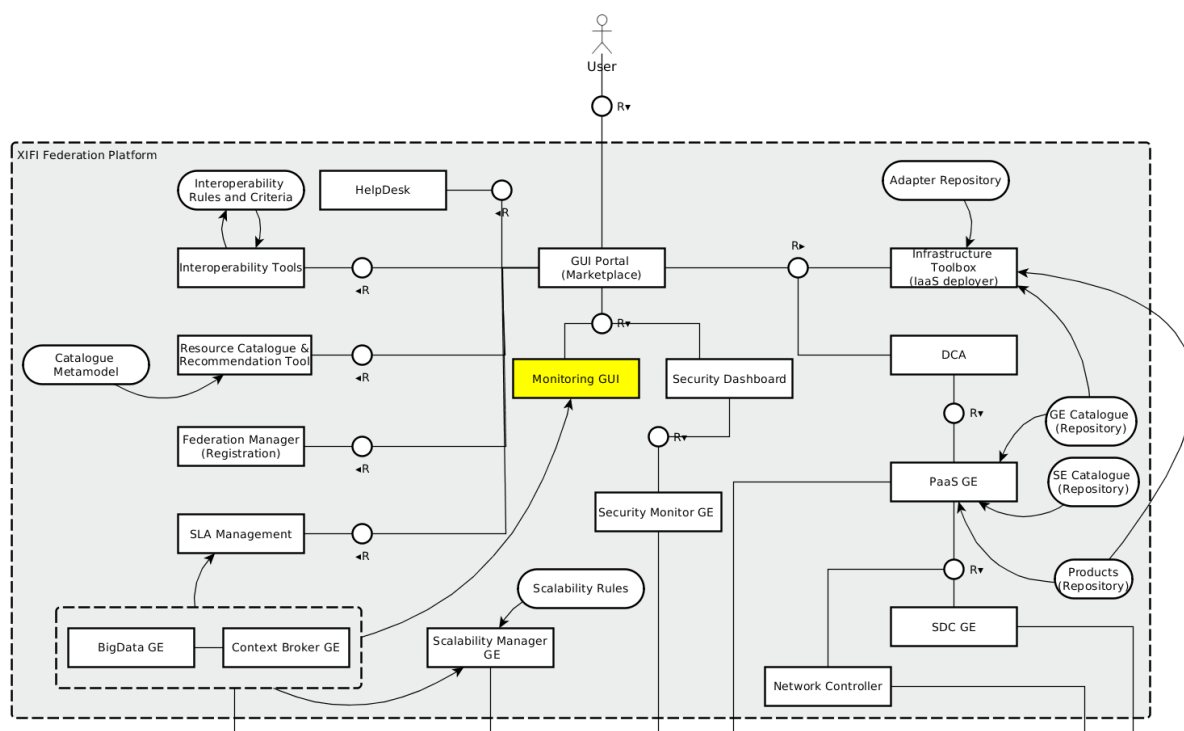


Figure 95: Location of the Monitoring Dashboard in the XIFI Reference Architecture

Reference Scenarios [3]	<ul style="list-style-type: none"> • UC-5 -Network and Data Centre operations [73]
Reference Stakeholders [6]	<ul style="list-style-type: none"> • Developers: that wants to monitor the status of their deployed resources. • Infrastructure Owners: that wants to monitor the status of their infrastructure components and their network connections.
Type of ownership	New Development
Original tool	N/A
Planned OS license	MIT License [122]
Reference OS community	FIWARE / OpenStack

Table 35: Summary Monitoring Dashboard.

Consists of

The monitoring Dashboard consists of two components: i) VMs monitoring tool ii) NAM monitoring tool.

- VMs monitoring tool (integrated in the Cloud Portal) is in charge of the Virtual Machines monitoring. It gets data from Federating Monitoring API in order to show the user the CPU, Memory and Disk status of its deployed Virtual Machines, either in real-time or historical data.
- NAM monitoring tool is oriented to XIFI IO, providing information about the Network status of the connections between XIFI nodes. It can obtain monitoring data about bandwidth, latency and packet loss in two ways: real data information and historical data information.

Depends on

- Federation Monitoring GE [91]
- Federation Identity Management GE [93]
- FIWARE Cloud Portal GE [section 8]
- Network Active Monitoring [109]

10.2 Component leaders

Developer	Email	Company
Álvaro Alonso	aalonsog@dit.upm.es	UPM-DIT

Table 36: List of responsible – Monitoring Dashboard.

10.3 Motivation

XIFI federation is composed of different nodes that provide cloud resources to the users. When a user deploys resources in the cloud infrastructure it is very interesting to have feedback about the status of them. And this status can be relevant in a live mode and as an historical summary in a determined period of time. So XIFI federation has to provide the users tools to be aware of the status of their resources in both modes. These resources are usually the Virtual Machines that the users have deployed in the Cloud Infrastructure.

But also for the Infrastructure Owners that are providing these resources to the users it is interesting to have tools to monitor the status of the infrastructure. This corresponds, for instance, to the status of the physical hosts in terms of CPU, Disk and Memory, the status of the network connections (inside the nodes and between them) or also the status of the Virtual Machines that they are provisioning.

The main objective of the Monitoring Dashboard tools is to provide the users and the Infrastructure Owners, Graphical User Interfaces in order to have an easy way to monitor the status of all these components.

10.4 User stories backlog

id	User story name	Actors	Description	Task id
1	Monitor NAM Bandwidth	Infrastructure Owner	NAM Dashboard should show to IO live and historical data about the bandwidth status between the Federation nodes.	1496

id	User story name	Actors	Description	Task id
2	Monitor NAM Latency	Infrastructure Owner	NAM Dashboard should show to IO live and historical data about the latency status between the Federation nodes.	1497
3	Monitor NAM Packet Loss	Infrastructure Owner	NAM Dashboard should show to IO live and historical data about the packet loss status between the Federation nodes.	1498
4	Monitor live VMs status	User	Cloud Portal monitoring module should show to users the status of their VMs in terms of Disk, DPU and Memory in live mode.	1499
5	Monitor historical VMs status	User	Cloud Portal monitoring module should show to users the status of their VMs in terms of Disk, DPU and Memory in historic mode.	1500
6	VMs status widgets	Developer	Developers should be able to include VMs monitoring widgets in their dashboards.	1501

Table 37: Monitoring Dashboard - User stories backlog

10.5 State of the art.

There are several Cloud Services that offer monitoring tools to the users.

- *Amazon Web Services* [103]: it offers Cloud Watch component for monitoring the Cloud Resources deployed in the service.
- *Heroku* [104]: offers some monitoring add-ons to get the status of the resources.
- *Joyent* [105]: offers cloud analytics and performance tools.
- *Google Cloud Platform* [106]: offers a Monitoring API that allows the users to read monitoring data such as response times, uptime, disk usage, MySQL queries.

