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

eXperimental Infrastructures for the Future Internet

D2.5: APIs and Tools for Infrastructure Federation v2

Revision: v1.0

Work package	WP 2
Task	Task T2.2
Due date	30/09/2014
Submission date	30/09/2014
Deliverable lead	Fraunhofer
Authors	Bernd Bochow (Fraunhofer, Editor), Daniel Nehls (TUB), Alaa Alloush (TUB), Maria Di Girolamo (ENG), Francesco Rossi (ENG), Mirko Ardinghi (ENG), Panos Trakadas (SYNELIXIS), Theodore Zahariadis (SYNELIXIS), Joaquin Iranzo (ATOS), Susana González (ATOS), Cyril Dangerville (THALES), Álvaro Alonso (UPM)
Reviewers	Federico Alvarez (UPM), Sean Murphy (ZAHW)

Abstract	This deliverable describes the set of APIs and tools implementing the federation layer that is to enable specific infrastructures to become nodes of the XIFI federation. Such APIs and tools include those needed to control and coordinate the infrastructures among them and their resources. It is accompanied by related software component design documents. This document updates D2.2 *APIs and Tools for Infrastructure Federation v1* with a focus on updated and newly introduced software component required to implement the federation and their interaction in the scope of building software sub-systems.
Keywords	Federation APIs, Federation Tools, Software Sub-systems, Specifications

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	17.09.2014	First complete draft for WP internal review	Bernd Bochow (Fraunhofer)
V0.2	19.09.2014	Added section on SLA manager and Resource Catalogue, completed formatting API docs, included comments received so far	Bernd Bochow (Fraunhofer)
V0.3	22.09.2014	Included comments received so far. Version for first internal review.	Bernd Bochow (Fraunhofer)
V0.4	28.09.2014	Included first comments from internal review and completed conclusions	Bernd Bochow (Fraunhofer)
V0.5	29.09.2014	Included first comment resolutions and started with exec summary	Bernd Bochow (Fraunhofer)
V0.6	30.09.2014	Included remaining comment resolutions, completed	Bernd Bochow (Fraunhofer)
V0.7	02.10.2014	Included some pending changes and updates (sect. 3.1 and 4.2.1)	Bernd Bochow (Fraunhofer)
V0.8	08.10.2014	Included final comment resolutions	Bernd Bochow (Fraunhofer)
V1.0	08.10.2014	Resolved an issue with page orientation changes	Bernd Bochow (Fraunhofer)

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 License[1].

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

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

Project co-funded by the European Commission in the 7th 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	

© 2013 - 2015 XIFI Consortium Parties

EXECUTIVE SUMMARY

This document concludes on the APIs and Tools for Infrastructure federation pursuing the work started with deliverable D2.2 [1]. The reader should be familiar with XIFI federation concepts introduced, outlined and initially documented by D2.2 since, in order to limit duplication of information across D2.2 and D2.5, this document aims to avoid repetitions by proper referencing rather than adapting source material.

This document is the final deliverable documenting the XIFI APIs and tools for infrastructure federation. The upcoming deliverable D2.6 will further evaluate on the infrastructure federation and will provide related test and validation results that may have an impact on the federation layer sub-systems in consequence.

This document utilizes the notion of a sub-system to document the function and interfaces of software components that are involved in implementing the XIFI federation layer and their interaction. Sub-systems have been introduced along the development of the federation layer as a convenient means to focus on software units that are independently testable, deployable and maintainable omitting unwanted complexity caused by the (intended) flexibility and versatility of the numerous underlying individual software components.

This documentation provides for each sub-system identified a functional description (incl. installation and user manual as well as test instructions), details on the components making up the sub-system and their interaction (incl. mandatory and optional components and their interfaces utilized in the process), and addresses requirements regarding platform services, supporting sub-systems or third-party components. It also details on published sub-systems APIs. The sub-systems covered have been conveniently summarized in Table 1: Sub-systems under consideration.

Although this document is the final documentation of the federation layer's APIs and tools, it is not the final word regarding the development of the federation layer. Besides test and validation in scope of D2.6 further evaluation, verification and refinement will take place in scope of federation operations and maintenance which strongly depends on the feedback on experience and requirements obtained from the individual infrastructure nodes implementing the federation layer "in real".

Bearing that in mind, the document also points to possible follow-up working items that could not be addressed in due detail throughout the development of the federation layer done so far. Two of these potential working items are in federation of non-conventional resources and services, while another working item is in high availability of federation services. Both are of due relevance for the upcoming large-scale operational phase of the XIFI federation.

© 2013 - 2015 XIFI Consortium Parties

TABLE OF CONTENTS

© 2013 - 2015 XIFI Consortium Parties

© 2013 - 2015 XIFI Consortium Parties

LIST OF FIGURES

© 2013 - 2015 XIFI Consortium Parties

© 2013 - 2015 XIFI Consortium Parties

LIST OF TABLES

© 2013 - 2015 XIFI Consortium Parties

ABBREVIATIONS

API	Application Programming Interface
DCA	Deployment and Configuration Adapter
DCRM	Data Center Resource Manager
DEM	Data Center and Enablers Monitoring
FM	Federation Manager
GE	Generic Enabler
GEi	Generic Enabler implementation
GUI	Graphical User Interface
HDFS	Hadoop Distributed File System
IDM	Identity Manager
IDP	Identity Provider
IO	Infrastructure Owner
MDVPN	Multi-Domain Virtual Private Network
MIB	Management Information Base
NAM	Network Active Monitoring
NGSI	Next Generation Service Interface
NPM	Network Passive Monitoring
ODC	OpenStack Data Collector
OSSIM	Open Source Security Information Management
PaaS	Platform as a Service
PEP	Policy Enforcement Point
PMI	PaaS Communication Service
SAML	Security Assertion Markup Language
SDC	Software Deployment and Configuration
SDCI	SDC Client Interface
SIEM	Security Information and Event Management
SLA	Service Level Agreement
SLS	Service Level SIEM
SNMP	Simple Network Management Protocol
SP	Service Provider
SSL	Secure Sockets Layer
URL	Uniform Resource Locator
VM	Virtual Machine
XACML	eXtensible Access Control Markup Language
XIMM	XIFI Infrastructure Monitoring Middleware

© 2013 - 2015 XIFI Consortium Parties

1 INTRODUCTION

This document concludes on the APIs and Tools for Infrastructure federation pursuing the work started with deliverable D2.2 [1]. The reader should be familiar with XIFI federation concepts introduced, outlined and initially documented by D2.2 since, in order to limit duplication of information across D2.2 and D2.5, this document aims to avoid repetitions by proper referencing rather than adapting source material. Although this is particularly true for D2.2 as referenced source material, this is also applying to a multiplicity of other references that provide detailed information and documentation for software components utilized by the federation but not developed with a particular focus on federation. In order to enhance usability of this deliverable, short summaries are provided in that case instead of a detailed description.

1.1 What has changed since Deliverable D2.2

Since submission of D2.2 there has been significant progress in the development of components both in terms of stability and functionality. Major achievements can be observed in the functional extension of the federated monitoring to handle historical data and to provide data persistency, in the Identity Management now considering external identity providers, thus allowing nodes in federation to maintain its authority on local users in particular regarding authorization to access node local non-conventional resources. More details on progress regarding component development are duly provided in scope of section 3 (description) and section 4 (APIs).

Besides due progresses in the development of components that implement the federation layer of XIFI, two major developments have taken place regarding APIs and tools for federation since D2.2 was submitted in M12:

1. Interaction – the notion of a sub-system has been introduced to create and describe separately testable and maintainable collections of components from a plethora of independent and versatile software components with a clear focus on the federation layer. To achieve this, some effort has been put in understanding the interaction of components that make up such a sub-system.
2. Integration – in the course of defining sub-systems, some consideration was given to the development process from component development and testing to sub-system integration and testing, sub-system deployment and sub-system maintenance.

While continuous integration strategies and software sub-system maintenance processes are clearly out of scope for this document, the sub-system approach was considered herein as a guiding principle for the API and tools documentation bridging a number of XIFI work packages from Architecture, to components development and integration towards node operations and maintenance, and finally reflects in the structure of this deliverable document.

A number of topics considered by D2.2 have not been taken up further in the scope of this document. It seems appropriate to conclude on these, since this document is in fact a final report and should not leave open any potentially important issue that was raised by earlier documents. Hence, in the following we provide more details on these topics and why they did not receive further attention. The topics introduced in D2.2 but not further addressed here are in particular:

- Help-desk integration
  In the D2.2 it was said to install an add-on to Jira called "Jira for Nagios". This add-on would have provided a widget integrating Nagios into Jira Dashboard. It was decided not to install it because:
    - This add-on is not open source and XIFI could not manage to support by its release plan.
    - Due to the limitation of accounts available in Jira, the key persons for whom this add-on would have been installed (FIWARE-LAB users) would have not benefit from this

© 2013 - 2015 XIFI Consortium Parties

effort.

- The service given would have been duplicated partly by those provided by the "Infographics and status pages"

Nevertheless, help desk integration is further elaborated in the scope of node operations and maintenance with a slightly different scope (supporting node maintainers instead of all federation users.

- Federation network management
  This has been brought (back) into the scope of component development. Solutions available are considered not yet sufficiently mature for integration with the federation layer. Nevertheless, the components and sub-systems described in this document are sufficiently stable (and flexible, if needed) to integrate with a new sub-system that potentially implements the federation network management, in that utilizing the sub-system APIs as described in this deliverable.
- Test tools
  Test tools have been considered by D2.2 in the light of supporting procedures to join and maintain the federation that is for verifying compliance and for fault detection. This concept is still followed upon but is implemented in scope of multiple other work packages with their distinct purpose:
  - Component API tests and test tools have been brought in scope of the particular component development and are adopted here for the purpose of sub-system testing;
  - Network and connectivity tests and test tools are now considered by node operations and maintenance related activities for practical reasons. These usually are not integrated with other subsystems (i.e. used as stand-alone tools or test suites), components or tools since there is no integrating “maintenance dashboard” yet.

  Regardless of these decisions, the federation layer still is integrating some of these tools or their results when used with or by its sub-systems, For example, by a quick online test and verifying the federation monitoring APIs of a new node right before and after issuing requests to join the federation.

1.2 Document structure

Following sections will first introduce and define the notion of a sub-system following up on the earlier use of this concept in scope of D2.3 [2].

Next, a description of sub-systems will be given outlining for each of them the components it consists of, their internal interaction, and their dependencies regarding a) functional support required to be provided by other sub-systems or platform services and b) third-party components, software packages or software frameworks utilized by the sub-system (i.e. its components).

The documentation of architecture and function, as well as installation, test and use of sub-systems is complemented by the documentation of their APIs in the next section. Modifications and enhancements to the APIs already described in D2.2 are given whenever suitable. Only APIs of core components and sub-systems (i.e. those with a clear focus on the federation aspect) are provided by this document. Many others are referenced or summarized only, in case they are essential for operating the federation or strongly interact or rely upon those core components or sub-systems. Detailed specifications can be obtained through these references.

A dedicated section has been added targeting the documentation of sub-system internal interfaces. The intention of this section was to enable documentation of sub-system characteristics and capabilities part of its internal functionality when, for example, adding an optional component to a sub-system that extends or modifies the sub-system API. This concept is fairly new and no details are currently captured in this deliverable. Nevertheless, the corresponding section has been upheld for discussing

© 2013 - 2015 XIFI Consortium Parties

the concept.

Finally, a detailed description of enhancements made to the identity manager has been provided as an annex. This update adds flexibility to the federated identity management as requested to give the federated infrastructures more flexibility to manage users and credentials by node-local means.

1.3 Document scope

The XIFI federated infrastructure presents itself in two ways: 1) as the FIWARE Lab it is visible to the user (i.e. the FI developer) through the cloud portal as a homogeneous cloud and 2) as an infrastructure of heterogeneous nodes joining in a federation. The latter is usually almost hidden to the less experienced user thanks to the unifying cloud portal. More experienced users in contrast can get full control on how features are deployed towards individual nodes.

This deliverable documents the results of the XIFI project's joint effort to implement a federation of heterogeneous infrastructure nodes suitable and sufficiently stable and robust for productive operations. The tools to achieve this are provided through the software sub-systems detailed here. As discussed above, documentation of the underlying architectural model, component functional description, component ownership, installation manual, user manual, and component test cases, sub-system integration and component interaction in scope of a sub-system are considered in scope of this deliverable. In contrast, conduction of component tests and sub-system tests, the outcome of tests already performed, continuous integration from component to node deployment and maintenance, as well as related topics in software maintenance are documented elsewhere.

The **Federation Manager** (cf. section 3.3.6) and **Federation Monitoring** (cf. section 3.1.1) are considered core sub-systems. Their APIs (sections 4.3.5 and 4.1, respectively) have been therefore provided in some detail.

The following sub-systems are considered as supporting sub-systems in the scope of this document. They are essential but are not core components. That is, they are either providing essential services or strongly depend on services provided by the core sub-systems:

- Identity and access management (section 3.2.1);
- SLA Manager (section 3.3.5);
- Deployment and Configuration Adapter (section 3.4.2);
- PaaS Manager (section 3.4.3);
- SDC (section 3.4.4).

These sub-systems have been developed in scope of other XIFI work packages and have been documented in scope of other deliverables already in detail. Related documentation thus is not duplicated here but is referenced while a short summary of their functionality and APIs is included to assist the reader to understand their role in the scope of the federation layer.

The following are considered supporting sub-systems since there is no implacable need for utilizing them in scope of basic federation mandatory capacities, but a strong requirement for their presence in a sustainable production environment. The documentation in scope of this deliverable thus is constrained and often provided by referencing source material only.

- Security Monitoring (section 3.2.1.5);
- Monitoring Dashboard (section 3.3.1);
- Security Dashboard (section 3.3.2);
- Infographics and status pages (section 3.3.3);
- Cloud portal (section 3.3.4);
- Interoperability tool (section 3.3.7);
- Resource Catalog (section 3.3.8);
- Infrastructure toolbox (section 3.4.1).

© 2013 - 2015 XIFI Consortium Parties

2 THE NOTION OF SUB-SYSTEMS IN XIFI

2.1 Definition of sub-systems

The notion of a sub-system has been introduced in the scope of D2.3 [2] to address the problem of describing collections of tightly interacting components that can be considered self-contained with minimum external dependencies and in consequence can be separately tested and deployed. The concept is under validation in scope of WP5 regarding the requirement also to have independently maintainable collections of components.

The subsystem view is an additional abstraction but is convenient in practice since it allows to handle multiple components jointly (e.g. as a unit in an installation or update procedure) not restricting the versatility gained by having a large number of distinct components at hand. A sub-system thus can be understood conveniently as a bundle of components strongly interacting and depending on each other to implement core functionality of the federation layer.

We define a sub-system as a self-contained unit consisting of one or more components as defined in the XIFI Wiki [3] and zero or more third-party tools or products that are mandatory for implementing all mandatory functions and interfaces of a sub-system. A sub-system implements some useful (potentially minimum) functionality. A sub-system may have different configurations (e.g. realized through optional components) or might provide baseline functionality through mandatory components and may add enhanced functionality through the use of optional components. For the definition of a sub-system the following constraints apply:

- A sub-system is the widest possible consist of components and third-party products that contains any mandatory component which is not a mandatory component of another sub-system – that is, *sub-systems must not have overlapping functionality*;
- A sub-system only relies on infrastructure services, platform services or other self-contained sub-systems (or components) not part of the sub-system under consideration (denoted here as 'supporting sub-system') – that is, *a sub-system may depend on other services (provided by the platform or by other sub-systems)*;
- Sub-systems interact with the platform, the infrastructure or other sub-systems through well-defined, compliant and verifiable interfaces – that is, *a sub-system must have well-defined published interfaces and may utilize a variety of other interfaces for internal purposes*.

Table 1 provides a list of sub-systems further considered by this document and detailed under several aspects in the XIFI Wiki [4].

The interaction of components is defined through the federation architecture; most recently outlined in scope of D1.5 [19], through the federation concepts outlined earlier by D2.2 [1], and through the component related user stories. The interaction of sub-systems clearly derives from the component interaction and produces a level of binding dependency between components which is driven by scope and purpose of a sub-system and how it is used for contributing to the federation. It thus emphasizes on the particular use case of a certain collection of components (here for the purposes of federating) and moves from the concept of a plethora of “general-use software components” towards a “particular application for a selected set of software-components”..

2.2 Managing and deploying sub-systems

The sub-system is a concept defined by the scope within that it is used. So far it has been shown that the same sub-system definition can be utilized for development, testing, deployment and maintenance objectives. From the development and integration perspective, sub-systems have been addressed by D2.3 [2] while from the maintenance perspective sub-systems are covered in D5.3 [6]. This is due to the fact that sub-system deployment, testing and maintenance do have very similar objectives and can be considered in line within the same continuous integration approach exemplary summarized (i.e.

© 2013 - 2015 XIFI Consortium Parties

partially implemented) below:

- Component development and test: Component owners develop (or manage development) and test their components within a proper test environment.

  Parts of this test environment are all the components that make up the sub-system the component under consideration belongs to. This environment might be maintained by the developer or by one or more infrastructures as a testbed. This has been outlined earlier in the scope of D2.3 [2]. In consequence, the component is tested in scope of its later operational environment and can be validated to be compliant with some well-defined (i.e. with proper versioning) set of mandatory, optional and supporting components as well as third-party software prior to deploying it to its operational environment.

- Component deployment: Component owners deploy their components to a software repository into a certain sub-system scope or bundle.

  Components must not be deployed directly to infrastructures in the XIFI federation to avoid potential hazards in node operation (e.g. functional, availability, security ..., which has been observed multiple times during set-up of the five initial nodes). Instead, a software repository is currently set-up acting as a deployment target (basically an Apache Maven [7] repository). Infrastructures then pull complete sub-system bundles from the repository (potentially after further verification through utilizing the testbed).

  Assuming that only one or few components of many are deployed in this step, it is identical with an update procedure. This is, for example, also part of maintenance procedures applied to software packages. It should be noted here that in course of software maintenance also a complete sub-system update might be needed (e.g. in the course of fault management an infrastructure or the federation maintainer may be originator of a sub-system update or rollback). Although not usually part of a development process, this may impact the developer's component test environment.

- Subsystem deployment: Sub-systems are deployed to the operational XIFI federation.

  It seems reasonable to protect the XIFI operational federation from unwanted alteration by immature components. In order to achieve that, the final deployment of sub-systems to the federation must be conducted by the infrastructure owner as part of a maintenance procedure defined in D5.3 [6]. Preferably, this is done by pulling a stable and validated sub-system from the software repository and applying its installation scripts in the course of a scheduled maintenance process.

- Sub-system testing: Automated tests may be applied directly to a sub-system in the software repository.

  As part of a continuous integration process, the sub-system may be scheduled for automated tests utilizing the XIFI testbed [2]. The basic procedure here is to download the full sub-system from the repository, deploy it to a testbed and apply a sequence of automated tests. As a maintenance procedure and depending on the test results, component maintainers or sub-system maintainers (as defined in D5.3 [6]) can be notified to take further actions.

  In the development cycle this may also apply in case of a version discrepancy detected, causing component tests to be repeated occasionally. Also, a component developer may decide to verify backward compatibility of his/her component by testing against different versions of the sub-system (or across sub-systems of distinct development status or version). It should be noted that, while maintenance is handling full sub-systems, developers will download a sub-system, will update the component under consideration in their local copy, will perform their actions on the sub-system, and then will update the sub-system in the repository with an updated component.

© 2013 - 2015 XIFI Consortium Parties

Sub-system		Sub-system components	Sub-system documentation
Monitoring		Context Broker GE, Big Data GE[1], NGSI Adapter, XIMM Adapters (NAM, NPM, DEM, OpenStack Data Collector)	Section 3.1.1 Section 3.1.2 (Components and Adapters) Section 4.1 (API)
Security	Identity and access management	Identity Management GE[2], Keystone proxy[3], Access Control GE[4]	Section 3.2.1 Section 4.2.1 (Identity Management API) Section 4.2.2 (Access Control API) Section A.1 (SAML Extension)
	Security monitoring	Security Monitoring GE, Service Level SIEM[2], SIEM agent[3]	Section 3.2.2 Section 4.2.3 (API)
User Oriented and GUI Subsystems	Monitoring Dashboard	Monitoring Resources Manager agents, Monitoring Configuration and Control tools agents, Monitoring Dashboard GUI, Monitoring Resources Manager, Monitoring Configuration and Control tools	Section 3.3.1 Section 4.3.1 (No published API)
	Security Dashboard	Security Dashboard GUI, Accountability Module	Section 3.3.2 (No published API)
	Infographics and status pages	GUI	Section 3.3.3 Section 4.3.2 (No published API)
	Cloud Portal	GUI	Section 3.3.4 Section 4.3.3 (No published API)
	SLA Manager	SLA dashboard, SLA negotiation, enforcement and decision components	Section 3.3.5 Section 4.3.4 (API)
	Federation Manager	Federation Manager GUI, Federation Manager Core, Quick Online Test	Section 3.3.6 Section 4.3.5 (API)
	Interoperability tool	Test pattern catalogue	Section 3.3.7 Section 4.3.6 (API)
	Resource Catalogue	Recommendation tool, Store - WStore GE, Repository GE	Section 3.3.8 Section 4.3.7 (API)
Deployment	Infrastructure	ITBox Interface, ITBox Middleware	Section 3.4.1

© 2013 - 2015 XIFI Consortium Parties

Sub-system		Sub-system components	Sub-system documentation
& Operations Subsystems	Toolbox		Section 4.4.1 (No published API)
	Deployment and Configuration Adapter	DCA component, Internal Information Repository	Section 3.4.2 Section 4.4.2 (API)
	PaaS Manager GE	PaaS Communication Service (PMI), PaaS Manager Core, SWInstallationManager	Section 3.4.3 Section 4.4.3 (API)
	SDC GE	SDC client interface, ProductManager, ProductInstanceManager, ConfigurationManager	Section 3.4.4 Section 4.4.4 (API)

Table 1: Sub-systems under consideration

© 2013 - 2015 XIFI Consortium Parties

Sub-systems are thus subject to the development process, where they provide a commendable test environment, and since they are defined that way, only interact with other sub-systems already deployed in the operational federation via published APIs. They are also portable environments, which allow performing component tests on different infrastructure, potentially differentiated by implementable test objectives.

2.3 Using sub-systems in the scope of federation APIs and tools

A main characteristic of a sub-system is that it "owns" multiple components interacting via internal interfaces and published and documented APIs that can be utilized by clients and other sub-systems. Clearly, this is due to the scoped view of components and their relation with sub-systems. While it is a valid assumption in the scope of development, testing, deployment and maintenance to consider only sub-system APIs, it may be different in another scope when components are used individually (which is the general use case for FI developers, i.e. XIFI users) or in different sub-system configurations. Any component API considered as sub-system internal before then may become a published or public API. In the definition of sub-systems used in the scope of this document care has been taken that all components are only considered part of exactly one sub-system. This may be needed to be reconsidered in future work.

For the reasons outlined above, sub-systems do not have any meaning in the user perspective but are internal to the XIFI federation. The main benefit is in a significant reduction of complexity since not any possible combination of components needs to be considered but only those that constitute a sub-system. This is a valid assumption for creating and maintaining the federation: sub-systems are tested and functional even if some of their components may fail later in other configurations. It should be noted that testing components in their sub-system environment hence is a scoped and incomplete test. A particular and more detailed view on sub-system testing is provided in scope of D2.3 [2] and will be concluded by the upcoming Deliverable D2.6 complementing D2.3 on test conduction and results. While these documents focus on test concepts, test requirements and test implementations, individual test descriptions are provided by the component’s documentation. For the time being it is assumed that conducting individual (pre-release) component tests is part of a proper development cycle, while the set-up of a sub-system environment for testing in a target environment is part of the validation, deployment and maintenance process mainly conducted by the XIFI infrastructure nodes.

In the scope of this document, API documentation is conveniently split between the two use cases: as a sub-system internal API or as a published or public API. Sub-systems are utilized by communicating through these APIs and implement their functionality through collaborating internal components.

Many sub-systems consist of optional components. They may enhance the capacity of a sub-system through

1. adding functionality without a change of the API or
2. adding capacity and new APIs.

Usually, the latter shows up as a secondary API URL, but it may also constitute as an API extension in case the main component (i.e., the one that implements the API) is capable to act as a proxy and can forward API requests. Both could realize static (i.e. by configuration) or dynamic (i.e. by starting new components) sub-system capacity enhancements.

© 2013 - 2015 XIFI Consortium Parties

3 SUB-SYSTEM DOCUMENTATION

According to the terminology on sub-systems introduced by section 2, and following the overview of sub-systems provided in Table 1 this section will provide the documentation of sub-systems. This section roughly follows the concept of first providing a functional description (incl. installation and user manual as well as test instructions, if applicable) for each sub-system in a dedicated sub-section, Next, the components making up the sub-system and their interaction (incl. mandatory and optional components and their interfaces utilized in the process, if applicable) are outlined. Finally, the corresponding subsection addresses platform services, supporting sub-systems or third-party components that are required for proper operation of a sub-system, Published sub-systems APIs are not included but are further addressed in section 4.

3.1 Monitoring

3.1.1 Federation Monitoring

3.1.1.1 Summary

Federation Monitoring aims at providing a common framework for storing, aggregating and publishing the monitored data collected by the different monitoring adapters provided by the XIMM module. This component is distributed on all the nodes of the federation and elaborates monitoring data leveraging on big data analysis techniques. The following figure highlights this component in the context of the XIFI architecture: the Federation Monitoring component is composed by the blocks in yellow (cf. Figure 1).

Reference Scenarios	UC-5 - Network and Data Centre operations
Reference Stakeholders	This component is aimed to provide monitoring data useful to Infrastructure Owners and FI Developers
Type of ownership	Deployment and Extension
Original tool	Based on FIWARE Context Broker GE and FIWARE Big Data GE
Planned OS license	Apache License Version 2.0[2]
Reference OS community	none at the moment

Consists of

- API specifications and implementation for accessing federation monitoring data
- Data aggregation functions implemented via MapReduce framework
- FI-WARE BigData Analysis - COSMOS[3]
- FI-WARE Publish/Subscribe Context Broker - Orion[4]

[2] http://www.apache.org/licenses/LICENSE-2.0.html

[3] http://catalogue.fi-ware.org/enablers/bigdata-analysis-cosmos

[4] http://catalogue.fi-ware.org/enablers/publishsubscribe-context-broker-orion-context-broker

© 2013 - 2015 XIFI Consortium Parties

- FI-WARE ContextBroker to BigData GEs connector - ngsi2cosmos[5]

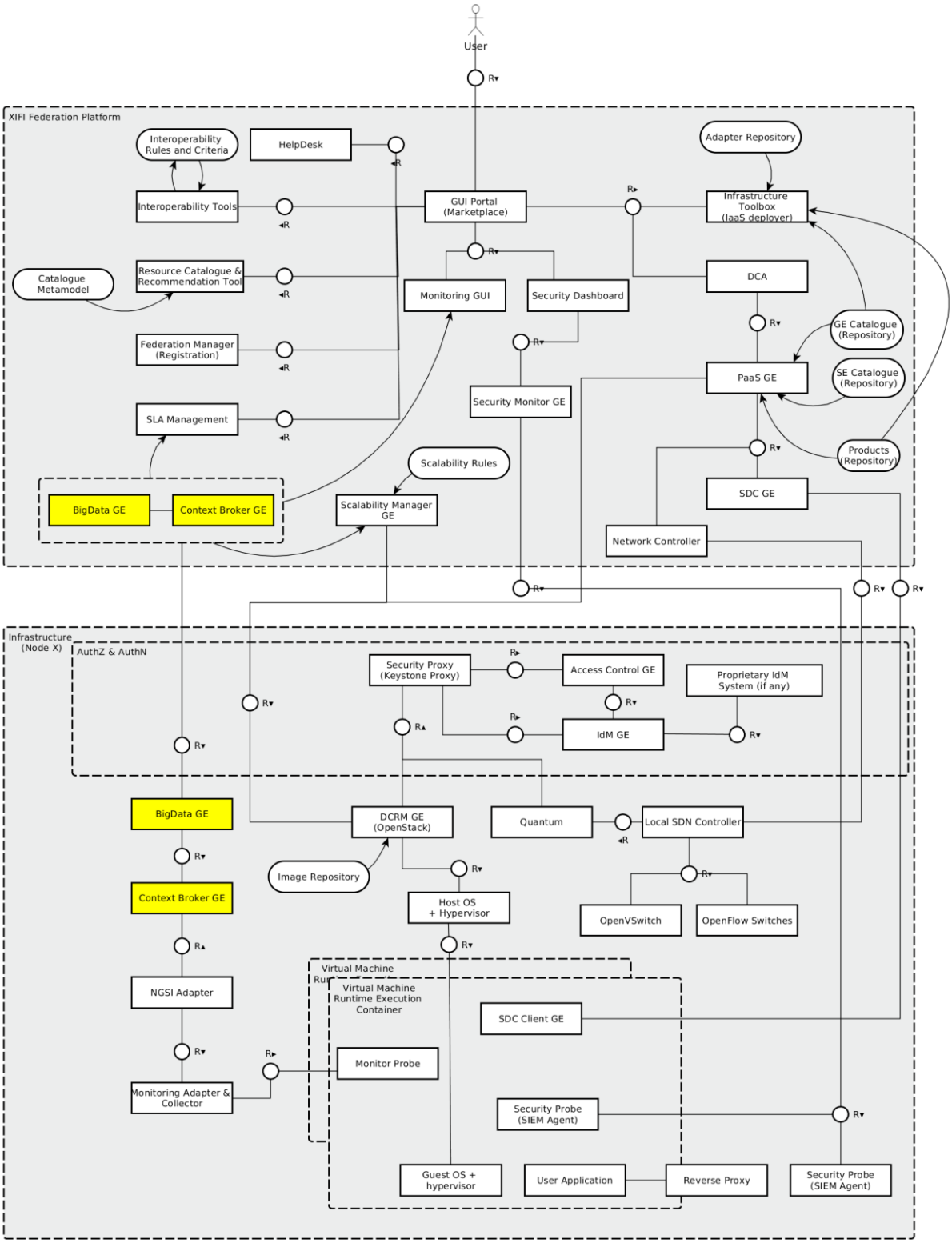


Figure 1: Federation Monitoring architectural context.

[5] https://github.com/telefonicaid/fiware-livedemoapp#ngsi2cosmos

Depends on

- XIMM modules (Network Active Monitoring-NAM[6], Network Passive Monitoring-NPM[7], Data Center & Enablers Monitoring-DEM[8]) – see D3.2 [5] and D3.5 [21]
- FI-WARE NGSI Adapter[9]

3.1.1.2 Component responsible

Developer	Email	Company
Attilio Broglio	abroglio@create-net.org	CREATE-NET

3.1.1.3 Motivation

This component implements the monitoring functionality at the federation level, providing a common view and data model of the monitoring data. This component elaborates the raw data collected by the individual monitoring adapters, provide historical support and persistence and offer some aggregation functions on the data.

3.1.1.4 User stories backlog

Table 2 provides the complete list of user stories. With respect to D2.2 [1] a user story has been added related to the power consumption.

User story id	User story name	Actors	Description	Task id
1	Provide OpenStack information for each region	Infographics and Status Pages Component	Federation Monitoring should be able to provide information coming from OpenStack Data Collector like #core, #ram, #disk, #VM, #users and status of xifi core services for each region of the federation	623
2	Provide OpenStack information aggregated at federation level	Infographics and Status Pages Component	Federation Monitoring should be able to provide information coming from OpenStack Data Collector like #core, #ram, #disk, #VM, #users aggregated at federation level (i.e. sum of the info collected for each region)	624
3	Provide historical host monitoring data aggregated on each region	Infographics and Status Pages Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each host of the federation, aggregated per region so as to have an	625

[6] http://wiki.fi-xifi.eu/Public:NAM

[7] http://wiki.fi-xifi.eu/Public:NPM

[8] http://wiki.fi-xifi.eu/Public:DEM

[9] http://catalogue.fi-ware.org/enablers/monitoring-ge-tid-implementation

© 2013 - 2015 XIFI Consortium Parties

User story id	User story name	Actors	Description	Task id
			averaged value of each host measure for the entire region	
4	Provide real time monitoring data for each host of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each host of the federation, in real time	626
5	Provide historical monitoring data for each host of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each host of the federation, aggregated on an hourly basis so as to have averaged values of each host measure	627
6	Provide real time monitoring data for each VM of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each VM of the federation, in real time	628
7	Provide historical monitoring data for each VM of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each VM of the federation, aggregated on an hourly basis so as to have averaged values of each VM measure	629
8	Provide real time monitoring data for each service of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each service of the federation, in real time	630
9	Provide historical monitoring data for each service of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each service of the federation, aggregated on an hourly basis so as to have averaged values of each service measure	631
10	Provide real time monitoring data for each network element (at the moment only *interfaces*) of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each network element of the federation, in real time	632
11	Provide historical monitoring data for each network element (at the moment only *interfaces*) of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each network element of the federation, aggregated on an hourly basis so as to have averaged values of each network element measure	633

© 2013 - 2015 XIFI Consortium Parties

User story id	User story name	Actors	Description	Task id
12	Handle active measures involving two hosts (both real time and historical)	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM - NAM module from couples of hosts in the federation, related to the links between two hosts	634
13	Provide historical monitoring data for other network elements (not only *interfaces*) of the federation	Monitoring Dashboard Component	Federation Monitoring should be able to provide monitoring data, collected by XIMM for each network element of the federation, aggregated on an hourly basis so as to have averaged values of each network element measure	635
14	Provide an extension of the previous functionalities (when appropriate) implementing multi tenancy	Monitoring Dashboard Component	A set of APIs should be developed in order to query the monitoring data on a per tenant basis. This means that when a request is made with a tenant_id as an argument, only the data pertaining to that tenant will be retrieved. This is in particular related to the VMs and services associated to a given tenant.	644
15	Provide accounting data for each tenant	Monitoring Dashboard Component	Federation Monitoring should provide accounting data collected by XIMM for each tenant. Accounting data should consider: number of virtual machines for each tenant, ram/cpu/disk usage for each VM of a tenant, uptime for each VMs of a tenant.	666
16	Provide aggregated accounting data for each tenant	Monitoring Dashboard Component	Federation Monitoring should provide aggregated accounting data collected by XIMM for each tenant. Aggregated accounting data should consider: number of virtual machines for each tenant, average ram/cpu/disk usage on all VMs of a tenant, sum of uptime for all VMs of a tenant.	667
17	Provide historical monitoring data on power consumption of the nodes	Monitoring Dashboard Component	Federation Monitoring should provide power consumption info for all the nodes of the federation. Data should be collected through monitoring adapters and provided to Federation Monitoring.	1037

Table 2: User stories (Federation Monitoring)

3.1.1.5 State of the art

This component is based on well-known big data analysis techniques offering distributed file system functionalities (HDFS - Hadoop Distributed File System). The components, used in order to implement Federation Monitoring, are provided by FI-WARE as Generic Enablers. Since XIFI is

© 2013 - 2015 XIFI Consortium Parties

committed to leverage on the FI-WARE GEs, we selected these GEs in order to implement this component.

The data model defined is quite simple but sufficient at the moment to model the relevant data of a data center. Some well-known "standards" as SNMP MIB and DMTF CIM[10] have been considered even though the derived model covers them only in a minimal part. In any case the model is not bounded to any specific technology and is quite extensible and, if needed, in the future it can be extended in order to monitor more resources.

3.1.1.6 Architecture design

The architectural design of this component is depicted in Figure 2. The design is mainly the one already presented in D2.2 [1]: the only changes are related with the Big Data Environment.

On each slave node of the federation the following component will be installed:

- Context Broker GE (Orion GEi)
- Big Data Environment (Big Data GE and ngsi2cosmos connector)

On each master node, in addition to the previous components, the following component will be installed:

- Context Broker GE (Orion GEi)
- API Server: in order to provide an implementation of the APIs for accessing monitoring data
- Big Data Environment (Big Data GE and ngsi2cosmos connector)

Each slave *Context Broker* is connected also to the master *Context Broker* in order to feed the master *Context Broker* with "raw" (i.e. not elaborated) monitoring data directly. In this way "real time" data is forwarded to the data consumers through the *API Server* avoiding any possible delay. On the other hand, each node elaborates and aggregates its monitoring data (in NGSI fashion) coming from the XIMM modules installed on the node, and finally stores its historical data. In order to do that each master or slave node has a *Big Data Environment* that periodically retrieves information from the local Context Broker and imports it in the HDFS file system. The Big Data Environment is composed by a name-node (the Hadoop master) and one or more data-nodes (Hadoop slaves) based on the Big Data GE.

Figure 3 provides an overview of the monitoring data model considered. The APIs developed for accessing the monitoring data are based on this model. With respect to the D2.2 [1], the data model has been modified in order to take into account the last user story added (i.e. 17).

Referring to the previous model, it is important to highlight how each object that provides monitoring data is derived from a generic *Monitored Object* that aggregates a set of *Measures* each one of them having a *Measure Type*. This means that each measure present in the different monitored objects under the word "*Measures:*" should be considered having the following structure:

- name (mandatory)
- value (mandatory)
- description (optional)
- aggregationOperator (optional)

The *aggregationOperator* identifies the operation of aggregation applied. At the moment we foresee three types of operations:

[10] http://www.dmtf.org/standards/cim

© 2013 - 2015 XIFI Consortium Parties

- sum
- average on time (per hour)
- average on the same family of resources (for example all the hosts belonging to a region)
- average on a per tenant basis

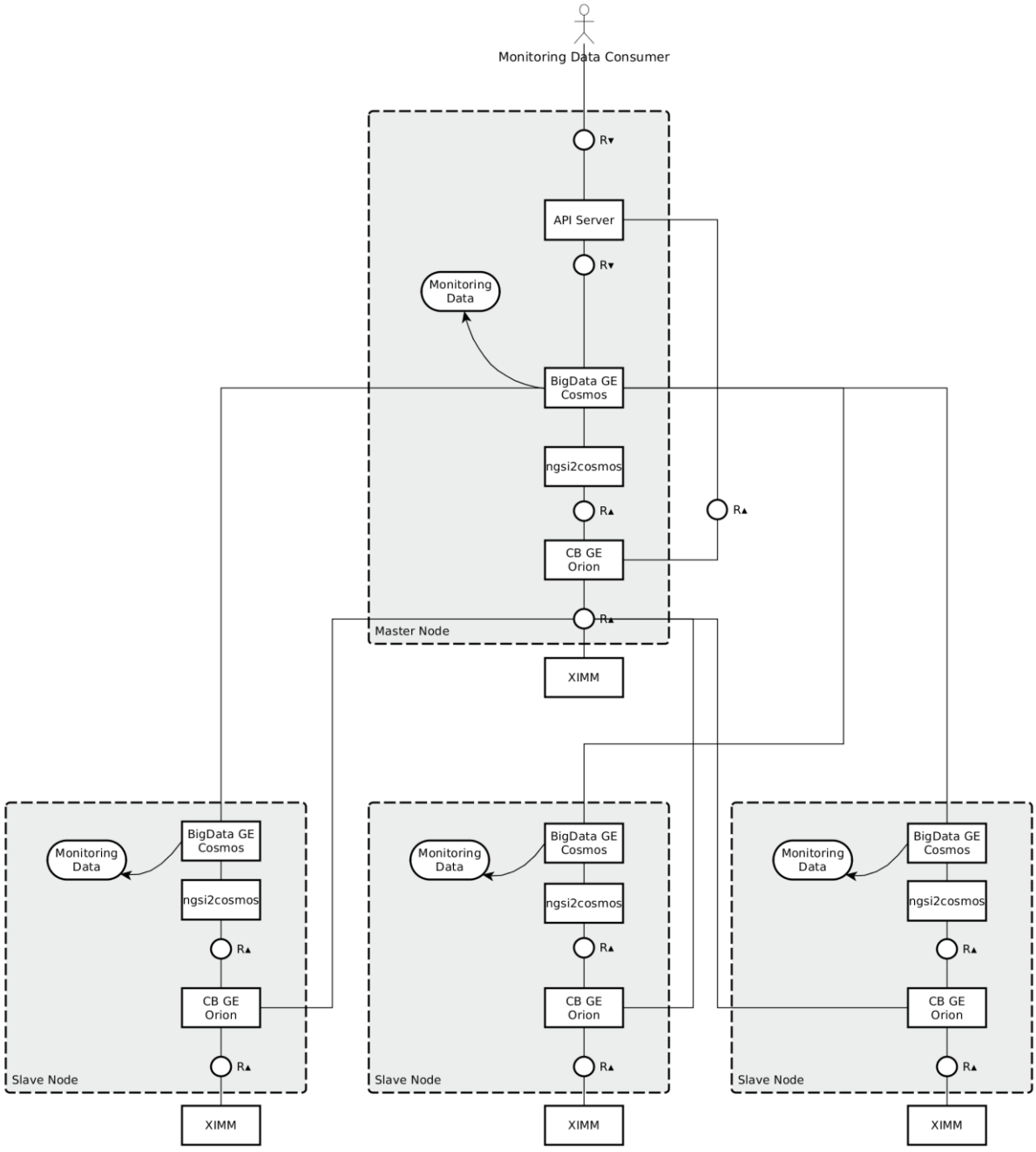


Figure 2: Federation monitoring architecture

The way a *Monitored Object* is identified deserves a clarification: each Monitored Object should be identified by an hierarchical id (e.g. for a VM the id should be composed by the regionid and vmid) in the form *parent-id:child-id*. A particular case is the the Monitored Object *Host2Host*: it represents a monitored connection between two hosts that could also be cross-region. The id of this object is to composition of the ids of two hosts in the form *regionid-hostid;regionid-hostid*.

© 2013 - 2015 XIFI Consortium Parties

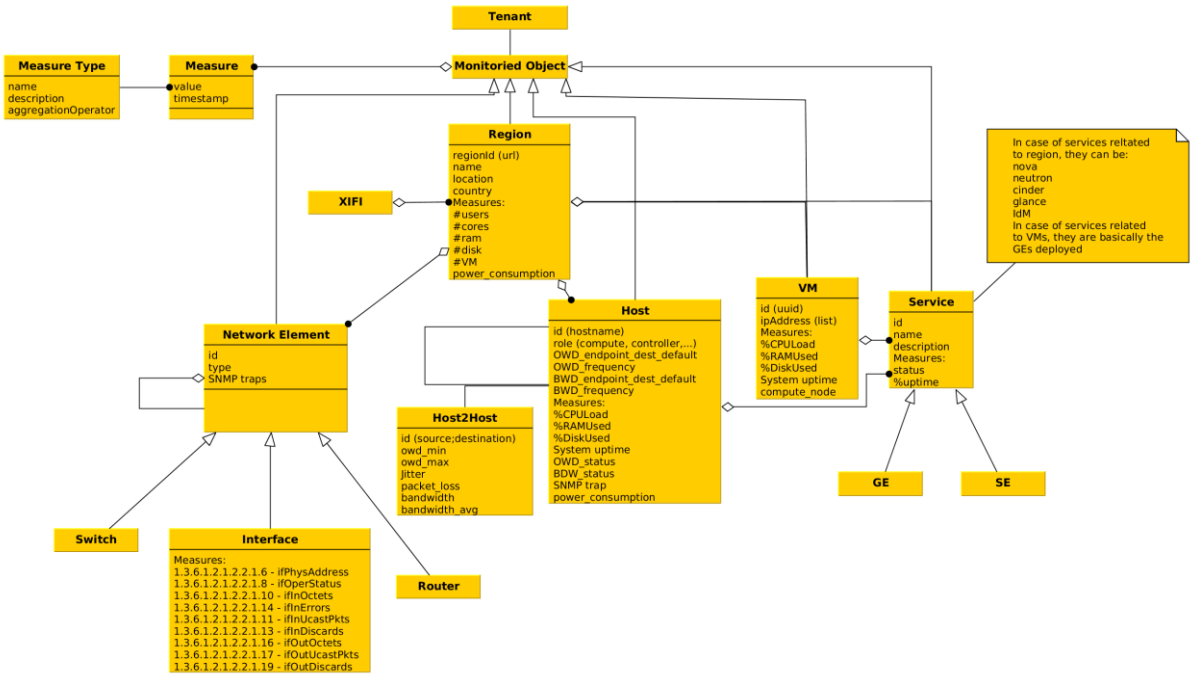


Figure 3: Federation monitoring data model

3.1.1.7 Release plan

Version Id	Milestone	User Stories
1.0	M9	1, 2, 4, 6, 8, 10
2.0	M12	3, 5, 7, 9, 11
3.0	M15	12, 13, 14
4.0	M18	15, 16, 17

3.1.1.8 Test cases

The test cases given in Table 3 can be replicated using a script available from https://github.com/SmartInfrastructures/xifi-script/testAPI.js.

In order to run it, you need to require and to set the ConsumerKey and the ConsumerSecret in the script.

The script can be launched with: *node testAPI.js <username> <password> <apiPath>*

Note that when in the "Expected results" column it is mentioned that the output expected is defined in the API definition (see User Manual), it is meant that the resulting output (int the form of a json snippet) is provided there together with the API definition.

© 2013 - 2015 XIFI Consortium Parties

Test id	Test description	Test script	Expected results
1	Test API /monitoring/regions	node testAPI.js .. /monitoring/regions	Output expected by the API under test - see User Manual section
2	Test API /monitoring/regions/{regionid}{?since} with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}{?since}	Output expected by the API under test - see User Manual section
3	Test API /monitoring/regions/{regionid}/hosts	node testAPI.js .. /monitoring/regions/{regionid}/hosts	Output expected by the API under test - see User Manual section
4	Test API /monitoring/regions/{regionid}/hosts/{hostid}{?since} with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}/hosts/{hostid}{?since}	Output expected by the API under test - see User Manual section
5	Test API /monitoring/regions/{regionid}/vms	node testAPI.js .. /monitoring/regions/{regionid}/vms	Output expected by the API under test - see User Manual section
6	Test API /monitoring/regions/{regionid}/vms/{vmid}{?since} with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}/vms/{vmid}{?since}	Output expected by the API under test - see User Manual section
7	Test API /monitoring/regions/{regionid}/services	node testAPI.js .. /monitoring/regions/{regionid}/services{?since}	Output expected by the API under test - see User Manual section
8	Test API /monitoring/regions/{regionid}/hosts/{hostid}/services	node testAPI.js .. /monitoring/regions/{regionid}/hosts/{hostid}/services	Output expected by the API under test - see User Manual section
9	Test API /monitoring/regions/{regionid}/hosts/{hostid}/services/{serviceName}?since with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}/hosts/{hostid}/services/{serviceName}?since	Output expected by the API under test - see User Manual section

© 2013 - 2015 XIFI Consortium Parties

Test id	Test description	Test script	Expected results
10	Test API /monitoring/regions/{regionid}/vms/{vmid}/services	node testAPI.js .. /monitoring/regions/{regionid}/vms/{vmid}/services	Output expected by the API under test - see User Manual section
11	Test API /monitoring/regions/{regionid}/vms/{vmid}/services/{serviceName}?{since} with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}/vms/{vmid}/services/{serviceName}?{since}	Output expected by the API under test - see User Manual section
12	Test API /monitoring/regions/{regionid}/nes	node testAPI.js .. /monitoring/regions/{regionid}/nes	Output expected by the API under test - see User Manual section
13	Test API /monitoring/regions/{regionid}/nes/{neid}{?since} with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}/nes/{neid}{?since}	Output expected by the API under test - see User Manual section
14	Test API /monitoring/host2hosts	node testAPI.js .. /monitoring/host2hosts	Output expected by the API under test - see User Manual section
15	Test API /monitoring/host2hosts/{source};{dest}{?since} with and without parameter {since}	node testAPI.js .. /monitoring/host2hosts/{source};{dest}{?since}	Output expected by the API under test - see User Manual section
16	Test API /monitoring/regions/{regionid}/services{?since} with and without parameter {since}	node testAPI.js .. /monitoring/regions/{regionid}/services{?since}	Output expected by the API under test - see User Manual section

Table 3: Test cases (Federation Monitoring)

© 2013 - 2015 XIFI Consortium Parties

3.1.1.9 Installation Manual

3.1.1.9.1 Prerequisites

- Ubuntu 12.04 as operating system
- See prerequisites for Context Broker GE (Orion)[11] and Big Data GE (COSMOS)[12].
- The hardware characteristics of the machine where to install a slave node is:

CPUs	4
RAM	16GB
Hard Disk	1TB

- For the master node it is suggested to increase the disk size to 2 TB.
- At least two VMs for the *BIG Data environment* for each node

VCPUs	2
RAM	4096MB
User Disk	30GB
Ephemeral Disk	30GB

3.1.1.9.2 Installation Procedure

On each node of the federation:

- Install and run the Context Broker GE (Orion) following the instruction available from http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker
- Contact the master-node administrator in order to subscribe *Context Broker* present on slave nodes with the master node *Context Broker*

Only on the master nodes:

- install Node.js[13]
- download *monitoringAPI.js* from here[14] and put it in a directory of your choice (suggestion */usr/local/monitoringAPI*)
- from the directory where you put *monitoringAPI.js*, run: *$INSTALL_DIR_NODEJS/bin/node monitoringAPI.js*, where *INSTALL_DIR_NODEJS* is the installation directory of Node.js (you can consider to run it as a daemon).

On each node:

- retrieve the two VM images (ambari-master, ambari-datanode) available from https://xifisvn.res.eng.it/wp2/software/trunk/Components/FederationMonitoring/API/.
- instantiate (at least two) VMs for the BIG Data Environment, one for the hadoop-namenode,

[11] http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker

[12] http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos

[13] http://nodejs.org/

[14] https://xifisvn.res.eng.it/wp2/software/trunk/Components/FederationMonitoring/API/

© 2013 - 2015 XIFI Consortium Parties

one or more for the hadoop-datanode, using the previous images

- Contact component owner in order to get username/password for the first login

Note1: perform this action as root

Note2: remember to set the region name, when requested, to the proper value. For example:

Trento	Berlin	Waterford	Spain	Lannion	Prague

On the hadoop-datanode/s:

- edit the file "/etc/hosts with the private IP of the (namenode and datanode)

```
vim /etc/hosts
10.10.10.10 testregion-master testregion-master.xifi
10.10.10.11 testregion-datanode1 testregion-datanode1.xifi
```

- set the hostname according to the previous parameters

```
hostname testregion-datanode1
```

- set the networking configuration file

```
vim /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=testregion-datanode1
```

On the hadoop-namenode:

- edit the file "/etc/hosts with the private IP of the (namenode and datanode)

```
vim /etc/hosts
10.10.10.10 testregion-master testregion-master.xifi
10.10.10.11 testregion-datanode1 testregion-datanode1.xifi
```

- set the hostname accordingly to the previous parameters

```
hostname testregion-master
```

- set the networking configuration file

```
vim /etc/sysconfig/network
NETWORKING=yes
HOSTNAME=testregion-master
```

- disable the iptables for the first installation

```
chkconfig iptables off
/etc/init.d/iptables stop
```

- check the ssh configuration

```
ssh root@testregion-datanode1
```

- On this node, it is installed a framework that helps to manage and setup the hadoop installation. This tool is ambari[15] and in order to make it working you have to:

```
ambari-server setup
  WARNING: SELinux is set to 'permissive' mode and temporarily
   disabled.
```

[15] http://ambari.apache.org

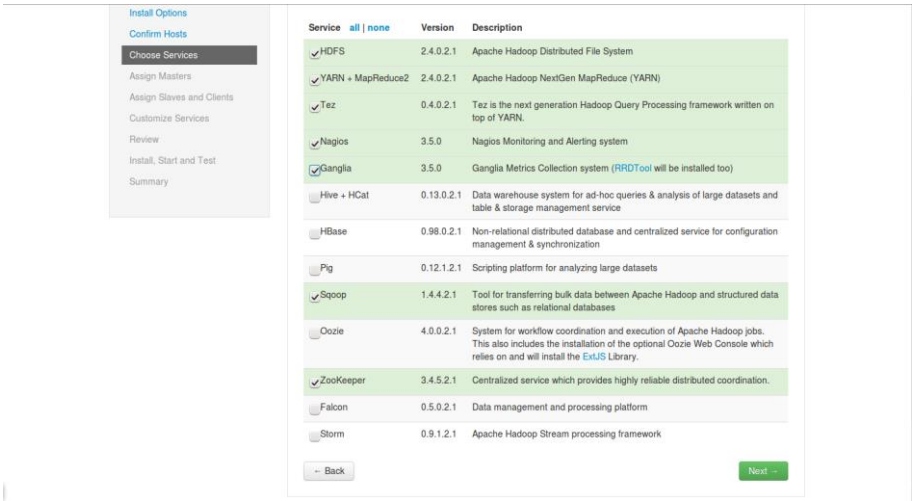
© 2013 - 2015 XIFI Consortium Parties

```
  OK to continue [y/n] (y)? y
Do you want to change Oracle JDK [y/n] (n)? y
  Enter choice (1): 1
Enter advanced database configuration [y/n] (n)? y
  Enter choice (1): 1
  Database Name (ambari): xyz
  Username (ambari): xyz
  Enter Database Password (bigdata): xyz
```

- restart ambari

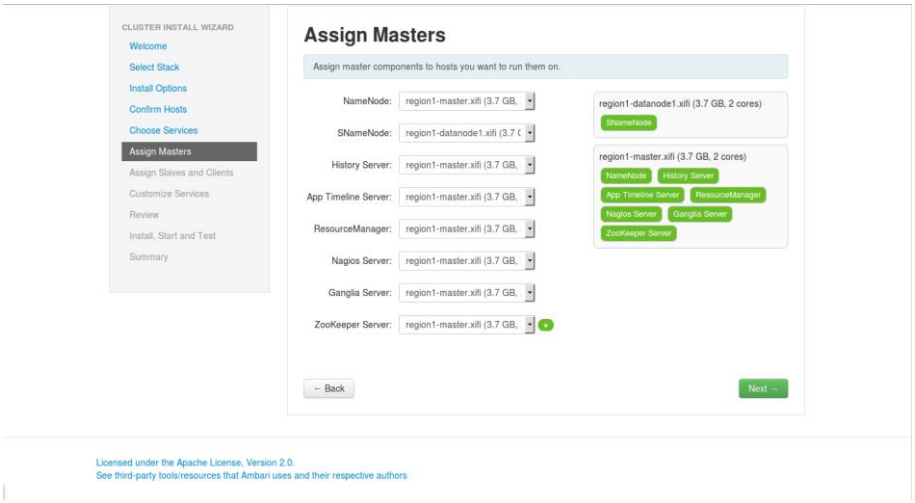
```
ambari-server restart
```

- Open a browser and reach the url namenode-publicIP:8080
  - login as admin/admin
  - set the region name
  - select HDP2.1
  - insert the hostname
    - testregion-master.xifi
    - testregion-datanode1.xifi
  - insert the ssh private key that you can find at the master node at "~/.ssh/id_rsa"
  - select a subset of service inside ambari:
    - HDFS
    - YARN
    - Tez
    - Nagios
    - Ganglia
    - Sqoop
    - ZooKeeper

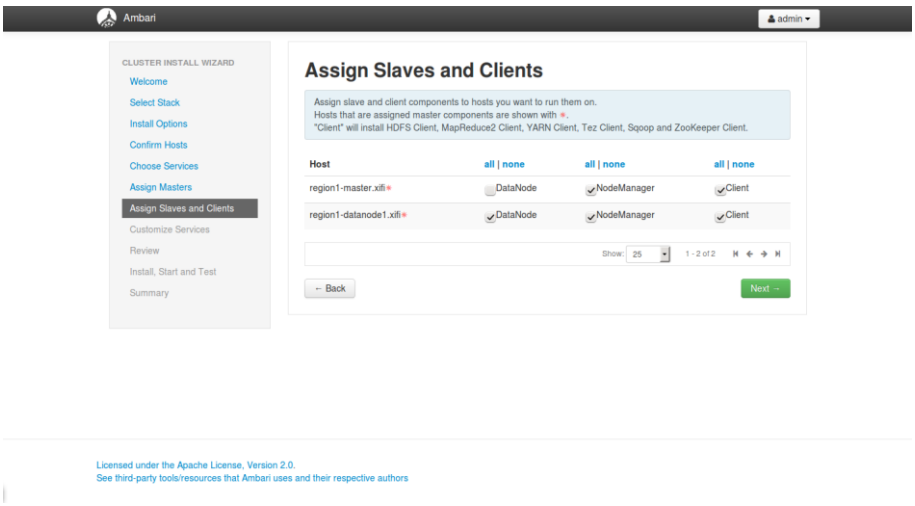


© 2013 - 2015 XIFI Consortium Parties

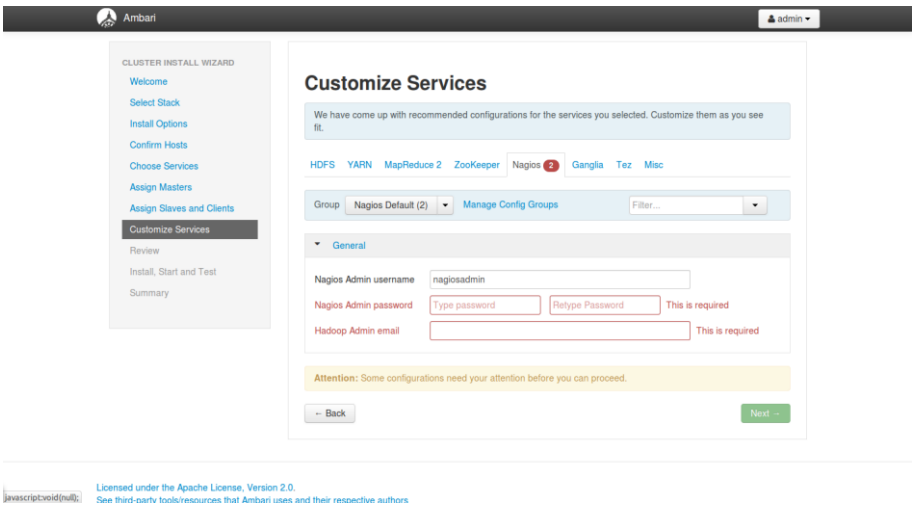
- set the namenode as the master for all the services:



- set the namenode to be (nodeManager and Client) and set the datanode to (DataNode, NodeManager and Client)



- create the nagios account



On the namenode:

- change the following parameter in the file /etc/init.d/ngsi2cosmos

```
[row 8] REGION="regionName"
```

- start the service /etc/init.d/ngsi2cosmos start
- configure these parameters in the script "/etc/monitoring/scheduledmonitoring.py"

```
[row 5] region='regionName'
[row 6] dbname='xyz'
[row 7] username='xyz'
[row 8] password='xyz'
ask to the component owner the dbname/username/password
```

- create the cronjob and schedule it as normal user (i.e. hdfs):

```
[30 0 * * * python /etc/monitoring/scheduledmonitoring.py]
```

- configure the parameters in the script /etc/monitoring/subscribeHadoop.py and run it only ONCE (python /etc/monitoring/subscribeHadoop.py)

```
[row 4] region="regionName"
  ''where CBurl is the ip address:port of the local contextBroker''
[row 5] CBurl="x.y.v.z:port"
  ''where hadoopMasterIP is the ip:5050 of the namenode''
[row 6] hadoopMasterIP="x.y.v.z:5050"
```

3.1.1.10 User Manual - Federation Monitoring API

The current and up to date version of the **XIFI Federation Monitoring** APIs can be found here[16]. The syntax used to define the APIs is the one proposed by apiary[17].

HOST (where the API are deployed): http://monitoring.fi-xifi.eu (not yet working from internet because hostname has to be registered). You can test them at 193.205.211.69:1026

A snapshot of the XIFI Federation Monitoring APIs can be found the API section of this document.

Note: all these APIs are now protected by a proxy that uses the OAuth2 authorization protocol through the IDM credential. So FIWARE Lab account is required in order to use them.

3.1.2 Sub-system components

3.1.2.1 NGSI Adapter

NGSI Adapter is the component in charge of parsing multiple data sources into an agreed standard NGSI-based format. In the context of the XIFI Monitoring Architecture, raw monitoring data will be published to the Orion Context Broker of the Federation Monitoring via this adapter. To find a full specification of the component the reader shall check M18 deliverable D3.5 [21], or its corresponding wiki page[18].

3.1.2.2 NAM Adapter

Network Active Monitoring (NAM) Adapter offers an end-to-end service to the federation in terms of monitoring. It manages cross-domain tests between the instances where it is deployed, providing a standard multi-domain monitoring mechanism able to handle latency and bandwidth-related tests.

[16] http://docs.federationmonitoring.apiary.io/

[17] http://apiary.io/

[18] http://wiki.fi-xifi.eu/Public:NGSIAdapter

© 2013 - 2015 XIFI Consortium Parties

Further description and details can be found in M18 deliverable D3.5 [21], or its corresponding wiki page[19].

3.1.2.3 DEM Adapter

Datacenter and Enablers Monitoring (DEM) adapter is responsible for gathering monitoring data from Virtual Machines (VMs) and Generic Enablers (GEs) deployed within the XIFI federated infrastructure and pass them to the Federation Monitoring. To find a full specification of the component the reader shall check M18 deliverable D3.5 [21], or its corresponding wiki page[20].

3.1.2.4 NPM Adapter

NPM Adapter is in charge of collecting performance data relative to passive monitoring of network devices of each infrastructure. Its specification can be found in M9 deliverable D3.2 [22], or its wiki page[21].

3.1.2.5 OpenStack Data Collector

This component collects capacity data from the OpenStack instance deployed in the infrastructure, such as number of cores installed, size of RAM and disk and number of virtual machines deployed. Further details can be found in M18 deliverable *D3.5- Infrastructures monitoring and interoperability adaptation components API v2*, or its corresponding wiki page[22].

3.1.3 Sub-system internal component interaction

Figure below depicts a general overview of how the different components belonging to this sub-system are related to each other.

1. NGSI Adapter:
    - As described above, it is the link between the Federation Monitoring and the components that collect the monitoring data. It acts as an Adaptation Layer that normalizes monitoring data into a common format (NGSI) before publishing to the Federation Monitoring.
    - It leverages on multiple parsers, integrated in the same software package, to standardize each particular format from heterogeneous sources.
2. NAM-DEM-NPM-ODC Adapters:
    - These components have no interaction between each other.
    - They are required to publish the data collected to the NGSI Adapter.
3. Federation Monitoring:
    - Orion Context Broker at each particular Infrastructure Node receives real time data (in NGSI format and following the data model presented in the previous section) originated by the monitoring adapters.
    - Connector "ngsi2cosmos" is subscribed to Orion Context Broker and is notified each

[19] http://wiki.fi-xifi.eu/Public:NAM

[20] http://wiki.fi-xifi.eu/Public:DEM

[21] http://wiki.fi-xifi.eu/Public:NPM

[22] http://wiki.fi-xifi.eu/Public:OpenstackDataCollector

© 2013 - 2015 XIFI Consortium Parties

time data is updated. Such updates become part of the historical data that is elaborated and written to storage using BigData GE.

- At Master Node, Orion Context Broker is subscribed to each of the Infrastructure Nodes' Context Brokers, so that it is also notified of updates and it aggregates monitoring data from the federation.

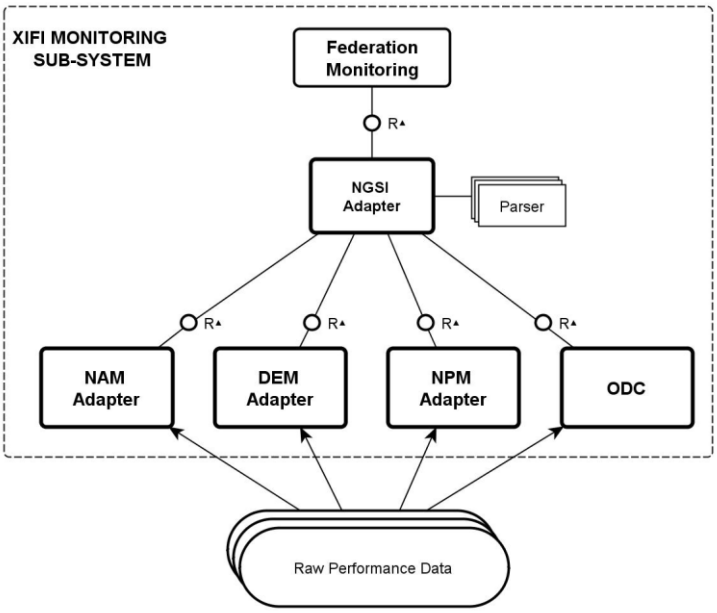


Figure 4: XIFI Monitoring General Architecture

3.1.4 Sub-system dependencies - Supporting sub-systems

1. Monitoring Dashboard:
   - Provides by graphical means the monitoring information collected and published by the Monitoring sub-system, both real-time and historical data.
2. Infographics and Status Page:
   - Same as the Monitoring Dashboard, this user oriented GUI is fed by the data of the Monitoring sub-system.
3. Infrastructure Toolbox:
   - Monitoring components will be embedded in the Infrastructure Toolbox to become avaible in the deployment of the node.
4. Identity and Access Management:
   - Monitoring uses the IDM component (Federated_Identity_Management[23]) in order to authenticate and authorize a user through the OAuth2 protocol (http://oauth.net/2/).

3.1.5 Sub-system dependencies - Third party components and frameworks

1. Monitoring Probes:
   - Monitoring Adapters require these probes which are the actual measurement tools, i.e.

[23] http://wiki.fi-xifi.eu/Public:Federated_Identity_Management

the performance data sources.

2. OpenStack Data Collector:
    - gather data from OpenStack (http://www.openstack.org/)

3.2 Security

3.2.1 Identity and access management

3.2.1.1 Components

The Identity and Access Management (IAM) subsystem consists of three components:

- Federated Identity Management;
- Security Proxy;
- Federated Access Control (optional).

3.2.1.2 Identity Management

As defined in the D2.2 [1] Identity Management (Federated) provides a federated management of identities for members of XIFI federation. It is distributed on all the nodes of the federation and stores identities of authorized users. The Federated Identity Management component is an extension the FI-WARE IdM GE to support XIFI requirements, for more details about the architecture, dependencies refer to D2.2.

Since submission of D2.2 an update of the IdM GE has been provided that focuses on the enhancement of the identity federation features and extends the component with SAML 2.0 protocol support.

This extension is considered relevant for the two main aspects described below

1. Make the GE more attractive by allowing to delegate authentication to an external party. Here the choice of a widespread and well accepted protocol as the SAML 2.0 is key in order to ease integration with existing authentication mechanisms.
2. Improve the identity federation across the XIFI nodes and distribute the functionality of user authentication: to each node might support local users and local IdM but, at the same time, each IdM instance can allow access to its node services to all the federation users by delegating authentication to the original user node.

As introduced in the deliverable D1.5 [19] and D2.4 [20], the main changes for this component are:

- SAML 2.0 support,
- implementation through FI-WARE KeyStone Proxy of the Security Proxy that provides proxy authentication to DCRM GE services toward the IdM GE. Federation
- implementation of HA (both backend and frontend)
- Keyrock IdM GEi architecture based on Galera
- implementation of the Virtual IP service

The first step was identified how SAML2.0 support the authentication for a Web Browser 2.0 and defined the architecture about IDM support SAML SP and SAML IDP.

Further information can be found in Appendix A.

© 2013 - 2015 XIFI Consortium Parties

3.2.1.3 Federated Access Control

Federated Access Control provides XACML-standard-compliant authorization services for each XIFI node. The service is usable via the FIWARE Access Control GE API, and the location of the GE instances in the global XIFI architecture is described in deliverable D1.5 [19]. This component is an optional component of the IAM subsystem which complements the Identity Management component efficiently in the domain of access control. Indeed, Federated Access Control proves to be useful when dynamic attribute-based access control matters, i.e. when authorization depends on attributes at the time of the access request, such as the requester (subject)'s attributes, the requested action, the requested resource, and possibly environment attributes. The component may be used in various ways to control access to critical infrastructure resources in the nodes, but also applications deployed by developers from use case projects. More details can be found in XIFI deliverables D1.5 [19] and D2.4 [20].

3.2.1.4 Sub-system internal component interaction

The figure below from deliverable D1.5 depicts a general overview of how the different components belonging to this sub-system interact with each other:

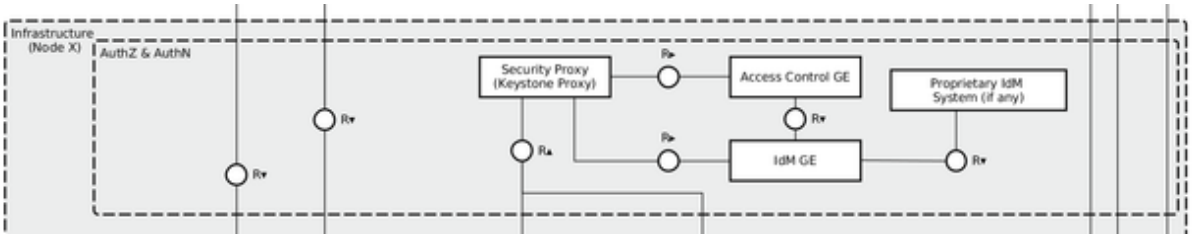


Figure 5: XIFI IAM subsystem architecture

1. Interaction between Security Proxy and IdM interaction

   When creating an Openstack Token (associated to a user and a tenant), the Security Proxy validates the OAuth2 token through the IdM in order to map it to the new Openstack token.

   When a Openstack service try to validates a token through Security Proxy, if valid, Security Proxy retrieves the user information from the IdM.

2. Interaction between Security Proxy and Federated Access Control

   The security proxy may play the role of XACML PEP (Policy Enforcement Point) and query the Access Control PDP (Policy Decision Point) API to get authorization decisions based on various subject, action, resource and/or environment attributes in the context of the access request.

3. Interaction between Federated Access Control and Federated IdM

   The Access Control GE may query the IdM API to get extra information about the user or client application making the access request.

3.2.1.5 Sub-system dependencies - Third-party components and frameworks

The IdM Keyrock GE depends basically of a ruby gem named social stream [27] that provides a social networking framework to manage users and organizations.

The Federation Access Control GE depends on a third-party component: a cluster file synchronization tool, such as csync2 [28], to keep the GE configuration directories synchronized across nodes. This synchronization supports the high-availability of the GE. As the Federation Access Control GE is an optional component, this third-party dependency is required only if the Federation Access Control GE is installed as part of the subsystem.

© 2013 - 2015 XIFI Consortium Parties

3.2.2 Security monitoring

As defined in the D2.2 [1], the security monitoring in the XIFI Federation provides the capabilities for the analysis of the events received from the Security Probes distributed through the nodes in the federation and to feed the Security Dashboard with the relevant security information to be shown to the user.

3.2.2.1 Components

The security monitoring in the XIFI Federation relies on the Security Monitoring GE [29] , one of the cornerstones of the FI-WARE Security Architecture[24]. The Security Monitoring GE integrates different components from the normalization and correlation of heterogeneous events to the risk analysis, decision making support and visualization and reporting. In the XIFI security monitoring deployment, only the Service Level SIEM (Security Information and Event Management) component of this Generic Enabler is used with the purpose of retrieving and tracing suspicious activity throughout the federation infrastructures. The result of this security analysis is reported through the XIFI Security Dashboard so Federation Managers and Infrastructures Owners can take the more suitable risk mitigation actions.

The Service Level SIEM is made up of a SIEM server installed in the master node and Security Probes installed in the different nodes or infrastructures of the federation. Each Security Probe consists of a set of plugins to read logs generated or received from different security data sources and normalize them into the predefined event format, and a SIEM Agent to communicate with the SIEM Server and send the security events collected. On the other hand, the SIEM server receives stores and correlates those events to evaluate the compliance with the security policies and directives that have been predefined in the federation.

More information about this component is described in D4.1 [23] and complemented in D4.2 [24].

3.2.2.2 Sub-system internal component interaction

The following interactions take place between the Security Probes on the slave nodes and the SIEM Server on the master node:

- The security events, already normalized to the OSSIM event format in the slave nodes, are sent via socket to the master node using the SSL (Secure Sockets Layer) protocol. Two processes are listening in the server side: one of them (by default in the port 41000/tcp) will pass the events for its processing in the Service Level SIEM correlation engine and the other (by default in the port 50001/tcp) is the Unix TCP port forwarder Socat, in charge of transferring the events to the OSSIM server installed in the master node for their storage.
- The SIEM Agent (process ossim-agent) establishes a connection with the port (by default 40001/tcp) where the OSSIM server installed in the master node. This connection is used to check if the agent is up and running.
- The SIEM Agent also establishes connections with the ports where the processes ossim-framework (by default 40002/tcp) and alienvault-idm (by default 40003/tcp).

[24] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Security_Architecture

© 2013 - 2015 XIFI Consortium Parties

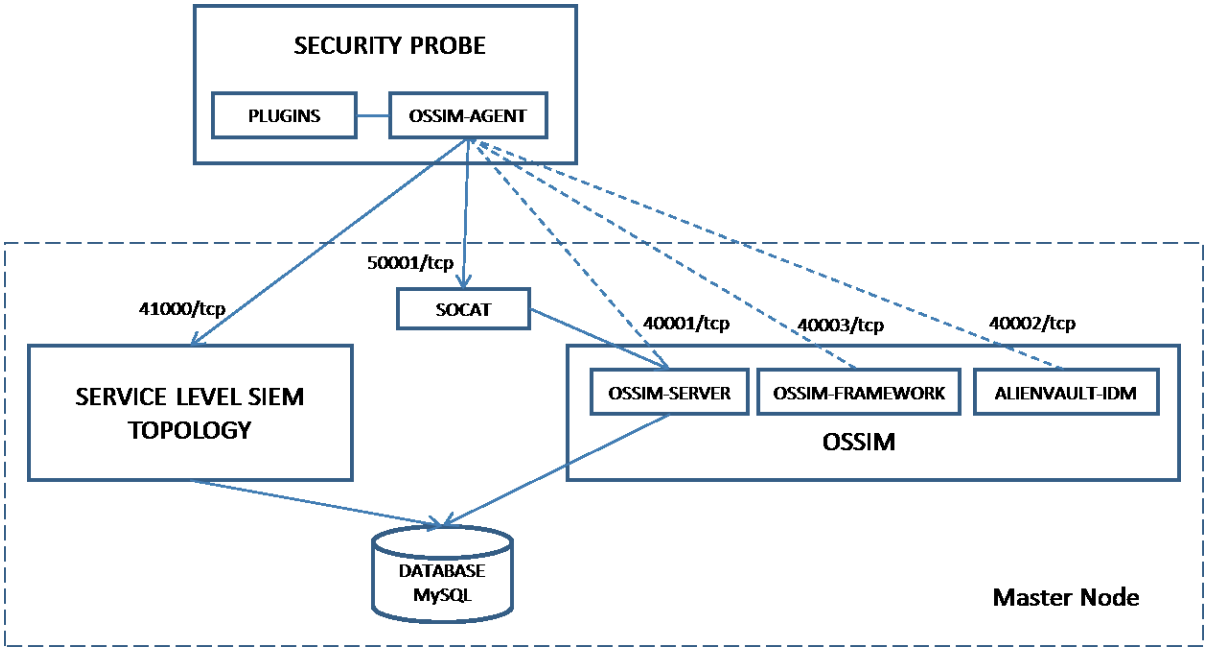


Figure 6: Security Monitoring Internal Component Interactions

More details are available in the page of the Security Monitoring component in the XIFI Wiki[25].

3.2.2.3 Sub-system dependencies - Supporting sub-systems

The first step required to perform the security monitoring is the data collection to feed the SIEM (Security Information and Event Management) system installed with the Security Monitoring GE in the XIFI Federation. This is done as a result of the Security Probes installed on the slave nodes, but this is not possible if there are no data sources to generate information relevant for the security monitoring.

In particular, the following data sources are currently considered to generate information relevant for the security monitoring in the XIFI Federation:

- FI-WARE Access Control GE

  The FI-WARE Access Control GE[26] is the data source for the accountability events generated in the different nodes of the XIFI Federation. These events are used in the XIFI security directives to detect not allowed access to sensitive information stored in the different nodes based on the rights defined by the node administrators to their infrastructure resources.

  With this data source, it is required to add an intermediary process to make the logs generated by this GE (which include complete XACML files) compatible for being processed by the plugins included in the Security Probes.

- Rsyslog

  Rsyslog[27] is one of the most extended open source software that implements the Syslog protocol (RFC 5424[28]) for forwarding log messages. This daemon will be used in XIFI for the

[25] http://wiki.fi-xifi.eu/Public:Security_Monitoring_GE

[26] http://catalogue.fi-ware.org/enablers/access-control-tha-implementation

[27] http://www.rsyslog.com/

[28] http://tools.ietf.org/html/rfc5424

© 2013 - 2015 XIFI Consortium Parties

reception of different logs generated throughout a slave node in the machine where the Security Probe is installed. In particular, the collection of the following information will be centralized through the use of rsyslog in the XIFI nodes mainly to detect failed ssh or login attempts and authentication failures, but it can also be used for logs sent from firewall or router devices (Cisco, Juniper, etc). An additional feature provided by Rsyslog and relevant for XIFI is that it allows the definition of filtering conditions which can be interesting to avoid that all the events of a type generated in a slave node are sent to the master node and only those ones relevant or with no personal data go out the own infrastructure.

- Nagios: Network/systems status monitoring daemon

  Nagios[29] probes are installed on the slave nodes for the Federation Monitoring but their logs can be also used by the Security Monitoring, mainly to perform a deeper analysis once detected an attack. For example, they could help to detect if a DoS (Denial of Service) attack has an internal source through the analysis of the nodes where it has been detected events where the CPU load or RAM status is critical. Specific service directives with Nagios events could also alert when it is detected that the host(s) where any of the security enablers included in the XIFI Federation (Identity Management, Access Control, Security Proxy…) are installed is down.

- Snort: Open source network intrusion detection system

  Snort[30] is one of the most extended open source network intrusion detection systems that can be used in the different XIFI nodes as a data source not only to read packets off of the network (sniffer mode) but also for the detection of threats and network attacks (Network Intrusion Detection System - NIDS mode). Furthermore, snort has the capacity of obfuscating IP addresses which is important to deal with legal restrictions that could have some of the nodes in the XIFI Federation. In particular, Snort can be used in XIFI for the detection of DoS (Denial of Access) attacks in those nodes where it is not possible to recover this type of information from other network devices such as routers or firewalls.

Other services that could be used as potential data sources in the future for the XIFI Federation are:

- pads: passive asset detection system
- tcptrack: monitor TCP connections on the network
- openVAS-Client: the client part of the OpenVAS Security Scanner
- ntop: network usage monitor
- arpwatch: monitor ethernet/ip address pairings
- p0f: identify remote systems passively

3.2.2.4 Sub-system dependencies - Third party components and frameworks

As it is described in FI-WARE Open Specifications[31], the Service Level SIEM (SLS) component included in the Security Monitoring GE is built on top of the **Open Source OSSIM SIEM**[32] (developed by Alienvault) and have its correlation engine within a topology running in a Storm[33]

[29] http://www.nagios.org/

[30] https://www.snort.org/

[31] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Security-Monitoring:_Service_Level_SIEM_Open_API_Specification

[32] https://www.alienvault.com/open-threat-exchange/projects

[33] http://storm-project.net/

© 2013 - 2015 XIFI Consortium Parties

cluster controlled by a Zookeeper[34] cluster to provide scalability and distributed real time computation.

By default, a standalone installation is considered in the XIFI master node where on a single machine will be deployed the OSSIM Framework, the Storm cluster only with a master node (called *nimbus*) and a single worker node (called *supervisor*), and a standalone Zookeeper cluster (with a single server) . New Storm workers or Zookeeper servers could be added if required to deal with the amount of data to be analysed and correlated in case the Federation grows up.

Since it is not the purpose of this deliverable to describe in detail these third party components, we will focus on explaining its use in the XIFI Federation for the security monitoring. You can find more information about them in the FI-WARE Open Specifications or in their own documentation.

- OSSIM Framework

  The OSSIM Framework integrated in the Security Monitoring GE is used in XIFI to manage the storage of the security events received in the master node in a MySQL database, the configuration of policies and data sources to be considered in XIFI Federation and to control the information to be shown in the Security Dashboard. It is also possible to define network directives using OSSIM rules. In particular, the following security directives are defined in the XIFI Federation to be detected with OSSIM:

  - Check if a node is up and running, monitoring Nagios events collected in the slave nodes and sent to the master node. This information is also provided in XIFI in the Infographics tool but could also be integrated to the Security Dashboard to make this knowledge available for the security administrators from a same interface.
  - Detect if a node is under a DoS Attack. This can be detected by statistical data on the network traffic of the nodes. Probes should be taken from network traffic to the internet and network traffic through the MDVPN. After this detection, it might be necessary to start a deeper investigation to isolate the source of the attack in order to take immediate mitigation actions such as isolate the source.
  - Detect if someone is trying to break into a node, detecting brute force attacks on passwords. This detection is done through the collection with the Security Probes installed on the nodes of syslog failed ssh/login attempts (by default in file /var/log/auth.log) and sending them to the server for its aggregation and generation of alerts.

  Besides, OSSIM includes the capability to integrate in the security monitoring the detection of known vulnerabilities and its cross correlation with security events received. For example, as a Federation Manager, it would be interesting to know that the nodes in the federation have not known vulnerabilities that could be used by attackers. This information would not be provided to someone else except to inform the node manager so that they can solve them. So, this checking would be done from the Security Monitoring GE in the master node, executing scan jobs against a predefined set of IPs and only in the case of slave node giving its previous approval for these scan jobs.

- Storm cluster

  In the Storm cluster is where the Service Level SIEM correlation engine included in FI-WARE Security Monitoring is running. This correlation engine is used in XIFI to detect the security service directives defined. This is the most interesting part in the security monitoring from the federation point of view because it allows the analysis among events coming from different slave nodes to generate alarms from a higher level. In particular, rules will be defined to detect if someone with no rights is accessing to sensitive information stored in a node.

[34] http://zookeeper.apache.org/

© 2013 - 2015 XIFI Consortium Parties

The administrators of the nodes will define through the Access Control GE the rights to access the resources and sensitive information stored in their nodes. With the logs generated by this Access Control GE will be received in the master node accountability events that help for example to know if there are suspicious access attempts (e.g. 5 denied access attempts to a same resource from the same user) or someone does any changes in the user rights of the node resources to detect bad behaviors or attempts to steal or modify data stored in the node. These alarms will be also stored in the same MySQL database managed by the OSSIM Framework, so they can be accessible from the XIFI Security Dashboard in a unified way with the rest of network alarms generated.

3.3 User oriented and GUI subsystems

3.3.1 Monitoring dashboard

The Monitoring Dashboard is a mashup of different monitoring tools for some of the XIFI components currently deployed. Nowadays it consists of two components:

1. VMs 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.

2. NAM monitoring tool.

   NAM monitoring tool is oriented to XIFI IO and provides them information about the Network status of the connections between XIFI nodes. They can obtain monitoring data about bandwidth, latency and packet loss in two ways: real data information and historical data information.