The VMs monitoring component of the Monitoring Dashboard is included in the Cloud Portal XIFI component. You can refer to the *Cloud Portal state of the art* [section 8.5] in order to know the state of the art of this one.

Regarding the NAM Dashboard, it is a Graphical User Interface designed following the MVC pattern for web services.

10.6 Architecture design

VMs Dashboard architecture

As VMs Monitoring Dashboard is included on Cloud Portal, inside the tab “Monitoring” the details of a VM can be displayed, refer to Cloud Portal architecture.

NAM Dashboard architecture

NAM Dashboard is a GUI that follows the MVC pattern as follows. The data is retrieved from Network Active Monitoring component.

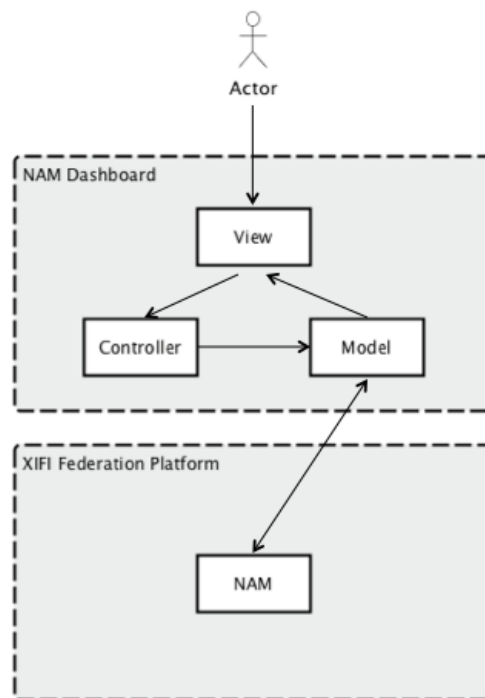


Figure 96: Monitoring Dashboard Architecture

10.7 Release plan

Version Id	Milestone	User Stories
1.0	22.09.2014	1, 2, 3, 4
1.1	01.11.2014	5, 6

Table 38: Monitoring Dashboard

10.8 Test cases

VMs Monitoring Dashboard tests

As VMs Monitoring Dashboard is included on Cloud Portal [section 8], refer to Cloud Portal test cases for installing this part.

NAM Dashboard tests

For test the service, perform the basic usage example described in the user manual [section 10.10] of this component.

10.9 Installation Manual

VMs Monitoring Dashboard installation

As VMs Monitoring Dashboard is included on Cloud Portal, inside the tab “Monitoring” the details of a VM can be displayed. Refer to Cloud Portal installation manual [section 8.9] for installing this part.

NAM Dashboard installation

Pre-requisites:

- nodejs
- npm

First, you have to clone the Github repository

```
git clone https://github.com/ging/fi-xifi-nam-dashboard.git
```

Then install the project dependencies

```
cd fi-xifi-nam-dashboard  
npm install
```

Before running the service, you have to set some configuration parameters

```
cp config.js.template config.js  
vi config.js
```

In this file you will see the following parameters

```
var config = {  
  config.oauth = {  
    account_server: ,  
    client_id: ,  
    client_secret: ,  
    callbackURL:  
  };  
  config.nam_server = ;  
  module.exports = config;
```

- *config.oauth* is the data of the application in order to make the single sign on with XIFI IdM GE. In order to use it you have to previous register this application there. Follow the instructions in IdM GE [93] to do that.
- *config.nam_server* is the nam server IP address in order to retrieve the monitoring data.

Once these parameters are configured you can start the service

```
sudo node server.js
```

10.10 User Manual

VMs Monitoring Dashboard User Manual

As VMs Monitoring Dashboard is included in Cloud Portal, inside the tab “Monitoring” the details of a VM can be displayed, refer to Cloud Portal user manual [section 8.10].

NAM Dashboard User Manual

This is the main view of the NAM Dashboard

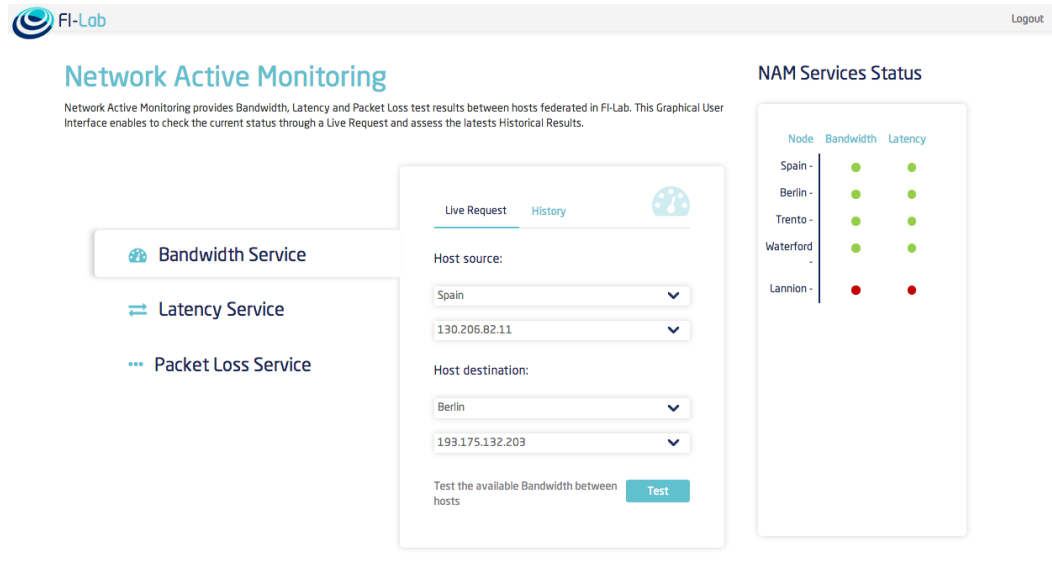


Figure 97: main view of NAM Dashboard

In the left menu you can select the type of the test to perform (bandwidth, latency or packet loss).

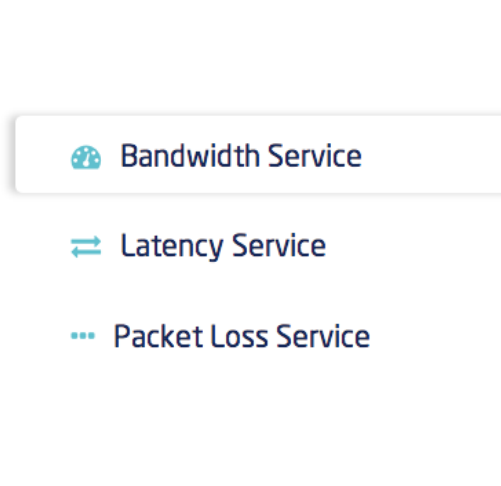


Figure 98: Type of test to perform NAM Dashboard

In one of each test type you can also select the mode (live or historical)

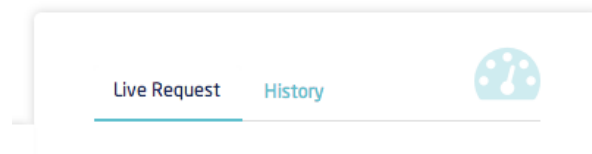


Figure 99: Live Request or History NAM Dashboard

Once you chose both options, you have to select the host source and destination using the select boxes

Host source:

Spain
Spain
Berlin
Trento
Waterford
Lannion

Berlin

193.175.132.203

Test the available Bandwidth between hosts

Figure 100: Test options for NAM Dashboard

Finally, for performing the test, click on the "Test" button

11 POLICY RECOMMENDATION SERVICE

11.1 Summary

The policy recommendation service allows experimenters to find suitable infrastructures within the federation that fulfil their specific privacy preferences. The service thus allows the identification of the most appropriate set of infrastructures for a concrete experiment. The Marketplace uses this privacy policy management tool (in conjunction with other XIFI tools) to allow experimenters/application providers to select the most suitable combination of infrastructures according to their privacy requirements following a guided process of recommendations. From the general architecture point of view, this Policy Recommendation Service belongs to the recommender’s tools, so it becomes part of the Marketplace/Resource Catalogue and Recommender tool boxes.

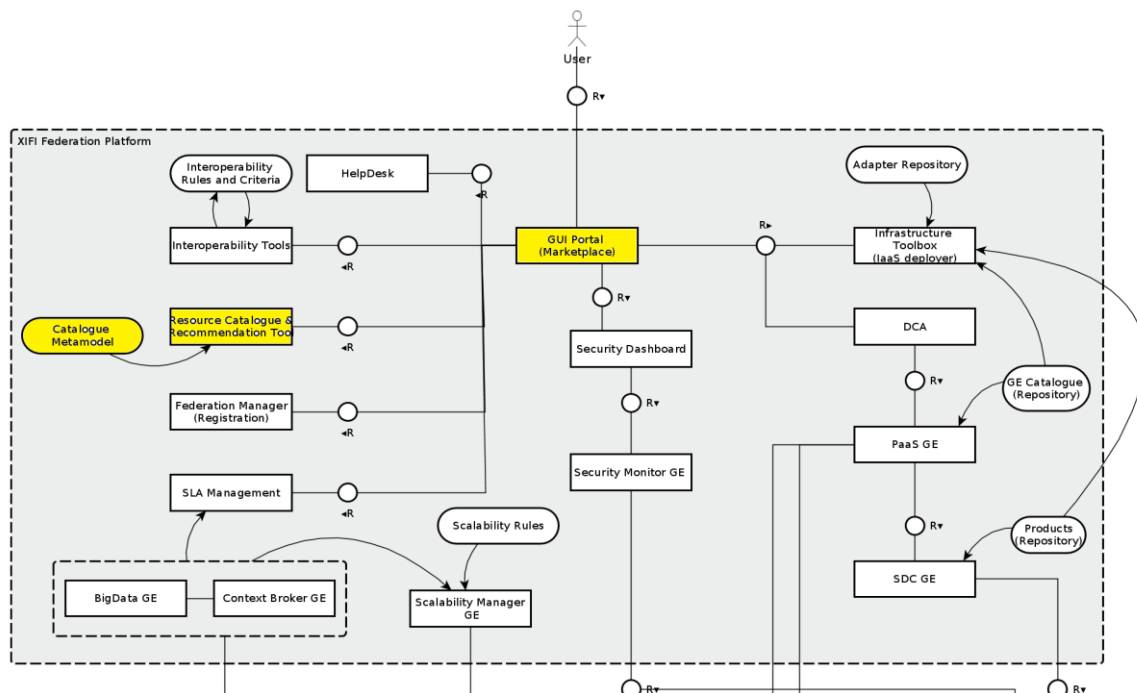


Figure 101: Location of the Policy Recommendation Service in the XIFI Reference Architecture

Reference Scenarios [3]	<ul style="list-style-type: none"> UC2 - Setup and use of development environment [8]
Reference Stakeholders [6]	<ul style="list-style-type: none"> As end user (Developers, End Users/Adopters; Sponsors/ Investors)
Type of ownership	New Development
Original tool	N/A
Planned OS license	LGPL version 3.0 [47]
Reference OS community	N/A

Table 39: Summary Policy Recommendation service.

Consist of

- Policy Recommendation service.

Depends on

- Federation Identity Management [93]
- Resource Catalogue & Recommendation Tool [section 2]

11.2 Component leaders

Developer	Email	Company
Bassem Nasser	bmn@it-innovation.soton.ac.uk	IT-Innovation

Table 40: List of responsible – Policy Recommendation Service.

11.3 Motivation

The XIFI federation brings together computational infrastructures (e.g. computational resources, networking resources, and storage resources) in order that they can be leveraged in combination by the developers of Future Internet technology. The central objective is to support industry stakeholders who require Future Internet resources to perform large-scale trials in order to evaluate and validate their technologies before they are transferred to market.

The technical compliance has been addressed by the interoperability tool in XIFI; however, the legal and ethical compliance is still an issue for experimenters and application providers mainly when their applications process data that is private (personal) or commercially sensitive. In such instances, the experimenter would currently be obliged to consider a hybrid cloud, with the private data used only within their private cloud. This may not, however, be practical or even desirable all the time and defeats the object of federated XIFI resource.

Before deploying their applications or their data onto the federation infrastructure, the experimenters and application providers will need to understand the different applicable privacy policies.

A XIFI privacy recommendation service allows these federation users to choose the nodes that match their privacy requirements.

This service requires the nodes' privacy policies to be available in machine readable format that can express a sophisticated set of terms and conditions associated with clear semantics for consistent processing.

The recommendation service can be applied in both scenarios:

- 1-Privacy policy be associated with the whole node
- 2-Privacy policy is associated with a specific service offered by the node.

11.4 User stories backlog

id	User story name	Actors	Description	Tasks id
1	Select privacy policies	Experimenter/ application developer/ application provider	The actor selects the different privacy policies to analyse. The privacy policies can be those of specific services or whole node.	1516
2	Specify privacy preference	Experimenter/ application developer/ application provider	The actor specifies their privacy preferences.	1517

id	User story name	Actors	Description	Tasks id
	Find compatible policies	Experimenter/ application developer/ application provider	Actor runs the policy analysis and obtains a list of compatible policies	1518

Table 41: Policy recommendation service - User stories backlog

11.5 State of the art.

Platform for Privacy Preferences (P3P) [110]

Platform for Privacy Preferences (P3P) is a W3C specification that enables Web sites to express their privacy policy in a machine readable, standard XML format. The goal is to automate the user decision-making process based on the privacy practices of the website. The specification is supposed to be interpreted by user agents installed in the browser (browser plug-ins, or proxy servers) so that the user does not need to read the privacy policy of each site they visit.