3.3.2 Security dashboard

3.3.2.1 Components

The Security Dashboard is composed of two main modules: **OSSIM Web Engine**, responsible for retrieving information from the SIEM included in the Security Monitoring GE to be shown into the dashboard, and the **Accountability Tools** to offer the user more advanced statistical tools about accountability events generated in the Federation.

Additionally, a **XIFI Adaptation Layer** has been developed in order to adapt the information provided by these modules to the XIFI Portal layout and to integrate and delegate the authentication to the XIFI Identity Management GE.

3.3.2.2 Sub-system internal component interaction

Figure 7 summarizes the sequence of component interactions:

© 2013 - 2015 XIFI Consortium Parties

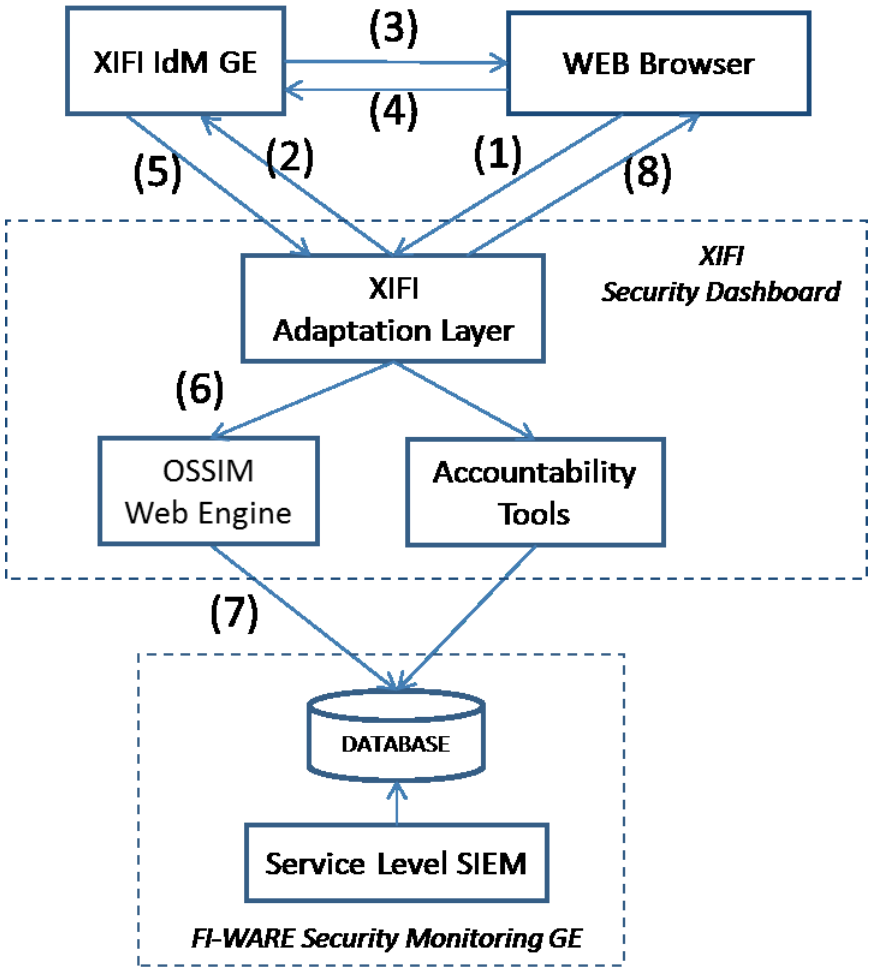


Figure 7: Security Dashboard Interactions

- The XIFI Adaptation Layer offers the first contact point to the user when the Security Dashboard endpoint is invoked from a web browser (1).
- If it is the first time the user accesses the Security Dashboard or the token has expired, a request is sent to the XIFI IdM GE (2). The XIFI IdM registration page is shown to the user to sign in (3). The user introduces his email and password, this information is authenticated by the IdM GE (4) and the user token is returned to the XIFI Adaptation Layer (5). More information about how this process is done can be found in the IdM documentation.
- Once the user has been authenticated (has a valid token), the XIFI Adaptation Layer will invoke the OSSIM Web Engine and the Accountability Tools (6). These modules will access to the MySQL database installed with the FI-WARE Security Monitoring and where the Service Level SIEM stores the security events and alarms as well as security directives and other SIEM configuration data.

3.3.2.3 Sub-system dependencies - Supporting sub-systems

The Security Dashboard delegates to the *Federated Identify Management* the identification of authorized users in the XIFI Federation.

The Security Dashboard application has been registered in the XIFI IdM and the following roles has been created and associated to the different functionalities provided:

- *FederationManager*: This role gives the administration rights to the user of the Security

© 2013 - 2015 XIFI Consortium Parties

Dashboard, which include all the dashboard management items: configuration of data sources, security directives, policies and actions, SIEM servers and security probes, or users. This application role is associated to the organizations with role *Federator*. As Federator, the user with this role can visualize all the security incidents generated in any of the nodes in the federation. There is no filtering applied when the information is recovered from the SIEM database.

- *IO*: With this role, the user of the Security Dashboard only will be able of visualizing different types of information but not configure the SIEM. This application role is associated to the organizations that represent an Infrastructure Owner.

The XIFI Adaptation Layer component of the Security Dashboard is the responsible for interacting with the IdM API. Apart from the identification of the user that is used to personalize the dashboard header, there are two pieces of user information relevant for the Security Dashboard: firstly, the organizations where the user belongs to, and secondly, for each organization it is recovered the roles assigned in order to determine if the user has some of the roles defined for the Security Dashboard application.

On the other hand, the *Federation Manager API* is used by the Security Dashboard to ensure the user belongs to a real node registered in the federation and to filter the user only will have access to the range of IP addresses defined for his/her node.

With this information, the XIFI Adaptation Layer does a mapping between the user registered in the IdM and the user access groups available in the Security Monitoring GE. Apart from the administrator user (selected in the case of Federation Manager role), there is an access group associated to each specific infrastructure or node in the XIFI Federation with the range of IP address allowed recovered from the Federation Manager API. This data can be modified or updated by the Federation Manager (as administrator of the federation) through the Security Dashboard.

3.3.2.4 Sub-system dependencies - Third party components and frameworks

3.3.2.4.1 FI-WARE Security Monitoring GE

All the data recovered by the OSSIM Web Engine and the Accountability Tool to be shown in the Security Dashboard need to be available in the MySQL database included with the Service Level SIEM component of the FI-WARE Security Monitoring GE. In particular, the following information will be used by the Security Dashboard:

- *Events*: collected from the different nodes throughout the XIFI Federation thanks to the SIEM agents installed with the Security Probes.
- *Alarms*: generated by the OSSIM server or with the Service Level SIEM correlation engine
- *SIEM Configuration*: users, networks defined, sensors (which represents where the SIEM Agents are up and running), data sources, security directives, policies and actions, etc.

3.3.3 Infographics and status pages

3.3.3.1 Components

The Infographics and Status Pages is a single component considered here as a dedicated sub-system that provides graphical 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 presents an innovative and intuitive way to publish high-level live information on the available infrastructure capacities in FIWARE Lab. Overall data are displayed in single sections (e.g. Users, Organizations, Regions, CPU, Ram, VM…). Opening each section, it is possible to see more detailed node data, also thanks to the use of maps.

© 2013 - 2015 XIFI Consortium Parties

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. Also if the Status page is a service normally offered by cloud providers and other services providers, there was not such service available as open source for OpenStack related infrastructures.

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

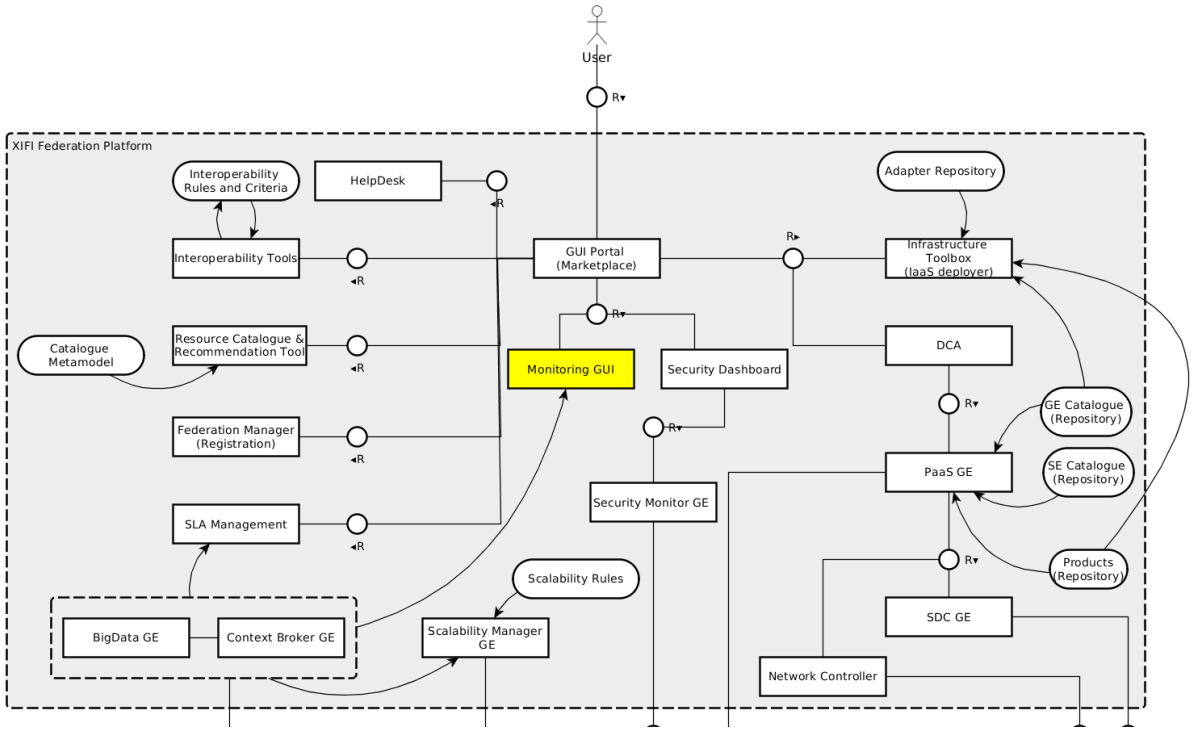


Figure 8: Infographics and status page architectural context.

Further details can be found in the corresponding wiki page[35].

3.3.3.2 Sub-system internal component interaction

None

3.3.3.3 Sub-system dependencies - Supporting sub-systems

The component depends on:

- Federation Monitoring
- Federated Identity Management

3.3.3.4 Sub-system dependencies - Third party components and frameworks

The component depends on:

- Jira

[35] http://wiki.fi-xifi.eu/Public:Infographics_and_Status_Pages

© 2013 - 2015 XIFI Consortium Parties

Infographics and Status Pages component strongly depends on APIs exposed by XIFI Federation Monitoring[36]. All real-time data about nodes and their services are collected by this component. XIFI Federation Monitoring APIs are protected by a proxy that uses the OAuth2 authorization protocol. In order to correctly access these APIs, Infographics and Status Pages component must first authenticate on the IDM component[37] gaining an access token.

This component offers also Jira support. In order to achieve this, it must use REST APIs exposed by Jira tracking software (https://www.atlassian.com/software/jira). Because of this, the Infographics and Status Pages component must be configured with appropriate Jira data before starting.

3.3.4 Cloud portal

The Cloud Portal provides support for the users of the 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.

The basic objective of the user portal is to facilitate the user of the cloud perform operations over the underlying infrastructure. This includes perform actions such as: create user, manage projects, lunch instances on a base of images, create images in the image repository, retrieve flavours from the resource, etc. Moreover the portal facilitates management of a Virtual Data centres (VDCs), Services and Service Data Centers (SDCs), PaaS management and will offer monitoring statistics of the physical and virtual machines.

3.3.5 SLA manager

3.3.5.1 Components

The SLA Manager provides mechanisms to support service level agreements management in the federated environment, based on WS-Agreement specification [30]. It is responsible to manage all the SLA lifecycle, creation of template for services, negotiation of the agreements and monitoring agreements compliance at runtime. Hence, the component allows: i) the direct interaction among the different actors through the graphical user interface and is designed to be part of XIFI portal ii) expose the APIs to be used by other components and interact with the Monitoring sub-system in order to collect the values of the associated metrics. The SLA Manager is composed by two main subcomponents:

- SLA Dashboard is focused on the interaction with the users, so it is responsible of the graphical user interfaces (Presentation layer). This module does not interact directly with the rest of components and it is not responsible to execute the functionalities. It delegates this action to the business layer or backend and it only need to communicate with the exposed APIs of the SLA Management.
- SLA Management (core) is focused on managing all the functionalities of the SLA lifecycle, templates, agreements, monitoring and violations (business layer or backend). It is 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...). Hence, the following section will be focus on the relationships with the other components. The SLA Management sub-system is divided in the

[36] http://wiki.fi-xifi.eu/Xifi:Wp2:d25#Federation_Monitoring

[37] http://wiki.fi-xifi.eu/Xifi:Wp2:d25#Federated_Identity_Management

© 2013 - 2015 XIFI Consortium Parties

following modules:

- service (REST interface): It is responsible to expose the backend functionalities through the REST interfaces;
- sla-management: it contains the backend business code (publish, negotiation, enforcement, recovery);
- repository: access to the main entities (Templates, Terms, Agreements and Violations);
- sla-personalization: it contains all the specific adapters for every project. This allows to change the adapters easily even if it is changed the origin of the monitoring, without affecting the sla-management modules.

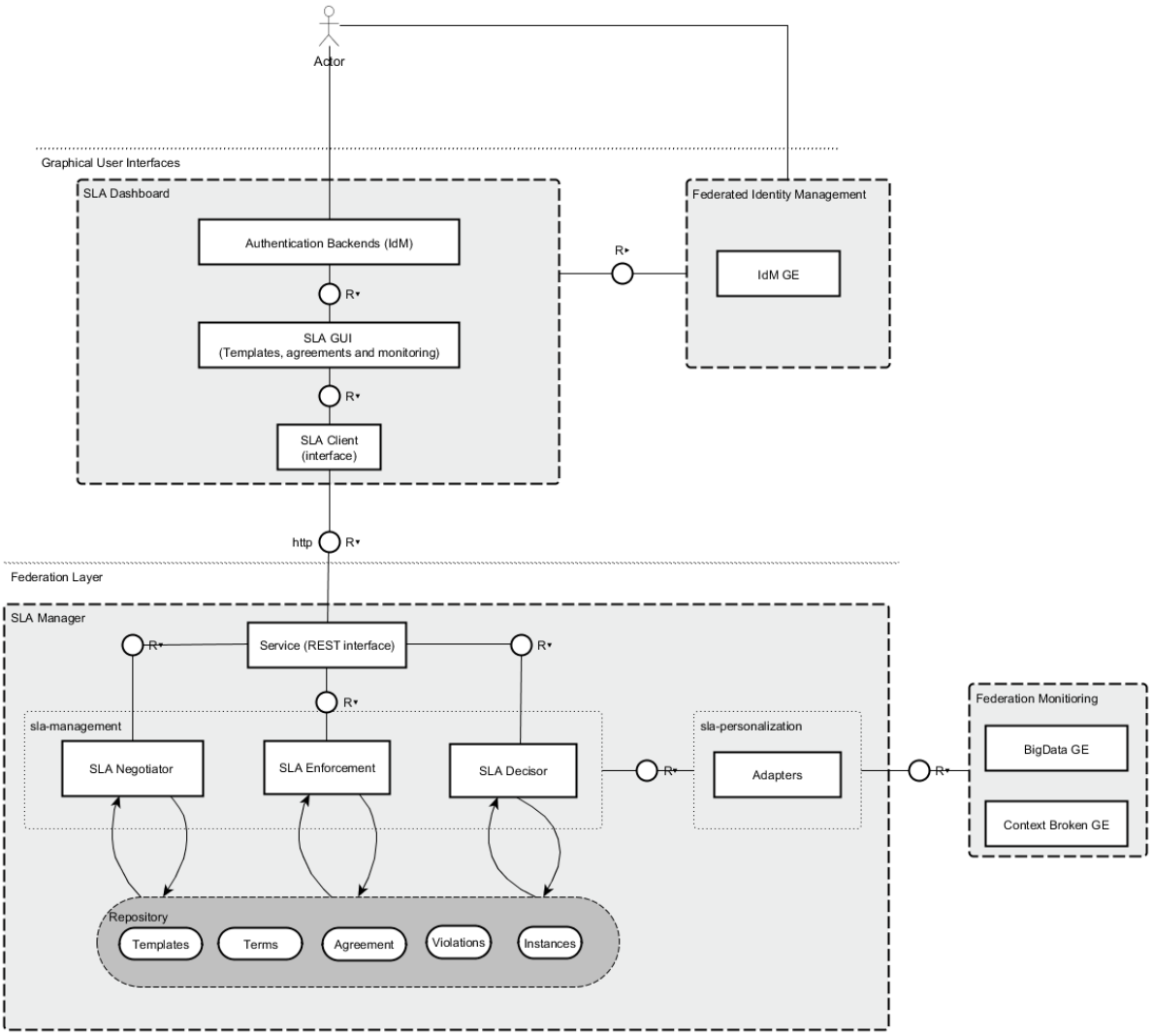


Figure 9: SLA Manager FMC compositional structure diagram.

Further details can be found in the corresponding [wiki page](#)[38] or in the D4.4 Baseline tools v2

[38] http://wiki.fi-xifi.eu/Xifi:Wp4:Resource_Catalogue&Recommender

© 2013 - 2015 XIFI Consortium Parties

3.3.5.2 Sub-system internal component interaction

The following sequence shows the internal interactions:

- The SLA Dashboard is the first contact point to the user, since he cannot interact directly with the SLA Management.
- The SLA Dashboard delegates the authorization of the user to the Federated Identity Management (Security Sub-system). Afterwards, the user can access to the component through the SLA Dashboard.
- Every user action is managed by the SLA Dashboard which is responsible to adapt the content in order to call the SLA Management.
- SLA Management performs the actions and interacts with the necessary components in order to provide the SLA functionalities. It is responsible to interact mainly with the exposed APIs Federation Monitoring. There is a configured scheduler that calls the APIs Federation Monitoring in order to collect the values of the associated metrics.

3.3.5.3 Sub-system dependencies - Supporting sub-systems

The subcomponent SLA Management (core) is responsible to support other components in order to allow accessing the SLA information. It exposes an API (described in the API sections of this document) that allows managing the SLA lifecycle. The components that interact with this component are:

- SLA Dashboard: it is the main component that interacts with the SLA Management (core). It is mandatory to deploy both components in order to allow the user to manage the SLA lifecycle through a graphical interface.
- Marketplace/Resource Catalogue and Recommend: There are several phases that interact with the SLA Management:
    - The agreement creation, when a user wants to register in a service as SaaS or when the user wants to instantiate a GE/SE as PaaS.
    - Analysis of the agreement, the component can provide a summary of the status of the user's agreements in the same place where the user has all the information of his services. Or, it can recommend through the stored data of the SLA.

3.3.5.4 Sub-system dependencies - Third party components and frameworks

- Federated Identity Management[39]: The SLA Management delegates the authentication to the Federated Identity Management component. So, the IdM is responsible to manage the lifecycle of the user registration. There are defined tree roles for the SLA Manager and every user has to have at least one of them.
    - **Consumer**: He wants to discover the services, register on them in order to use them (indicating what he wants to monitor), and manage his services.
    - **Provider Infrastructure Owner (Federation Member)**: is a service provider in our federated environment. They have to configure, deploy and manage the services which have been published.

[39] http://wiki.fi-xifi.eu/Xifi:Wp2:Identity_Management_GE

© 2013 - 2015 XIFI Consortium Parties

- **Federator**: He is responsible for providing all the necessary processes and tools in order to cover the deploying and managing of the services.

- Federation Monitoring API[40]: In order to obtain the values of the metrics, which have been defined in the agreements, the SLA Management (core) uses the APIs[41] of the Federation Monitoring. This can provide to types of metrics in different levels.

  Historical data monitoring: This is provided by the BigData GE and it is historical (average of the real values). These metrics can be directly an average or can aggregated complexity to the basic metrics. The SLA Management can use this aggregated metrics to reduce the complexity of the validators of the metrics. Although, the SLA Manage can create its own validators (ImetricsValidator (cf. the Architecture wiki[42]), it is more efficient to delegate this to the Federation Monitoring.

  Real time monitoring data: Federation Monitoring provides monitoring data in real time, through the Context Broker GE. "Real time" data is forwarded to the data consumers through the API avoiding any possible delay. This data are directly collected from the different nodes and it is not aggregated.

  There are several levels to be applied the basic metrics as Host, Virtual Machine and Services. Not all the metrics are available in all the levels and they are based in the monitoring data model diagram[43].

The SLA Management only has to include in the templates and agreements the metrics that can be identified and measured by the Federation Monitoring, both the historical and the real time. If the metrics cannot be measured, they shouldn't be available for the users.

3.3.6 Federation manager

The Federation Manager component developed in the scope of the XIFI project provides access to the federation. This component is the central registration point for new infrastructures and their services. Its role in the XIFI architecture can be referred from Figure 10.

The component consists of two parts, the Federation Manager GUI and the Federation Manager Core. Compared to its description in D2.2 [1] several points have changed, ranging from changes in the installation procedure and configuration, to the interaction with the IDM for authentication purposes and API extensions and modifications. These changes are described in more detail in the corresponding sections in this document.

Reference Scenarios	UC-1 Joining the Federation.
Reference Stakeholders	Infrastructure owners, Federator.
Type of ownership	New component
Original tool	None

[40] http://wiki.fi-xifi.eu/Public:Federation_Monitoring

[41] http://docs.federationmonitoring.apiary.io/

[42] http://wiki.fi-xifi.eu/Public:SLA_Manager

[43] http://wiki.fi-xifi.eu/Public:Federation_Monitoring#Architecture_design

© 2013 - 2015 XIFI Consortium Parties

Planned OS license	Apache License Version 2.0.
Reference OS community	None

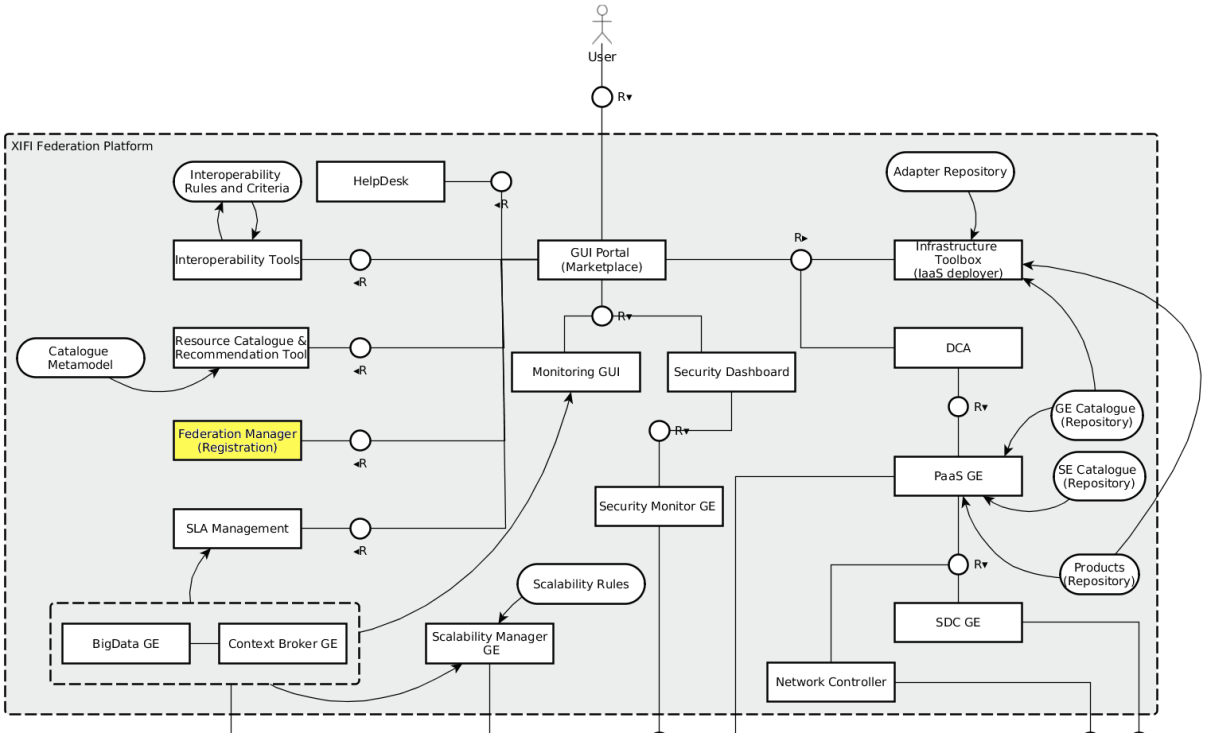


Figure 10: Federation Manager Architectural Context

3.3.6.1 Components

- Federation Manager Core
- Federation Manager GUI

3.3.6.2 Component responsible

Developer	Email	Company
Daniel Nehls	daniel.nehls@tu-berlin.de	TUB
Alexander Willner	alexander.willner@tu-berlin.de	TUB

3.3.6.3 Motivation

The XIFI federation follows a one stop shop model architecture.As such, a central registration unit for infrastructures and their services is needed and shall be provided by this component. Besides its pure registry functionality, the Federation Manager shall support the “Join-the-Federation” process by keeping track of the current status of an infrastructure.

3.3.6.4 State of the art

Federation of infrastructures can be achieved by manifold architecture models (cf. D1.1 [25]), thus registry mechanisms which are implemented for one example of testbed federation cannot be simply adopted by the XIFI federation. Nevertheless this short state of the art analysis on testbed registry

© 2013 - 2015 XIFI Consortium Parties

mechanisms in some important federation projects has been analysed in D2.2 [1].

3.3.6.5 Sub-system internal component interaction

As stated above the Federation Manager Component consists of two parts as depicted in Figure 11.

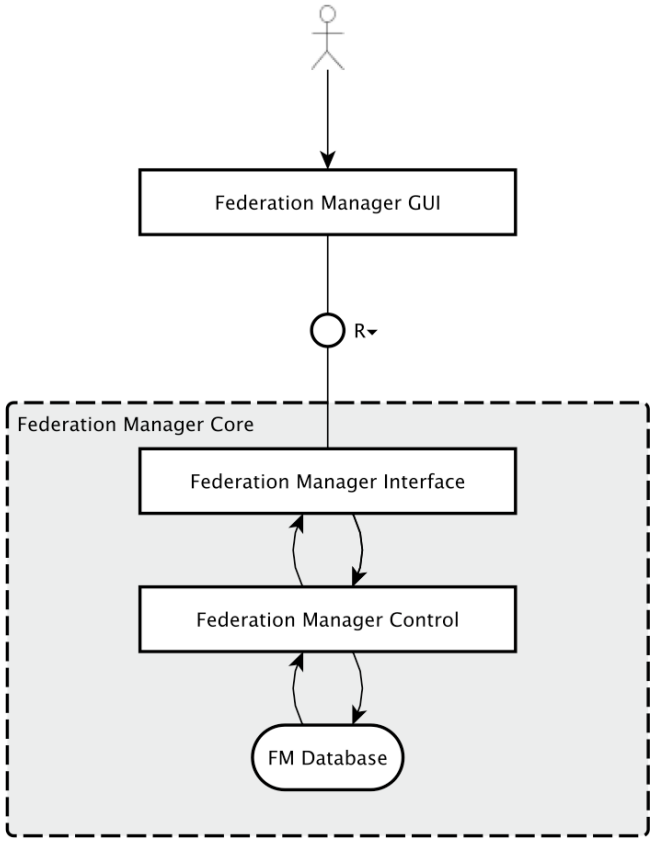


Figure 11: Federation Manager Architecture.

The GUI presents views for each the federation administrator and the owner/admin of the new infrastructure. For the owner of a new infrastructure it encompasses forms to provide information about the infrastructure, such as support contact data or location information, and offers the download of the Infrastructure Toolbox. The federation administrator part of the GUI lists all infrastructures together with their status in the "Join-the-Federation" process and enables remote testing of the components installed.

The Federation Manager Core stores the information and provides a REST based interface.

3.3.6.6 Sub-system dependencies - Supporting sub-systems

Based on the requirements, there are no specific functionalities to be provided by the the FM so that it needs to use the monitoring system, but if any, the FM is extendable to interact with the monitoring system.

The Federation Manager component depends on:

- Federated Identity Management
- Following interaction is planned:
- Federated Access Control

© 2013 - 2015 XIFI Consortium Parties

- Data-push to Keystone Proxy

The API will be used by several components, e.g. the Security Dashboard for retrieval of IP range information for specific regions.

3.3.6.7 Sub-system dependencies - Third party components and frameworks

The FM depends on four third-party software components: Java 7, any database (the current implementation uses MySQL), any J2EE server (the current implementation uses WildFly 8) and Maven (version used: 3.05).

3.3.6.8 Installation Manual

The Federation Manager is implemented as Java Webarchive and makes use of J2EE technologies. In the following an installation guide for running the FM on a WildFly Application Server on a UNIX based machine is provided.

Prerequisites

- Java 7
- Maven
- WildFly 8 AS avaliable at http://www.wildfly.org
- Federation Manager source code available at https://xifisvn.res.eng.it/wp2/software/trunk/Components/FederationManager/

Installation

- Before running WildFly for the first time an adminstrator user has to be created:

```
<wildfly_installation_folder>/bin/add-user.sh
```

  and follow the instructions prompted.

- run WildFly AS:

```
<wildfly_installation_folder>/bin/standalone.sh
```

Compile

```
cd FederationManager
mvn clean install
```

MySql Database Setup

- log into WildFly AS at localhost:9990 with username and password provided during the step above
- under the Profile tab add a new datasource and follow the steps shown in Figure 12, Figure 13 and Figure 14.

  The datasource has to be enabled afterwards
- In FM source folder edit the *src/main/webapp/WEB-INF/classes/META-INF/persistence.xml* according to the values provided in the WildFly management GUI (do not edit the name of the persistence unit! Otherwise the references in the source have to be edited as well)

© 2013 - 2015 XIFI Consortium Parties

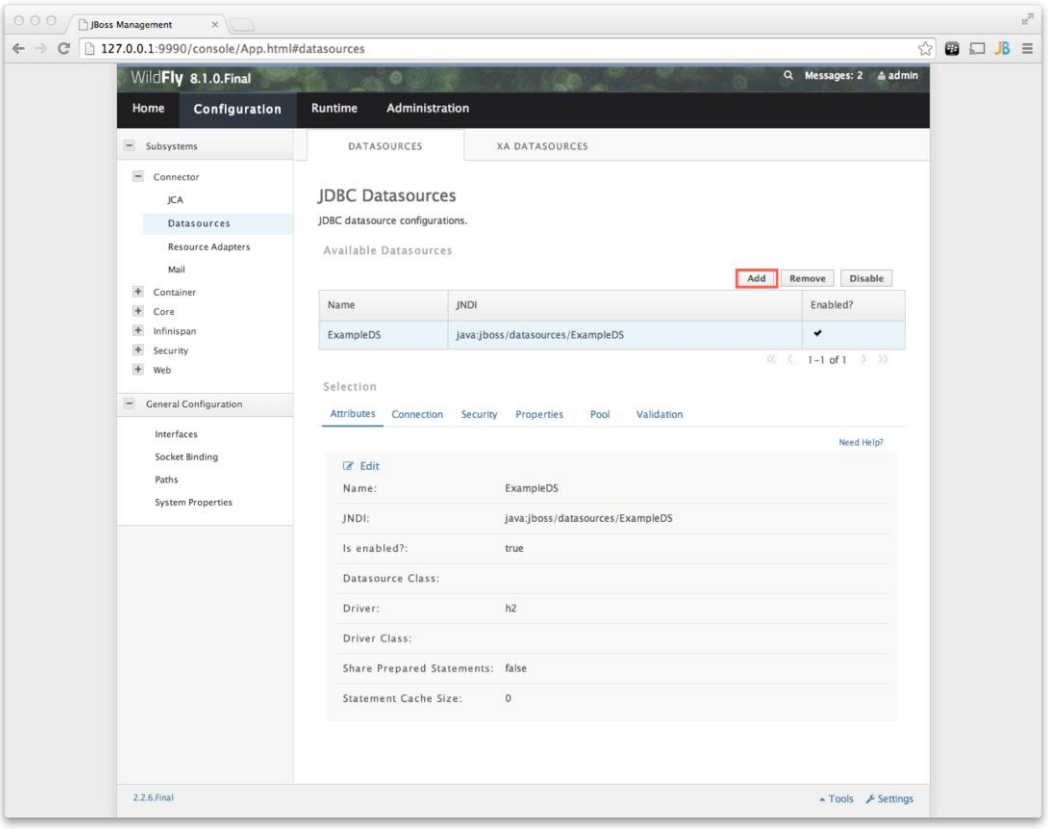


Figure 12: Federation Manager Add Datasource.

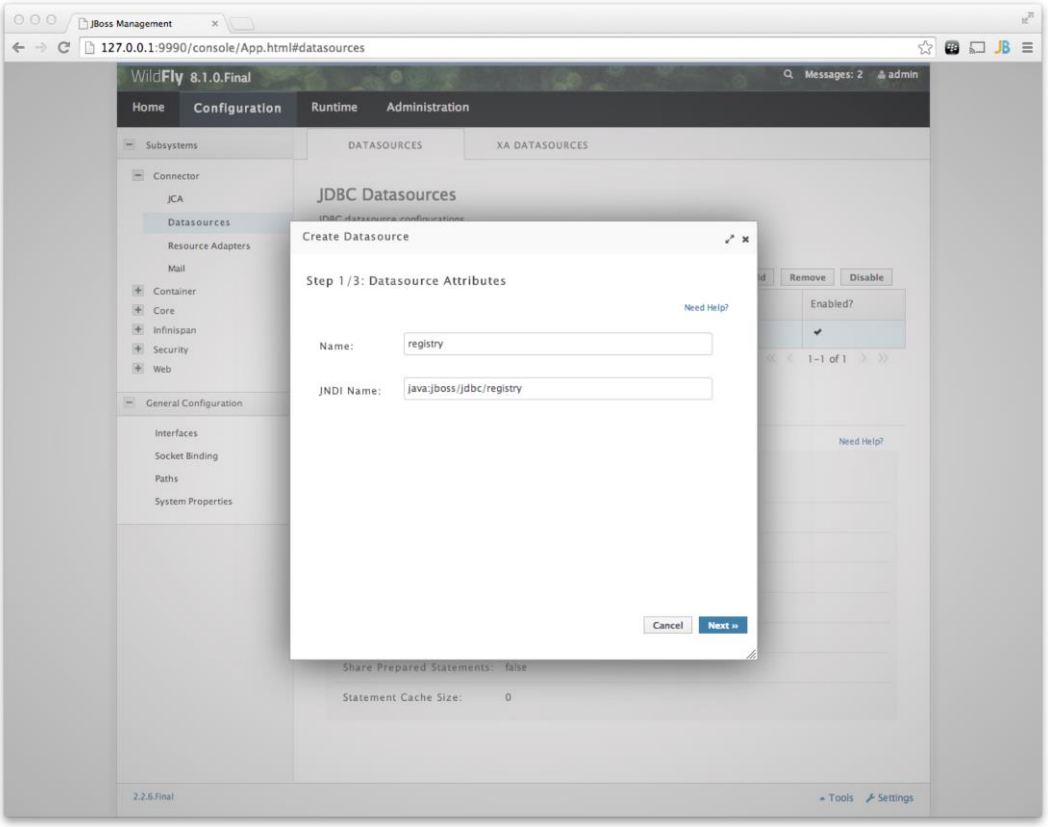


Figure 13: Federation Manager Datasource Attributes.

© 2013 - 2015 XIFI Consortium Parties

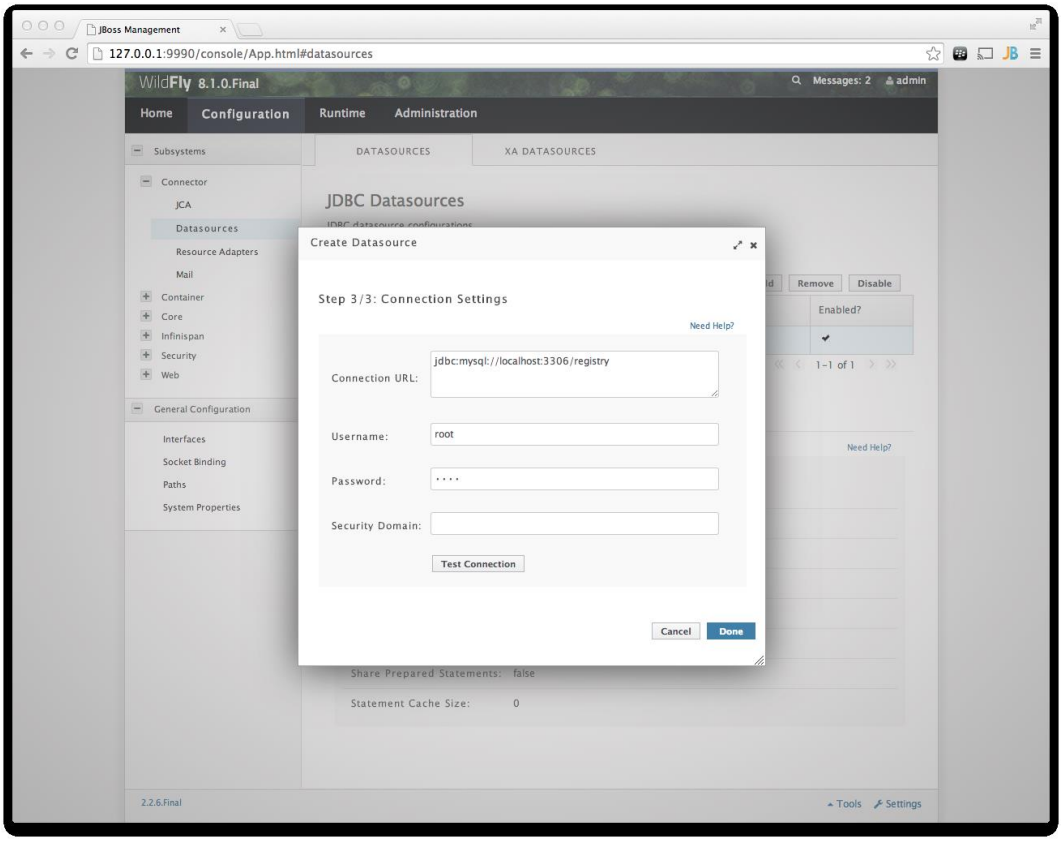


Figure 14: Federation Manager Datasource Connection Settings.

Preferences

The Federation Manager application defines a set of preferences for configuration. The location of the preferences file depends on the OS used.

- tokenLocation - url to fetch oauth access token, for example https://account.lab.fi-ware.org/oauth2/token
- redirectURI - callback url registered at IDM
- userLocation - url to fetch user based on access token
- clientId - client id assigned by IDM
- environment - defines the role of the user to use, either "dev_member", "dev_admin", "dev_fm" for development environment or "production" if to be used with IDM authentication/authorization
- errorPage - defines the URL for an error page
- clientSecret - client secret assigned by IDM
- authLocation - url of oauth entry point

Start the application

copy the build war-file (step "compile") to `server/wildfly/standalone/deployments/`. By default, the application runs in development mode, this means no outh identificaton is required. To change the role of the user connected, edit the preference `environment`.

- dev_member: A user which is a member of an organization, without rights to edit or create new infrastructures.

© 2013 - 2015 XIFI Consortium Parties

- dev_admin: A user which is given the admin role by its organization. Can create, edit or delete infrastructures.
- dev_fm: A user which belongs to the FederationManager organization. Can see, edit and delete all infrastructures registered.
- production: This value enables oauth identification, other preferences must be adjusted as well. See the documentation of the [IDM](https://github.com/ging/fi-ware-idm/wiki/Using-the-FI-LAB-instance "KeyRock IDM")for details.

3.3.6.9 Installation Test

The files referenced can be found in the according svn project under src/main/resources/test_examples. When **not** running in development mode (see Installation Guide), the **Authorization** header must be set to a valid value. The test cases and a script how to perform these tests can be found in Table 4.

3.3.6.10 User Manual

The registration process has been modified since its introduction in D2.2 [1]. The major change is the integration of the OAUTH mechanism in order to use the IDM for authentication. When a non-registered infrastructure owner (IO) visits the Federation Manager website (FM GUI) he will be asked if he and the organization he is acting on behalf on, is already registered at the federation identity management system (IDM). When denied, the visitor is redirected to the IDM to complete this mandatory registration. After registration at the IDM, the IO has to answer the Quick Online Test. This test provides an questionnaire where the IO has to provide information regarding legal, technical and operational compliance, as well as information about the organization, which will operate the node. The data is transmitted to the Federation Manager (FM), who can then add the organization to the list of authorized entities for the Federation Manager application (FM App). The node owner will then be informed and can proceed in the process. The IO can now login to the FM GUI, add information about the infrastructure and download the Infrastructure Toolbox software package.

After the successful installation and configuration, the infrastructure owner has to provide information on the federation services now running on his infrastructure. When this data is committed the federation administrator can trigger remote tests on these services. The specification of these tests is ongoing and will be further detailed in upcoming documents.

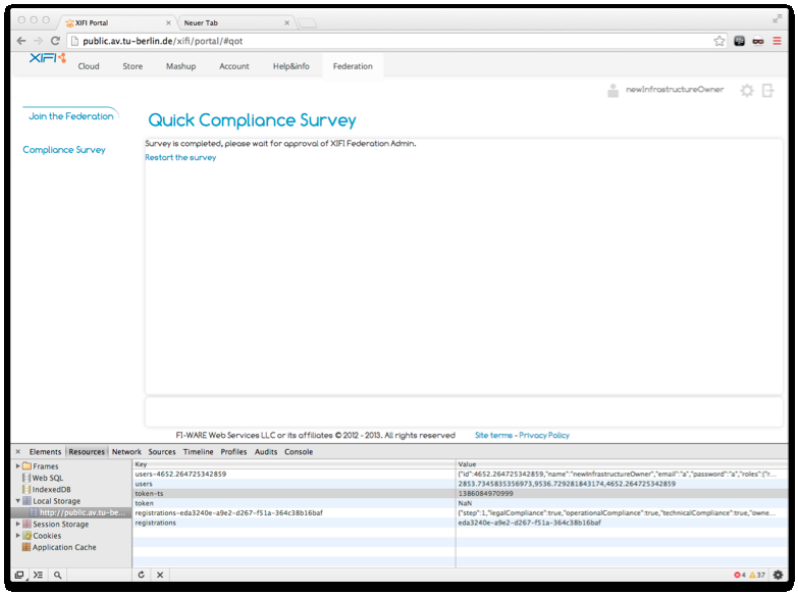


Figure 15: Federation Manager GUI - Infrastructure Owner Compliance Survey

Test id	Test description	Test script	Expected results
1	Test POST: /v3/regions	curl -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @postRegion.txt http://localhost:8080/federationManager/api/v3/regions	Output expected by the API under test - see API section
2	Test GET: /v3/regions	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions	Output expected by the API under test - see API section
3	Test GET: /v3/regions/{regionid}	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions/{regionId}	Output expected by the API under test - see API section
4	Test PATCH: /v3/regions/{regionid}	curl -X PATCH -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @patchRegion.txt http://localhost:8080/federationManager/api/v3/regions/{regionId}	Output expected by the API under test - see API section
5	Test DELETE: /v3/regions/{regionid}	curl -X DELETE -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions/{regionId}	Output expected by the API under test - see API section
6	Test GET: /v3/regions/{regionid}/status	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions/{regionId}/status	Output expected by the API under test - see API section
7	Test PATCH: /v3/regions/{regionid}/status	curl -X PATCH -H "Content-Type:application/json" --data @patchRegionStatus.txthttp://localhost:8080/federationManager/api/v3/regions/{regionId}/status	Output expected by the API under test - see API section
8	Test GET: /v3/regions/{regionid}/contacts	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions/{regionId}/contacts	Output expected by the API under test - see API section
9	Test POST: /v3/regions/{regionid}/contacts	curl -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @postRegionContact.txthttp://localhost:8080/federationManager/api/v3/regions/{regionId}/contacts	Output expected by the API under test - see User Manual section

© 2013 - 2015 XIFI Consortium Parties

Test id	Test description	Test script	Expected results
10	Test GET: /v3/regions/{regionid}/contacts/{contactId}	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions/{regionId}/contacts/{contactId}	Output expected by the API under test - see User Manual section
11	Test PATCH: /v3/regions/{regionid}/contacts/{contactId}	curl -X PATCH -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @patchRegionContact.txthttp://localhost:8080/federationManager/api/v3/regions/{regionId}/contacts/{contactId}	Output expected by the API under test - see User Manual section
12	Test DELETE: /v3/regions/{regionid}/contacts/{contactId}	curl -X DELETE -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/regions/{regionId}/contacts/{contactId}	Output expected by the API under test - see User Manual section
13	Test POST: /v3/services	curl -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @postService.txt http://localhost:8080/federationManager/api/v3/services	Output expected by the API under test - see User Manual section
14	Test GET: /v3/services	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/services	Output expected by the API under test - see User Manual section
15	Test GET: /v3/services/{serviceId}	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/services/{serviceId}	Output expected by the API under test - see User Manual section
16	Test PATCH: /v3/services/{serviceId}	curl -X PATCH -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @patchService.txt http://localhost:8080/federationManager/api/v3/services/{serviceId}	Output expected by the API under test - see User Manual section

© 2013 - 2015 XIFI Consortium Parties

Test id	Test description	Test script	Expected results
17	Test DELETE: /v3/services/{serviceId}	curl -X DELETE -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/services/{serviceId}	Output expected by the API under test - see User Manual section
18	Test POST: /v3/endpoints	curl -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @postEndpoint.txt http://localhost:8080/federationManager/api/v3/endpoints	Output expected by the API under test - see User Manual section
19	Test GET: /v3/endpoints	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/endpoints	Output expected by the API under test - see User Manual section
20	Test GET: /v3/endpoints/{endpointId}	curl -v -H "Content-Type:application/json+hal" -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/endpoints/{endpointId}	Output expected by the API under test - see User Manual section
21	Test PATCH: /v3/endpoints/{endpointId}	curl -X PATCH -H "Content-Type:application/json" -H "Authorization: Bearer developer" --data @patchEndpoint.txt http://localhost:8080/federationManager/api/v3/endpoints/{endpointId}	Output expected by the API under test - see User Manual section
22	Test DELETE: /v3/endpoints/{endpointId}	curl -X DELETE -H "Authorization: Bearer developer"http://localhost:8080/federationManager/api/v3/endpoints/{endpointId}	Output expected by the API under test - see User Manual section

Table 4: Federation Manager Installation Test

.

© 2013 - 2015 XIFI Consortium Parties

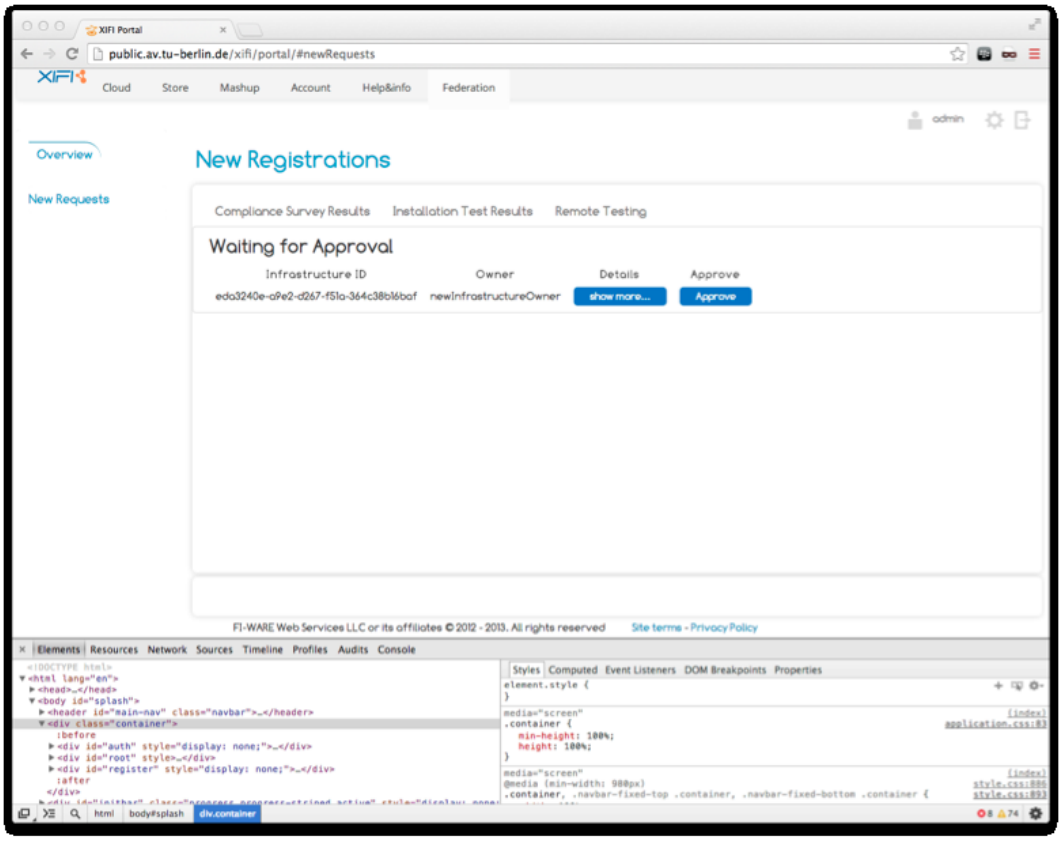


Figure 16: Federation Manager GUI - Federation Admin New Requests

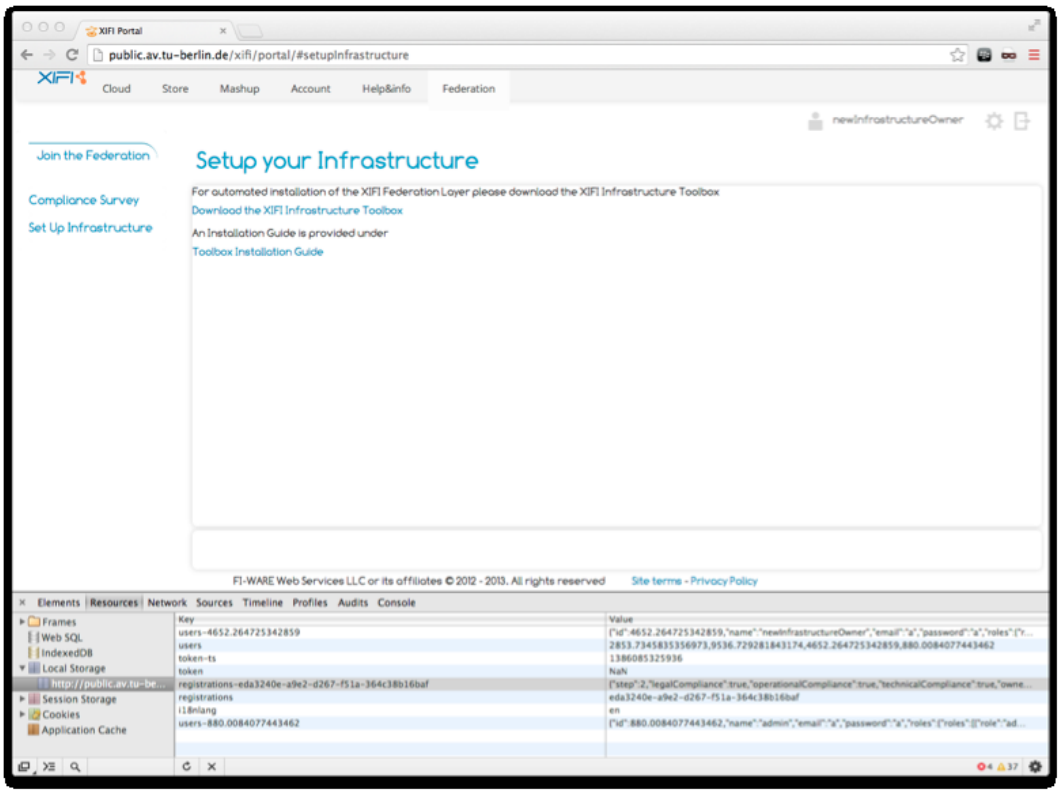


Figure 17: Federation Manager GUI - Download of the Infrastructure Toolbox

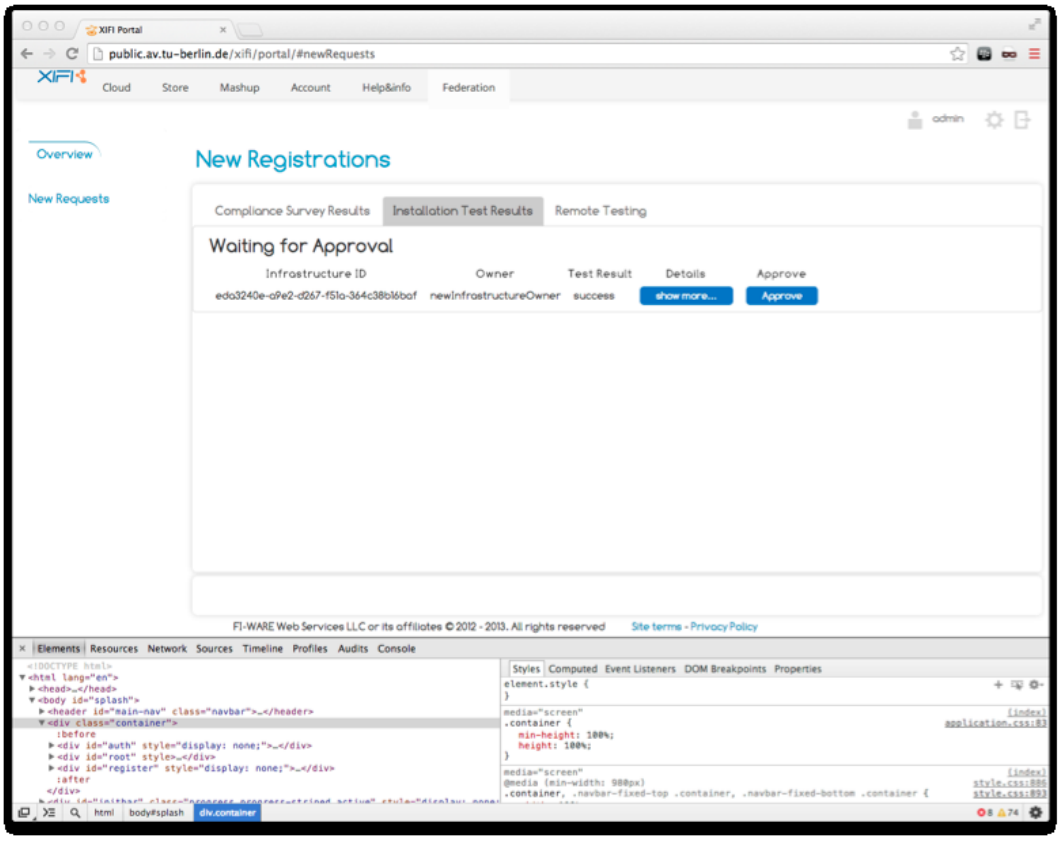


Figure 18: Federation Manager GUI - Infrastructure Toolbox Test Results

3.3.7 Interoperability tool

The Interoperability Tool is a software engineering tool that supports development and testing of FI-WARE 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. 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 usage of GEs; therefore, they can quickly build their applications based upon these established patterns of behavior.

Infrastructure owners can also use the tool for verifying compliance of developed and deployed services. That is, if a GE complies with the FI-WARE 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.

Currently, the Cloud Portal (cf. section 3.3.4) is the sole client to the Interoperability Tool.

Further documentation can be found at the corresponding wiki page[44] and in D4.4 [26].

[44] http://wiki.fi-xifi.eu/Xifi:Wp4:Interoperability_Tool

3.3.8 Resource catalogue

3.3.8.1 Components

Resource Catalogue and Recommender, integrated in the XIFI portal, allows developers and experimenters managing instances by presenting a catalogue with the full-offering of resources available through a graphical user interface. The integrated Marketplace allows identifying, searching, using, and comparing different services while providing recommendations, for example, selecting the more convenient node to be used depending on the user profile, skills or location. Besides, software providers (Infrastructure Owners and Technological providers) can advertise their resources/services in order to allow finding and accessing their XIFI federated resources/services.

The Resource Catalogue component composed by the "Resource Catalogue and Recommendation tool" and "Catalogue Metamodel" architectural components and integrated with the "GUI Portal". It is decouple in two layers: "Presentation Layer", responsible of the visualization and the user interface, and "Business layer and Data Layer", responsible of managing the three main functionalities: search, offering and recommendation.

Further details can be found in the corresponding wiki page[45] or in the D4.4 [26].

3.3.8.2 Sub-system internal component interaction

The following sequence shows the interactions:

- The Presentation Layer is the first contact point to the user, since he cannot interact directly with the rest of components.
- If the user has not been identified in the federation, the Presentation Layer delegates the authorization of the user to the Federated Identity Management (Security Sub-system). Afterwards, the user can access to component through the Presentation Layer.
- Every user action is managed by the Presentation Layer which is responsible to adapt the content in order to call the Business Layer.
- Business Layer performs the called action and interacts with the necessary components in order to provide the functionality. It is responsible to interact with other sub-systems (Security, Deployment and Operation) and third party components (FIWARE Catalogue).

3.3.8.3 Sub-system dependencies - Supporting sub-systems

The component depends on:

- FI-WARE Repository GE[46]: The component uses the Repository GE to manage the Catalogue Metamodel which is based on the USDL standard. Hence, it is necessary to install and configure the GE to store the data of the resources.
- Federated Identity Management[47]: The component delegates the authentication to the Federated Identity Management component. So, the IdM is responsible to manage the lifecycle of the user registration. It is necessary to create the following roles when the "Resource Catalogue and Recommender" is registered in the IdM:
    - Developer: They are identified as simple users that want to use the platform as an

[45] http://wiki.fi-xifi.eu/Xifi:Wp4:Resource_Catalogue%26Recommender

[46] http://catalogue.fi-ware.eu/enablers/repository-sap-ri

[47] http://wiki.fi-xifi.eu/Xifi:Wp2:Identity_Management_GE

© 2013 - 2015 XIFI Consortium Parties

individual relationship (developers, FI Experimenters, end users….).

- Infrastructure Owners: They are federation members that provide their infrastructures and publish their resources/services (Trento, Lannion, Berlin…).
- Technological Provider: they want to publish/share part of their portfolio in the XIFI federation environment (GE/SE developers, SMEs, services providers…).
- Federator: It is the responsible to the maintenance of the component and the validation of the catalogue publication.

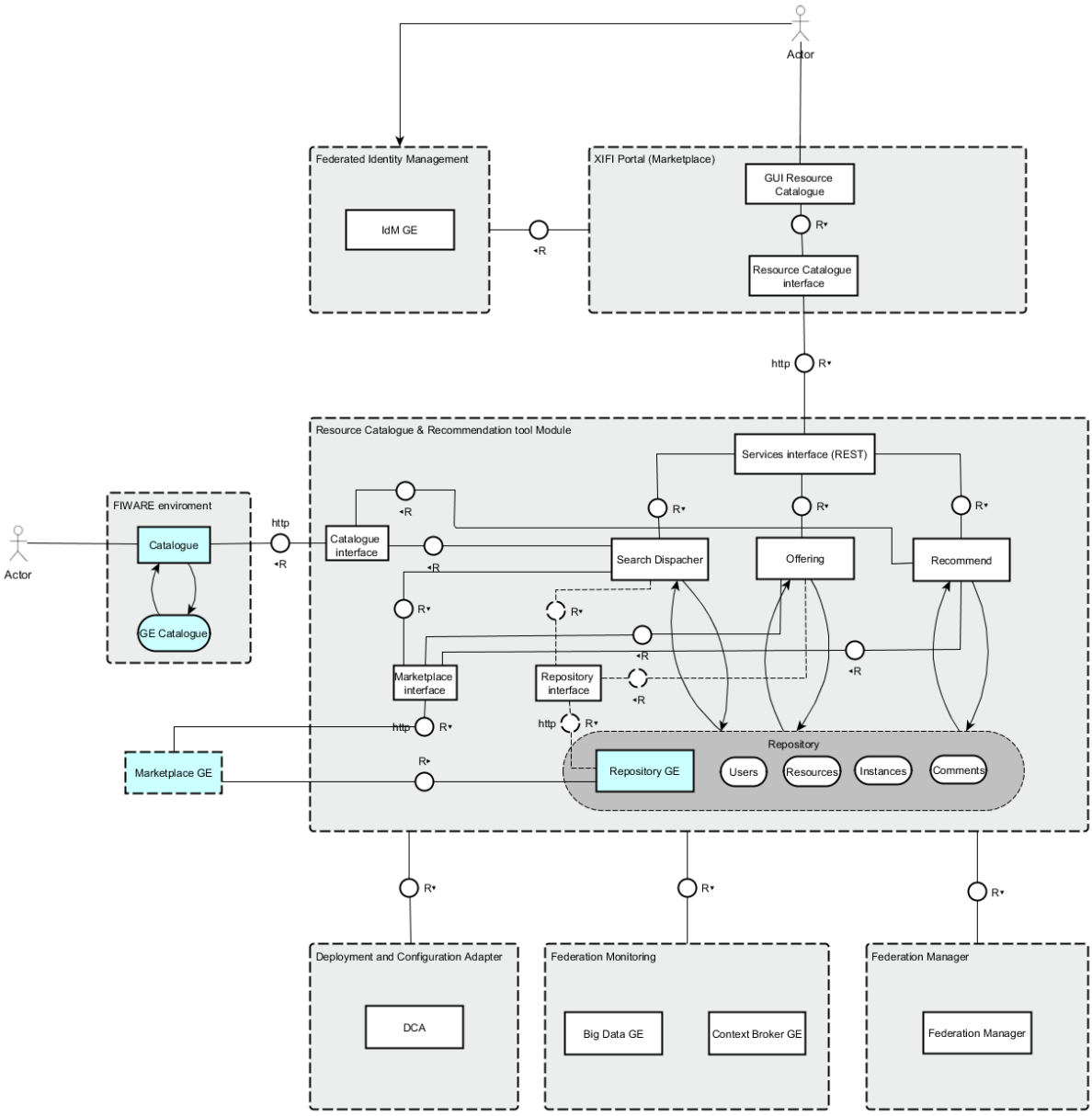


Figure 19: Resource Catalogue FMC compositional structure diagram

- FIWARE Catalogue[48]: It is a third-party component which is responsible to manage the publication of the GE. This component exposes an API in order to find the GEs and their

[48] http://catalogue.fi-ware.org/

description. The Resource Catalogue shows, homogeneously, both the GE data and the information of the instances and his status per node.

- Deployment and Configuration Adapter – DCA[49]: The component uses this APIs DCA to obtain the correlation between all the entities: instances of GE/SE, users, nodes, tenant… in the federation.
- Federation Monitoring API[50]: The resource catalogue uses this APIs component in order to show the status of the instances and to recommend the best nodes based on the user preferences.
- Federation Manager[51] The resource Catalogue and Recommender uses this APIs exposed by the Federation Manager mainly to collect the data of the nodes.
- FI-WARE MarketPlace GE[52] (Optional): This GE allows creating a common marketplace between different repositories or stores. It is not necessary to install it if you don't need this behavior. So far, there are not requirements that indicate the activation, nevertheless it is already integrated, so, it is only necessary to install it and use it to cover them.

The Resource Catalogue and Recommender component has many dependencies, and it integrates with different sub-systems to reduce the complexity and harmonize the federated environment. Further details about the configuration of the Resource Catalogue and Recommender are in the wiki page[53]

3.3.8.4 Sub-system dependencies - Third party components and frameworks

This sub-system does not depend on further third-party components and frameworks.

3.4 Deployment and operations

3.4.1 Infrastructure toolbox

ITBox is described in the component's page at http://wiki.fi-xifi.eu/Xifi:Wp3:Components:InfrastructureToolbox. It does not depend on any other internal or external sub-system, and also does not depend any other third party components or frameworks.

3.4.2 Deployment and Configuration Adapter

The Deployment and Configuration Adapter (DCA) is the XIFI component that caters for the enhanced deployment functionality, as needed by the project users forming in parallel a Deployment Registry. For the purpose of WP2, DCA communicates with PaaS and Cloud Portal in order to retrieve information regarding the deployment of VMs/GEs that have taken place within XIFI federation. In this perspective the information processed and stored in DCA can be queried by other components (i.e. SLA Manager) via a RESTful API. In particular, the DCA provides:

1. Deployment of multiple GEs and XiFi components upon XiFi infrastructure

   The DCA supports the deployment and configuration of multiple GEs in a batch mode (as images, through recipes or in a combination), allowing the user to select details (including the

[49] http://wiki.fi-xifi.eu/Xifi:Wp3:Components:DCA

[50] http://wiki.fi-xifi.eu/Public:Federation_Monitoring

[51] http://wiki.fi-xifi.eu/wiki/index.php?title=Xifi:Wp2:Federation_ManagerFederation&action=edit&redlink=1

[52] http://catalogue.fi-ware.eu/enablers/marketplace-sap-ri

[53] http://wiki.fi-xifi.eu/Public:Resource_Catalogue%26Recommender#Installation_Manual

© 2013 - 2015 XIFI Consortium Parties

sequential or parallel deployment and the notification capabilities). Such multi-GE deployment can take place in a single node or upon federated XiFi nodes. The DCA can also be used to deploy XiFi components upon the infrastructure.

2. Check of Available Resources prior to the Deployment

   The DCA performs check on the resources that are available to the user, prior to the deployment of one or multiple GEs and according to the documented hardware requirements of the GEs. This functionality can protect the user from receiving errors (by the platform) after invoking the deployment procedure. The resource availability check is performed considering the user's quota upon the XiFi infrastructure, the resources that have been already reserved by the user and the hardware needs of the GEs under deployment (currently quantified in CPU cores, memory and storage). The checks can be performed per node and / or federated nodes.

3. Persistency of information related to the deployed GE instances

   The DCA holds all pertinent information from the whole lifecycle of GE deployment. This includes the requests on behalf of the users (through the portal) and the system responses as related to the GE instances (going well beyond the typical awareness of the VM instances). This information is adapted and then exposed upon request to the components of WP4, through a set of meaningful queries.

DCA API (http://docs.dca.apiary.io/) has been extended (compared to D2.2 [1]) to include requirements arising from XIFI components that are making use of DCA functionality.

To find a full specification of the component the reader shall check DCA corresponding wiki page[54].

3.4.3 PaaS Manager

The purpose of FIWARE PaaS Manager GE is to support the Cloud Portal (User Oriented and GUI Subsystem) by easing the way Application Developers carry out the task of deploying applications on the federated infrastructure. PaaS Manager helps developers in the definition of blueprint templates to orchestrate the interaction with DCRM for the deployment of the hardware resources (virtual machines or servers and networks) and with the SDC GE for the installation and configuration of software (for more details, please refer to the FIWARE specification[55]).

PaaS comprises three main components:

- PaaS Communication Service (PMI): it's responsible for offering a RESTful interface to the PaaS GE users
- PaaS Manager Core: build up from several components responsible of managing the different entities in charge of executing the tasks
- SWInstallationManager: it provides the mechanisms to coordinate with the **SDC GE** the installation of products and artifacts on VMs or servers, and to configure them.

[54] http://wiki.fi-xifi.eu/Public:Deployment_and_Configuration_Adapter

[55] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.PaaS

© 2013 - 2015 XIFI Consortium Parties

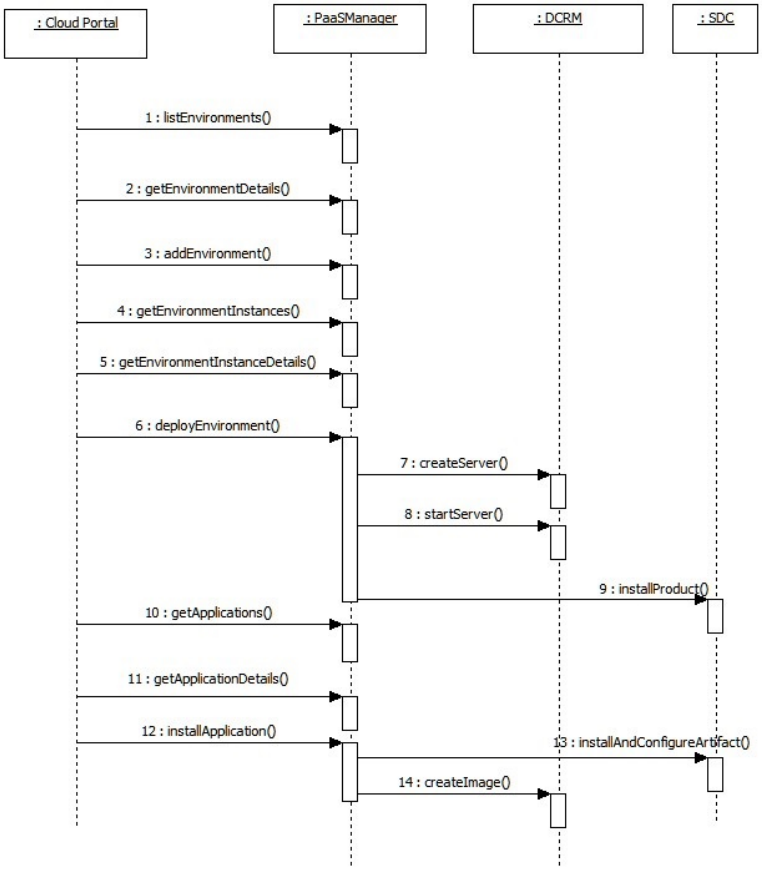


Figure 20: PaaS-DCRM-SDC Interaction

3.4.4 SDC

FIWARE SDC (Software Deployment and Configuration) GE is an enabler used to support automated deployment (installation and configuration) of software on running virtual machines. PaaS Manager relies on this component when orchestrating blueprint-based deployments on the federated infrastructure (for more details, please refer to the FIWARE specification[56]).

SDC consists of:

- SDC Client Interface (SDCI): used to obtain information from a VM or server and request Configuration agent executions.
- ProductManager: manages the catalogue of products.
- ProductInstanceManager: in charge of managing and executing the installation of products and artifacts through the Configuration Engine.
- ConfigurationManager: manages the communication with the Configuration Engine for the assignation of new recipes to the REC execution queue, and to trigger the actual execution through the SDC client in the REC.

[56] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Cloud.SDC

© 2013 - 2015 XIFI Consortium Parties

4 SUB-SYSTEM APIS

This section details on the published sub-system APIs complementing the functional description given by the previous section. It concludes on the monitoring, federation manager and SLA APIs while emphasis is put on the changes with respect to D2.2. Other APIs are referenced but not detailed further in the scope of this document. It should be noted that the API documentation includes a number of examples that are referred to by some of the test case descriptions provided in the previous section.

4.1 Monitoring

4.1.1 Federation Monitoring API

HOST: http://monitoring.fi-xifi.eu (Note: not yet working from the Internet at the time of writing because the hostname has not yet being registered. You can test them at 193.205.211.69:1026).

These are the **XIFI Federation Monitoring** APIs.

Note: all these APIs are now protected by a proxy that uses the OAuth2 authorization protocol through the IDM credential. So a FIWARE Lab account is required in order to use them.

4.1.1.1 Monitoring API Root [/]

API entry point.

4.1.1.1.1 Retrieve Entry Point [GET]

Response 200 (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/" },
        "regions": { "href": "/monitoring/regions", "templated": true
   }
        "host2hosts": { "href": "/monitoring/host2hosts",
   "templated": true }
    }
}
```

4.1.1.2 Group Region

Region related APIs

4.1.1.2.1 Regions [/monitoring/regions]

Retrieve all regions.

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/regions" }
    },
    "_embedded": {
        "regions": [
            {
```

```
                "_links" : {
                    "self": { "href": "/monitoring/regions/Trento" }
                },
                "id": "Trento"
            }
        ]
    },
    "total_nb_users": "10",
    "total_nb_organizations": "10",
    "total_nb_cores": "100",
    "total_nb_cores_enabled": "100",
    "total_nb_ram": "1000",
    "total_nb_disk": "10000",
    "total_nb_vm": "100000"
}
```


4.1.1.2.2 List All Regions [GET]

Response 200 [Regions][]

4.1.1.2.3 Region [/monitoring/regions/{regionid}{?since}]

Retrieve a region. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise metrics referred to the last monitored interval are provided. Measures are aggregated on an hourly basis.

Parameters

```
regionid (string)
```

Model (application/hal+json)

Body


```
{
    "_links": {
        "self": { "href": "/monitoring/regions/Trento" },
        "hosts": { "href": "/monitoring/regions/Trento/hosts" }
    },
    "id": "Trento",
    "name": "Trento",
    "country": "Italy",
    "latitude": "xyz",
    "longitude": "xyz",
    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "nb_cores": "100",
            "nb_cores_enabled": "100",
            "nb_ram": "1000",
            "nb_disk": "10000",
            "nb_vm": "100000",
            "power_consumption": "123",
            "percCPULoad": {
                "value": "123",
                "description": "avarage of the percCPULoad for all
   the hosts"
            },
            "percRAMUsed": {
                "value": "123",
                "description": "avarage of the percCPULoad for all
```

```
    the hosts"
                },
             "percDiskUsed": {
                "value": "123",
                "description": "avarage of the percCPULoad for all
    the hosts"
                }
            }
         ]
}
```

4.1.1.2.4 Retrieve a Region [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [Region][]

4.1.1.3 Group Host

Host related APIs.

4.1.1.3.1 Hosts [/monitoring/regions/{regionid}/hosts]

Retrieve all hosts in a given region.

Parameters

```
regionid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/regions/{regionid}/hosts" }
    },
    "hosts": [
        {
            "_links" : {
            "self": { "href":
   "/monitoring/regions/Trento/hosts/12345" }
            },
            "id": "12345"
        }
        ]
}
```

4.1.1.3.2 List All Hosts for a given Region [GET]

Response 200 [Hosts][]

4.1.1.3.3 Host [/monitoring/regions/{regionid}/hosts/{hostid}{?since}]

Retrieve an host. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise metrics referred to the last monitoried interval are provided. Measures are aggregated on an hourly basis.

© 2013 - 2015 XIFI Consortium Parties

Parameters

```
regionid (string)
hostid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/regions/Trento/hosts/12345" },
        "services": { "href":
   "/monitoring/regions/Trento/hosts/12345/services" }
    },
    "regionid": "Trento",
    "hostid": "12345",
    "ipAddresses": [
            {
               "ipAddress": "1.2.3.4"
            }
        ],
    "owd_endpoint_dest_default": "xyz",
    "bwd_endpoint_dest_default": "xyz",
    "owd_frequency": "123",
    "bwd_frequency": "123",
    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "percCPULoad": {
                "value": "123",
                "description": "desc"
                },
            "percRAMUsed": {
                "value": "123",
                "description": "desc"
                },
            "percDiskUsed": {
                "value": "123",
                "description": "desc"
                },
            "sysUptime": {
                "value": "123",
                "description": "desc"
                },
            "owd_status": {
                "value": "123",
                "description": "desc"
                },
            "bwd_status": {
                "value": "123",
                "description": "desc"
                }
            }
        ],
    "traps": [
        {
            "description": "desc"
        }
    ]
}
```

© 2013 - 2015 XIFI Consortium Parties

4.1.1.3.4 Retrieve an host [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [Host][]

4.1.1.4 Group VM

VM related APIs.

4.1.1.4.1 VMs [/monitoring/regions/{regionid}/vms]

Retrieve all VMs on a given region.

Parameters

```
regionid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/regions/{regionid}/vms" }
    },
    "vms": [
        {
            "_links" : {
            "self": { "href": "/monitoring/regions/Trento/vms/54321"
   }
            },
            "id": "54321"
        }
        ]
}
```

4.1.1.4.2 List All VMs for a given Region on a given Host [GET]

Response 200 [VMs][]

4.1.1.4.3 VM [/monitoring/regions/{regionid}/vms/{vmid}{?since}]

Retrieve a VM. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise metrics referred to the last monitoried interval are provided. Measures are aggregated on an hourly basis.

Parameters

```
regionid (string)
vmid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href":
   "/monitoring/regions/Trento/hosts/12345/vms/54321" },
```

© 2013 - 2015 XIFI Consortium Parties

```
        "services": { "href":
   "/monitoring/regions/Trento/vms/54321/services" }
    },
    "regionid": "Trento",
    "vmid": "54321",
    "ipAddresses": [
            {
               "ipAddress": "1.2.3.4"
            }
        ],
    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "percCPULoad": {
                "value": "123",
                "description": "desc"
                },
            "percRAMUsed": {
                "value": "123",
                "description": "desc"
                },
            "percDiskUsed": {
                "value": "123",
                "description": "desc"
                },
            "sysUptime": {
                "value": "123",
                "description": "desc"
                }
            }
        ],
    "traps": [
        {
            "description": "desc"
        }
    ]
}
```

4.1.1.4.4 Retrieve a vm [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [VM][]

4.1.1.5 Group Service

Service related APIs.

4.1.1.5.1 Services4Host [/monitoring/regions/{regionid}/hosts/{hostid}/services]

Retrieve all Services running on a given host.

Parameters

```
regionid (string)
hostid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href":
   "/monitoring/regions/{regionid}/hosts/{hostid}/services" }
    },
    "services": [
        {
            "_links" : {
            "self": { "href":
   "/monitoring/regions/Trento/hosts/12345/services/apache2" }
            },
            "id": "apache2"
        }
        ]
}
```

4.1.1.5.2 List all Services running on a given Host [GET]

Response 200 [Services4Host][]

4.1.1.5.3 Service4Host [/monitoring/regions/{regionid}/hosts/{hostid}/services/{serviceName}?since]

Retrieve a service. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise measures referred to the last monitored interval are provided. Measure status is always referred to the current status. Measure percUptime is present only if since is specified and is aggregated on an hourly basis.

Parameters

```
regionid (string)
hostid (string)
serviceName (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href":
   "/monitoring/regions/Trento/hosts/12345/service/apache2" }
    },
    "regionid": "Trento",
    "hostid": "12345",
    "serviceName": "apache2",
    "description": "this is apache2 service!",
    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "status": {
                "value": "green/yellow/red",
                "description": "status of the service. In case of
   realtime, it could be only red or green whereas in case of
   historical data it could be green if %upTime > 90%,
   50%<=yellow<=90%, red < 50%"
                }
```

© 2013 - 2015 XIFI Consortium Parties

```
            }
        ]
}
```

4.1.1.5.4 Retrieve a service belonging to an host [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [Service4Host][]

4.1.1.5.5 Services4VM [/monitoring/regions/{regionid}/vms/{vmid}/services]

Retrieve all Services running on a given VM.

Parameters

```
regionid (string)
vmid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href":
   "/monitoring/regions/{regionid}/vms/{vmid}/services" }
    },
    "services": [
        {
            "_links" : {
            "self": { "href":
   "/monitoring/regions/Trento/vms/54321/services/apache2" }
            },
            "id": "apache2"
        }
        ]
}
```

4.1.1.5.6 List all Services running on a given VM [GET]

Response 200 [Services4VM][]

4.1.1.5.7 Service4VM [/monitoring/regions/{regionid}/vms/{vmid}/services/{serviceName}?{since}]

Retrieve a service belonging to a VM. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise measures referred to the last monitored interval are provided. Measure status is always referred to the current status. Measure percUptime is present only if since is specified and is aggregated on an hourly basis.

Parameters

```
regionid (string)
vmid (string)
serviceName (string)
```

© 2013 - 2015 XIFI Consortium Parties

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href":
   "/monitoring/regions/Trento/vms/54321/service/apache2" }
    },
    "regionid": "Trento",
    "vmid": "54321",
    "serviceName": "apache2",
    "description": "this is apache2 service!",
    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "status": {
                "value": "green/yellow/red",
                "description": "status of the service. In case of
   realtime, it could be only red or green whereas in case of
   historical data it could be green if %upTime > 90%,
   50%<=yellow<=90%, red < 50%"
            }
        }
        ]
}
```

4.1.1.5.8 Retrieve a service belonging to an vm [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [Service4VM][]

4.1.1.5.9 Services4Region [/monitoring/regions/{regionid}/services?{since}]

Retrieve the current status of the services running on a given region. If parameter {since} is specified, the the measures represents the %uptime of each service. + Parameters + regionid (string)

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/regions/{regionid}/services" }
    },

     "measures": [
            {
            "timestamp" : "2013-12-20 12.00",
            "novaServiceStatus": {
                "value": "green/yellow/red",
                "description": "status of the service. In case of
   realtime, it could be only red or green whereas in case of
   historical data it could be green if %upTime > 90%,
   50%<=yellow<=90%, red < 50%"
                },
             "neutronServiceStatus": {
                "value": "green/yellow/red",
                "description": "status of the service. In case of
```

```
    realtime, it could be only red or green whereas in case of
    historical data it could be green if %upTime > 90%,
    50%<=yellow<=90%, red < 50%"
                    },
                "cinderServiceStatus": {
                    "value": "green/yellow/red",
                    "description": "status of the service. In case of
    realtime, it could be only red or green whereas in case of
    historical data it could be green if %upTime > 90%,
    50%<=yellow<=90%, red < 50%"
                    },
                "glanceServiceStatus": {
                    "value": "green/yellow/red",
                    "description": "status of the service. In case of
    realtime, it could be only red or green whereas in case of
    historical data it could be green if %upTime > 90%,
    50%<=yellow<=90%, red < 50%"
                    },
                "KPServiceStatus": {
                    "value": "green/yellow/red",
                    "description": "status of the service. In case of
    realtime, it could be only red or green whereas in case of
    historical data it could be green if %upTime > 90%,
    50%<=yellow<=90%, red < 50%"
                    },
                "OverallStatus": {
                    "value": "green/yellow/red",
                    "description": "calculated from the previous statuses
    in this way: all green => green, > 50% green => yellow, <= 50%
    green => red"
                    }
                }
            ]
}
```

4.1.1.5.10 List all Services running on a given region [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [Services4Region][]

4.1.1.5.11 Group Network Element

Network Element related APIs.

4.1.1.5.12 NEs [/monitoring/regions/{regionid}/nes]

Retrieve all NEs monitored in a given region.

Parameters

```
regionid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
```

```
        "self": { "href": "/monitoring/regions/{regionid}/nes" }
    },
    "nes": [
        {
            "_links" : {
            "self": { "href": "/monitoring/regions/Trento/12345" }
            },
            "id": "12345"
        }
        ]
}
```

4.1.1.5.13 List all NEs for a given Region on a given Host [GET]

Response 200 [NEs][]

4.1.1.5.14 NE [/monitoring/regions/{regionid}/nes/{neid}{?since}]

Retrieve a network element. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise metrics referred to the last monitored interval are provided. Measures are aggregated on an hourly basis.