In addition to the data-collection and data-usage practices, the specification provides a means of associating privacy policies with Web pages or sites, and cookies and a mechanism for transporting P3P policies over the web.

In the P3P context, any data that can be used reasonably by a data controller or any other person to identify an individual is considered to be identifiable data.

A policy reference file is used to locate the privacy policy documents associated with websites or resources at the same website. The policy reference file may be located in a predefined location or sent across via the html header or in an html “link” tag.

The main content of a privacy policy is the following:

- Policy identifier: this includes the name of the policy, URI of the natural language privacy statement, etc.
- Entity: identifies the legal entity making the representation of the privacy practices contained in the policy.
- Access: refers to the ability of the individual to view identified data and address questions or concerns to the service provider. Service providers must disclose one value for the access attribute.
- Disputes: this refers to one way the entity offers or acknowledges for a user to resolve disputes about the entity's privacy practices or alleged protocol violations (e.g. customer service, independent organisation, court or applicable law). It should also include remedies that the entity offers to the identified dispute resolution procedures (e.g. correct the error, fine or compensation, legal redress)
- Policy statement: is a container that encapsulates the following information:
 - Consequence: a short summary or explanation of the data practices described in this policy statement that can be shown to a human user
 - Purpose: purposes for data processing. This is mandatory when the collected data is “identifiable”.
 - Recipient: lists who is receiving the collected data. This can be the entity itself, legal entities following the same practices, public, unrelated third parties, etc.
 - Retention: the type of retention policy in effect. This can include, for instance, no retention, or retention for the stated purpose, legal requirement, business practices,

etc.

- Data: this includes information about the data to be transferred (e.g. computer information) or inferred (e.g. location information inferred from the IP address). The data types (e.g. business info, IP address, home info, etc) and the categories of the data (e.g. "financial" for Financial Information, "computer" for Computer Information, "demographic" for Demographic and Socioeconomic Data, etc) can also be included here.

A data schema and an XML policy schema are available to be used for defining privacy policies.

A P3P Preference Exchange Language (APPEL)

APPEL [111] is a W3C working draft specification that complements P3P and allows users to express their privacy preferences in a set of preference-rules which allows the user agent to make automated or semi-automated decisions regarding the acceptability of machine-readable privacy policies from P3P enabled Web sites.

The preferences are encapsulated in a “ruleset” which includes a series of rules. Each rule contains conditions under which a behaviour (e.g. block, prompt) should be carried out.

APPEL allows users to express such preferences as:

1. Requests for personal information that will be given out to 3rd parties should be blocked.
2. The user does not mind revealing click-stream and user agent information to sites that collect no other information.
3. The user is comfortable with giving out the first and last name, as long as it is for non-marketing purposes. The user requires assurances (i.e., dispute information) from both "PrivacyProtect" and "TrustUs" and to be explicitly prompted before actually accessing such a page.
4. When interacting with her bank's Web site at <http://www.my-bank.com>, she accepts any data request as long as her data is not redistributed to other recipients.

Rule 3 can be encoded as:

```
<appel:RULE
behavior="request" prompt="yes"
  promptmsg="Service only collects your name
            for non-marketing purposes (assured)
            Do you want to continue?">
  <p3p:POLICY>
  <p3p:STATEMENT>
  <p3p:PURPOSE appel:connective="or-exact">
    <p3p:current/>
    <p3p:admin/>
    <p3p:customization/>
    <p3p:develop/>
  </p3p:PURPOSE>
  <p3p:DATA-GROUP appel:connective="or-exact">
    <p3p:DATA ref="#user.name.*"/>
  </p3p:DATA-GROUP>
  </p3p:STATEMENT>
  <p3p:DISPUTES-GROUP>
  <p3p:DISPUTES service="http://www.privacyprotect.com"/>
  <p3p:DISPUTES service="http://www.trustus.org"/>
  </p3p:DISPUTES-GROUP>
  </p3p:POLICY>
</appel:RULE>
```

Policy example

The following is an example taken from the P3P specification that shows how an English-language privacy policy can be encoded as a P3P policy. The example is provided in both English and as a more formal description using P3P element and attribute names.

CatalogExample's Privacy Policy for Browsers

At CatalogExample, we care about your privacy. When you come to our site to look for an item, we will only use this information to improve our site and will not store it with information we could use to identify you.

CatalogExample, Inc. is a licensee of the PrivacySealExample Program. The PrivacySealExample Program ensures your privacy by holding Web site licensees to high privacy standards and confirming with independent auditors that these information practices are being followed.

Questions regarding this statement should be directed to:

CatalogExample
4000 Lincoln Ave.
Birmingham, MI 48009 USA
email: catalog@example.com
Telephone 248-EXAMPLE (248-392-6753)

If we have not responded to your inquiry or your inquiry has not been satisfactorily addressed, you can contact PrivacySealExample at <http://www.privacyseal.example.org>. CatalogExample will correct all errors or wrongful actions arising in connection with the privacy policy.

What We Collect and Why:

When you browse through our site we collect:

- the basic information about your computer and connection to make sure that we can get you the proper information and for security purposes.
- aggregate information on what pages consumers access or visit to improve our site.

Data retention:

We purge every two weeks the browsing information that we collect.

The formal description is shown below, using the P3P element and attribute names [with the section of the spec that was used cited in brackets for easy reference]:

- Disclosure URI: <http://www.catalog.example.com/PrivacyPracticeBrowsing.html> [3.2.2 Policy]
- Entity: CatalogExample [3.2.5 Entity]
 4000 Lincoln Ave.
 Birmingham, MI 48009
 USA
 catalog@example.com
 +1 (248) 392-6753
- Access to Identifiable Information: None [3.2.6 Access]
- Disputes: [3.2.7 Disputes]
 resolution type: independent
 service: <http://www.privacyseal.example.org>
 description: PrivacySealExample
- Remedies: we'll correct any harm done wrong. [3.2.8 Remedies]

- We collect: [[5.5 Base data schema](#)]
dynamic.clickstream
dynamic.http
- For purpose: Web site and system administration, research and development. [[3.3.5 Purpose](#)]
- Recipients: Only ourselves and our agents. [[3.3.6 Recipients](#)]
- Retention: As long as appropriate for the stated purposes. [[3.3.7 Retention](#)]

(Note also that the site's human-readable privacy policy MUST mention that data is purged every two weeks, or provide a link to this information.)