Parameters

```
regionid (string)
neid (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/regions/Trento/nes/54321" }
    },
    "regionid": "Trento",
    "neid": "54321",
    "neType": "network element type",
    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "ifPhysAddress": {
                "value": "123",
                "description": "desc"
                },
            "ifOperStatus": {
                "value": "123",
                "description": "desc"
                },
            "ifInOctects": {
                "value": "123",
                "description": "desc"
                },
            "ifInErrors": {
                "value": "123",
                "description": "desc"
                },
            "ifInUCastPkts": {
                "value": "123",
                "description": "desc"
                },
```

© 2013 - 2015 XIFI Consortium Parties

```
            "ifInDiscard": {
                "value": "123",
                "description": "desc"
                },
            "ifOutOctects": {
                "value": "123",
                "description": "desc"
                },
            "ifOutErrors": {
                "value": "123",
                "description": "desc"
                },
            "ifOutUCastPkts": {
                "value": "123",
                "description": "desc"
                },
            "ifOutDiscard": {
                "value": "123",
                "description": "desc"
                }
            }
        ],
    "traps": [
        {
            "description": "desc"
        }
    ]
}
```

4.1.1.5.15 *Retrieve a network element [GET]*

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [NE][]

4.1.1.6 Group Host2Host

Host related APIs.

4.1.1.6.1 *Host2Hosts [/monitoring/host2hosts]*

Retrieve all host2hosts.

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/host2hosts" }
    },
    "host2hosts": [
        {
            "_links" : {
            "self": { "href": "/monitoring/host2hosts/Trento-
   12345;Berlin-5678" }
            },
            "id": "Trento-12345;Berlin-5678"
        }
```

```
        ]
}
```

4.1.1.6.2 List All Host2Hosts [GET]

Response 200 [Host2Hosts][]

4.1.1.6.3 Host2Host [/monitoring/host2hosts/{source};{dest}{?since}]

Retrieve a host2host object representing a connection between two hosts identified by the parameters {source} and {dest}, each one in the form regionid-hostid. Parameter {since} is optional. If present, the measures after {since} are provided, otherwise metrics referred to the last monitored interval are provided. Measures are aggregated on an hourly basis.

Parameters

```
source (string)
dest (string)
```

Model (application/hal+json)

Body

```
{
    "_links": {
        "self": { "href": "/monitoring/host2hosts/Trento-
   12345;Berlin-5678" }
    },
    "source": "Trento-12345",
    "dest": "Berlin-5678",

    "measures": [
        {
            "timestamp" : "2013-12-20 12.00",
            "owd_min": {
                "value": "123",
                "description": "desc"
                },

            "owd_max": {
                "value": "123",
                "description": "desc"
                },
            "bandwidth": {
                "value": "this is a string",
                "description": "desc"
                },
            "jitter": {
                "value": "123",
                "description": "desc"
                },
            "packet_loss": {
                "value": "123",
                "description": "desc"
                },
            "bandwidth_avg": {
                "value": "123",
                "description": "desc"
                }
            }
        ]
```

© 2013 - 2015 XIFI Consortium Parties

```
}
```

4.1.1.6.4 Retrieve an host2host [GET]

Parameters

```
since (optional, string) ... in the format `YYYY-MM-DD HH:MM'
```

Response 200 [Host2Host][]

4.2 Security

4.2.1 Identity management

Identity Management exposes three different APIs in order to manage the resources users, organizations, applications, etc.

- REST API: allows users to manage IdM resources from a REST API. It includes users, organizations and applications management.
- OAuth2 API: allows external applications (previously registered in the IdM) to login users using their IdM accounts.
- SCIM 2.0 API: allows administration users to perform massive requests to the IdM resources (users and organizations).

The Identity management sub-system published APIs are documented in its wiki page[57] and component page[58], and are detailed in the corresponding FI-WARE KeyRock[59] documentation.

4.2.2 Federated Access Control

The Access Control API complements the IdM API as an extension and is visible to the FIWARE developer as a part of the Identity and Access Management (IAM) sub-system API. The Access Control API is multi-tenant, i.e. multiple organizations, projects (also called tenants in Openstack or other cloud terminology) can manage and enforce access control policies at the same time, with complete isolation from each other. Each tenant is provided with the same API. This API is divided into two main parts:

- The Policy Administration Point (PAP) API

  The PAP API allows users to manage (create/update) the set of access control (XACML) policies for their tenant.

- Policy Decision Point (PDP) API

  The PDP API is typically used by PEP to request (XACML) authorization decision based on the XACML request sent by the PEP (all authorization attributes in the context of the access request) and the access control policy set in force for the tenant.

[57] http://wiki.fi-xifi.eu/Xifi:Wp2:Identity_Management_GE

[58] https://github.com/ging/fi-ware-idm/wiki

[59] https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Identity_Management_-_KeyRock_-_User_and_Programmers_Guide

© 2013 - 2015 XIFI Consortium Parties

The API is fully described in the FIWARE GE manual of the Access Control GE[60].

4.2.3 Security monitoring

The FI-WARE Security Monitoring GE does not provide a REST API for the data collection. The collection is done by SIEM (Security Information and Event Management) agents installed in the nodes with the Security Probes. These agents send the security events, already normalized to a common format, to predefined ports where the server is listening on the master node.

More information about the normalized event format and the different collection methods can be found in the FI-WARE Service Level SIEM Open API Specification[61].

4.3 User oriented and GUI subsystems

4.3.1 Monitoring dashboard

The Monitoring dashboard sub-system does not publish an API. It relies on other sub-systems and third-party components as documented in its corresponding wiki page[62]

4.3.2 Infographics and status pages

The Infographics and Status Pages sub-system does not publish an API. It relies on other subsystems as documented on in D4.4 [26] and its corresponding wiki page[63].

4.3.3 Cloud portal

The Cloud Portal sub-system does not publish an API. It relies on other sub-systems and third-party components as documented in its corresponding wiki page[64].

4.3.4 SLA manager

The subcomponent SLA Management exposes an API, which is responsible of providing the SLA lifecycle. The following list provides a summary of the set of interfaces that allow registering and listing the templates, agreements, enforcement and violations.

This new release has been extended and modified with respect the one described in the deliverable D2.2 [1]:

1. the message can be serialized by both xml and json;
2. increase the errors control and his notification in the message;
3. automatic creation of the enforcement, iv) refine the messages content for the template and the agreements.

---

[60] http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Access_Control_-_User_and_Programmers_Guide#XACML_Access_Control_Asset

[61] https://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Security-Monitoring:_Service_Level_SIEM_Open_sAPI_Specification

[62] http://wiki.fi-xifi.eu/Xifi:Wp4:_Monitoring_Dashboard

[63] http://wiki.fi-xifi.eu/Public:Infographics_and_Status_Pages

[64] http://wiki.fi-xifi.eu/Xifi:Wp4:Cloud_Portal

© 2013 - 2015 XIFI Consortium Parties

Further detailed information is provided in the API description wiki[65].

List of interfaces for the different resources:

Providers

The 'Providers' store all the providers included in the SLA Manager that can manage his own SLAs (templates and agreements).

- GET /providers/{uuid} Retrieves a specific provider identified by uuid
- GET /providers Retrieves the list of all providers
- POST /providers Creates a provider. The uuid is in the file beeing send
- DELETE /providers/{uuid} Removes the provider identified by uuid.

Templates

The 'Templates' provides methods to create, read, update, delete and list the templates of the SLA Manager.

- GET /templates/{TemplateId} Retrieves a template identified by TemplateId.
- GET /templates{?serviceId} Retrievers templates where serviceId is the id of service that is associated to the template.
- 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.
- DELETE /templates/{TemplateId} Removes the template identified by TemplateId.

Agreements

The 'Agreements' provides method to create, read, update, delete and list the agreements of the SLA Manager.

- GET /agreements/{AgreementId} Retrieves an agreement identified by AgreementId.
- GET /agreements/ Retrieves the list of all agreements.
- GET /agreements{?consumerId,providerId,active} Retrieves the list of all agreements where i) consumerId: uuid of the consumer; ii)providerId: uuid of the provider; iv)active: boolean value (value in {1,true,0,false}); if true, agreements currently enforced are returned.
- POST /agreements it creates 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.
- DELETE /agreements/{AgreementId} Removes the agreement identified by AgreementId.
- GET /agreements/active Returns the list of active agreements.
- GET /agreements/{AgreementId}/context Only the context from the agreement identified by AgreementId is returned.
- GET /agreements/{AgreementId}/guaranteestatus Gets the information of the status of the different Guarantee Terms of an agreement.

[65] http://wiki.fi-xifi.eu/Public:SLA_Manager#API_Specification

© 2013 - 2015 XIFI Consortium Parties

Enforcement Jobs

The '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 enforcement.

- GET /enforcements/{AgreementId} Retrieves an enforcement job identified by AgreementId.
- GET /enforcements Retrieves the list of all enforcement job.
- POST /enforcements Creates and enforcement job. Not required anymore. The enforcement job is automatically generated when an agreement is created.
- PUT /enforcements/{AgreementId}/start Starts an enforcement job.
- PUT /enforcements/{AgreementId}/stop Stops an enforcement job

Violations

The "Violation" indicates when a guarantee terms have not fulfilled and automatically a violations has been created. Hence, it is only possible to consult them.

- GET /violations/{uuid} Retrieves information from a violation identified by the uuid.
- GET /violations{?agreementId,guaranteeTerm,providerId,begin,end} Retrieves information from a violation identified by:
    - 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.

4.3.5 Federation manager

The following listings provide the APIs defined by the Federation Manager component and callable by the users of this component. The syntax used to define the APIs is the one proposed by apiary. The Federation Manager API consists of a set of interfaces that allow registering and listing regions, services and their endpoints offered by XIFI nodes. It is based on the OpenStack Identity Service API v3. Compared to the first release of this API described in D2.2 [1], the API has been extended and slightly modified.

4.3.5.1 Modifications

All API calls now require an **Authorization** header field to be set. This header must follow the format: "Authorization: Bearer *access_token*". This access token has to be obtained from the IDM Oauth API Other modifications that have been done in the API are concerning the region model. These are as follows:

- The attribute "adminUsername" has been replaced by "organizationId". A region is now not anymore bound to a single person but a whole organization instead.
- The attribute lists "federationIP_ranges", "publicIP_ranges" and "privateIP_ranges" have been added to cover requests by the Security Dashboard
- The attribute "publicNode" was added, to differentiate between public and private regions.

© 2013 - 2015 XIFI Consortium Parties

- The "uuid" attribute has been added to provide a numeric identifier for the region

4.3.5.2 Extensions

There are new extensions to the API compared to the status of the API already reported in D2.2. These are as follows:

- /v3/regions/{region_uuid}/members

  With this extension an IO can provide information on authorized members of his private region.

- /v3/regions

  Three optional query parameters where added to this API:

  - public (true/false)

    With this query parameter the caller gets only public (if public=true) or private (public=false) regions

  - status

    Restricts the response to regions with a specified status.

  - organizationId

    With this parameter the caller get only regions which are owner by the organization specified by the ID.

4.3.5.3 Service Catalogue [/v3/services]

The Service Catalogue stores information about types of Federation Services available. It provides methods to retrieve, update, and delete Service types.

4.3.5.3.1 Lists services [GET]

Parameters

```
type (optional, string)
page (optional, string)
per_page (optional, string)
```

Response 200 (application/hal+json)

```
[
  {
        "id": "--service-id--",
        "links": {
          "self": "http://identity:35357/v3/services/--service-id--"
   },
     "type": "volume"
   },
   {
     "id": "--service-id--",
     "links": {
       "self": "http://identity:35357/v3/services/--service-id--"
     },
     "type": "identity"
  }
]
```

4.3.5.3.2 Add a service [POST]

Request (application/json)

```
{
   "type": "volume"
}
```

Response 201 (application/hal+json)

```
{
   "service": {
      "id": "--service-id--",
      "type": "volume"
   }
}
```

4.3.5.4 Service Catalogue [/v3/services/{service_id}]

4.3.5.4.1 Get service instance [GET]

Response 200 (application/hal+json)

```
{
   "id": "--service-id--",
   "links": {
      "self": "http://identity:35357/v3/services/--service-id--"
   },
   "type": "identity"
}
```

4.3.5.4.2 Updated a specified service [PATCH]

Request (application/json)

```
{
   "type": "volume"
}
```

Response 201 (application/hal+json)

```
{
   "service": {
      "id": "--service-id--",
      "type": "volume"
   }
}
```

4.3.5.4.3 Delete a service [DELETE]

Response 204

4.3.5.5 Endpoints Catalogue [/v3/endpoints]

The Endpoints Catalogue stores information about deployments and accessibility of Federation Services. An Endpoint 'connects' a Service to a Region.

© 2013 - 2015 XIFI Consortium Parties

4.3.5.5.1 Lists endpoints [GET]

Parameters

```
service_id (optional, string)
region (optional, string)
interface (optional, string)
page (optional, string)
per_page (optional, string)
```

Response 200 (application/hal+json)

```
[
   {
      "id": "--endpoint-id--",
      "interface": "public",
      "links": {
        "self": "http://identity:35357/v3/endpoints/--endpoint-id--"
      },
      "name": "the public volume endpoint",
      "service_id": "--service-id--",
      "region": "--region-id--"
   },
   {
      "id": "--endpoint-id--",
      "interface": "internal",
      "links": {
        "self": "http://identity:35357/v3/endpoints/--endpoint-id--"
      },
      "name": "the internal volume endpoint",
      "service_id": "--service-id--",
      "region": "--region-id--"
   }
]
```

4.3.5.5.2 Add an endpoint [POST]

Request (application/json)

```
{
   "interface": "[admin|public|internal]",
   "name": "name",
   "url": "...",
   "region" : "...",
   "service_id" : "..."
}
```

Response 201 (application/hal+json)

```
{
   "id": "--endpoint-id--",
   "interface": "internal",
   "links": {
      "self": "http://identity:35357/v3/endpoints/--endpoint-id--"
   },
   "name": "the internal volume endpoint",
   "region" : "...",
   "service_id": "--service-id--"
}
```

© 2013 - 2015 XIFI Consortium Parties

4.3.5.6 Endpoints Catalogue [/v3/endpoints/{endpoint_id}]

4.3.5.6.1 Get endpoint instance [GET]

Response 200 (application/hal+json)

```
{
   "id": "--endpoint-id--",
   "interface": "internal",
   "links": {
      "self": "http://identity:35357/v3/endpoints/--endpoint-id--"
   },
   "name": "the internal volume endpoint",
   "region" : "...",
   "service_id": "--service-id--"
}
```

4.3.5.6.2 Update an endpoint [PATCH]

Request (application/json)

```
{
   "id": "--endpoint-id--",
   "interface": "internal",
   "links": {
      "self": "http://identity:35357/v3/endpoints/--endpoint-id--"
   },
   "name": "the internal volume endpoint",
   "region" : "...",
   "service_id": "--service-id--"
}
```

Response 201 (application/hal+json)

```
{
   "id": "--endpoint-id--",
   "interface": "internal",
   "links": {
      "self": "http://identity:35357/v3/endpoints/--endpoint-id--"
   },
   "name": "the internal volume endpoint",
   "region" : "...",
   "service_id": "--service-id--"
}
```

4.3.5.6.3 Delete an endpoint [DELETE]

Response 204

4.3.5.7 Regions Catalogue [/v3/regions]

The Regions Catalogue provides methods to create, read, update and delete regions/nodes to/from the federation. If a region corresponds to a node is still under discussion. Additionally the Regions Catalogue provides information on the registration status of a region and the contact information of e.g. the region administration, the support or the management.

© 2013 - 2015 XIFI Consortium Parties

4.3.5.7.1 Lists regions [GET]

Parameters

```
country (optional, string)
status (optional, string)
public (optional, string)
organizationId (optional, string)
page (optional, string)
per_page (optional, string)
```

Response 200 (application/hal+json)

```
[
   {
      "uuid": "--region-uuid--",
                    "id": "--region-id--",
                    "country": "isocode",
                    "latitude": "latitude",
                    "longitude": "longitude",
                    "public": "true/false",
                    "organizationId" :"organizationId",
                    "federationIP_ranges":
   ["10.10.10.12/24","10.10.11.12/32"],
                    "publicIP_ranges":
   ["130.10.10.12/24","130.10.11.12/32"],
                    "privateIP_ranges":
   ["192.10.10.12/24","192.10.11.12/32"],
                    "links": {
                        "self": "http://identity:35357/v3/regions/--
   region-uuid--"
                    }
   },
   {
      "uuid": "--region-uuid--",
                    "id": "--region-id--",
                    "country": "isocode",
                    "latitude": "latitude",
                    "longitude": "longitude",
                    "public": "true/false",
                    "organizationId" :"organizationId",
                    "federationIP_ranges":
   ["10.10.13.12/24","10.10.14.12/32"],
                    "publicIP_ranges":
   ["130.10.15.12/24","130.10.17.12/32"],
                    "privateIP_ranges":
   ["192.10.14.12/24","192.10.17.12/32"],
                    "links": {
                        "self": "http://identity:35357/v3/regions/--
   region-uuid--"
                    }
   }
]
```

4.3.5.7.2 Add a region [POST]

Request (application/json)

```
{
    "id": "--region-id--",
    "country": "isocode",
```

```
    "latitude": "latitude",
    "longitude": "longitude",
    "public": "true/false",
    "organizationId" :"organizationId",
    "federationIP_ranges": ["10.10.10.12/24","10.10.11.12/32"],
    "publicIP_ranges": ["130.10.10.12/24","130.10.11.12/32"],
    "privateIP_ranges": ["192.10.10.12/24","192.10.11.12/32"]
}
```

Response 201 (application/hal+json)

```
{
           "uuid": "--region-uuid--",
                "id": "--region-id--",
                "country": "isocode",
                "latitude": "latitude",
                "longitude": "longitude",
                "public": "true/false",
                "organizationId" :"organizationId",
                "federationIP_ranges":
   ["10.10.10.12/24","10.10.11.12/32"],
                "publicIP_ranges":
   ["130.10.10.12/24","130.10.11.12/32"],
                "privateIP_ranges":
   ["192.10.10.12/24","192.10.11.12/32"],
                "links": {
                    "self": "http://identity:35357/v3/regions/--
   region-uuid--"
                }
}
```

4.3.5.8 Regions Catalogue [/v3/regions/{region_uuid}]

Get region instance [GET]

Response 200 (application/hal+json)

```
{
           "uuid": "--region-uuid--",
                "id": "--region-id--",
                "country": "isocode",
                "latitude": "latitude",
                "longitude": "longitude",
                "public": "true/false",
                "organizationId" :"organizationId",
                "federationIP_ranges":
   ["10.10.10.12/24","10.10.11.12/32"],
                "publicIP_ranges":
   ["130.10.10.12/24","130.10.11.12/32"],
                "privateIP_ranges":
   ["192.10.10.12/24","192.10.11.12/32"],
                "links": {
                    "self": "http://identity:35357/v3/regions/--
   region-uuid--"
                }
}
```

4.3.5.8.1 Update a region [PATCH]

Request (application/json)

```
{

                "id": "--region-id--",
                "country": "isocode",
                "latitude": "latitude",
                "longitude": "longitude",
                "public": "true/false",
                "organizationId" :"organizationId",
                "federationIP_ranges":
   ["10.10.10.12/24","10.10.11.12/32"],
                "publicIP_ranges":
   ["130.10.10.12/24","130.10.11.12/32"],
                "privateIP_ranges":
   ["192.10.10.12/24","192.10.11.12/32"],
}
```

Response 201 (application/hal+json)

```
{
           "uuid": "--region-uuid--",
                "id": "--region-id--",
                "country": "isocode",
                "latitude": "latitude",
                "longitude": "longitude",
                "public": "true/false",
                "organizationId" :"organizationId",
                "federationIP_ranges":
   ["10.10.10.12/24","10.10.11.12/32"],
                "publicIP_ranges":
   ["130.10.10.12/24","130.10.11.12/32"],
                "privateIP_ranges":
   ["192.10.10.12/24","192.10.11.12/32"],
                "links": {
                    "self": "http://identity:35357/v3/regions/--
   region-uuid--"
                }
}
```

4.3.5.8.2 Delete a Region [DELETE]

Response 204

4.3.5.9 Regions Catalogue [/v3/regions/{region_uuid}/status]

4.3.5.9.1 Get region status [GET]

Response 200 (application/hal+json)

```
{
   "region": "--region-id--",
   "timestamp": "--update--timestamp--",
   "status": "maintenance/active/registered"
}
```

4.3.5.9.2 Update a region [PATCH]

Request (application/json)

```
{
   "status": "maintenance/active/registered"
```

© 2013 - 2015 XIFI Consortium Parties

```
}
```

Response 201 (application/hal+json)

```
{
   "region": "--region-id--",
   "timestamp": "--update--timestamp--",
   "status": "maintenance/active/registered"
}
```

4.3.5.10 Regions Catalogue [/v3/regions/{region_uuid}/contacts]

Get region contacts [GET]

Response 200 (application/hal+json)

```
 [
   {
      "id": "1"
      "name": "someone",
      "country" : "de",
      "fax" : "faxit",
      "phone" : "phoneno",
      "email" : "somemail",
      "type" : "organization",
      "address" : "somewhere"
   },
   {
      "id": "2"
      "name": "someone else",
      "country" : "de",
      "fax" : "faxit",
      "phone" : "phoneno",
      "email" : "somemail",
      "type" : "support",
      "address" : "somewhere"
   },
   {
      "id": "3"
      "name": "anotherone",
      "country" : "de",
      "fax" : "faxit",
      "phone" : "phoneno",
      "email" : "somemail",
      "type" : "management",
      "address" : "somewhere"
   }
]
```

4.3.5.10.1 Add a region contact [POST]

Request (application/json)

```
{
   "name": "something",
   "country" : "de",
   "fax" : "faxit",
   "phone" : "phoneno",
   "email" : "somemail",
   "type" : "organization",
   "address" : "somewhere"
```

© 2013 - 2015 XIFI Consortium Parties

```
}
```

Response 201 (application/hal+json)

```
{
   "id": "1"
   "name: "something"
   "country": "de"
   "fax": "faxit"
   "phone": "phoneno"
   "email": "somemail"
   "type": "organization"
   "address": "somewhere"
   _links: {
      self: [1]
            0:  {
              href: "http://localhost:8080/federationManager/api/
                        v3/regions/1/contacts/1"
            }
   }
}
```

4.3.5.11 Regions Catalogue [/v3/regions/{region_uuid}/contacts/{contact_id}]

4.3.5.11.1 Get region contact [GET]

Response 200 (application/hal+json)

```
{
   "id": "1"
   "name: "something"
   "country": "de"
   "fax": "faxit"
   "phone": "phoneno"
   "email": "somemail"
   "type": "organization"
   "address": "somewhere"
   _links: {
      self: [1]
            0:  {
              href: "http://localhost:8080/federationManager/api/v3/
                        regions/1/contacts/1"
            }
   }
```

4.3.5.11.2 Update a region contact [PATCH]

Request (application/json)

```
{
   "name": "my name"
}
```

Response 201 (application/hal+json)

```
{
   "id": "1"
   "name: "my name"
   "country": "de"
   "fax": "faxit"
   "phone": "phoneno"
```

© 2013 - 2015 XIFI Consortium Parties

```
    "email": "somemail"
    "type": "organization"
    "address": "somewhere"
    _links: {
       self: [1]
             0:  {
               href: "http://localhost:8080/federationManager/api/
                         v3/regions/1/contacts/1"
             }
    }
}
```

4.3.5.11.3 Delete a contact [DELETE]

Response 204

4.3.5.12 Regions Catalogue [/v3/regions/{region_uuid}/members

4.3.5.12.1 Get region members [GET]

Response 200 (application/hal+json)

```
[
    {
        "id": "1",
        "tenand_id": "000000113",
        "links": {
                "self":
   "http://identity:35357/v3/regions/1/members/1"
        }
    },
    {
        "id": "2",
        "tenand_id": "000000256",
        "links": {
                "self":
   "http://identity:35357/v3/regions/1/members/2"
        }
    },
    {
        "id": "3",
        "tenand_id": "000000253",
        "links": {
            "self": "http://identity:35357/v3/regions/1/members/3"
        }
    }
]
```

4.3.5.12.2 Add a region member [POST]

Request (application/json)

```
{
         "tenand_id": "000000113"
    }
```

Response 201 (application/hal+json)

```
{
    "id": "3",
```

© 2013 - 2015 XIFI Consortium Parties

```
    "tenand_id": "000000253",
    "links": {
        "self": "http://identity:35357/v3/regions/1/members/3"
    }
}
```

4.3.5.13 Regions Catalogue [/v3/regions/{region_uuid}/members/{member_id}]

4.3.5.13.1 Get region member [GET]

Response 200 (application/hal+json)

```
{
    "id": "3",
    "tenand_id": "000000253",
    "links": {
        "self": "http://identity:35357/v3/regions/1/members/3"
    }
}
```

4.3.5.13.2 Update a region member [PATCH]

Request (application/json)

```
{
    "tenand_id": "000000253"
}
```

Response 201 (application/hal+json)

```
{
    "id": "3",
    "tenand_id": "000000253",
    "links": {
        "self": "http://identity:35357/v3/regions/1/members/3"
    }
}
```

4.3.5.13.3 Delete a member [DELETE]

Response 204

4.3.6 Interoperability tool

The interoperability tool publishes an API as documented in D4.4 [26] and its corresponding wiki page[66]. Currently, the Cloud Portal (cf. section 3.3.4) is its sole client.

4.3.7 Resource catalogue

The Resource Catalogue and Recommender is a graphical web client interface that does not provide a distinct API.

[66] http://wiki.fi-xifi.eu/Xifi:Wp4:Interoperability_Tool

© 2013 - 2015 XIFI Consortium Parties

4.4 Deployment and Operations

4.4.1 Infrastructure toolbox

The Infrastructure toolbox is a dedicated tool for setting-up an infrastructure node from "bare metal". It currently does not provide a distinct API.

4.4.2 Deployment and Configuration Adapter

DCA API is a set of APIs that allows for deploying and/or polling (possibly multiple) GEs, located at different XIFI regions (DCRM instances).

The interested user may also see the link http://docs.dca.apiary.io/ for more information.

4.4.2.1 Endpoints

Add a new identity management endpoint to the DCA database. The endpoint actually represents a new DCRM instance within the context of XIFI federation.

4.4.2.1.1 Add Endpoint

Creates an endpoint for use by DCA.

POST /dca/addEndpoint

Request

```
curl --include \
--request POST \
--header "Content-Type: application/json" \
--data-binary '{"region":"A_Region", "url":"http://hostname.example.com
/keystone/v2.0"}' \
http://dca.apiary-mock.com/dca/addEndpoint
```

Response

```
201 (Created)
Content-Type: application/json
{"url":"http://hostname.example.com/keystone/v2.0", "region":"A_Region"}
```

4.4.2.2 Flavors

Polls a specific DCRM instance for information related to the VM flavors that can be used to deploy new VMs/GEs.

4.4.2.2.1 Flavors

The available flavors for use by the user in a specific DCRM instance.

Note: This operation requires definition of the appropriate management endpoint in DCA database.

GET /dca/flavors/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/flavors/region/
```

Response

```
200 (OK)
```

© 2013 - 2015 XIFI Consortium Parties

```
Content-Type: application/json
{
"flavors":[
   {
      "id":"1",
      "name":"m1.tiny",
      "vcpus":"1",
      "ram":512,
      "disk":"0",
      "swap":"",
      "links":[
         {
            "rel":"self",
            "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/1",
            "type":null
         },
         {
            "rel":"bookmark",
            "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/1",
            "type":null
         }
      ],
      "public":null,
      "OS-FLV-EXT-DATA:ephemeral":0,
      "rxtx_factor":1.0,
      "OS-FLV-DISABLED:disabled":null,
      "rxtx_quota":null,
      "rxtx_cap":null,
      "os-flavor-access:is_public":null
   },
   {
      "id":"2",
      "name":"m1.small",
      "vcpus":"1",
      "ram":2048,
      "disk":"10",
      "swap":"",
      "links":[
         {
            "rel":"self",
            "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000001807/flavors/2",
            "type":null
         },
         {
            "rel":"bookmark",
            "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000001807/flavors/2",
            "type":null
         }
      ],
      "public":null,
      "OS-FLV-EXT-DATA:ephemeral":20,
      "rxtx_factor":1.0,
      "OS-FLV-DISABLED:disabled":null,
      "rxtx_quota":null,
      "rxtx_cap":null,
```

```
            "os-flavor-access:is_public":null
        },
        {
            "id":"3",
            "name":"m1.medium",
            "vcpus":"2",
            "ram":4096,
            "disk":"10",
            "swap":"",
            "links":[
                {
                    "rel":"self",
                    "href":"http://cloud.lab.fi-
    ware.eu:8774/v2/00000000000000000000000000001807/flavors/3",
                    "type":null
                },
                {
                    "rel":"bookmark",
                    "href":"http://cloud.lab.fi-
    ware.eu:8774/00000000000000000000000000001807/flavors/3",
                    "type":null
                }
            ],
            "public":null,
            "OS-FLV-EXT-DATA:ephemeral":40,
            "rxtx_factor":1.0,
            "OS-FLV-DISABLED:disabled":null,
            "rxtx_quota":null,
            "rxtx_cap":null,
            "os-flavor-access:is_public":null
        }
    ]
    }
```


4.4.2.3 Keypairs

Polls a specific DCRM instance for information related to the keypairs that the user has created in the context of the specific DCRM instance.

4.4.2.3.1 Keypairs

The available flavors for use by the user in a specific DCRM instance.

Note: This operation requires definition of the appropriate management endpoint in DCA database.

GET /dca/keypairs/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/keypairs/region/
```

Response


```
200 (OK)
Content-Type: application/json
{
"keypairs": [
                {
                    "name":"panos",
```

```
"fingerprint":"22:ec:58:0a:c5:f5:c6:d4:7b:91:64:26:5f:17:e9:8b",
                    "user_id":null,
                    "public_key":"ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAAAgQDBgi/eZqP7IKqygkvTn2pkrlPn3LKg57SU8j
yRQNxMq37fG6RZUtfTSZsaJs0lnQTQvFhfuuzRXs4/9eYQ2CD82BcFOqQr6p2CyhgY
zMnEv4xXIz53fUFXGqSjsvUb+YbR6fbSibl3OaXLkNhg4FQH4lGQHDEVQEABCyagyQ
TuPQ== nova@gcsic001.ifca.esn",
                    "private_key":null
                }
            ]
}
```

4.4.2.4 Security Groups

Polls a specific DCRM instance for information related to the security groups that the user has created in the context of the specific DCRM instance.

4.4.2.4.1 Security Groups

The available security groups for use by the user in a specific DCRM instance.

Note: This operation requires definition of the appropriate management endpoint in DCA database.

GET /dca/securitygroups/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/securitygroups/region/
```

Response

```
200 (OK)
Content-Type: application/json
{
"security_groups":[
   {
      "id":2372,
      "name":"context_broker",
      "description":"context broker security group",
      "rules":[
         {
            "id":4211,
            "name":null,
            "group":{
               "name":null,
               "tenant_id":null
            },
            "parent_group_id":2372,
            "from_port":1026,
            "to_port":1026,
            "ip_protocol":"tcp",
            "ip_range":{
               "cidr":"0.0.0.0/0"
            }
         },
         {
            "id":4212,
            "name":null,
            "group":{
               "name":null,
```

```
                    "tenant_id":null
                },
                "parent_group_id":2372,
                "from_port":8,
                "to_port":0,
                "ip_protocol":"icmp",
                "ip_range":{
                    "cidr":"0.0.0.0/0"
                }
            },
            {
                "id":4213,
                "name":null,
                "group":{
                    "name":null,
                    "tenant_id":null
                },
                "parent_group_id":2372,
                "from_port":0,
                "to_port":0,
                "ip_protocol":"icmp",
                "ip_range":{
                    "cidr":"0.0.0.0/0"
                }
            }
        ],
        "links":null,
        "tenant_id":"00000000000000000000000000001807"
    },
    {
        "id":2078,
        "name":"default",
        "description":"default",
        "rules":[
            {
                "id":3425,
                "name":null,
                "group":{
                    "name":null,
                    "tenant_id":null
                },
                "parent_group_id":2078,
                "from_port":22,
                "to_port":22,
                "ip_protocol":"tcp",
                "ip_range":{
                    "cidr":"0.0.0.0/0"
                }
            }
        ],
        "links":null,
        "tenant_id":"00000000000000000000000000001807"
    },
    {
        "id":2079,
        "name":"monitoring",
        "description":"For FI-WARE monitoring",
        "rules":[
            {
                "id":4137,
```

```
                "name":null,
                "group":{
                   "name":null,
                   "tenant_id":null
                },
                "parent_group_id":2079,
                "from_port":1026,
                "to_port":1026,
                "ip_protocol":"tcp",
                "ip_range":{
                   "cidr":"0.0.0.0/0"
                }
             },
             {
                "id":4141,
                "name":null,
                "group":{
                   "name":null,
                   "tenant_id":null
                },
                "parent_group_id":2079,
                "from_port":80,
                "to_port":80,
                "ip_protocol":"tcp",
                "ip_range":{
                   "cidr":"0.0.0.0/0"
                }
             }
          ],
          "links":null,
          "tenant_id":"00000000000000000000000000001807"
       }
    ]
    }
```


4.4.2.5 Images

Polls a specific DCRM instance for information related to the images of GEs and other VMs available to the user in the context of the specific DCRM instance.

4.4.2.5.1 Images

The images available to the user in a specific DCRM instance.

Note: This operation requires definition of the appropriate management endpoint in DCA database.

GET /dca/images/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/images/region/
```

Response


```
200 (OK)
Content-Type: application/json
{
 "images":[
  {
     "status":"active",
```

```
            "name":"CentOS_6.4",
            "deleted":false,
            "container_format":"bare",
            "created_at":"2013-11-04T19:13:46",
            "disk_format":"iso",
            "updated_at":"2013-11-04T19:14:01",
            "id":"79e9245c-3a02-48af-8dd2-ada0d893331e",
            "min_disk":0,
            "protected":false,
            "min_ram":0,
            "checksum":"4a5fa01c81cc300f4729136e28ebe600",
            "owner":"e9312e85dde04636a63c1b340f89242a",
            "is_public":true,
            "deleted_at":null,
            "properties":{
            },
            "size":358959104
        },
        {
            "status":"active",
            "name":"Cirros 0.3.1",
            "deleted":false,
            "container_format":"bare",
            "created_at":"2013-10-23T14:28:21",
            "disk_format":"qcow2",
            "updated_at":"2013-10-23T14:28:22",
            "id":"c4c6463f-0acb-4e06-8051-1e14070c154d",
            "min_disk":0,
            "protected":false,
            "min_ram":0,
            "checksum":"d972013792949d0d3ba628fbe8685bce",
            "owner":"e9312e85dde04636a63c1b340f89242a",
            "is_public":false,
            "deleted_at":null,
            "properties":{
            },
            "size":13147648
        },
        {
            "status":"active",
            "name":"orion",
            "deleted":false,
            "container_format":"bare",
            "created_at":"2013-11-20T18:06:47",
            "disk_format":"qcow2",
            "updated_at":"2013-11-20T18:12:22",
            "id":"791c1279-4d38-4f89-a33b-a3a93a29e75c",
            "min_disk":0,
            "protected":false,
            "min_ram":0,
            "checksum":"d60b7cfc2f87a9b69095cbd627805bf0",
            "owner":"e9312e85dde04636a63c1b340f89242a",
            "is_public":false,
            "deleted_at":null,
            "properties":{
                "instance_uuid":"a9259820-dd5e-4fb4-9581-09acaddf199b",
                "image_location":"snapshot",
                "image_state":"available",
                "instance_type_memory_mb":"2048",
                "instance_type_swap":"0",
```


© 2013 - 2015 XIFI Consortium Parties

```
            "instance_type_vcpu_weight":"None",
            "image_type":"snapshot",
            "instance_type_id":"5",
            "ramdisk_id":null,
            "instance_type_name":"m1.small",
            "instance_type_ephemeral_gb":"0",
            "instance_type_rxtx_factor":"1",
            "kernel_id":null,
            "instance_type_flavorid":"2",
            "instance_type_vcpus":"1",
            "user_id":"752627b3561f46b08bc27726ce2630ce",
            "instance_type_root_gb":"20",
            "base_image_ref":"79e9245c-3a02-48af-8dd2-ada0d893331e",
            "owner_id":"e9312e85dde04636a63c1b340f89242a"
        },
        "size":358875136
    }
  ]
  }
```


4.4.2.6 Networks

Polls a specific DCRM instance for information related to the networks that are available to the user in the context of the specific DCRM instance.

4.4.2.6.1 Networks

The available networks for use by the user in a specific DCRM instance.

Note: This operation requires definition of the appropriate management endpoint in DCA database.

GET /dca/networks/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/networks/region/
```

Response


```
200 (OK)
Content-Type: application/json
{
  "networks":[
    {
      "status":"ACTIVE",
      "subnets":[
         "3b873329-6469-4d44-9814-93be7b6d7eb2"
      ],
      "name":"net-0",
      "id":"a714bca6-7f2d-4181-91fb-44e6b603a7e4",
      "shared":"false",
      "provider: physical_network":null,
      "admin_state_up":true,
      "tenant_id":"e9312e85dde04636a63c1b340f89242a",
      "provider: network_type":"gre",
      "router: external":"false",
      "provider: segmentation_id":"1"
    },
    {
      "status":"ACTIVE",
```


© 2013 - 2015 XIFI Consortium Parties

```
        "subnets":[
           "ed1fa327-81ba-40ca-b523-aa64e45ba091"
        ],
        "name":"ext_net",
        "id":"c3a72055-71ac-4cc7-afad-4869dc602b19",
        "shared":"false",
        "provider: physical_network":null,
        "admin_state_up":true,
        "tenant_id":"e9312e85dde04636a63c1b340f89242a",
        "provider: network_type":"gre",
        "router: external":"true",
        "provider: segmentation_id":"2"
     }
  ]
  }
```

4.4.2.7 Servers

Deploys single or groups of servers (either simple VMs or GEs) in one or more DCRM instances, also providing capabilities for polling servers state either actively, or passively.

4.4.2.7.1 List server operations

The servers that the user has created in the specified region. All servers (either plain VMs or GEs) are homogeneously treated.

GET /dca/servers/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/region/
```

Response

```
200 (OK)
Content-Type: application/json
[
{
   "id":"6fda6be8-0e70-4d08-9731-c858d6a27f50",
   "name":"test-xifi-proton",
   "nid":"146",
   "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
   "flavorRef":"2",
   "keyName":"xifi",
   "securityGroups":"default,cep",
   "created":1390051880000,
   "status":null,
   "tenantId":"00000000000000000000000000000101",
   "userId":"john-smith",
   "region":"RegionOne"
},
{
   "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
   "name":"test-xifi-proton-r1",
   "nid":"146",
   "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
   "flavorRef":"2",
   "keyName":"xifi",
   "securityGroups":"default,cep",
   "created":1390135132000,
```

© 2013 - 2015 XIFI Consortium Parties

```
    "status":null,
    "tenantId":"00000000000000000000000000000101",
    "userId":"john-smith",
    "region":"RegionOne"
}
]
```

4.4.2.7.2 List servers per user

List severs of a spesific user. Note that the DCRM instances are not queried on this method invocation.

GET /dca/servers/user/{user_id}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/user/john-smith
```

Response

```
200 (OK)
Content-Type: application/json
[
{
    "id":"6fda6be8-0e70-4d08-9731-c858d6a27f50",
    "name":"test-xifi-proton",
    "nid":"146",
    "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
    "flavorRef":"2",
    "keyName":"xifi",
    "securityGroups":"default,cep",
    "created":1390051880000,
    "status":null,
    "tenantId":"00000000000000000000000000000101",
    "userId":"john-smith",
    "region":"RegionOne"
},
{
    "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
    "name":"test-xifi-proton-r1",
    "nid":"146",
    "imageRef":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
    "flavorRef":"2",
    "keyName":"xifi",
    "securityGroups":"default,cep",
    "created":1390135132000,
    "status":null,
    "tenantId":"00000000000000000000000000000101",
    "userId":"john-smith",
    "region":"RegionTwo"
}
]
```

4.4.2.7.3 Create server

Create a new server operation

Note: This operation requires definition of the appropriate management endpoint in DCA database.

POST /dca/servers/

© 2013 - 2015 XIFI Consortium Parties

Request

```
curl --include \
--request POST \
--header "Content-Type: application/json" \
--data-binary '{ "name":"test-3", "imageRef":"90d4865d-5e7b-4d95-
af2c-69753e1740d6", "flavorRef":"2", "keyName":"xifi",
"securityGroups":[ { "name":"default" }, { "name":"cep" }],
"region":"RegionOne" }' \
http://dca.apiary-mock.com/dca/servers/
```

Response

```
201 (Created)
Content-Type: application/json
{
   "id":"ffae6295-9aba-48a8-908d-4eecb71bd79c",
   "name":null,
   "addresses":null,
   "links":[
      {
         "rel":"self",
         "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000000101/servers/ffae6295-9aba-48a8-908d-
4eecb71bd79c",
         "type":null
      },
      {
         "rel":"bookmark",
         "href":"http://cloud.lab.fi-
ware.eu:8774/00000000000000000000000000000101/servers/ffae6295-9aba-48a8-908d-
4eecb71bd79c",
         "type":null
      }
   ],
   "image":null,
   "flavor":null,
   "accessIPv4":null,
   "accessIPv6":null,
   "configDrive":null,
   "status":null,
   "progress":null,
   "fault":null,
   "tenantId":null,
   "userId":null,
   "keyName":null,
   "hostId":null,
   "updated":null,
   "created":null,
   "metadata":null,
   "securityGroups":null,
   "taskState":null,
   "powerState":null,
   "vmState":null,
   "host":null,
   "instanceName":null,
   "hypervisorHostname":null,
   "diskConfig":"MANUAL",
   "availabilityZone":null,
   "uuid":null,
   "adminPass":"hRmXkm97DRqY"
```

© 2013 - 2015 XIFI Consortium Parties

```
}
```

4.4.2.7.4 Delete server

Remove server from DCA database

DELETE /dca/servers/{id}

Request

```
curl --include \
 --request DELETE \
http://dca.apiary-mock.com/dca/servers/6fda6be8-0e70-4d08-9731-c858d6a27f50
```

Response

```
200 (OK)
Content-Type: application/json
{ "deleted":"d5c22170-38d6-4344-9d62-c9770e550cee" }
```

4.4.2.7.5 Live server information

Shows the status of a server, after polling the host DCRM instance

Note: This operation requires definition of the appropriate management endpoint in DCA database.

GET /dca/servers/live/region/{region_name}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/live/region/
```

Response

```
200 (OK)
Content-Type: application/json
{
   "list":[
      {
         "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
         "name":"test-xifi-proton-r1",
         "addresses":{
            "addresses":{
               "private":[
                  {
                     "macAddr":null,
                     "version":"4",
                     "addr":"10.0.1.202",
                     "type":null
                  }
               ]
            }
         },
         "links":[
            {
               "rel":"self",
               "href":"http://cloud.lab.fi-
ware.eu:8774/v2/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-
355160bc93cc",
               "type":null
            },
            {
```

```
                "rel":"bookmark",
                "href":"http://cloud.lab.fi-ware.eu:8774/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-355160bc93cc",
                "type":null
            }
        ],
        "image":{
            "id":"90d4865d-5e7b-4d95-af2c-69753e1740d6",
            "status":null,
            "name":null,
            "progress":null,
            "minRam":null,
            "minDisk":null,
            "created":null,
            "updated":null,
            "size":null,
            "metadata":null,
            "links":[
                {
                    "rel":"bookmark",
                    "href":"http://cloud.lab.fi-ware.eu:8774/00000000000000000000000000000101/images/90d4865d-5e7b-4d95-af2c-69753e1740d6",
                    "type":null
                }
            ]
        },
        "flavor":{
            "id":"2",
            "name":null,
            "vcpus":null,
            "ram":null,
            "disk":null,
            "ephemeral":null,
            "swap":null,
            "rxtxFactor":null,
            "disabled":null,
            "rxtxQuota":null,
            "rxtxCap":null,
            "links":[
                {
                    "rel":"bookmark",
                    "href":"http://cloud.lab.fi-ware.eu:8774/00000000000000000000000000000101/flavors/2",
                    "type":null
                }
            ],
            "public":null
        },
        "accessIPv4":"",
        "accessIPv6":"",
        "configDrive":"",
        "status":"ACTIVE",
        "progress":0,
        "fault":null,
        "tenantId":"00000000000000000000000000000101",
        "userId":"john-smith",
        "keyName":"xifi",
```

```
"hostId":"c0d2f2faa797bfb8be05680a6d2f31ba4553f7219e35a13011cb22d1
",
          "updated":"2014-01-19T12:39:05Z",
          "created":"2014-01-19T12:38:52Z",
          "metadata":{
          },
          "securityGroups":null,
          "taskState":null,
          "powerState":"1",
          "vmState":"active",
          "host":"gcsic022.ifca.es",
          "instanceName":"instance-00000ad2",
          "hypervisorHostname":null,
          "diskConfig":"MANUAL",
          "availabilityZone":null,
          "uuid":null,
          "adminPass":null
       }
    ]
 }
```

4.4.2.8 GE-related queries

Allows for more fine-grained monitoring of the federation, via enabling context-aware queries related to the deployment region of a GE and/or its functional description.

GET /dca/servers/ge/region/{region_name}/desc/{ge_desc}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/ge/region//desc/
```

Response

```
 200 (OK)
 Content-Type: application/json
[
{
   "id":"1ae2d8d8-e8ed-4e4e-88bd-4209b808eeda",
   "name":"test-xifi-proton-r2",
   "nid":"146",
   "imageRef":"7b001833-5eaa-4a4d-84fa-803e3c59377d",
   "flavorRef":"2",
   "keyName":"xifi",
   "securityGroups":"default,cep",
   "created":1390052553000,
   "status":null,
   "tenantId":"00000000000000000000000000000158",
   "userId":"john-smith",
   "region":"RegionTwo"
}
]
```

4.4.2.8.1 Popular GEs

Returns a list of the most popular GEs in a region.

GET /dca/servers/ge/popular/region/{region}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/ge/popular/region/RegionOne
```

Response

```
200 (OK)
Content-Type: application/json
[
{
"name": "Marketplace - SAP RI",
"nid": "95",
"deployments": 2
},
{
"name": "Complex Event Processing (CEP) - IBM Proactive Technology
Online",
"nid": "146",
"deployments": 2
},
{
"name": "Repository - SAP RI",
"nid": "58",
"deployments": 1
},
{
"name": "Publish-Subscribe Context Broker - Orion Context Broker",
"nid": "344",
"deployments": 1
}
]
```

4.4.2.8.2 Active GEs in a specific time period

Returns a list of active GEs in a specific time period.

GET /dca/servers/ge/region/{region}/desc/{desc}/time/{time1}/{time2}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/ge/region/RegionOne/desc/344/time/2014-07-
14%2000%3A00%3A00/2014-08-14%2000%3A00%3A00
```

Response

```
200 (OK)
Content-Type: application/json
[
{
"id": "125bf922-dbb4-4693-9f53-e3c37880d82f",
"name": "CEP_1",
"nid": "146",
"imageRef": "262a9194-f00f-4c0d-8f3e-f331c80070c8",
"flavorRef": "m1.small (13)",
"keyName": "user1key",
"securityGroups": "default",
"created": 1407231675000,
"deleted": null,
"status": "P_DELETED",
"tenantId": "00000000000000000000000000002782",
"userId": "user1",
"region": "region"
```

© 2013 - 2015 XIFI Consortium Parties

```
    },
    {
    "id": "4bb0ec63-14e3-4448-86a5-62d56b11657f",
    "name": "CEP_2",
    "nid": "146",
    "imageRef": "262a9194-f00f-4c0d-8f3e-f331c80070c8",
    "flavorRef": "m1.small (13)",
    "keyName": "user2key",
    "securityGroups": "default",
    "created": 1408339950000,
    "deleted": null,
    "status": "ACTIVE",
    "tenantId": "00000000000000000000000000000125",
    "userId": "user2",
    "region": "Region"
    }
    ]
```

4.4.2.9 Combinatorial GE-related queries

Allows for searching DCRM instances that host a combination of (active) GEs.

GET /dca/servers/ge/complex/{ge_desc1}/{ge_desc2}

Request

```
curl --include \
http://dca.apiary-mock.com/dca/servers/ge/complex/%22Publish-
Subscribe%20Context%20Broker%20-
%20Orion%20Context%20Broker%22%20or%20344/Marketplace%20-
%20SAP%20RI%20or%2095
```

Response

```
200 (OK)
Content-Type: application/json
[ "RegionOne" ]
```

4.4.2.9.1 Multiple server deployment

Deploys multiple servers, at possibly different regions, with a single request.

Note: This operation requires definition of the appropriate management endpoint in DCA database.

POST /dca/multiple

Request

```
 curl --include \
 --request POST \
 --data-binary '{
"list": [
    {
        "name": "test-xifi-proton-r1",
        "imageRef": "90d4865d-5e7b-4d95-af2c-69753e1740d6",
        "flavorRef": "2",
        "keyName": "xifi",
        "securityGroups": [
            {
                "name": "default"
            },
            {
```

```
                    "name": "cep"
                }
            ],
            "region": "RegionOne"
        },
        {
            "name": "test-xifi-orion-r2",
            "imageRef": "02fdb0bc-6b47-4af4-ab13-95508033cdb4",
            "flavorRef": "2",
            "keyName": "xifi",
            "securityGroups": [
                {
                    "name": "default"
                }
            ],
            "region": "RegionTwo"
        }
    ]
    }' \
     http://dca.apiary-mock.com/dca/multiple
```

Response

```
    200 (OK)
    Content-Type: application/json
    [
    {
       "id":"a335265d-777e-42fa-8f5a-355160bc93cc",
       "name":null,
       "addresses":null,
       "links":[
          {
             "rel":"self",
             "href":"http://cloud.lab.fi-
    ware.eu:8774/v2/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-
    355160bc93cc",
             "type":null
          },
          {
             "rel":"bookmark",
             "href":"http://cloud.lab.fi-
    ware.eu:8774/00000000000000000000000000000101/servers/a335265d-777e-42fa-8f5a-
    355160bc93cc",
             "type":null
          }
       ],
       "image":null,
       "flavor":null,
       "accessIPv4":null,
       "accessIPv6":null,
       "configDrive":null,
       "status":null,
       "progress":null,
       "fault":null,
       "tenantId":null,
       "userId":null,
       "keyName":null,
       "hostId":null,
       "updated":null,
       "created":null,
```

```
      "metadata":null,
      "securityGroups":null,
      "taskState":null,
      "powerState":null,
      "vmState":null,
      "host":null,
      "instanceName":null,
      "hypervisorHostname":null,
      "diskConfig":"MANUAL",
      "availabilityZone":null,
      "uuid":null,
      "adminPass":"ZGs4QxAvZc8c"
   },
   {
      "id":"29c1ee5e-fac8-451d-85d9-963ed7ae2386",
      "name":null,
      "addresses":null,
      "links":[
         {
            "rel":"self",

    "href":"http://130.206.80.11:8774/v2/00000000000000000000000000000158/servers/29c1ee5e
    -fac8-451d-85d9-963ed7ae2386",
            "type":null
         },
         {
            "rel":"bookmark",

    "href":"http://130.206.80.11:8774/00000000000000000000000000000158/servers/29c1ee5e-
    fac8-451d-85d9-963ed7ae2386",
            "type":null
         }
      ],
      "image":null,
      "flavor":null,
      "accessIPv4":null,
      "accessIPv6":null,
      "configDrive":null,
      "status":null,
      "progress":null,
      "fault":null,
      "tenantId":null,
      "userId":null,
      "keyName":null,
      "hostId":null,
      "updated":null,
      "created":null,
      "metadata":null,
      "securityGroups":null,
      "taskState":null,
      "powerState":null,
      "vmState":null,
      "host":null,
      "instanceName":null,
      "hypervisorHostname":null,
      "diskConfig":"MANUAL",
      "availabilityZone":null,
      "uuid":null,
      "adminPass":"P8inbZ7LERa8"
   }
```

```
]
```

4.4.2.10 VMs

Informs DCA for VMs deployed or deleted by another component.

4.4.2.10.1 New VM Created

Informs DCA for the succefull deloyment of a new VM

POST /dca/vm

Request

```
curl --include \
  --request POST \
  --data-binary '{
 "name": "test-3",
 "imageRef": "90d4865d-5e7b-4d95-af2c-69753e1740d6",
 "flavorRef": "2",
 "keyName": "xifi",
 "securityGroups": [{"name": "default"}, {"name": "cep"}],
 "region": "RegionOne",
 "links": [
             {
                 "rel": "self",
                 "href": "http://cloud.lab.fi-ware.eu:8774/v2/00000000000000000000000000000101/servers/ffae6295-9aba-48a8-908d-4eecb71bd79c",
                 "type": null
             },
             {
                 "rel": "bookmark",
                 "href": "http://cloud.lab.fi-ware.eu:8774/00000000000000000000000000000101/servers/ffae6295-9aba-48a8-908d-4eecb71bd79c",
                 "type": null
             }
         ],
 "userID": "1234567890"
}' \
  http://dca.apiary-mock.com/dca/vm
```

Response

```
200 (OK)
Content-Type: application/json
[
{"imported":"ffae6295-9aba-48a8-908d-4eecb71bd79c"}
]
```

4.4.2.10.2 Delete a VM

Informs DCA that a server was deleted.

DELETE /dca/vm/{id}

Request

```
curl --include \
  --request DELETE \
```

© 2013 - 2015 XIFI Consortium Parties

http://dca.apiary-mock.com/dca/vm/6fda6be8-0e70-4d08-9731-c858d6a27f50

Response

```
200 (OK)
Content-Type: application/json
 { "deleted":"d5c22170-38d6-4344-9d62-c9770e550cee" }
```

4.4.3 PaaS Manager

The full PaaS Manager API documentation can be obtained from the corresponding FIWARE documentation at
http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/PaaS_Manager_-_User_and_Programmers_Guide#Developer_Guide

It consists of 5 basic primitives:

Get the environment list from the catalogue (GET /paasmanager/rest/catalog/environment)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <environmentDtoes>
      <environment>
          <environmentType>
             <id>24</id>
             <name>java_web_server</name>
             <description>An environment for Java applications
   deployed on a the web server</description>
          </environmentType>
          <name>java_tomat</name>
          <tiers>
             <initial_number_instances>1</initial_number_instances>
             <maximum_number_instances>1</maximum_number_instances>
             <minimum_number_instances>1</minimum_number_instances>
             <name>tomcat_tier</name>
             <productReleases>
                <product>tomcat</product>
                <version>7.0</version>
                <description>Tomcat 7.0</description>
                <productType>
                   <id>6</id>
                   <name>ApplicationWebServer</name>
                   <description>Application Web Server
   description</description>
                </productType>
             </productReleases>
          </tiers>
       </environment>
   </environmentDtoes>
```

Deploy an environment (POST /paasmanager/rest/org/{org-id}/vdc/{vdc-id}/environmentInstance/)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <task href="http:/130.206.80.112:8080/paasmanager/rest/org/{org-
   id}/vdc/{vdc-id}/task/{task-id}" startTime="2012-11-
   08T09:13:18.311+01:00" status="RUNNING">
      <description>Deploy environment {environment-
   name}</description>
      <vdc>{vdc-id}</vdc>
   </task>
```

Get information about the installed environment (GET /paasmanager/rest/org/{org-

id}/vdc/{vdc-id}/environmentInstance/{environmentInstance-id})

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <environmentInstanceDtoes>
      <environmentInstance>

   <environmentInstanceName>java_tomat_instance</environmentInstanceN
   ame>
        <vdc>{vdc-id}</vdc>
        <environment>
           <name>java_tomat</name>
           <tiers>
              <initial_number_instances>1</initial_number_instances>
              <maximum_number_instances>1</maximum_number_instances>
              <minimum_number_instances>1</minimum_number_instances>
              <name>tomcat_tier</name>
              <productReleases>
                 <product>tomcat</product>
                 <version>7.0</version>
                 <description>Tomcat 7.0</description>
                 <productType>
                    <id>6</id>
                    <name>ApplicationWebServer</name>
                    <description>Application Web Server
   description</description>
                 </productType>
              </productReleases>
           </tiers>
        </environment>
        <tierInstances>
            <id>138</id>
            <date>2012-11-29T13:03:36.056+01:00</date>
            <status>INSTALLED</status>
            <tier>
               <name>tomcat</name>
               <productReleases>
                  <product>test</product>
                  <version>0.1</version>
               </productReleases>
            </tier>
            <currentNumberInstances>1</currentNumberInstances>

   <fqn>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.re
   plicas.1</fqn>
            <name>java_tomat_instance_tomcat</name>
            <productInstances>
               <date>2012-11-29T13:03:27.834+01:00</date>

   <name>4caast.customers.test4.services.testtomcatsap9.vees.tomcat.r
   eplicas.1_tomcat_7.0</name>
               <status>INSTALLED</status>
               <productRelease>
                   <product>tomcat</product>
                   <version>7.0</version>
               </productRelease>
               <vm>
                  <hostname>tomcat</hostname>
                  <ip>130.206.80.114</ip>
               </vm>
            </productInstances>
```

© 2013 - 2015 XIFI Consortium Parties

```
         </tierInstance>
      </environmentInstance>
   </environmentInstanceDtoes>
```

Deploy an application in an environment already installed (POST /paasmanager/rest/org/{org-id}/vdc/{vdc-id}/environmentInstance/{environmentInstance-id}/applicationInstance)

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <task href="http:/130.206.80.112:8080/paasmanager/rest/org/{org-
   id}/vdc/{vdc-id}/task/{task-id}" startTime="2012-11-
   08T09:13:18.311+01:00" status="RUNNING">
      <description>Deploy application {application-
   name}</description>
      <vdc>{vdc-id}</vdc>
   </task>
```

Get information about an application already deployed (GET /paasmanager/rest/org/{org-id}/vdc/{vdc-id}/environmentInstance/{environmentInstance-id}/applicationInstance/{applicationInstance-id})

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <applicationInstances>
   <id>206</id>
      <date>2012-12-03T08:42:21.294+01:00</date>
      <name>warapplication_instance</name>
      <status>INSTALLED</status>    <vdc>{vdc-id}</vdc>
      <applicationName>warapplication</applicationName>
      <version>1.0</version>
      <artifacts>
        <artifact>
          <name>thewarfile</name>
          <attributes>
             <key>webapps_url</key>

   <value>http://Artefacts/WAR/tomcatFixedLocalPostgresDB/flipper.war
   </value>
           </attributes >
           <productRelease>
                <version>7.0</version>
                <product>tomcat</product>
             </productRelease>
          </artifact>
       </artifacts>
       <environmentInstance>

   <environmentInstanceName>java_tomat_instance</environmentInstanceN
   ame>
          <vdc>{vdc-id}</vdc>
          <environment>
             <name>java_tomat</name>
             <tiers>

   <initial_number_instances>1</initial_number_instances>

   <maximum_number_instances>1</maximum_number_instances>

   <minimum_number_instances>1</minimum_number_instances>
                <name>tomcat_tier</name>
                <productReleases>
                   <product>tomcat</product>
                   <version>7.0</version>
```

© 2013 - 2015 XIFI Consortium Parties

```
                    <description>Tomcat 7.0</description>
                    <productType>
                       <id>6</id>
                       <name>ApplicationWebServer</name>
                       <description>Application Web Server
   description</description>
                    </productType>
                 </productReleases>
              </tiers>
           </environment>
      </environmentInstance>
   </applicationInstances>
```

4.4.4 SDC TID

The full SDC API documentation can be obtained from the corresponding FIWARE documentation at http://forge.fi-ware.org/plugins/mediawiki/wiki/fiware/index.php/Software_Deployment_%26_Configuration_-_User_and_Programmers_Guide#Developer_Guide

It consists of 5 basic primitives:

Retrieve the product list from the catalogue (GET /sdc/rest/catalog/product):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <products>
     <product>
       <name>tomcat</name>
       <description>tomcat J2EE container</description>
     </product>
   </products>
```

Install a product in a VM (POST /sdc/rest/vdc/{VDC_ID}/product):

```
<productInstanceDto>
     <vm>
       <ip>{NODE_IP}</ip>
       <fqn>fqn</fqn>
     </vm>
     <product>
       <productDescription/>
       <productName>tomcat</productName>
       <version>6</version>
    </product>
   </productInstanceDto>
```

Retrieve information about the product installed (GET /sdc/rest/vdc/{VDC_ID}/product/{PRODUCT_ID}):

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
   <productInstance>
     <id>{PRODUCT_ID}</id>
     <date>2012-11-07T15:56:30.590+01:00</date>
     <status>INSTALLED</status>
     <vm>
       <ip>130.206.80.114</ip>
       <hostname>rhel-5200ee66c6</hostname>
       <fqn>fqn</fqn>
       <osType>95</osType>
     </vm>
     <vdc>{vdc-id}</vdc>
```

© 2013 - 2015 XIFI Consortium Parties

```
      <product>
        <releaseNotes>Tomcat server 6</releaseNotes>
        <version>6</version>
        <product>
          <name>tomcat</name>
          <description>tomcat J2EE container</description>
          <attributes>
             <key>port</key>
             <value>8080</value>
           </attributes>
        </product>
       </product>
    </productInstance>
```

Update product version in the VM (PUT /sdc/rest/catalog/product/{PRODUCT_ID}/{VERS})

Reconfigure a product already installed (PUT /sdc/rest/catalog/product/{PRODUCT_ID})