The XML encoding of the policy:

```
<POLICIES xmlns="http://www.w3.org/2002/01/P3Pv1"
  xmlns:p3p11="http://www.w3.org/2006/01/P3Pv11">
  <POLICY name="forBrowsers"
    discuri="http://www.catalog.example.com/PrivacyPracticeBrowsing.html"
    xml:lang="en">
    <ENTITY>
      <EXTENSION>
        <p3p11:data-group>
          <p3p11:datatype>
            <p3p11:business>
              <p3p11:orgname>CatalogExample</p3p11:orgname>
              <p3p11:contact-info>
                <p3p11:postal>
                  <p3p11:street>4000 Lincoln Ave.</p3p11:street>
                  <p3p11:city>Birmingham</p3p11:city>
                  <p3p11:state>MI</p3p11:state>
                  <p3p11:postalcode>48009</p3p11:postalcode>
                  <p3p11:country>USA</p3p11:country>
                </p3p11:postal>
                <p3p11:online>
                  <p3p11:email>catalog@example.co.uk</p3p11:email>
                </p3p11:online>
                <p3p11:telecom>
                  <p3p11:telephone>
                    <p3p11:intcode>1</p3p11:intcode>
                    <p3p11:lococode>248</p3p11:lococode>
                    <p3p11:number>3926753</p3p11:number>
                  </p3p11:telephone>
                </p3p11:telecom>
              </p3p11:contact-info>
            </p3p11:business>
          </p3p11:datatype>
        </p3p11:data-group>
      </EXTENSION>
      <DATA-GROUP>
        <DATA ref="#business.name">CatalogExample</DATA>
        <DATA ref="#business.contact-info.postal.street">4000 Lincoln Ave.</DATA>
        <DATA ref="#business.contact-info.postal.city">Birmingham</DATA>
        <DATA ref="#business.contact-info.postal.stateprov">MI</DATA>
        <DATA ref="#business.contact-info.postal.postalcode">48009</DATA>
        <DATA ref="#business.contact-info.postal.country">USA</DATA>
        <DATA ref="#business.contact-info.online.email">catalog@example.com</DATA>
        <DATA ref="#business.contact-info.telecom.telephone.intcode">1</DATA>
        <DATA ref="#business.contact-info.telecom.telephone.lococode">248</DATA>
        <DATA ref="#business.contact-info.telecom.telephone.number">3926753</DATA>
      </DATA-GROUP>
    </ENTITY>
  </POLICY>
</POLICIES>
```

```

</ENTITY>
<ACCESS><nonident/></ACCESS>
<DISPUTES-GROUP>
  <DISPUTES resolution-type="independent"
  service="http://www.PrivacySeal.example.org"
  short-description="PrivacySeal.example.org">
    <IMG src="http://www.PrivacySeal.example.org/Logo.gif" alt="PrivacySeal's logo"/>
  <REMEDIES>
    <correct/>
  </REMEDIES>
</DISPUTES>
</DISPUTES-GROUP>
<STATEMENT>
  <PURPOSE>
    <admin/>
    <develop/>
  </PURPOSE>
  <RECIPIENT>
    <ours/>
  </RECIPIENT>
  <RETENTION>
    <stated-purpose/>
  </RETENTION>
  <!-- Note also that the site's human-readable
  privacy policy MUST mention that data
  is purged every two weeks, or provide a
  link to this information. -->
  <EXTENSION>
    <p3p11:data-group>
      <p3p11:datatype>
        <p3p11:dynamic>
          <p3p11:clickstream/>
          <p3p11:http/>
        </p3p11:dynamic>
      </p3p11:datatype>
    </p3p11:data-group>
  </EXTENSION>
  <DATA-GROUP>
    <DATA ref="#dynamic.clickstream"/>
    <DATA ref="#dynamic.http"/>
  </DATA-GROUP>
</STATEMENT>
</POLICY>
</POLICIES>

```

11.6 Architecture design

The architecture should be flexible to address infrastructure privacy as well as service privacy. The following diagram sketches the interactions of the recommendation service:

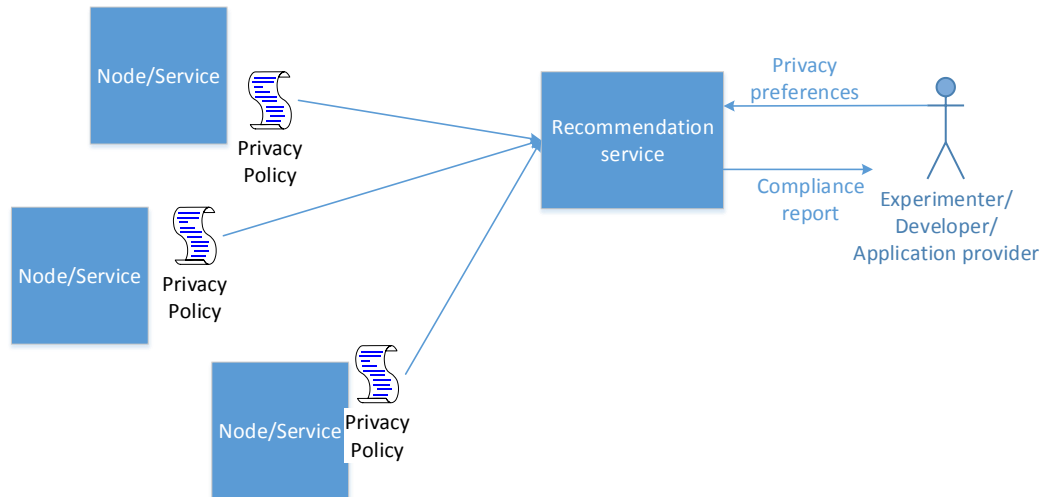


Figure 102: Policy Recommendation service.

Infrastructures publish a machine readable privacy policy that is accessible by the recommendation service. The experimenter provides their privacy preferences to the recommendation service which compares it with the different privacy policies (whether node or service policy). The recommendation service produces a compliance report detailing how each infrastructure covers the set of preferences.

The recommendation service building blocks are shown in the following figure:

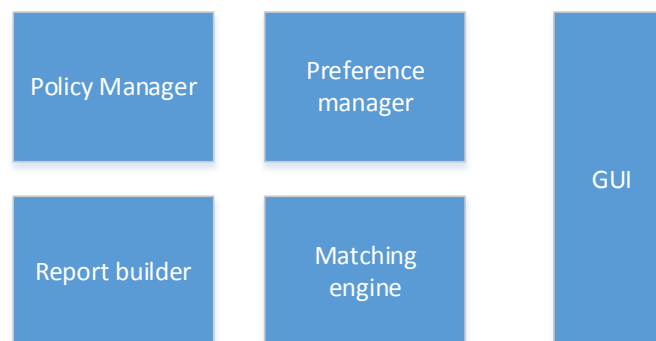


Figure 103: Policy Recommendation service components.