```
<attributes>
     <key>port</key>
     <value>8082</value>
     <description>The listen port</description>
   </attributes>
```

© 2013 - 2015 XIFI Consortium Parties

5 SUB-SYSTEM INTERNAL APIS

The specification of sub-system internal APIs is an ongoing task. It strongly correlates with the definition of sub-system internal test procedures.

All components described in the scope of this document provide their particular component functional description, installation and configuration instructions and dedicated user manuals and have been properly referenced. Those component descriptions additionally define and document their particular APIs, which defines the super-set of sub-system internal APIs.

So far, there has no particular case identified where a distinct description of sub-system APIs would be required. But it is recognized that in course of federation operations and maintenance it might be needed to reflect a particular use of component APIs in scope of a sub-system would enable specific "maintenance" manuals, covering for example distinct deployment and test cases that may better support (i.e. more focused) the needs of node and federation maintainers. The notion of "sub-system internal APIs" will then form a comfortable basis for "best-practice" manuals, as already suggested in D2.2 [1].

© 2013 - 2015 XIFI Consortium Parties

6 CONCLUSIONS

This document grounds on the work done for deliverable D2.2 [1] and elaborates further on the APIs and Tools for Infrastructure federation. It newly introduced the notion of a sub-system adopted from D2.3 [2] where it was used in the scope of implementing the federation to describe independently testable and deployable software sub-systems each consisting of a collection software components. It aligns in this paradigm with D5.3 [6] that uses the concept to describe independently maintainable software sub-systems. Although introducing a further level of abstraction, the notion of sub-systems has proven convenient in practice since it allows to focus on the use of a "bundle" of components for a particular use and its interfaces relevant for exactly that use, omitting any details or capacities that are not of relevance for implementing a particular purpose – which is the implementation of the federation layer in scope of this document.

It has to be emphasized that this document is by intention not self-contained (and cannot be self-contained) since it has to reference a multitude of source documents that provide a more detailed documentation on components jointly utilized to the XIFI federation layer. Whenever suitably supporting its readability and clarity, this document provided a short self-contained summary of a referenced source in addition.

Although this document provides the concluding documentation for APIs and Tools for Infrastructure federation, it does not mark the final point of development. There will be more development effort required, for example in the course of enhancing the federation capacity in terms of functionality and scalability, or in making it sufficiently robust for a productive operational phase. But this development will take place on the level of single components or sub-systems and will not affect concepts and architecture as outlined here.

Practical experience with the current implementation of the XIFI federation layer has been partly addressed in scope of D1.5 [19]. It shows that major concerns of the infrastructure nodes can and have been addressed by achievements performed since the submission of D2.2. A further review can be expected from the upcoming D5.5 when elaborating on the operations and maintenance of the federation. Until then the federation layer will undergo constant revision and verification and remaining weaknesses will be addressed:

- So far, non-conventional resources have not been addressed in the scope of tools and APIs for infrastructure federation. More attention must be paid in due course of implementing node-local services, in particular regarding access control, usage policies, monitoring and SLA support. Operational level agreements and non-conventional services monitoring are current working items.
- High availability has not been addressed so far for the federation layer in whole but only for few components of the federation sub-systems. Since the XIFI federation strongly depends on its master node and some centralized components, it is assumed that increasing availability of those central components will also increase availability of the federation in whole. In fact this assumption still has to be verified.

Concluding, this document is the final documentation of the federation layer's APIs and tools but it is not the final word regarding the development of the federation layer. Further evaluation, verification and refinement will take place in scope of federation operations and maintenance, which strongly depends on the feedback on experience and requirements obtained from the individual infrastructure nodes.

© 2013 - 2015 XIFI Consortium Parties

REFERENCES

[1] THE XIFI CONSORTIUM. (2014). *Deliverable D2.2: APIs and Tools for Infrastructure Federation*. [Online] Available from: https://bscw.fi-xifi.eu/pub/bscw.cgi/d59218/XIFI-D2.2-APIs_and_Tools_for_Infrastructure_Federation_v1.pdf [Accessed: 12th September 2014]

[2] THE XIFI CONSORTIUM. (2014). *Deliverable D2.3: Test and Validation Report v1*. Available from: (URL missing) [Accessed: 12th September 2014]

[3] THE XIFI CONSORTIUM. (2014). *XIFI Components*. Available from: http://wiki.fi-xifi.eu/XiFi:Components [Accessed: 12th September 2014]

[4] THE XIFI CONSORTIUM. (2014). *XIFI Sub-systems*. Available from: http://wiki.fi-xifi.eu/Xifi:Subsystems#Definition_of_software_sub-systems_based_on_the_component_definitions [Accessed: 12th September 2014]

[5] THE XIFI CONSORTIUM. (2014). *Deliverable D3.2: Infrastructures monitoring and interoperability adaptation components toolkit and API*. [Online] Available from: https://bscw.fi-xifi.eu/pub/bscw.cgi/d58608/XIFI-D3.2-Infrastructures_monitoring_and_interoperability_adaptation_components_toolkit_and_API.pdf [Accessed: 12th September 2014]

[6] THE XIFI CONSORTIUM. (2014). *Deliverable D5.3: 1st Report on XIFI nodes operation, maintenance, assistance and procedures updates*. [Online] Available from: (URL missing) [Accessed: 12th September 2014]

[7] THE APACHE SOFTWARE FOUNDATION (2014). *Apache Maven Project*. Available from: http://maven.apache.org/index.html [Accessed: 12th September 2014]

[8] For the Shibboleth Official documentation refer to: https://shibboleth.net

[9] For the OpenLDAP Bycript : https://github.com/dnwright/openldap-bcrypt

[10] For Identity Management's Devise to interact with the SP:https://github.com/plataformatec/devise

[11] For authentication module the sub module omniauth-shibboleth to interact with the SP and use of omniauth-shibboleth gem to implement the omniauth shibboleth strategy:https://github.com/toyokazu/omniauth-shibboleth

[12] https://www.phusionpassenger.com/

[13] For the authentication module:https://github.com/intridea/omniauth

[14] For Brycpt functionalities: http://bcrypt.sourceforge.net/

[15] For OpenlDap documentationhttp://www.openldap.org/

[16] For the Tomcat Official documentation:http://tomcat.apache.org/

[17] For Rubyonrails documentation:http://guides.rubyonrails.org/v3.2.13/migrations.html

[18] For the Shibboleth documentation:https://wiki.shibboleth.net/confluence/display/SHIB2/Metadata

[19] THE XIFI CONSORTIUM. (2014). *Deliverable D1.5: Federated Platform Architecture Version 2*. [Online] Available from: (URL missing) [Accessed: 17th September 2014]

[20] THE XIFI CONSORTIUM. (2014). *Deliverable D2.4: XIFI Handbook v2*. [Online] Available from: (URL missing) [Accessed: 17th September 2014]

[21] THE XIFI CONSORTIUM. (2014). *Deliverable D3.5: Infrastructures monitoring and interoperability adaptation components API v2*. [Online] Available from October 2014: (URL will be provided)

[22] THE XIFI CONSORTIUM. (2014). *Deliverable D3.2: Infrastructures monitoring and*

© 2013 - 2015 XIFI Consortium Parties

interoperability adaptation components toolkit and API. [Online] Available from:https://bscw.fi-xifi.eu/pub/bscw.cgi/d58608/XIFI-D3.2-Infrastructures_monitoring_and_interoperability_adaptation_components_toolkit_and_API.pdf

[23] THE XIFI CONSORTIUM. (2014). *Deliverable D4.1: Services and tools specification.* [Online] Available from: https://bscw.fi-xifi.eu/pub/bscw.cgi/d44635/XIFI-D4.1-Services_and_tools_specification.pdf [Accessed: 17th September 2014]

[24] THE XIFI CONSORTIUM. (2014). *Deliverable D4.2: Baseline Tools v1*. [Online] Available from: https://bscw.fi-xifi.eu/pub/bscw.cgi/d58659/XIFI-D4.2-Baseline_Tools_v1.pdf [Accessed: 17th September 2014]

[25] THE XIFI CONSORTIUM. (2013). *Deliverable D1.1b: XIFI core concepts, requirements and architecture draft.* [Online] Available from: https://bscw.fi-xifi.eu/pub/bscw.cgi/d44695/XIFI-D1.1b-XIFI_core_concepts_requirements_and_architecture_draft.pdf [Accessed: 17th September 2014]

[26] THE XIFI CONSORTIUM. (2014). *Deliverable D4.4: Baseline Tools v2.* [Online] Available from October 2014: (URL will be provided)

[27] Social Stream Ruby Gem: https://github.com/ging/social_stream

[28] Csync2 documentation: http://www.openldap.org/

[29] Public link for Security Monitoring: http://wiki.fi-xifi.eu/Public:Security_Monitoring_GE

[30] Web Services Agreement Specification (WS-Agreement): http://ogf.org/documents/GFD.192.pdf

[31] Public link for Federation Manager Documentation: http://wiki.fi-xifi.eu/Public:Federation_Manager

[32] http://guides.rubyonrails.org/v3.2.13/migrations.html

© 2013 - 2015 XIFI Consortium Parties

Appendix A Update on the Identity Management

A.1 TOP level architecture for web browser 2.0 support

Figure 21 shows the top level view of a SAML 2.0 authentication flow for a Web browser based scenario.

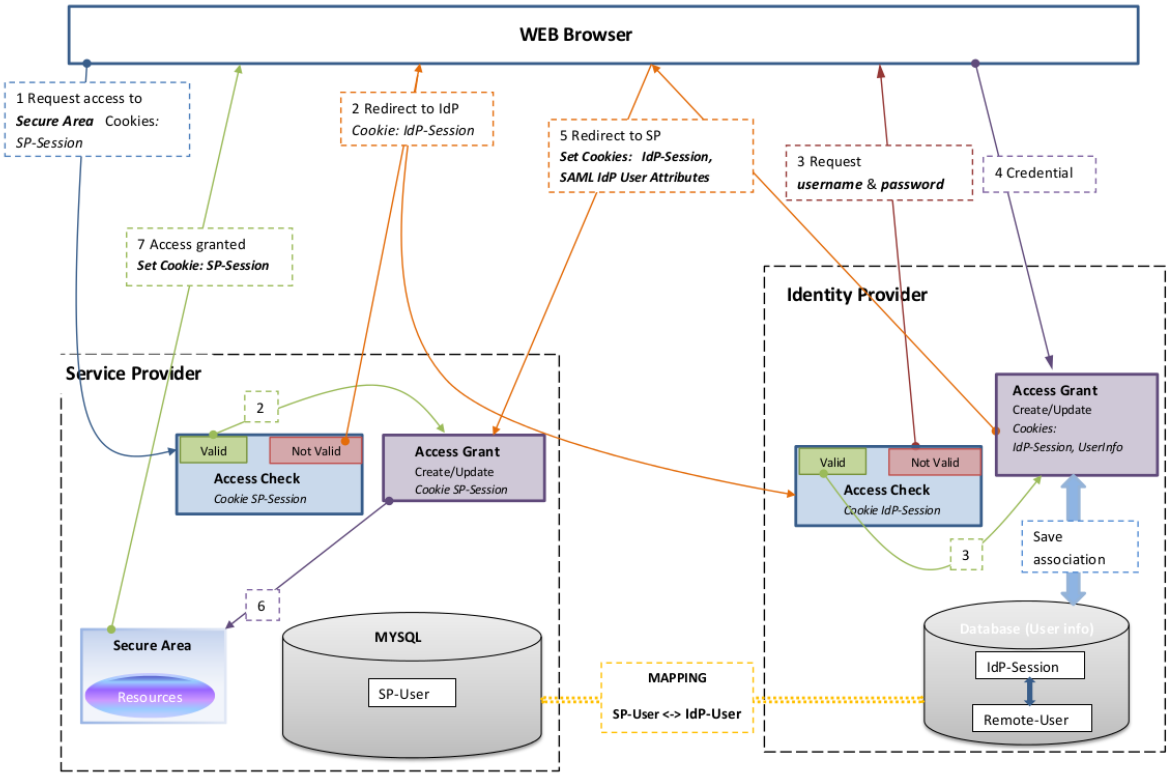


Figure 21: Web Browser SAML 2.0 Architecture.

Particularly the depicted scenario has the following features relevant for the XIFI IDM scenario:

1. The Service Provider does maintain (local) session information (SP-Session in Figure 21).
2. The protected resources do have a relation with the service user (in other words the SP-Session is linked to a given user).
    a. The user information is also kept locally accessible to the Service Provider.
        i. The information does not (necessarily) include the authentication credentials.
        ii. This also means that user entries in the IdP database have a profiling "local" to the SP and the information present in both databases has to be correlated.
3. Once the user has been authenticated by the IdP (using his credentials), then user attributes (e.g. mail address) have to be sent back to the SP in order to establish a link with the SP-Session (see above point b.). This information is protected inside the SAML assertions in order to protect the user privacy.

To support the flow depicted in Figure 21 the IDM is extended to provide to provide SAML 2.0 support as both the Identity Provider and the Service Provider.

The basic extension to the architecture is found in Figure 22 and makes use of Shibboleth [8] that is the most widely adopted Open Source solution for Identity Federation compliant with the SAML 2.0

© 2013 - 2015 XIFI Consortium Parties

specification.

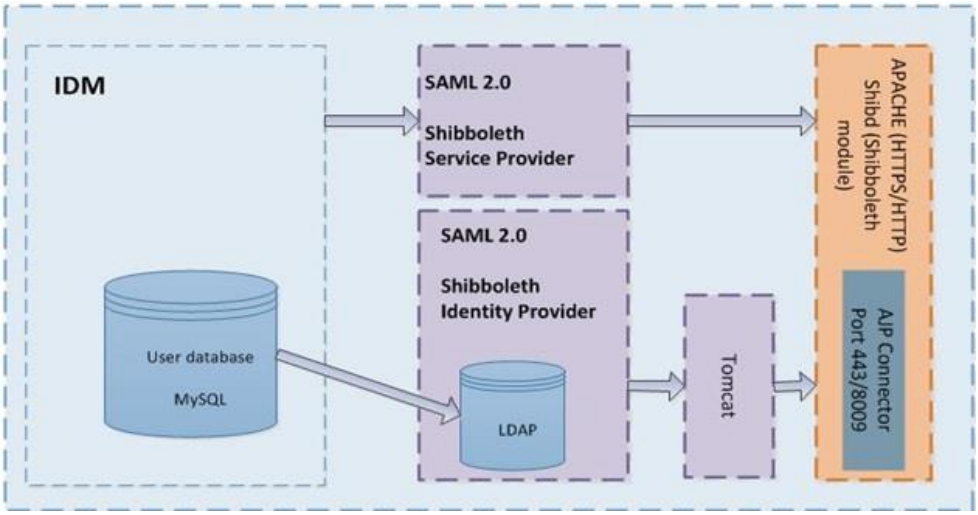


Figure 22: Extended IDM basic architecture

Details of the architecture for SAML SP and SAML IdP extensions are then provided in following section.

A.2 IDM integration with SAML SP

This section describe how to modify the IDM to integrate the Shibboleth Service Provider module

A.2.1 IDM Architecture to support SAML SP

Figure 23 shows how the IDM can be extended to act also as a SAML 2.0 SP.

The main necessary modifications are the following:

1. Install the Shibboleth Service Provider (SP) in the IDM's machine.
2. Modify the Apache config file to add a new route for the SP
3. Add to the IDM's Devise authentication module [10] the submodule omniauth-shibboleth [11] to interact with the SP.
4. Modify IDM's database to add the external users references.
5. Modify IDM's web interface with the external IDP reference (one or more).

External user profiling local to SP As mentioned above a profiling of external users (IdP users) locally to the SP (IDM) is necessary. As a consequence also a correlation mean between the two sources of information for the same user has to be present.

The following constrains apply:

1. external users have the same permissions as the local users;
2. external users are persistent in the SP (IDM) database.

Below two possible solutions for the user local profiling are provided and at the end of this analysis a comparison is done leading to select one solution:

1. extension of external mail address field;
2. keeping same mail address field for external users.

Both solutions are clarified with the help of the following three case examples.

© 2013 - 2015 XIFI Consortium Parties

1. First login of an external user;
2. First login with external users with same e-mail of a local user;
3. Local user registration with email equal to the one of an external user.

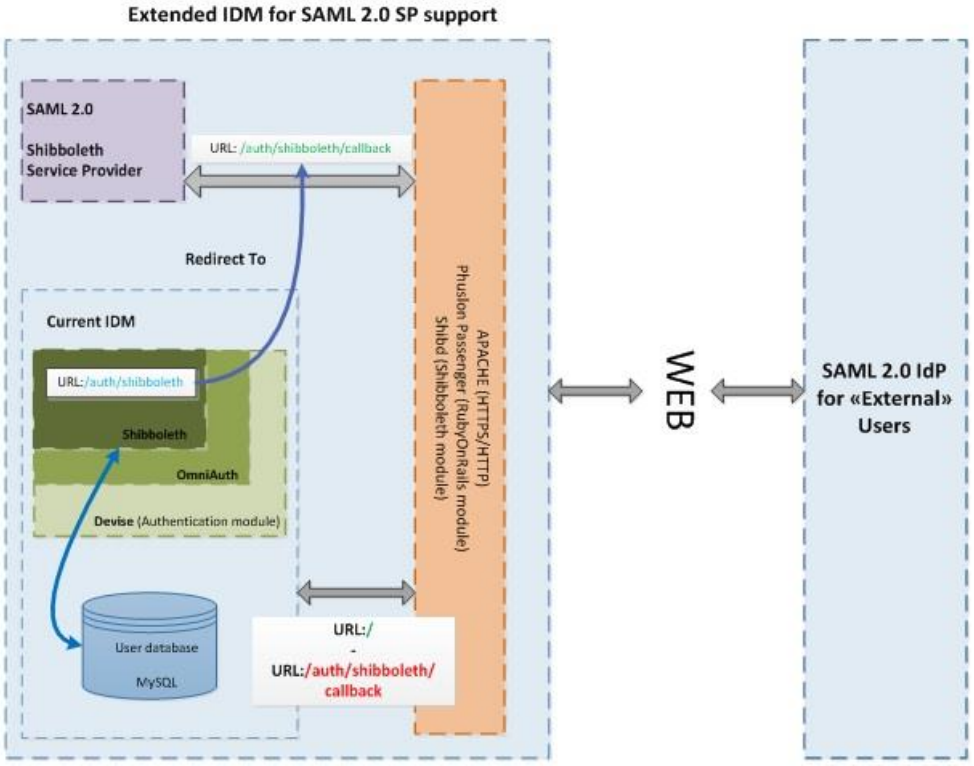


Figure 23: Extended IDM for SAML SP

Solution 1: extension of external mail address field

- Database schema

  Figure 24 shows how the external users are discriminated from the local users thanks to the idp_signature. In this solution the original mail (of the external user) is changed by attaching the idp_signature as suffix. E.g. the mail test@test.com come from the external IDP idp.test.com (with signature TEST) become test@test.com_TEST.

- Case example 1: First login of an external user

  A user tries to login through an external IDP (identified by IDPSAML signature) for the first time.

  The user is correctly registered in the external IDP service and, as shown in Figure 25, is redirected for authentication to the IDP. Inside the IDM no information is stored yet for this user, indeed no login has been already attempted yet.

  After a successful authentication the external user's mail john.doe@unknown.org is transformed in john.doe@unknown.org_IDPSAML into the IdM (cf. Figure 26) and is saved into the database.

- Case Example 2: First login with external users with same e-mail of a local user

  In this case example we have a first login with a local user's mail equal to an external user's mail.

  Before attempting the first login the mail of the user sam.gamgee@lotr.org is already registered locally as shown in Figure 27.

© 2013 - 2015 XIFI Consortium Parties

After a successful authentication the external user's mail sam.gamgee@lotr.org will be transformed in sam.gamgee@lotr.org_IDPSAML (cf. Figure 28) in order not to collide with the local mail entry.

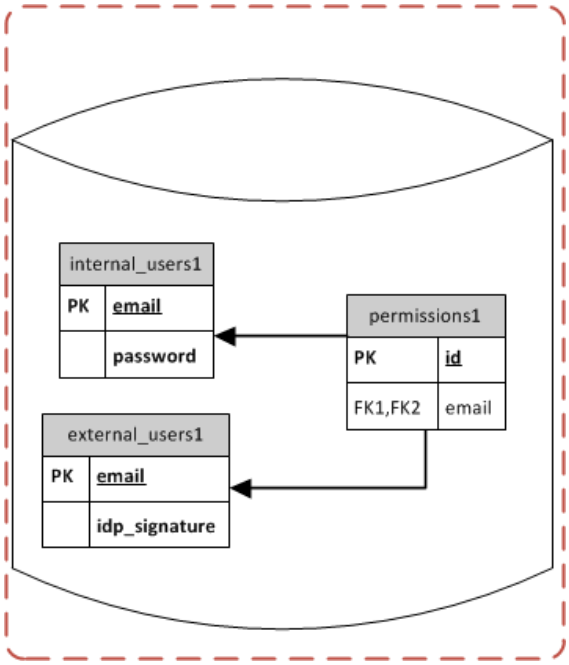


Figure 24: Database schema (solution 1)

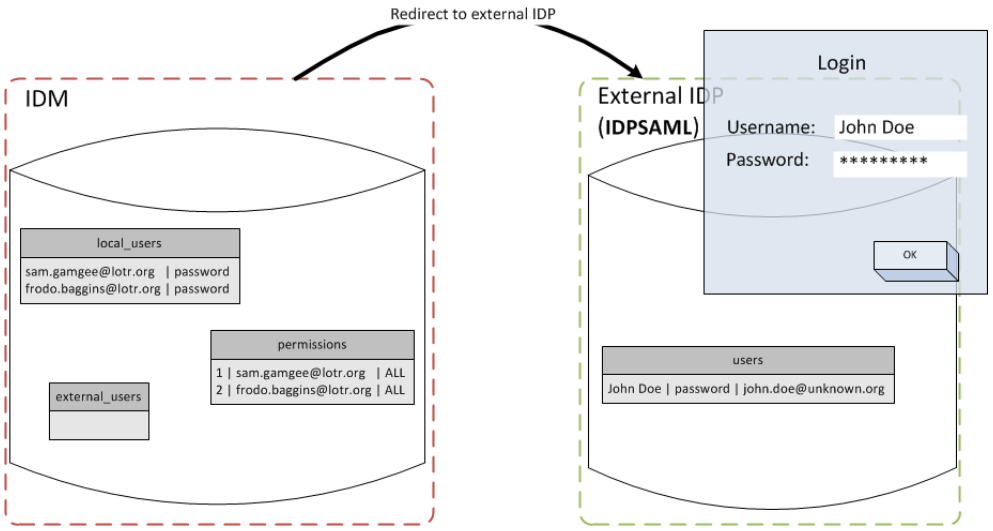


Figure 25: First login of an external user – Part 1 (solution 1, case1)

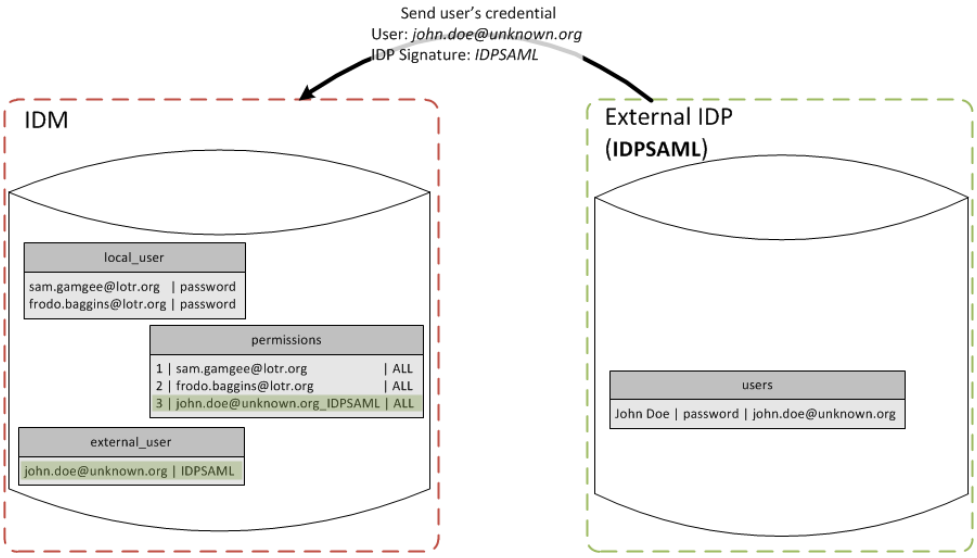


Figure 26: First login of an external user – Part 2 (solution 1, case1)

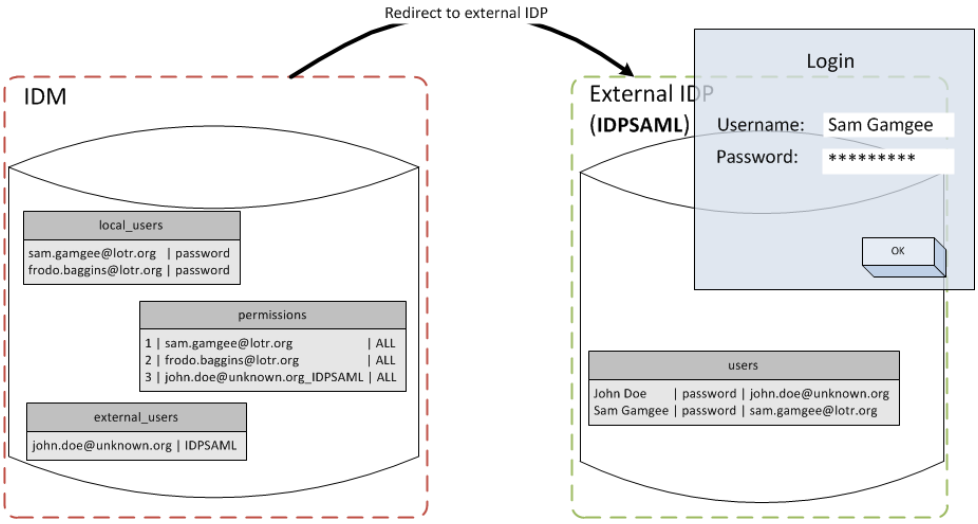


Figure 27: External users with same e-mail of a local user - Part 1 (solution 1, case2)

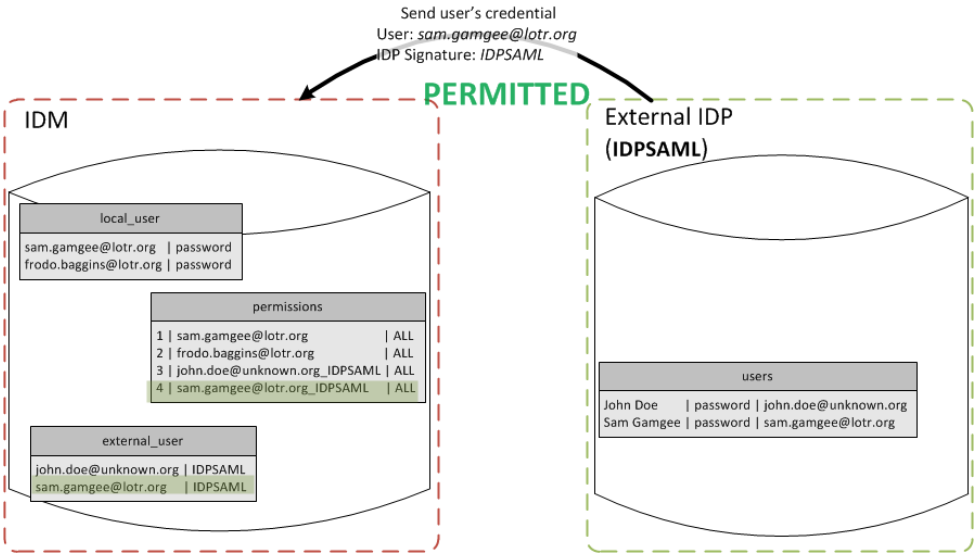


Figure 28: External users with same e-mail of a local user - Part 2 (solution 1, case2)

© 2013 - 2015 XIFI Consortium Parties

- Case Example 3: Local user registration with email equal to the one of an external user

  This example analyzes the opposite scenario with respect to the one described in Case example 2.

  Now we have a local user that tries to sign up with an email equal to the one of an external user that has been previously authenticated, and then that has a local profile (cf. Figure 29).

  As in the previous case the mail will not collide because the external user's mail was transformed as john.doe@unknown.org_IDPSAML.

Solution 2: keeping same mail address field for external users

- Database schema

  Figure 30 shows how the external users are discriminated among the local users by the idp_signature. In this solution the original mail (of the external user) is not modified by attaching the idp_signature as suffix. In this solution the external users is identified by his email plus the idp signature. The internal user has the idp signature field equal to NULL.

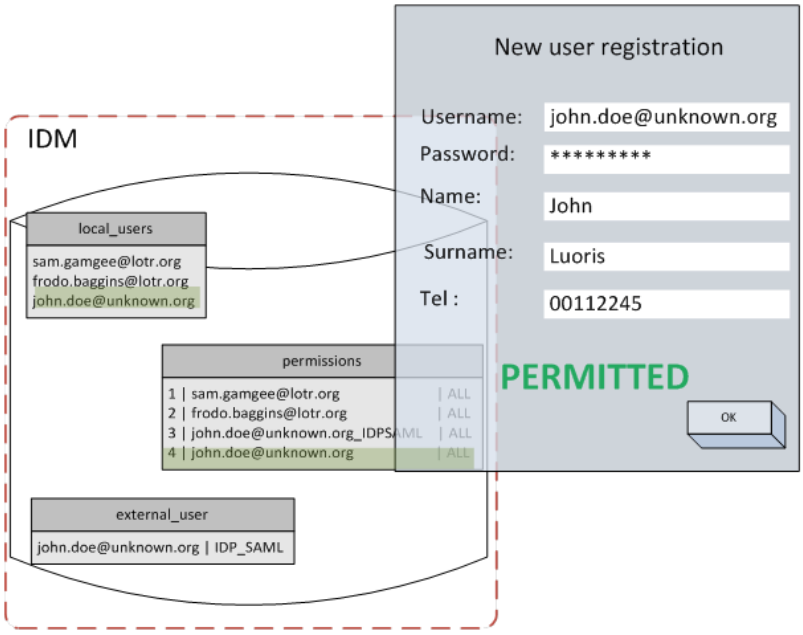


Figure 29: Local registration with same e-mail of external user (solution 1, case 3)

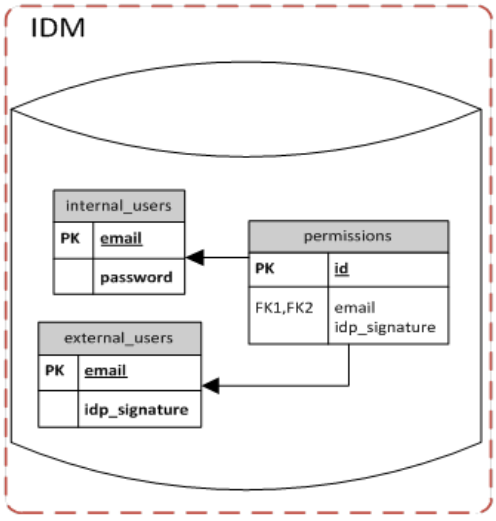


Figure 30: Database schema (solution 2)

- Case example 1: First login of an external user

  A user tries to login through an external IDP (identified by IDPSAML signature) for the first time. The user is correctly registered in the external IDP service and, as shown in Figure 31, is redirected for authentication to the IDP. Inside the IDM no information is stored yet for this user, indeed no login has been already attempted yet.

  After a successful authentication the external user's mail john.doe@unknown.org is kept without modifications and the external user is identified by a combination of his mail plus an IDP tag (cf. Figure 32)

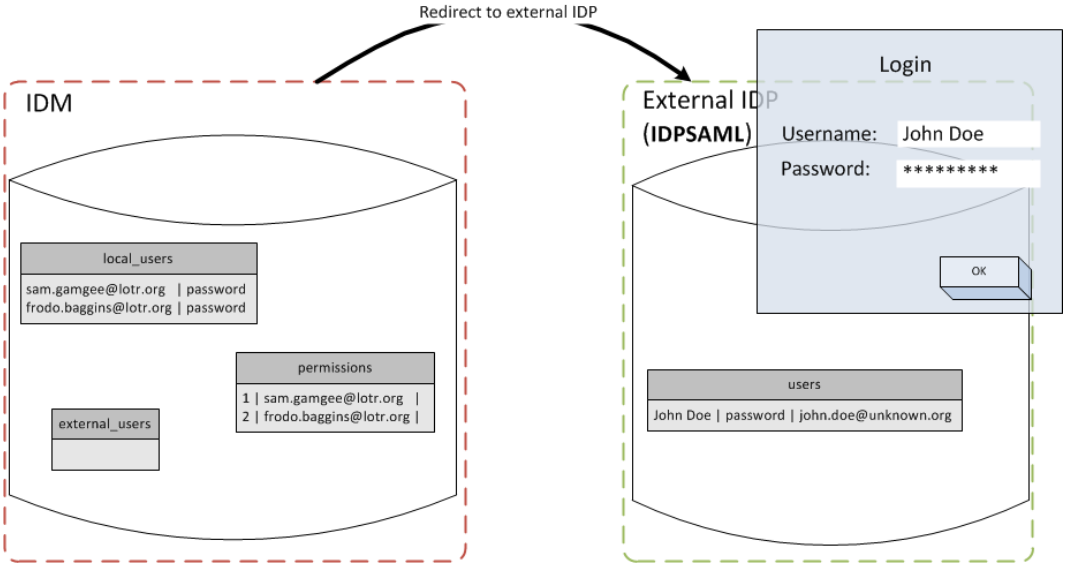


Figure 31: First login of an external user – Part 1 (solution 2, case 1).

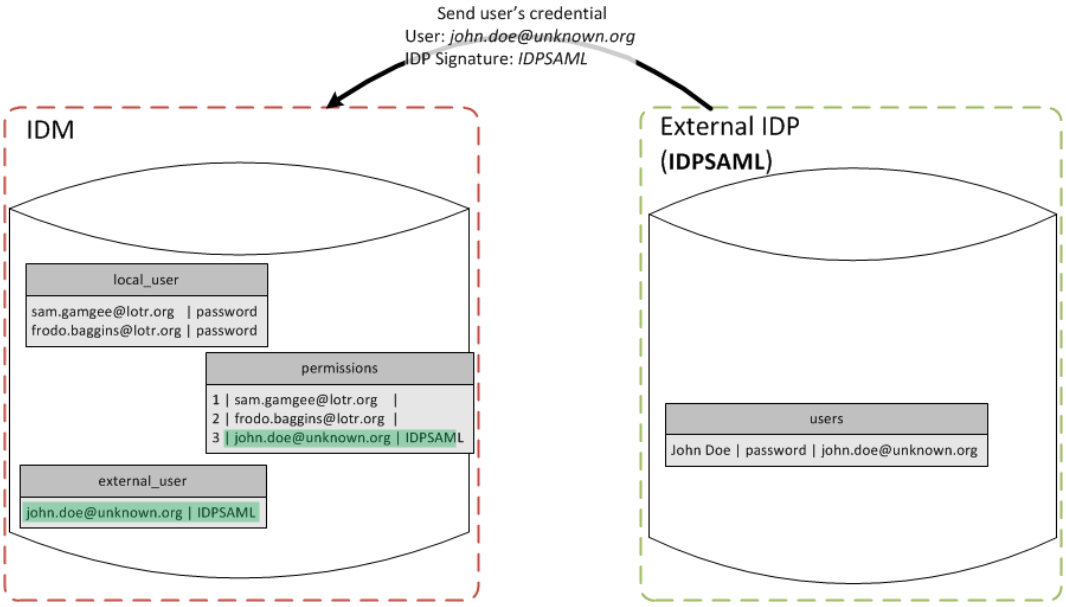


Figure 32: First login of an external user – Part 2 (solution 2, case 1).

- Case example 2: First login with external users with same e-mail of a local user

  In this case example we have a first login with local user's mail equal to an external user's mail.

  Before attempting the first login the mail of the user sam.gamgee@lotr.org is already

© 2013 - 2015 XIFI Consortium Parties

registered locally as shown in Figure 33.

After a successful authentication the external user's mail sam.gamgee@lotr.org is kept without modifications but the external user will not collide with the local user because in addition the IDP tag is also associated to his profile (cf. Figure 34).

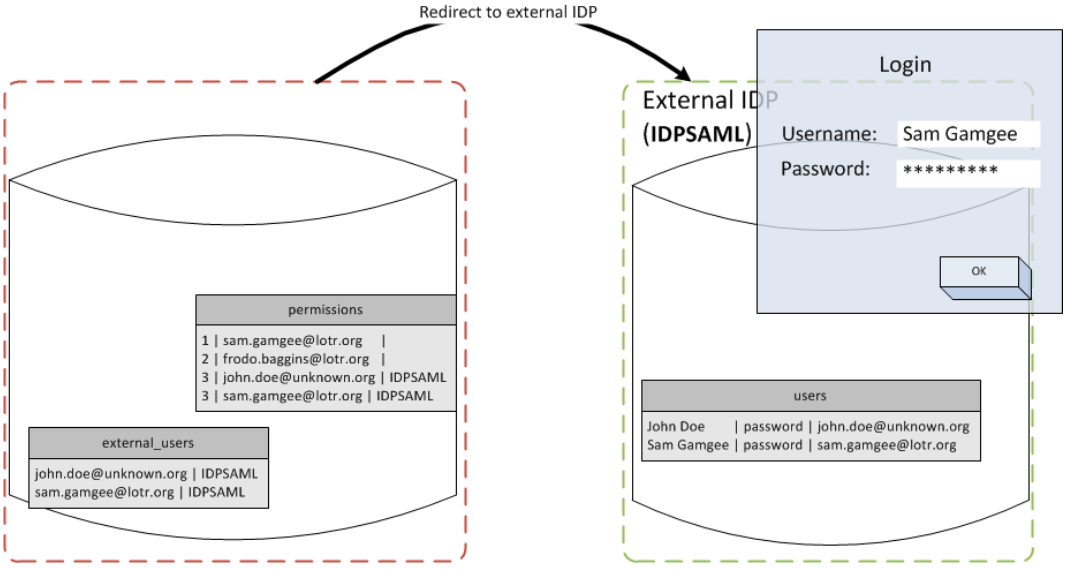


Figure 33: External users with same e-mail of a local user - Part 1 (solution 2, case 2).

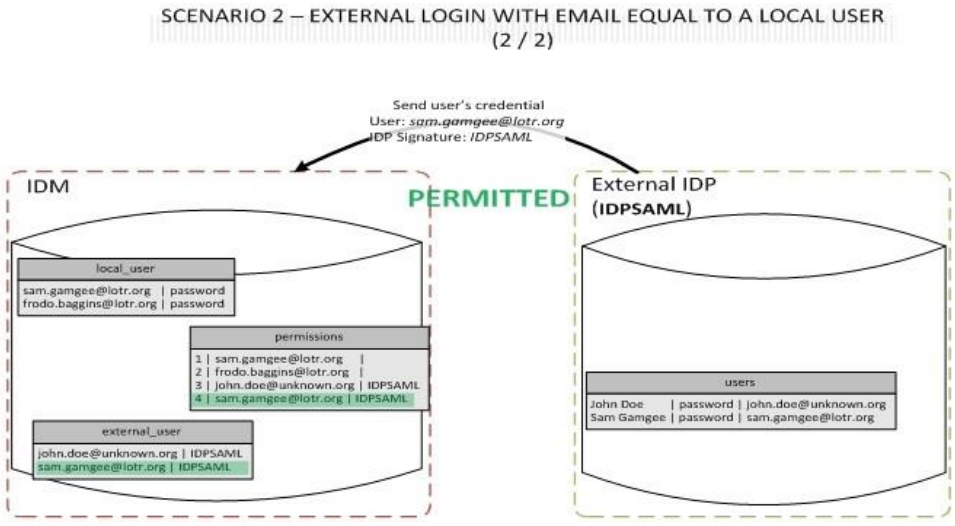


Figure 34: External users with same e-mail of a local user - Part 2 (solution 2, case 2)

- Case example 3: Local user registration with email equal to the one of an external user

  This example analyzes the opposite scenario with respect to the one described in Case example 2. Now we have a local user that tries to sign up with an email equal to the one of an external user that has been previously authenticated, and then that has a local profile.

  As in the previous case the local user john.doe@unknown.org doesn't collide with the external user because the local user doesn't have an IDP tag (cf. Figure 35).

- Conclusion - Comparison and solution selection

  Saying that there are no functional differences between the two proposed solutions choice has fallen on solution 2 described in previous above.

  Reason is mainly is to minimize integration issue, indeed applying solution 2 the IdM logic

© 2013 - 2015 XIFI Consortium Parties

can use the mail address field of "external" users similarly to the one of usual users without any constrain being appliedAs an example consider the case when sending mail to the user is necessary: with this solution no other modifications to this mail transmission logic is necessary.

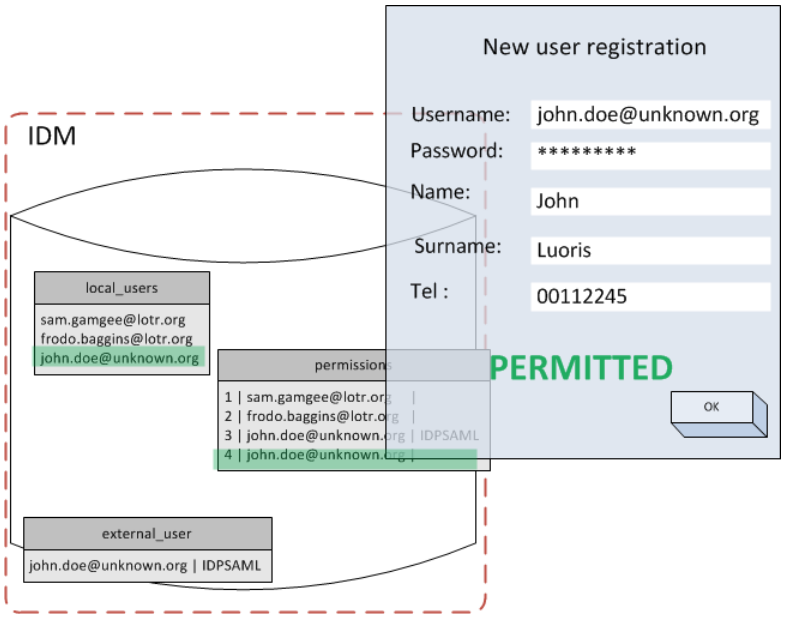


Figure 35: Registration with same e-mail of external user (solution 2, case 3).

- Support of multiple IdPs

  Figure 36 shows how the original architecture of Figure 23 can be extended to support multiple IDPs.

  The IDM will able to manage different routes towards different external IDP.

  The IDM doesn't know the real IDP route but has a local 'url' that identifies the IDP (/auth/shibboleth in the following figure).

  The routes (local and real) are mapped inside the Apache configuration files (e.g. /auth/shibboleth -> https://shibboleth_idp1).

  This mapping is updated dynamically by a dedicated page inside the IDM.

© 2013 - 2015 XIFI Consortium Parties

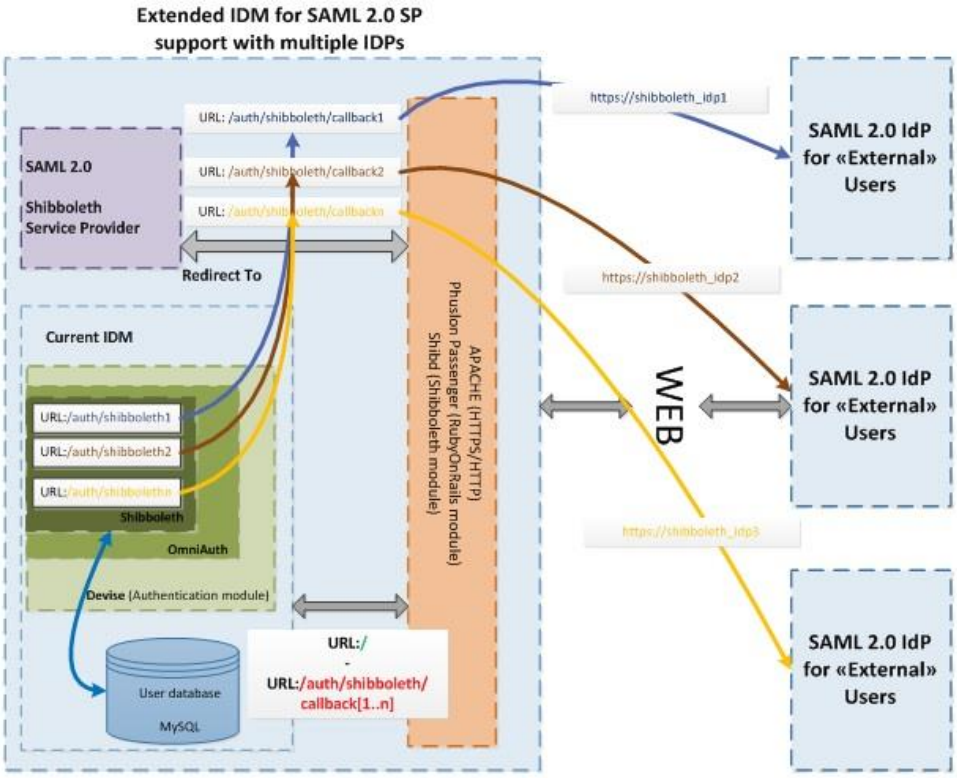


Figure 36: Support of multiple IdPs

A.2.2 IDM Update to support SAML SP

This section reports the necessary changes to be made in order for the IDM to act as a SAML SP.

Particularly the changes are grouped and described as follows:

- Shibboleth SP: installation and configuration
- Apache configuration
- Update done to KeyRock
- Update for the user interface

A.2.2.1 Shibboleth SP installation

- Install the packages

```
sudo apt-get install apache2 libapache2-mod-shib2 openssl php5 ntp
```

After this operation the shibboleth provider will be installed into the /etc/shibboleth directory

A.2.2.2 Shibboleth SP configuration

1. Modify the file /etc/hosts

```
127.0.0.1   <keyrock machine's hostname>
```

2. Get the GARR-CA.pem certificate

```
wget https://ca.garr.it/mgt/CAcert.pem -O /etc/shibboleth/GARR-CA.pem
```

verify the authenticity comparing the following values with those present to this url https://ca.garr.it/mgt/getCA.php

```
openssl x509 -in GARR-CA.pem -fingerprint -sha1 –noout
openssl x509 -in GARR-CA.pem -fingerprint -md5 -noout
```

© 2013 - 2015 XIFI Consortium Parties

modify the GARR-CA.pem permissions

```
chmod 444 /etc/shibboleth/GARR-CA.pem
```

3. Create one certificate and one key (both self-signed) with the following command

```
/usr/sbin/shib-keygen
```

Will be created the files /etc/shibboleth/sp-key.pem e /etc/shibboleth/sp-cert.pem

Create the folder /etc/shibboleth/cert-from-CA :

```
mkdir /etc/shibboleth/cert-from-CA
```

and move in /etc/shibboleth/cert.from-CA folder both certificate and key (renaming them)

```
mv /etc/shibboleth/sp-key.pem /etc/shibboleth/cert.from-CA/ssl-
   key.pem
mv /etc/shibboleth/sp-cert.pem /etc/shibboleth/cert.from-CA/ssl-
   cert.pem
```

4. Download the TERENA-chain.pem file and put into the /etc/shibboleth/cert-from-CA folder

```
cd /etc/shibboleth/cert-from-CA
wget https://ca.garr.it/mgt/Terena-chain.pem
```

5. Modify the /etc/shibboleth folder permissions to permit to the "_shibd" user to write in it:

```
chown -R _shibd /etc/shibboleth/
```

6. Make the following links

```
ln -sf /etc/shibboleth/cert.from-CA/ssl-key.pem /etc/shibboleth/sp-
   key.pem
ln -sf /etc/shibboleth/cert.from-CA/ssl-cert.pem /etc/shibboleth/sp-
   cert.pem
```

7. Create the folder /etc/shibboleth/metadata where collect the IDP's metadata
8. Modify the /etc/shibboleth/shibboleth2.xml file
    - change the ApplicationDefaults tag

```
<ApplicationDefaults entityID="https://spd.example.org/shibboleth"
   REMOTE_USER="eppn persistent-id targeted-id">
```

as follows

```
<ApplicationDefaults entityID="<keyrock machine's
   hostname>/shibboleth" REMOTE_USER="eppn persistent-id targeted-
   id">
```

    - Modifiy the SSO tag

```
<SSO entityID="https://idp.example.org/shibboleth" discoveryProtocol="SAMLDS"
       discoveryURL="https://ds.example.org/DS/WAYF">
SAML2 SAML1
</SSO>
```

as follows

```
<SSO>
SAML2
</SSO>
```

© 2013 - 2015 XIFI Consortium Parties

- Modify the Errors tag as follows

```
<Errors supportContact="<email@support.com>"
logoLocation="/usr/share/shibboleth/logo.jpg"
styleSheet="/usr/share/shibboleth/main.css"/>
```

- Modify the MetadataProvider tag to include the IDP's metadata (two in this example)

```
<MetadataProvider type="Chaining">
        <MetadataProvider type="XML"
   path="/etc/shibboleth/metadata/idp1-metadata.xml"/>
        <MetadataProvider type="XML"
   path="/etc/shibboleth/metadata/idp2-metadata.xml"/>
</MetadataProvider>
```

9. Modify the /etc/shibboleth/attribute-map.xml file to enable the identification of the attributes 'mail' and 'cn' required from the KeyRock for the external users authentication.

- Uncomment these lines

```
<Attribute name="urn:mace:dir:attribute-def:cn" id="cn"/>
<Attribute name="urn:mace:dir:attribute-def:mail" id="mail"/>
<Attribute name="urn:oid:2.5.4.3" id="cn"/>
<Attribute name="urn:oid:0.9.2342.19200300.100.1.3" id="mail"/>
```

A.2.2.3 Apache configuration

1. Modify the /etc/apache2/sites-available/default-ssl file (the following modifies have to apply inside the tag <VirtualHost _default_:443>

- Take off the url /Shibboleth.sso from the Phusion Passenger control

```
<Location /Shibboleth.sso>
    PassengerEnabled off
     Options +Indexes
</Location>
```

- add the following code as many as the differents IDP.

```
<Location /users/auth/shibboleth_idp[1..n]/callback>
    Options -MultiViews
              AuthType shibboleth
```

ShibRequestSetting entityID <url IDPn>/idp/shibboleth ShibRequestSetting requireSession On require valid-user

```
</Location>
```

- under the line "SSLEngine on" add:

```
SSLProtocol all -SSLv2
SSLCipherSuite ALL:!aNULL:!ADH:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:!MEDIUM
```

and

```
SSLCertificateFile /etc/ssl/certs/ssl-cert-snakeoil.pem
SSLCertificateKeyFile /etc/ssl/private/ssl-cert-snakeoil.key
#SSLCertificateChainFile /etc/apache2/ssl.crt/server-ca.crt
```

become

```
SSLCertificateFile /etc/shibboleth/cert-from-CA/ssl-cert.pem
SSLCertificateKeyFile /etc/shibboleth/cert-from-CA/ssl-key.pem
SSLCertificateChainFile /etc/shibboleth/cert-from-CA/Terena-chain.pem
```

© 2013 - 2015 XIFI Consortium Parties

2. Enable the shibboleth apache modules

```
a2enmod shib2
```

3. Restart the services

```
service apache2 restart
service shibd restart
```

A.2.2.4 KeyRock updates

To enable the SAML support, the IDM has to be modified in three virtual "sections":

- the controller section,
- the database section,
- the layout section.

Controller changes

KeyRock, for the authentication procedure, uses the devise gem [6] and devise uses the omniauth gem [11].

For omniauth there are several gems to cover a lot of different authentication strategies. In particular we have used the omniauth-shibboleth [11] gem that implement the omniauth shibboleth strategy .

The steps to add omniauth-shibboleth to KeyRock are the following:

1. Add the omniauth-shibboleth gem to the Gemfile

```
gem 'omniauth-shibboleth'
```

and install it with command

```
bundle install
```

2. Add the follow omniauth.rb configuration file in the <project home>/config/initialize folder

```
# initializers/omniauth.rb
module OmniAuth::Strategies
  class ShibbolethIdp1 < Shibboleth
    def name
      :shibboleth_idp1
    end
  end
  class ShibbolethIdp2 < Shibboleth
    def name
      :shibboleth_idp2
    end
  end
  class ShibbolethIdp3 < Shibboleth
    def name
      :shibboleth_idp3
    end
  end
  class ShibbolethIdp4 < Shibboleth
    def name
      :shibboleth_idp4
    end
  end
end
```

in the above configuration we have provided four different class that correspond to four different IDP.

© 2013 - 2015 XIFI Consortium Parties

3. Add the following lines at the end of the devise.rb file in the <project home>/config/initialize folder

```
config.omniauth :shibboleth_idp1, {:info_fields => {:email =>
  'mail', :name => 'cn'},:debug => false }
config.omniauth :shibboleth_idp2, {:info_fields => {:email =>
  'mail', :name => 'cn'},:debug => false }
config.omniauth :shibboleth_idp3, {:info_fields => {:email =>
  'mail', :name => 'cn'},:debug => false }
config.omniauth :shibboleth_idp4, {:info_fields => {:email =>
  'mail', :name => 'cn'},:debug => false }
```

According to this configuration a mapping of the IDP's SAML attributes is done. These attributes will be used to save the user's credential into the database.

4. Modify the route.rb file:

modify the line

```
devise_for :users, :controllers => {:omniauth_callbacks =>
   'omniauth_callbacks', registrations: 'user_registrations'}
```

as follows

```
devise_for :users, :controllers => {:omniauth_callbacks =>
   'omniauth_callbacks', registrations: 'user_registrations',
   sessions:  'user_sessions'}
```

5. Add a new controller OmniauthCallbacks

```
class OmniauthCallbacksController <
   Devise::OmniauthCallbacksController
  # Shibboleth provider
  def shibboleth_idp1
  # Check if the access request is made by an authorized IDP
   autenticate_sso_user(request)
   redirect_to :home
 end
 # Shibboleth provider
 def shibboleth_idp2
   # Check if the access request is made by an authorized IDP
   autenticate_sso_user(request)
   redirect_to :home
 end
 # Shibboleth provider
 def shibboleth_idp3
   # Check if the access request is made by an authorized IDP
   autenticate_sso_user(request)
   redirect_to :home
 end
 # Shibboleth provider
 def shibboleth_idp4
   # Check if the access request is made by an authorized IDP
   autenticate_sso_user(request)
   redirect_to :home
 end
private
  def autenticate_sso_user(requestVar)
   # Removed callback path
   info_idp =
   ExternalIdp.find_by_route(requestVar.env['REQUEST_URI'].to_s.sub('
   /callback',))
   if !info_idp.blank?
```

© 2013 - 2015 XIFI Consortium Parties

```
      if info_idp[:enabled]
        begin
          email = requestVar.env['omniauth.auth']['info']['email']
        rescue
          email = nil
        end
        if !email.nil?
          id_user = authorized_external_user(email,
   requestVar.env['omniauth.auth']['info']['name'], info_idp[:id])
          if id_user > 0
            sign_in User.find(id_user)
            set_flash_message :notice, :success, :kind =>
   info_idp[:description]
          end
        end
      end
    end
  end
  def authorized_external_user(email, name, idp_mark)
    # check if the user already exists
    user = User.find(:first, :include => :actor, :conditions =>
    {'actors.email' => email, :ext_idp => idp_mark})
    if !user.nil? # Find it!
      return user.id
    end
    password = Devise.friendly_token.first(12) # Generic password
    user = User.new
    user.email = email
    user.password = password
    user.password_confirmation = password
    user.skip_confirmation!
    user.name = name
    user.ext_idp = idp_mark
    user.save
    return user.id
  end
end
```

This controller inherits from Devise::OmniauthCallbacksController and is used by omniauth-shibboleth gem; through this controller we are able to intercept the IDP's callback and, in particular, through the authorized_external_user method it is possible to save the external user into the database.

These users are marked as 'external' and each user has assigned a tag that identifies the IDP to which it belongs.

6. Add a new controller UserSessions

```
class UserSessionsController < Devise::SessionsController
 include FiWareIdm::Controllers::Shibboleth
 def destroy
   shibboleth_logout
   super
 end
end
```

7. Amend the controller UserRegistrations with the follows method

```
def destroy
  shibboleth_logout
  super
```

© 2013 - 2015 XIFI Consortium Parties

```
end
```

with the steps 6 and 7 it has been added the capability, to both controllers, to make a SSO logout via the shibboleth_logout method (defined into the shibboleth.rb module (see step 8 in section A.2.2.2).

8. Add a new ruby module into the folder <project home>/lib/fi_ware_idm/controllers called shibboleth.rb

```
module FiWareIdm
 module Controllers
   module Shibboleth
    def shibboleth_logout
       if current_user.ext_idp?
         # Recover shibboleth session cookie
         cookies.each do |cookie|
           if ! cookie[0].index("_shibsession_").nil? # Find it! Make
   a GET for shibboleth Logout
              shib_cookie = {"Cookie" => cookie[0] + "=" + cookie[1]}
              RestClient.get(root_url +
   "Shibboleth.sso/Logout",shib_cookie)
              # Delete cookie
              cookies[cookie[0]] = { :expires => Time.at(0) }
           end
         end
       end
     end
   end
 end
end
```

9. Modify the ruby file user.rb (in the folder <project home>/lib/fi_ware_idm/models) adding, the following code

```
   . . . . . .
```

included do

```
   validate :checkEmail
    # Only for local users
   class << self
  %w( email slug name ).each do |a|
     eval <<-EOS
   def find_by_#{ a }(#{ a })
     find :first,
          :include => :actor,
          :conditions =>
            { 'actors.#{ a }' => #{ a }, :ext_idp => nil }
   end
     EOS
   end
 end
  def checkEmail
   . . . . . .
```

NOTE: Take care that these changes (underlined text) are such that controls over registration data is done only for “local” users (meaning not external users).

Database changes

To update the database the changes to be applied are described below.

© 2013 - 2015 XIFI Consortium Parties

1. Create a new table called external_idps

```
CREATE TABLE `external_idps` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `route` varchar(255) NOT NULL,
 `enabled` tinyint(1) DEFAULT '1',
 `mark` varchar(255) NOT NULL,
 `description` varchar(255) DEFAULT ,
 `created_at` datetime NOT NULL,
 `updated_at` datetime NOT NULL,
 `url` varchar(255) NOT NULL DEFAULT ,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
```

This table will be used to save the info about the external IDPs.

2. Create the ExternalIdp model

```
class ExternalIdp < ActiveRecord::Base
 attr_accessible :enabled, :route, :url
end
```

3. c.Add a new column and a foreign key to the users table

```
ALTER TABLE `users`
ADD `ext_idp` int(11) DEFAULT NULL
ALTER TABLE users
ADD CONSTRAINT `fk_users_external_idps`
FOREIGN KEY (`ext_idp`)
REFERENCES `external_idps`(`id`)
```

This column will be used to discriminate the “local” users (ext_idp = NULL) among “external” users.

An alternative procedure is to adopt the rail migration system to implement the same above described changes to the database [32].

A.2.2.5 Layout changes

The changes made to the layout of the pages are here described.

1. Add the remove_account_settings.rb file in the <project home>/app/override/setting/_index folder

```
Deface::Override.new(:virtual_path => "settings/_index",
                     :name => "remove_account_settings_start_if",
                     :insert_before => "code[erb-
   loud]:contains('devise/registrations/edit_user')",
                     :text => "<% if !current_user.ext_idp? %>")
Deface::Override.new(:virtual_path => "settings/_index",
                     :name => "remove_account_settings_end_if",
                     :insert_before => "code[erb-
   loud]:contains('render partial: \"language\"')",
                     :text => "<% end %>")
```

Doing so the account configuration for external users has been disabled.

2. Add a new area in the home page with the list of external IDPs:

   a. create the _external_login.html.erb file in the <project home>/app/view/frontpage folder

```
 <section id="external_sign_in">
  <h3>
     <%= t('external_IDP') %>
```

© 2013 - 2015 XIFI Consortium Parties

```
  </h3>
  <article class="login_data">
    <%- if devise_mapping.omniauthable? %>
      <%- resource_class.omniauth_providers.each do |provider|
          info_idp =
   ExternalIdp.find_by_route(omniauth_authorize_path(resource_name,
   provider))
          if !info_idp.blank? && info_idp[:enabled] -%>
          <%= link_to image_tag("#{ provider }.png"),
   omniauth_authorize_path(resource_name, provider) %>
        <% end -%>
      <% end -%>
    <% end -%>
  </article>
 </section>
```

b. create the _form.html.erb file in the <project home>/app/view/devise/session folder

```
 <div id="area_login">
  <section id="sign_in">
  <h3>
     <%= t('sign_in') %>
  </h3>
  <%= form_for(resource, :as => resource_name, :url =>
   session_path(resource_name)) do |f| %>
    <%= devise_error_messages! if controller.controller_name ==
   'sessions' %>
    <article class="login_data">
      <%= f.label :email %>
      <%= f.email_field :email %>
      <%= f.label :password %>
      <%= f.password_field :password %>
      <% if devise_mapping.rememberable? -%>
        <div id="remember">
          <%= f.check_box :remember_me %>
          <%= f.label :remember_me %>
        </div>
      <% end %>
      <%= f.submit t('sign_in'), :class =>"btn" %>
      <hr class="soften">
      <%= render :partial => "devise/shared/links" %>
    </article>
  <% end %>
  </section>
  <%= render :partial => 'frontpage/external_login' %>
  </div>
```

c. modify the _frontpage_and_devise.css.sass into <project home>/app/assets/stylesheets/fi-ware_idm/frontpage_and_device/layout folder

```
. . . . . .
#login
 input
   &[type="submit"]
     @include btn-primary
 #area_login
   float: right
   width: 450px
   #sign_in
     width: 88%
   #external_sign_in
```

© 2013 - 2015 XIFI Consortium Parties

```
    @include box-shadow(0px 0px 13px rgba(50, 50, 50, 0.15))
    @include border-radius (15px)
    background-color: $white
    color: #8c8c8c
    width: 89%
    display: inline-block
    padding: 5px 15px
    margin: 27px 2% 0 14px
    h3
      margin-top: 0px
      margin-bottom: 0px
      text-align: right
      font-size: 22px
```

d. create the _links.erb file into the <project home>/app/views/devise/shared folder