The recommendation service consists of a GUI that allows users to specify their privacy preferences. These preferences are managed by the “preference manager”. The “matching engine” uses the “policy manager” to collect the different privacy policies and compares them to the user’s privacy preferences. It then uses the “report builder” to provide the user with a report of the results to be displayed at the GUI.

The technology used in developing the privacy recommendation service is shown in red in the figure below.

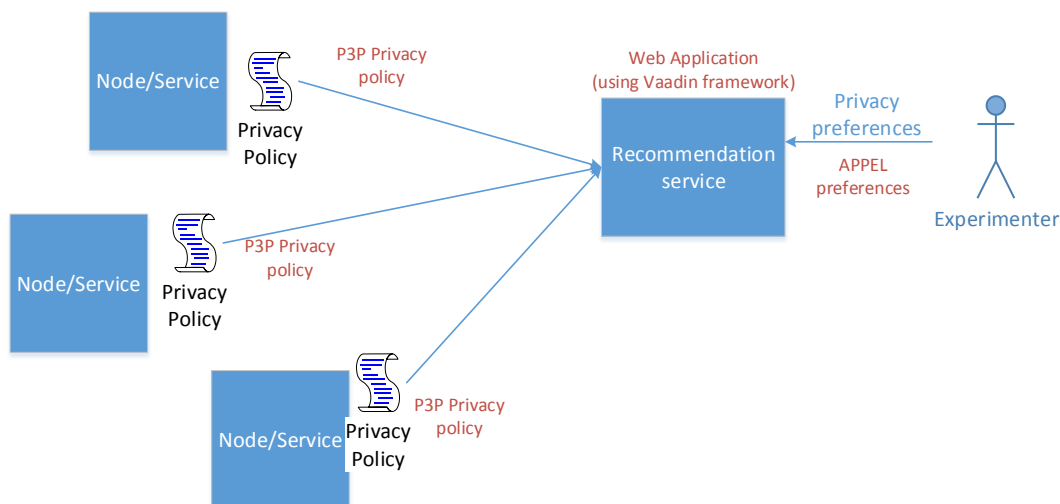


Figure 104: Technology used in Policy Recommendation Service components.

The privacy policies are encoded based on the W3C P3P recommendation for privacy policies. These are to be created by the different infrastructure providers for the node as whole or particular services. The location of these policies should be known (either a shared well-known location or a policy-reference file can be used at the root of the web server to indicate where the privacy policy is). In order to generate the privacy policies, there are online services/tools (e.g. p3pedit [112]) that provide services to specify P3P 1.0 and P3P 1.1 privacy policies. Moreover, the JRC Policy Workbench is an open source [Under the COMMON PUBLIC LICENSE [113]] tool for creating a P3P policy and associating it to specific services hosted at the same website.

The Recommendation service is developed as a Web application using the Vaadin [114] framework. It provides a GUI for the user to specify and evaluate their privacy preferences. These preferences are encoded according to the APPEL specification.

11.7 Release plan

Version Id	Milestone	User Stories
1.0	30/11/2014	1,2,3

Table 42: Policy Recommendation Service Release plan

11.8 Test cases

This section will be provided through the component description in the wiki, when its release 1.0 will be available.

11.9 Installation Manual

Pre-requisite software

- Java Development Kit (JDK 7 and later) [115]
- Maven (Version 2 and later) [116]
- Apache Tomcat server (8 or later) [117]

Pre-requisite software

The recommendation service is provided as a Web application jar file. It can be deployed into containers like Tomcat by simply putting the jar file in Tomcat/Webapps or via the Tomcat administrator interface.

11.10 User Manual

This section will be provided through the component description in the wiki, when its release 1.0 will be available.

12 CONCLUSIONS

This deliverable provides the second version of services and tools described in the framework of WP4. It is also the reference document that describes how to use such software and is designed for all users in order to be able to understand, install and use the following components:

The Resource Catalogue and Recommendation Tool, the SLA Manager, the Interoperability tool, the Security Dashboard, the Infographics and Status Pages, the Security monitoring Generic Enabler, the Cloud portal, the FIWARE LAB APP Template, the Monitoring Dashboard and the Privacy Recommendation Service.

This document includes all the sections per component, even when there are not changes compared to the previous version, to facilitate the comprehension and the use of the components. Hence, this document allows the readers to understand these components through the different sections, from the motivations to the technical details. Due to the same structure, all the components can be analyzed in the same manner: i) a brief description of all components, where they are in the XIFI architecture and what are their dependencies and implications; ii) information on the component leaders (key people to contact in case of questions); iii) the state of the art that provides the analysis of the existing solutions in order to select the components to be used, and based on this, the motivation why the design and development was done; iv) the user stories and when these will be implemented (release plan); v) detailed architecture design that shows the relations among components and the technical details; vi) the test that the developer must execute in order to be sure that the component works properly; vii) the installation manual explains all the steps in order to install the component; viii) reference manual to use the component that includes a description of the API (focused on the developers) and the description of the GUI (more oriented to the End User)

This release has focused on finishing the different components and integrating them with the federated components. The components have been defined to be integrated with the portal, nevertheless the components need to be aligned with the XIFI portal, which will allow accessing information on the offered services by the federation through a single entry point. This integration will be reflected in D4.5 XIFI Marketplace implementation v2 (M21).

The components have increased the relation with the definition of the showcases and the relationships with the business point of view that will be reflected through more interesting showcases, defined in WP6. Furthermore, these components are the basis of part of the learning content, which is defined in WP7. So keeping updated content, in the relevant XIFI wiki pages, is deemed necessary in order to collaborate transversely (cross-WP). This allows emphasizing the assets and competitive advantages of the components. During the next months, the interaction with the others WPs will be the main target.

REFERENCES

- [1] Subversion: <https://xifisvn.res.eng.it/wp4>
- [2] D4.1b- Services and tools specification: [https://bscw.fi-xifi.eu/pub/bscw.cgi/d64426/XIFI-D4.1b-Services and tools specification.pdf](https://bscw.fi-xifi.eu/pub/bscw.cgi/d64426/XIFI-D4.1b-Services%20and%20tools%20specification.pdf)
- [3] D1.1b XIFI core concepts, requirements and architecture draft: [https://bscw.fi-xifi.eu/pub/bscw.cgi/d44695/XIFI-D1.1b-XIFI core concepts requirements and architecture draft.pdf](https://bscw.fi-xifi.eu/pub/bscw.cgi/d44695/XIFI-D1.1b-XIFI%20core%20concepts%20requirements%20and%20architecture%20draft.pdf)
- [4] D2.2 APIs and Tools for Infrastructure Federation v1: [https://bscw.fi-xifi.eu/pub/bscw.cgi/d59218/XIFI-D2.2-APIs and Tools for Infrastructure Federation v1.pdf](https://bscw.fi-xifi.eu/pub/bscw.cgi/d59218/XIFI-D2.2-APIs%20and%20Tools%20for%20Infrastructure%20Federation%20v1.pdf)
- [5] D1.3 Federated Platform Architecture v1: [https://bscw.fi-xifi.eu/pub/bscw.cgi/d58595/XIFI-D1.3-Federated Platform Architecture v1.pdf](https://bscw.fi-xifi.eu/pub/bscw.cgi/d58595/XIFI-D1.3-Federated%20Platform%20Architecture%20v1.pdf)
- [6] XIFI Stakeholders: <https://www.fi-xifi.eu/about-xifi/stakeholders.html>
- [7] Apache License Version 2.0: <http://www.apache.org/licenses/LICENSE-2.0.html>
- [8] UC-2: Setup and use of development environment: [http://wiki.fi-xifi.eu/Xifi:Wp1:requirements:Setup and use of development environment](http://wiki.fi-xifi.eu/Xifi:Wp1:requirements:Setup%20and%20use%20of%20development%20environment)
- [9] Store – Wstore: <http://catalogue.FIWARE.eu/enablers/store-wstore>
- [10] Repository GE: <http://catalogue.FIWARE.eu/enablers/repository-sap-ri>
- [11] Marketplace GE: <http://catalogue.FIWARE.eu/enablers/marketplace-sap-ri>
- [12] Application/Services Ecosystems and Delivery Framework: http://catalogue.FIWARE.eu/enablers?chapter_tid=1
- [13] Revenue Settlement and Sharing System: <http://catalogue.FIWARE.eu/enablers/revenue-settlement-and-sharing-system>
- [14] Application Mashup – Wirecloud: <http://catalogue.FIWARE.eu/enablers/application-mashup-wirecloud>
- [15] Business Calculator GE - iMinds (Belgium): <http://catalogue.FIWARE.eu/enablers/business-calculator-ge-iminds-belgium>
- [16] RuleML: http://wiki.ruleml.org/index.php/RuleML_Home
- [17] Business Modeler: <http://catalogue.FIWARE.eu/enablers/business-modeler>
- [18] Identity Management GE: <http://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.IdentityManagement>
- [19] Dublin Core: <http://dublincore.org/>
- [20] Internet of services, USDL: <http://www.internet-of-services.com/index.php?id=264>
- [21] Linked-USDL: <http://linked-usdl.org/>
- [22] Z39.50: <http://www.loc.gov/z3950/agency/Z39-50-2003.pdf>
- [23] Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH): <http://www.openarchives.org/OAI/2.0/openarchivesprotocol.htm>
- [24] Open Archives Initiative: <http://www.openarchives.org/>
- [25] Content Management Interoperability Services (CMIS): <http://docs.oasis-open.org/cmisis/CMIS/v1.0/os/cmisis-spec-v1.0.pdf>

- [26] ReSTful AtomPub: <http://tools.ietf.org/html/rfc5023>
- [27] WStore Installation and Administration Guide: [https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Store - W-Store - Installation and Administration Guide](https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_Installation_and_Administration_Guide)
- [28] Source of Resource Catalogue: <https://xifisvn.res.eng.it/wp4/Marketplace/trunk/wstore-0.2/>
- [29] Example remote catalogue Search URL: http://130.206.80.235/rest_node_api/search_node/retrieve.json?keys=XX%20type:enabler
- [30] Example remote catalogue URL Base http://130.206.80.235/rest_node_api/enabler_list.json?page=0&field_api_chapter_tid=1
- [31] WStore User & Programmer Guide: [https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Store - W-Store - User and Programmer Guide](https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Store_-_W-Store_-_User_and_Programmer_Guide)
- [32] Remote FIWARE catalogue: <http://catalogue.FIWARE.eu/>
- [33] WS-Agreement standard is a full recommendation of the Open Grid Forum (OGF) defined by the Grid Resource Allocation Agreement Protocol working group (GRAAP--WG) http://www.gridforum.org/Public_Comment_Docs/Documents/2011-03/WS-Agreement-Negotiation+v1.0.pdf
- [34] The Future Internet Public-Private Partnership Programme, <http://www.fi-ppp.eu>
- [35] Modacloud-Analysis of the state of the art and scope definition: <http://www.modaclouds.eu/publications/public-deliverables/>
- [36] WSLA: <http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf>
- [37] WS-Agreement Structure: <http://www.mcs.anl.gov/~keahey/Meetings/GRAAP/WS-Agreement%20Structure.pdf>
- [38] S. Becker, “Quality of service modeling language,” in Dependability Metrics, ser. Lecture Notes in Computer Science, I. Eusgeld, F. Freiling, and R. Reussner, Eds. Springer Berlin Heidelberg, 2008, vol. 4909, pp. 43–47. [Online]. Available: <http://dx.doi.org/10.1007/978-3-540-68947-8>
- [39] D. D. Lamanna, J. Skene, and W. Emmerich, “Slang: A language for defining service level agreements,” in Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, ser. FTDCS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 100–. [Online]. Available: <http://dl.acm.org/citation.cfm?id=795675.797134>
- [40] A. Paschke, “Rbsla a declarative rule-based service level agreement language based on ruleml,” in Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on, vol. 2, nov. 2005, pp. 308–314.
- [41] SLA@SOI: <http://sla-at-soi.eu>
- [42] OPTIMIS: <http://www.optimis-project.eu>
- [43] J. L. Prieto and F. J. Nieto, “A bidirectional trust model for service and infrastructure providers in clouds.”
- [44] Cloud4SOA: <http://www.cloud4soa.eu/>
- [45] Contrail: <http://contrail-project.eu/>
- [46] Mosaic: <http://www.mosaic-fp7.eu/>
- [47] LGPL versión 3.0: <http://www.gnu.org/licenses/lgpl.html>
- [48] Maarten van Steen, Andrew S. Tanenbaum. Distributed systems: principles and paradigms.