```
<article class="options">
 <%- if devise_mapping.registerable? && controller_name !=
   'registrations' %>
   <%= link_to t('devise.links.sign_up'),
   new_registration_path(resource_name) %>
 <% end -%>
 <%- if devise_mapping.recoverable? && controller_name !=
   'passwords' %>
   <%= link_to t('devise.links.forgot_password'),
   new_password_path(resource_name) %>
 <% end -%>
 <%- if devise_mapping.confirmable? && controller_name !=
   'confirmations' %>
   <%= link_to t('devise.links.confirmation_instructions'),
   new_confirmation_path(resource_name) %>
 <% end -%>
 <%- if devise_mapping.lockable? &&
   resource_class.unlock_strategy_enabled?(:email) &&
   controller_name != 'unlocks' %>
   <%= link_to t('devise.links.unlock_instructions'),
   new_unlock_path(resource_name) %>
 <% end -%>
 </article>
```

e. create the add_shibboleth_idp_url.html.erb.deface file into the <project home>/app/override/layouts/application folder

```
<%= include_shibbolet_idp %>
```

f. modify the application_helper.rb into <project home>/app/helpers folder

```
. . . . . .
 if options[:https].present?
     protocol += 's'
   end
   "#{ protocol }://#{ domain }"
 end
 def include_shibbolet_idp
   # For SSO IDP side
   script = "<script>var idp_logout_url ="
   if user_signed_in? and current_user.ext_idp?
     script += "'" + ExternalIdp.find(current_user.ext_idp).url +
   "/profile/Logout'";
   else
     script += ""
   end
```

© 2013 - 2015 XIFI Consortium Parties

```
    script += "</script>"
    script.html_safe
  end
end
```

g. modify the signOut.js.rb into <project home>/app/assets/javascripts folder

```
. . . . . .
var developmentCall = function(currentPortal) {
    var url = location.protocol + '//' + window.location.host +
    portals[currentPortal].path;
    $.ajax(url, {
      type: portals[currentPortal].verb,
      success: function() {
        if (idp_logout_url != )
        {
          window.location.replace(idp_logout_url);
        }
        else
        {
          window.location.replace(location.protocol + '//' +
    window.location.host);
        }
      }
    });
  };
. . . . . .
```

h. modify the devise.en.yml into <project home>/config/locale folder

```
. . . . . .
  devise:
    user_sessions:
      user:
        signed_in: 'Signed in successfully.'
        signed_out: 'Signed out successfully.'
    failure:
. . . . . .
```

i. for each external IDP we need to add a new logo image into the <project home>/app/assets/images called shibboleth_idp[1-4].png

Note that the current configuration support 4 IDP

A.2.3 Dynamic IDP

The list of available external IDP is not static but administrators can be updated it by a dedicated page inside the IDM.

© 2013 - 2015 XIFI Consortium Parties

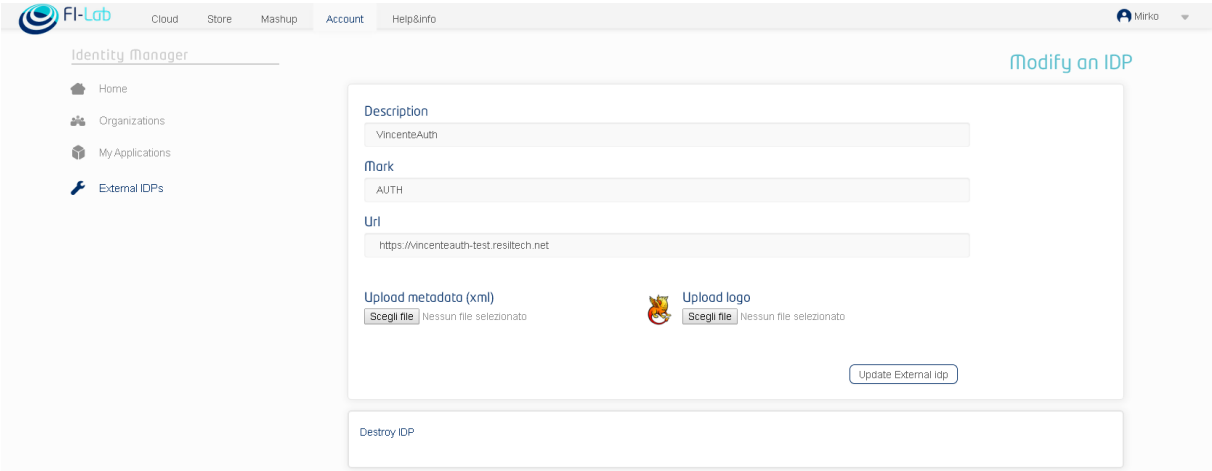


Figure 37: Page for insert/edit/destroy an IDP

The list of external IDP will be available at the following url **Error! Hyperlink reference not valid.**<hostname>/external_idps

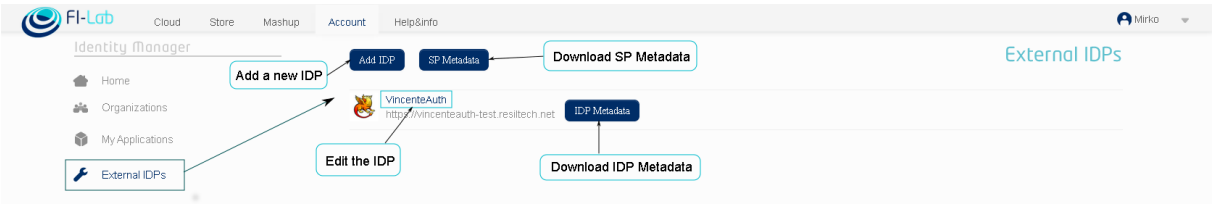


Figure 38: IDP list

In this page will be possible to download the idp's metadata and the sp's metadata in addition to the standard operation (insert and modify operation).

The available idp will be shown as link in the homepage (Figure 39).

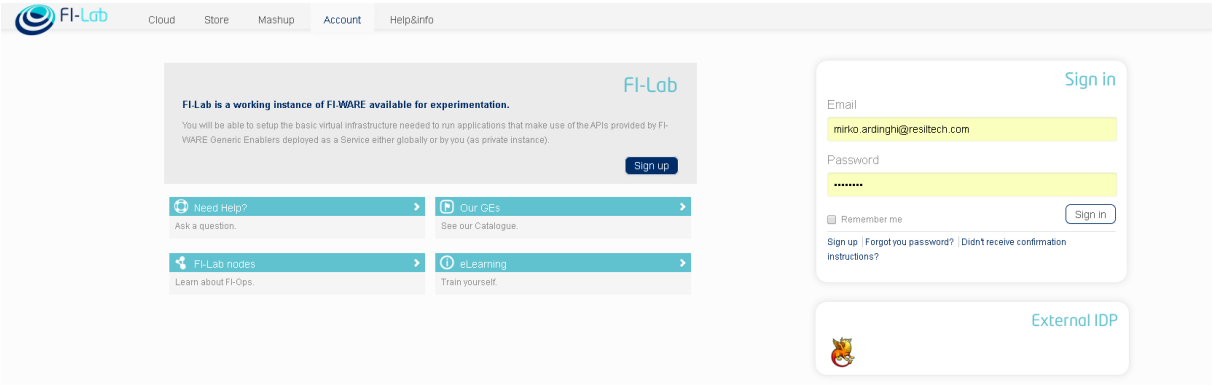


Figure 39: IDP Homepage

To implement the dynamic idp it was necessary to significantly alter the previous idp structures (model/view/controller).

Database changes

It was added two new fields to the external_idps table • logo (string): it will be stored the path of the idp's logo • metadata (string): it will be stored the path of the idp's metadata

MVC changes

It was added a dedicated controller for the external idp and the views to edit/add/delete it

© 2013 - 2015 XIFI Consortium Parties

```
class ExternalIdpsController < ApplicationController

 # GET /external_idps
 def index
   @external_idps = ExternalIdp.all
   respond_to do |format|
     format.html # index.html.erb
   end
 end
 # GET /external_idps/1
 def show
   @external_idp = ExternalIdp.find(params[:id])
   respond_to do |format|
     format.html # show.html.erb
   end
 end
 # GET /external_idps/new
 def new
   @new_idp = true # For the view
   @external_idp = ExternalIdp.new
   respond_to do |format|
     format.html # new.html.erb
   end
 end
 # GET /external_idps/1/edit
 def edit
   @new_idp = false # For the view
   @external_idp = ExternalIdp.find(params[:id])
 end
 # POST /external_idps
 def create
   @external_idp = ExternalIdp.new(params[:external_idp])
   respond_to do |format|
     if @external_idp.save && !@external_idp.errors.any?
       format.html { redirect_to external_idps_url, notice:
   t('idp.form.new') }
     else
       @new_idp = true
       format.html { render action: "new" }
     end
   end
 end
 # PUT /external_idps/1
 def update
   @external_idp = ExternalIdp.find(params[:id])
   respond_to do |format|
     res_update =
   @external_idp.update_attributes(params[:external_idp])
     if res_update && !@external_idp.errors.any?
       format.html { redirect_to external_idps_url, notice:
   t('idp.form.update') } # Tutto ok
     else
       @new_idp = false
       format.html { render action: "edit" }
     end
   end
 end
 # DELETE /external_idps/1
 def destroy
   @external_idp = ExternalIdp.find(params[:id])
```

© 2013 - 2015 XIFI Consortium Parties

```
    @external_idp.destroy
    respond_to do |format|
      format.html { redirect_to external_idps_url, notice:
    t('idp.delete')  }
    end
  end
  #def download
  #end
end
```

Below the form details to add/edit external idp (used by the external idp views)

```
 <section id="external_idp_data" class="zona">
  <%= form_for(@external_idp) do |f| %>
  <%if @external_idp.errors.any?%>
  <div id="error_explanation">
    <h2><%=  pluralize(@external_idp.errors.count,"error") %>
      prohibited this product from being saved:
    </h2>
    <ul>
      <%  @external_idp.errors.full_messages.each do |msg| %>
        <li> <%=  msg %> </li>
      <%  end %>
    </ul>
  </div>
  <%end%>
  <section class="form_section description">
    <%= f.label :description %>
    <%= f.text_field :description, :required =>  "" %>
  </section>
  <section class="form_section">
    <%= f.label :mark %>
    <%= f.text_field :mark, :required =>  "" %>
  </section>
  <section class="form_section">
    <%= f.label :url %>
    <%= f.text_field :url, :required =>  "" %>
  </section>
  <section class="form_section" id="idp_upload_area">
    <div id="idp_upload_metadata">
      <%= f.label t('idp.form.metadata.label') %>
      <%=
      if @new_idp
        f.file_field :metadata, :required =>  ""
      else
        f.file_field :metadata
      end
      %>
    </div>
    <div id="idp_upload_logo_area">
      <div id="idp_upload_logo_img">
        <%= if @external_idp.logo_url.nil?
              image_tag "/assets/logos/small/site.png"
            else
              image_tag @external_idp.logo_url
            end %>
      </div>
      <div id="idp_upload_logo_button">
        <%= f.label t('idp.form.logo.label') %>
        <%= f.file_field :logo %>
      </div>
```

© 2013 - 2015 XIFI Consortium Parties

```
    </div>
  </section>
  <div class="actions">
    <%= f.submit :class => 'btn' %>
  </div>
  <% end %>
  </section>
 </section>
```

Every time that an idp is modified/added/removed the IDM runs a ruby script (see **Ruby Script** below) to apply the new changes. To do this it was necessary to modify some files that had already been modified to implement the management of the idp statically.

IDM configuration changes

1. File omniauth.rb (see step 2 under **Controller changes** in section A.2.2.4) become

```
# initializers/omniauth.rb
module OmniAuth::Strategies
 class ShibbolethIdp1 < Shibboleth
   def name
     :shibboleth_idp1
   end
 end
 class ShibbolethIdp2 < Shibboleth
   def name
     :shibboleth_idp2
   end
 end
 class ShibbolethIdp3 < Shibboleth
   def name
     :shibboleth_idp3
   end
 end
 class ShibbolethIdp4 < Shibboleth
   def name
     :shibboleth_idp4
   end
 end

end
```

2. In file devise.rb (see step 2 under **IDM configuration changes**) the new lines like the following

```
 config.omniauth :shibboleth_idp1, {:info_fields => {:email =>
   'mail', :name => 'cn'},:debug => false }
```

   are removed

File OmniauthCallbacksController (see step 5 under **Controller changes** in section A.2.2.4) become:

```
class OmniauthCallbacksController <
   Devise::OmniauthCallbacksController
# Shibboleth provider
 def shibboleth_idp1
   # Check if the access request is made by an authorized IDP
   authenticate_sso_user(request)
   redirect_to :home
 end
 # Shibboleth provider
 def shibboleth_idp2
   # Check if the access request is made by an authorized IDP
```

© 2013 - 2015 XIFI Consortium Parties

```
  autenticate_sso_user(request)
  redirect_to :home
 end
 # Shibboleth provider
 def shibboleth_idp3
  # Check if the access request is made by an authorized IDP
  autenticate_sso_user(request)
  redirect_to :home
 end
 # Shibboleth provider
 def shibboleth_idp4
  # Check if the access request is made by an authorized IDP
  autenticate_sso_user(request)
  redirect_to :home
 end

 private
  def autenticate_sso_user(requestVar)
   # Removed callback path
   info_idp =
   ExternalIdp.find_by_route(requestVar.env['REQUEST_URI'].to_s.sub('
   /callback',))
   if !info_idp.blank?
     if info_idp[:enabled]
       begin
         email = requestVar.env['omniauth.auth']['info']['email']
       rescue
         email = nil
       end
       if !email.nil?
         id_user = authorized_external_user(email,
   requestVar.env['omniauth.auth']['info']['name'], info_idp[:id])
         if id_user > 0
           sign_in User.find(id_user)
           set_flash_message :notice, :success, :kind =>
   info_idp[:description]
         end
       end
     end
   end
 end
 def authorized_external_user(email, name, idp_mark)
  # check if the user already exists
   user = User.find(:first, :include => :actor, :conditions =>
   {'actors.email' => email, :ext_idp => idp_mark})
   if !user.nil? # Find it!
     return user.id
   end
   password = Devise.friendly_token.first(12) # Generic password
   user = User.new
   user.email = email
   user.password = password
   user.password_confirmation = password
   user.skip_confirmation!
   user.name = name
   user.ext_idp = idp_mark
   user.save
   return user.id
 end
end
```

Apache configuration changes

A. The part of configuration that concern the idp (see step 1 under **Apache configuration** in section A.2.2.2) must be removed

```
<Location /users/auth/shibboleth_idp[1..n]/callback>
Options -MultiViews
AuthType shibboleth
ShibRequestSetting  entityID <url IDPn>/idp/shibboleth
ShibRequestSetting requireSession On
require valid-user
</Location>
```

Shibboleth SP configuration changes

The configuration file (see step 8 in section A.2.2.2) is been modified as below

```
<MetadataProvider type="Chaining">

        <MetadataProvider type="XML"
    path="/etc/shibboleth/metadata/idp1-metadata.xml"/>
        <MetadataProvider type="XML"
    path="/etc/shibboleth/metadata/idp2-metadata.xml"/>

</MetadataProvider>
```

Ruby script

All the parts removed in omniauth.rb, devise.rb, OmniauthCallbacksController (see steps 1, 2 and 3 under **IDM configuration changes**), Apache (see **Apache configuration changes**) and Shibboleth SP (see **Shibboleth SP configuration changes**) will be applied dynamically by the ruby script. The script (multiple-idp-configure.rb) is located in <project home>/lib/scripts. Is possible to change some settings by editing the file <project home>/lib/scripts/multiple-idp-configure.ini.

```
[sp]
configuration_path = /etc/shibboleth/
[apache]
configuration_path = /etc/apache2/sites-available/
[idm]
path = /home/mirko/progetti/fi-ware-idm/
```

The ruby script (and his shell execution script <project home>/lib/scripts/multiple-idp-configure.sh) must be executed with root permissions.

SSO Logout

A brief consideration about the SSO Logout is reported in this section.

When an user is logged in KeyRock through an external IDP the logout button will redirect the browser to the IDP logout url (see step 2g in section A.2.2.5), furthermore KeyRock makes a REST operation towards the SP logout route (see step 8 under **Controller changes** in section A.2.2.4).

These operations are necessary in order to avoid that the current user is logged with the credentials of the previous user who logged in with the same browser (with cookies enabled).

This is why the system will still use the SSO data/credential of the previous user (saved inside the cookies).

User interface

The following describes the KeyRock behaviour after the SP implementation.

Figure 40 shows the new form with the external IDPs area.

© 2013 - 2015 XIFI Consortium Parties

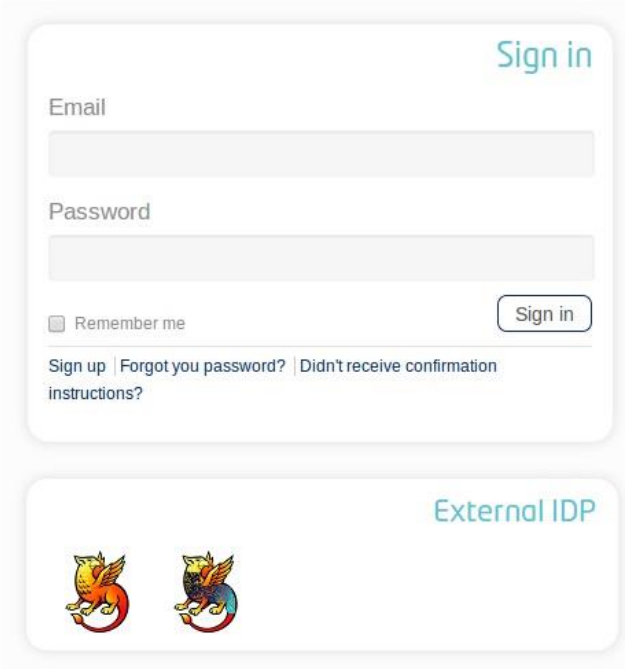


Figure 40: KeyRock login form

When the user selects one of the shibboleth images he will redirect to the login form of the selected IDP, see Figure 41

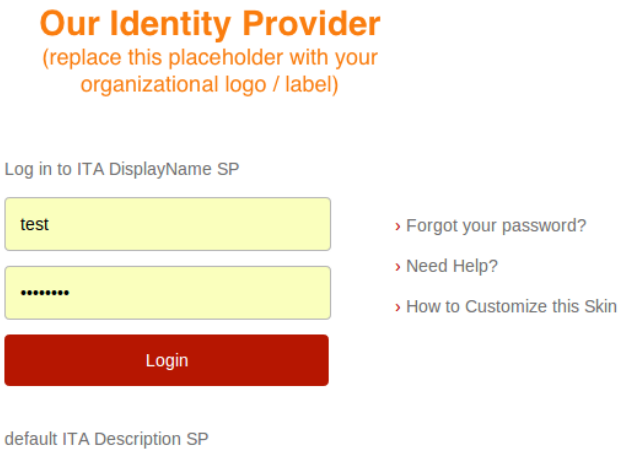


Figure 41: IDP login form

After a successful log in the user will be redirect to the KeyRock homepage.

When the user does logout he will be redirected to the IDP logout page, see Figure 42.

© 2013 - 2015 XIFI Consortium Parties

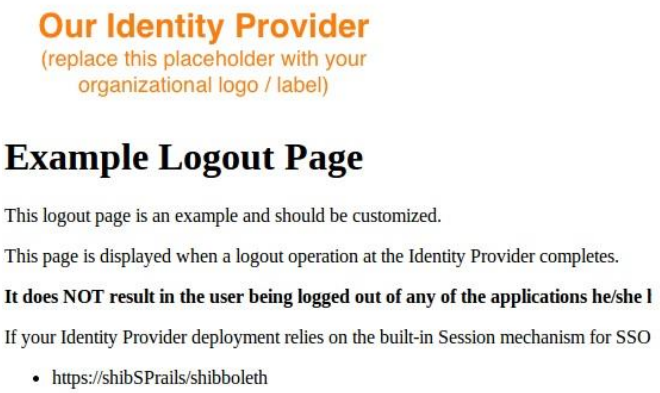


Figure 42: I IDP logout form

Figure 43 shows how the external users are saved into the database with the 'ext_idp' field different from NULL.

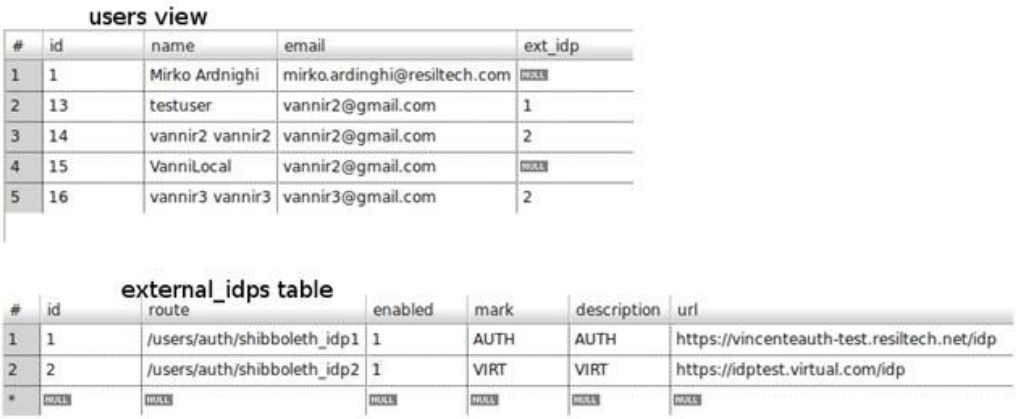


Figure 43: External users in the database

A.3 IDM integration with SAML IDP

This section describe how to modify the IDM to integrate the Shibboleth Identity Provider module

A.3.1 IDM Architecture to support SAML IDP

This section proposes two solutions to extend the IDM in order to act also as a SAML 2.0 IdP.

1. Here database synchronization toward the original user MySQL database is suggested.
2. Here the MySQL database is directly made available for external authentication without the need for specific synchronization logic.

Solutions are compared and one is then selected for implementation.

Solution 1: MySQL-LDAP syncro solution

Error! Reference source not found. shows the IDM modifications where a database synchronization s necessary in order to interface with the LDAP backend adopted by Shibboleth.

The main necessary modifications are the following.

1. Install the Shibboleth Identity Provider (IDP) in the IdM's machine and his dependencies

© 2013 - 2015 XIFI Consortium Parties

(LDAP, Tomcat)

2. Modify the Apache config file to add the IDP configuration
3. Connect LDAP with the IDM's database:
    a. This is necessary to make the user credentials “available” to the Shibboleth Identity Provider for authentication.
    b. This is done thanks to a dedicated sync module.

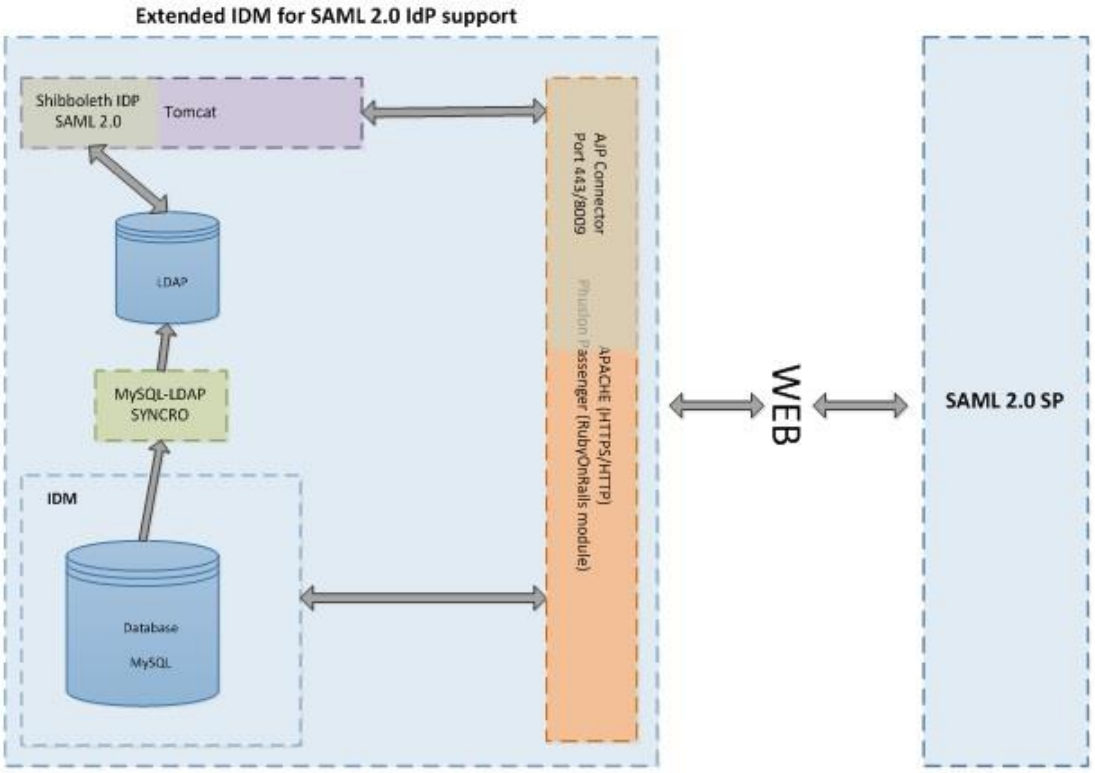


Figure 44: Extended IDM for SAML IdP with database synchronization

Solution 2: LDAP using MySQL as backend

Error! Reference source not found. shows the IDM modifications where the connections between DAP and the IDM database is done using ODBC driver.

The main necessary modifications are the following.

1. Install the Shibboleth Identity Provider (IDP) in the IdM's machine and his dependencies (LDAP, Tomcat)
2. Modify the Apache config file to add the IDP configuration
3. Connect LDAP with the IDM's database as backend
    a. Install SQL support for OpenLDAP.
    b. Install the BCrypt module for OpenLDAP [14].

© 2013 - 2015 XIFI Consortium Parties

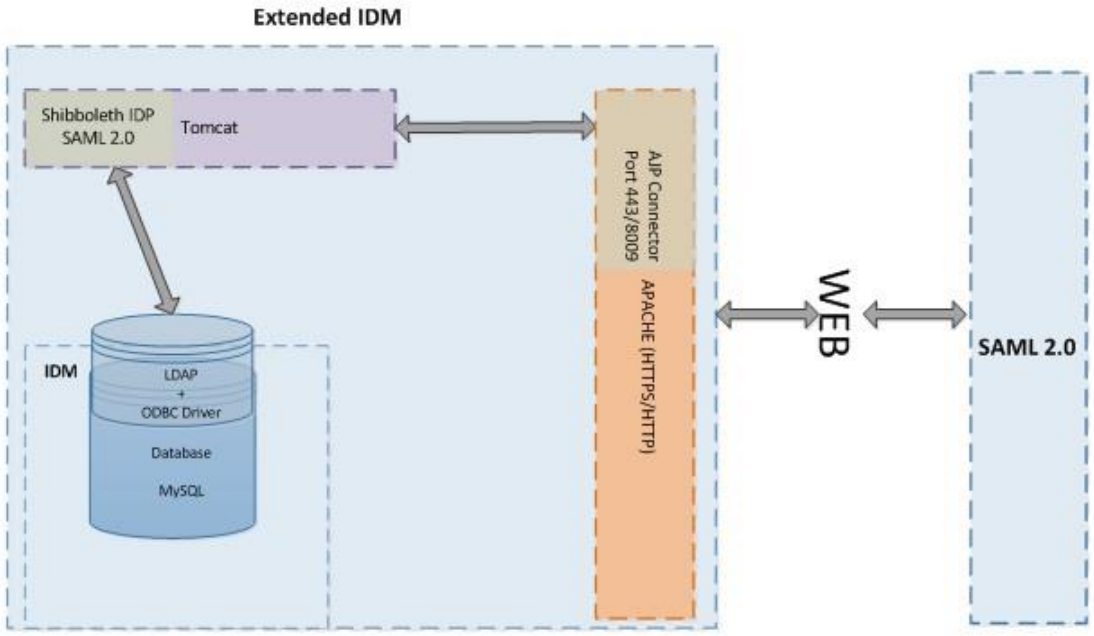


Figure 45: Extended IDM for SAML IdP with ODBC driver

- Conclusion - Comparison and solution selection

  Saying that there are no functional differences between the two proposed solutions choice has fallen on solution 2 described above.

  Main reason is for adopting solution 2 is to avoid the issues of database synchronization that is not necessary in this case. From implementation point of view solution 2 is simpler than solution1 and any potential mis-alignment of databases is avoided.

A.3.2 IDM Update to support SAML SP

To integrate the Shibboleth IDP it is necessary define an interaction mean between LDAP (used by Shibboleth IDP) and MySQL (used by KeyRock), because the shibboleth IDP must use the users (identified by their email + password) saved into the MySQL database.

This integration need generate the following issue to solve: KeyRock saves the user's password as an hash value (calculated with the BCrypt algorithm) but unfortunately LDAP doesn't support Bcrypt [14] natively.

IDP installation

Note that Shibboleth IDP requires Tomcat [16] and LDAP [9] to run.

Shibboleth IDP installation

1. Download the shibboleth IDP

```
wget http://www.shibboleth.net/downloads/identity-provider/2.4.0/shibboleth-identityprovider-2.4.0-bin.zip
unzip shibboleth-identityprovider-2.4.0-bin.zip
cd shibboleth-identityprovider-2.4.0
```

2. Modify $IDP_SRC/src/main/webapp/WEB-INF/web.xml with own CIDR

```
<servlet>
<servlet-name>Status</servlet-name>
<servlet-class>
edu.internet2.middleware.shibboleth.idp.StatusServlet
</servlet-class>
```

© 2013 - 2015 XIFI Consortium Parties

```
<init-param>
<param-name>AllowedIPs</param-name>
<param-value>
127.0.0.1/32 ::1/128 #your IP range#
</param-value>
</init-param>
<load-on-startup>2</load-on-startup>
</servlet>shibboleth-identityprovider-2.4.0-bin.zip
```

3. Install Shibboleth IDP with the command "sh install.sh" (choose you FQDN , store the password and don't change the default installation path /opt/shibboleth)
4. Copy the Xerces and Xalan libraries in $TOMCAT_HOME

```
cp -r $IDP_SRC/endorsed/ $CATALINA_HOME
```

5. Modify the IDP's folder permission and the credentials files permissions

```
chown tomcat7 $IDP_HOME/logs/
chown tomcat7 $IDP_HOME/metadata/
chown tomcat7 $IDP_HOME/credentials/
```

```
chmod 400 $IDP_HOME/credentials/idp.key
chmod 644 $IDP_HOME/credentials/idp.crt
chown tomcat7 $IDP_HOME/credentials/idp.key
chown tomcat7 $IDP_HOME/credentials/idp.crt
```

and remove the "<Connector port=8080 …"

Tomcat configuration

1. Modify the server.xml file in $TOMCAT_HOME/conf/server.xml .

   Add the follow connector

```
<Connector port="8009"
protocol="AJP/1.3"
redirectPort="443"
address="127.0.0.1"
enableLookups="false"
tomcatAuthentication="false" />
```

   and remove the "<Connector port=8080 …"

2. Modify the /etc/default/tomcat7 file:

   the line

```
#AUTHBIND=no
```

   become

```
AUTHBIND=yes
```

3. Put the private key and the certificate in the $IDP_HOME/credentials folder
4. Deploy the IDP's war file (in $IDP_HOME/war/) into tomcat
5. Enable the SSL and the proxy_ajp apache's modules

```
sudo a2enmod ssl proxy_ajp ; service apache2 restart
```

6. Modify the apache configuration file

```
<VirtualHost _default_:443>
```

© 2013 - 2015 XIFI Consortium Parties

```
ServerName idp.example.org:443
ServerAdmin admin@example.org
DocumentRoot /var/www
<Proxy ajp://localhost:8009>
Allow from all
</Proxy>
ProxyPass /idp ajp://localhost:8009/idp retry=5
ProxyPassReverse /idp ajp://localhost:8009/idp retry=5
SSLEngine On
SSLCipherSuite HIGH:MEDIUM:!aNULL:!MD5
SSLProtocol all -SSLv2
SSLCertificateFile /opt/shibboleth-idp/certs/cert-server.pem
SSLCertificateKeyFile /opt/shibboleth-idp/certs/key-server.pem
SSLCertificateChainFile /opt/shibboleth-idp/certs/Terena-Chain.pem
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>
<VirtualHost _default_:8443>
ServerName idp.example.org:8443
ServerAdmin admin@example.org
DocumentRoot /var/www
<Proxy ajp://localhost:8009>
Allow from all
</Proxy>
ProxyPass /idp ajp://localhost:8009/idp retry=5
ProxyPassReverse /idp ajp://localhost:8009/idp retry=5
SSLEngine On
SSLCipherSSLProtocol all -SSLv2
SSLVerifyClient optional_no_ca
SSLVerifyDepth 10
SSLCertificateFile /opt/shibboleth-idp/credentials/idp.crt
SSLCertificateKeyFile /opt/shibboleth-idp/credentials/idp.key
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>Suite HIGH:MEDIUM:!aNULL:!MD5
```

7. Add the port 8443 to the apache file ports.conf

Shibboleth configuration

1. Modify the $IDP_HOME/conf/handler.xml file: Disable RemoteUser block (comment it), enable UsernamePassword block (uncomment it)
2. Modify the $IDP_HOME/conf/login.config file (add ldap connection info)

```
edu.vt.middleware.ldap.jaas.LdapLoginModule required
ldapUrl="ldap://ldap.example.org:389"
baseDn="dc=garr,dc=it"
bindDn="cn=ldapadmin,dc=garr,dc=it"
bindCredential="password_serviceUser"
ssl="false"
userFilter="uid={0}"
subtreeSearch="true";
```

© 2013 - 2015 XIFI Consortium Parties

3. Inside the $IDP_HOME/conf/attribute-resolver.xml file uncomment all <resolver:AttributeDefinition .... > occurrences. Don't uncomment the follows part

```
<!-- Do NOT use the version of eduPersonTargetedID defined below
  unless you
understand why it was deprecated and know that this reason does not
  apply to
you.
<!--
<resolver:AttributeDefinition xsi:type="ad:Scoped" idID.old"
  scope="unimo.it"
sourceAttributeID="persistentID">
<resolver:Dependency ref="storedID" />
<resolver:AttributeEncoder xsi:type="enc:SAML1ScopedString"
name="urn:mace:dir:attribute-def:eduPersonTargetedID" />
</resolver:AttributeDefinition>
-->
```

4. Set up the LDAP connection info

```
<!-- Example LDAP Connector -->
<resolver:DataConnector id="myLDAP"
xsi:type="LDAPDirectory"
ldapURL="ldap://ramo1.di.ldap.da.reperire
ldap://ramo2.di.ldap.da.reperire
ldap://ramoN.di.ldap.da.reperire"
baseDN="dc=dc_di_ldap,dc=dc_di_ldap"
useStartTLS="true" /* se LDAP con TLS */
/* Parameters that can be omitted */
principal="cn=admin_di_ldap,dc=garr,dc=it"
principalCredential="password_principal">
/************************/
<dc:FilterTemplate>
<![CDATA[
(uid=$requestContext.principalName)
]]>
</dc:FilterTemplate>
</resolver:DataConnector>
<resolver:Dependency ref="myLDAP" />
```

5. Modify the attributes file (attribute-filter.xml). We have added the attributes for the user's registration in the Shibboleth SP.

```
<afp:AttributeFilterPolicy id="specifiedSPexample">
        <afp:PolicyRequirementRule xsi:type="basic:ANY"/>
        <afp:AttributeRule attributeID="commonName">
                <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
        <afp:AttributeRule attributeID="email">
                <afp:PermitValueRule xsi:type="basic:ANY" />
        </afp:AttributeRule>
</afp:AttributeFilterPolicy>
```

6. Modify the $IDP_HOME/conf/relying-party.xml as follows

```
   <rp:AnonymousRelyingParty provider="https://<my idp
  url>/idp/shibboleth" defaultSigningCredentialRef="IdPCredential"/>
   <rp:DefaultRelyingParty provider="https://<my idp
  url>/idp/shibboleth" defaultSigningCredentialRef="IdPCredential">
        <!--
            Each attribute in these profiles configuration is set to
  its default value,
```

© 2013 - 2015 XIFI Consortium Parties

```
            that is, the values that would be in effect if those
   attributes were not present.
            We list them here so that people are aware of them (since
   they seem reluctant to
            read the documentation).
         -->
         <rp:ProfileConfiguration xsi:type="saml:ShibbolethSSOProfile"
   includeAttributeStatement="false"
                                  assertionLifetime="PT5M"
   signResponses="conditional" signAssertions="never"
                                  includeConditionsNotBefore="true"/>
  . . . . . .
 <metadata:MetadataProvider xsi:type="FilesystemMetadataProvider"
       xmlns="urn:mace:shibboleth:2.0:metadata" id="SPMETADATA"
        metadataFile="/opt/shibboleth-idp/metadata/<sp metadata>.xml"
   />
 . . . . . .
```

7. Copy the SP's metadata "/opt/shibboleth-idp/metadata/<sp metadata>.xml"

LDAP installation

For LDAP implementation OpenLDAP [15] has been used.

OpenLDAP has been compiled from scratch using the last stable version (2.4.39).

OpenLDAP compile configuration has been defined to be strictly adapted to the scope of its usage in this context, indeed it has been compiled enabling the support of SQL and modules.

The configuration setting is reported below:

```
./configuration -enable-sql -enable-modules
```

SQL supports OpenLDAP back-end based on any relational database which has ODBC driver. In *nix platform the most common ODBC implementation is "unixODBC"; drivers available for "unixODBC" encompasse all the most common DBMS, ranging form MySQL to Oracle DB.

In the context of this project, it isn't required any specific features from "unixODBC", thus it can be installed directly from the sources, or using the package manager of the current *nix platform.

The tests were conducted with "unixODBC" 2.2.14p2-5ubuntu3. KeyRock relies upon MySQL thus "unixODBC" was configured to use MySQL driver. The configuration is reported below.

```
[openldap]
Description     = MySQL based OpenLDAP's backend
Driver          = MySQL
Database        = openldap
Servername      = localhost
UserName        = root
Password        = resiltech
ReadOnly        = No
RowVersioning   = No
ShowSystemTables= No
ShowOidColumn   = No
FakeOidIndex    = No
ConnSettings    =
OPTION          = 3407872
SOCKET          = /var/run/mysqld/mysqld.sock
```

The tests were conducted with a database hosted in the local machine whose access credentials were "root" as username and "resiltech" as password. "ReadOnly" option must be set to "No" in order to avoid any kind of unbearable caching policy, the other options can be set following the user needs and *nix platform.

© 2013 - 2015 XIFI Consortium Parties

It is worth to mention that the "option" key was the subject of a deeply study because caching policy did affect correctness of OpenLDAP results. Thus to avoid any kind of caching, the use of forward cursors was forced as well as its not caching.

The final step was to let KeyRock and OpenLDAP sharing the same tables, to do so OpenLDAP tables were re-designed as view of KeyRock tables. The newly defined views did have the very same structure of the tables that they had replaced.

Furthermore KeyRock tables were enriched with the fields "ext_idp" column for managing users from different IdMs as well as to define a finer control over each user that can be authenticated by the local OpenLDAP instance.

All these considerations led to replacing the openldap entities table, called "ldap_entries", with the view defined as follows:

```
select `openldap`.`ldap_entries_template`.`id` AS
   `id`,`openldap`.`ldap_entries_template`.`dn` AS
   `dn`,`openldap`.`ldap_entries_template`.`oc_map_id` AS
   `oc_map_id`,`openldap`.`ldap_entries_template`.`parent` AS
   `parent`,`openldap`.`ldap_entries_template`.`keyval` AS `keyval`
   from `openldap`.`ldap_entries_template` union select (11 +
   `persons`.`id`) AS
   `id`,concat('uid=',`persons`.`uid`,',ou=users,dc=example,dc=com')
   AS `dn`,1 AS `oc_map_id`,9 AS `parent`,`persons`.`id` AS `keyval`
   from `openldap`.`persons`
```

An interesting observation of "ldap_entries" is that it was defined as a compounded view made up of a static part, defined by the table "ldap_entries_template" and a variable part which relies upon the "persons" view which in turn come after the "users" and "actors" tables of KeyRock. The static part, defined by the table "ldap_entries_template", provides the structural information that Shibboleth needs to invoke correctly OpenLDAP; meanwhile the variable part provides the merged joint between OpenLDAP and KeyRock.

The "persons" view was defined as follows:

```
select `keyrock_db`.`users`.`id` AS `id`,`keyrock_db`.`actors`.`name`
   AS `name`,`keyrock_db`.`actors`.`name` AS
   `surname`,concat('{BCRYPT}',`keyrock_db`.`users`.`encrypted_passwo
   rd`) AS `password`,`keyrock_db`.`actors`.`name` AS
   `uid`,`keyrock_db`.`actors`.`email` AS `email` from
   (`keyrock_db`.`users` join `keyrock_db`.`actors`
   on((`keyrock_db`.`users`.`actor_id` =
   `keyrock_db`.`actors`.`id`))) WHERE `keyrock_db`.`users`.`ext_idp`
   IS NULL
```

The "users" and "actors" KeyRock tables were merged and rearrenged to be shaped for its subsequent use in "ldap_entries" view creation query.

Furthermore some fields were reworked, such as the "password" field, in order to have a format OpenLDAP compliant.

Finally the WHERE clause provides the filtering logic that allowed to authenticate the exact set of users managed by the local KeyRock instance.

A.3.3 BCrypt problem solution

Bcrypt functionalities are not natively supported in OpenDLAP 2.4.39 [15], thus the encryption and decryption capabilities were added by the means of an external module. For security reasons, the default compile settings doesn't encompass the use of external modules, thus it was necessary to enable it manually by "–enable-modules" parameter.

The Bcrypt functionalities were provided by [14] which don't require anything more than following

© 2013 - 2015 XIFI Consortium Parties

the compiling instructions.

[end of document]

© 2013 - 2015 XIFI Consortium Parties