Pearson Prentice Hall. 2007. ISBN: 0-13-239227-5

- [49] RESTAssured: <https://code.google.com/p/rest-assured/%7C>
- [50] OSSIM(Open Source Security Information Management): <http://www.alienvault.com/open-threat-exchange/projects>
- [51] Prelude: <http://www.prelude-ids.com/index.php/uk>
- [52] OSSIM: Open Source Security Information Management - <http://www.ossim.net>
- [53] Logalyze: <http://www.logalyze.com>
- [54] Cyberoam: <http://www.cyberoam.com>
- [55] FIWARE Security Monitoring GE Open Specifications: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.SecurityMonitoring>
- [56] Storm:<http://storm-project.net>
- [57] MySQL: <http://www.mysql.com>
- [58] FIWARE Service Level SIEM Open Specifications : <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Security-Monitoring:ServiceLevelSIEMOpenAPISpecification>
- [59] Sanity Check Procedures: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/SecurityMonitoring/ServiceLevelSIEM-InstallationandAdministrationGuide#Sanitycheckprocedures%7C>
- [60] FIWARE Security Monitoring Installation and Administration Guide: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/SecurityMonitoring-InstallationandAdministrationGuide>
- [61] Service Level SIEM User and Programmers Guide: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/SecurityMonitoring/ServiceLevelSIEM-UserandProgrammersGuide%7CFIWARE>
- [62] Service Level SIEM installation and administration guide: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/SecurityMonitoring/ServiceLevelSIEM-InstallationandAdministrationGuide>
- [63] Apache Zookeeper: <http://zookeeper.apache.org/>
- [64] ZeroMQ: <http://www.zeromq.org>
- [65] Daemontools: <http://cr.yip.to/daemontools.html>
- [66] FIWARE Open API Specification: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Security-Monitoring:ServiceLevelSIEMOpenAPISpecification>
- [67] Service Level SIEM User and Programmers Guide: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/SecurityMonitoring/ServiceLevelSIEM-UserandProgrammersGuide%7CFIWARE>
- [68] Cloud Portal Open Specification: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.SelfServiceInterfaces>
- [69] Cloud Portal installation manual: <https://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/SelfServiceInterfaces-InstallationandAdministrationGuide>
- [70] Cloud Portal User manual:

- [http://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Self_Service_Interfaces -
_User and Programmers Guide](http://forge.FIWARE.eu/plugins/mediawiki/wiki/fiware/index.php/Self_Service_Interfaces_-_User_and_Programmers_Guide)
- [71] Hibernate : <http://www.hibernate.org/>
 - [72] Interoperability tool SVN: <https://xifisvn.res.eng.it/wp4/InteroperabilityTool>
 - [73] UC-5: User Support Scenario: http://wiki.fi-xifi.eu/Xifi:Wp1:requirements:User_Support
 - [74] Rackspace: <https://status.rackspace.com/>
 - [75] Amazon: <http://status.aws.amazon.com/>
 - [76] Statshboard: <http://www.stashboard.org/>
 - [77] Overseer: <https://github.com/disqus/overseer>
 - [78] FIWARE lab-infographics source: <https://github.com/SmartInfrastructures/fi-lab-infographics>
 - [79] FIWARE lab-infographics source release: <https://github.com/SmartInfrastructures/fi-lab-infographics/releases/>
 - [80] Phusion Passenger on Apache :
<http://www.modrails.com/documentation/Users%20guide%20Apache.html>
 - [81] Infrastructure Management Scenario : http://wiki.fi-xifi.eu/Xifi:Wp1:requirements:Infrastructure_Management_ScenarioD
 - [82] Keyrock IdM GEi: <https://github.com/ging/FIWARE-idm>
 - [83] Django: <https://www.djangoproject.com/>
 - [84] FIWARE: <http://www.FIWARE.eu/>
 - [85] Apache ALOIS: <http://incubator.apache.org/alouis/>
 - [86] OpenStack Horizon Dashboard: <http://docs.openstack.org/developer/horizon/>
 - [87] FIWARE Lab : <http://lab.FIWARE.eu>
 - [88] jQuery : <http://jquery.com/>
 - [89] UC-1: Joining the federation : [http://wiki.fi-xifi.eu/Xifi:Wp1:requirements:Joining the Federation scenario](http://wiki.fi-xifi.eu/Xifi:Wp1:requirements:Joining_the_Federation_scenario)
 - [90] D3.3: Infrastructures Management Toolkit API: https://bscw.fi-xifi.eu/pub/bscw.cgi/d58587/XIFI-D3.3-Infrastructures_Management_Toolkit_API.pdf
 - [91] Federation Monitoring API: http://wiki.fi-xifi.eu/Public:Federation_Monitoring
 - [92] Python: <https://www.python.org/>
 - [93] Federation Identity Management: http://wiki.fi-xifi.eu/Public:Federated_Identity_Management
 - [94] Federation Manager: http://wiki.fi-xifi.eu/Public:Federation_Manager
 - [95] D4.2 Baseline Tools v1: https://bscw.fi-xifi.eu/pub/bscw.cgi/d58659/XIFI-D4.2-Baseline_Tools_v1.pdf
 - [96] PHP OAuth Client Library, GitHub: <https://github.com/fkooman/php-oauth-client>
 - [97] FIWARE Security Monitoring GE: <http://catalogue.FIWARE.org/enablers/security-monitoring>
 - [98] FIWARE Security Monitoring GE Terms & Conditions: <http://catalogue.fi-ware.org/enablers/terms-and-conditions-4>
 - [99] FIWARE Cloud Portal: <https://cloud.lab.FIWARE.org>
 - [100] Access Control GE: http://wiki.fi-xifi.eu/Public:Access_Control_GE

- [101] OpenStack: www.openstack.org
- [102] Dashboard OpenStack: <https://www.openstack.org/software/openstack-dashboard/>
- [103] Amazon Web Services: <http://aws.amazon.com/es/>
- [104] Heroku: <https://www.heroku.com/>
- [105] Joyent: <http://www.joyent.com/>
- [106] Google Cloud Platform: <https://cloud.google.com/>
- [107] Self-Service Interfaces GE: <http://catalogue.fi-ware.org/enablers/self-service-interfaces-cloud-portal-upm>
- [108] FIWARE Lab App Template: http://wiki.fi-xifi.eu/Public:FI-Lab_App_Template
- [109] Network Active Monitoring: <http://wiki.fi-xifi.eu/Public:NAM>
- [110] Platform for Privacy Preferences (P3P): <http://www.w3.org/TR/2006/NOTE-P3P11-20061113/>
- [111] APPEL : <http://www.w3.org/TR/P3P-preferences/>
- [112] P3pedit: <http://p3pedit.com/>
- [113] COMMON PUBLIC LICENSE: <http://www.eclipse.org/legal/cpl-v10.html>
- [114] Vaadin: <https://vaadin.com>
- [115] Java Development Kit: <http://www.oracle.com/technetwork/java/javase/overview/index.html>
- [116] Maven: <http://maven.apache.org/download.cgi>
- [117] Apache Tomcat server: <http://tomcat.apache.org/>
- [118] Access Control GE: http://wiki.fi-xifi.eu/Public:Access_Control_GE
- [119] FIWAREOps: <http://www.fi-ware.org/fiware-operations/>
- [120] FIWARELab: <http://www.fi-ware.org/lab/>
- [121] XIFI Redmine : <https://redmine.fi-xifi.eu/>
- [122] MIT License : <http://opensource.org/licenses/MIT>