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

eXperimental Infrastructures for the Future Internet

D2.2: APIs and Tools for Infrastructure Federation v1

Revision: v1.0

Work package	WP2
Task	Task T2.2
Due date	31/12/2013
Submission date	20/02/2014
Deliverable lead	Fraunhofer
Authors	Bernd Bochow (Fraunhofer, Editor), Daniel Nehls, Alexander Willner (TUB), Silvio Cretti, Attilio Broglio, Federico Facca (CREATE-NET), Panos Trakadas, Andreas Papadakis (SYNELIXIS), Maria Di Girolamo, Laura Pucci, Davide Dalle Carbonare (ENG), Susana Gonzalez Zarzosa, Joaquin Iranzo (ATOS), Pablo Rodríguez (TID), Ian Mills (WIT), Federico Alvarez, Jose Gonzalez (UPM), Riwal Kerherve (ImaginLab)
Reviewers	Federico Alvarez (UPM), Massimo Vecchio (CREATE-NET)

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 design documents. This design document focuses on the **federation concepts and architecture**, on how to **manage the federation** when established and on **procedures** to create and maintain the federation. It documents **APIs and other interfaces** utilized by the tools and portals that provide access to the federation for infrastructure owners and users.
Keywords	Federation APIs, Federation Tools, Specifications

Document Revision History

Version	Date	Description of change	List of contributor(s)
V0.1	29.01. 2014	First integrated version for internal circulation	Bernd Bochow (Fraunhofer)
V0.2	31.01.2014	Second integrated version for internal review	Bernd Bochow (Fraunhofer)
V0.3	03.02.2014	First complete of deliverable ready for review	Bernd Bochow (Fraunhofer)
V0.4	10.02.2014	First reviewed version	Bernd Bochow (Fraunhofer)
V0.5	17.02.2014	Reverted to include "Figure 6: Federation Monitoring Architecture" upon request Added text provided by ATOS to the sect. 2.5 introduction.	Bernd Bochow (Fraunhofer)
V0.6	20.02.2014	Implemented changes requested by the XIFI TM and the WP2 lead. Included contributions from ENG, CREATE-NET, ATOS, ImaginLab, WIT,TUB	Maria Di Girolamo (ENG), Federico Facca, Silvio Cretti(CREATE-NET), Riwal Kerherve (ImaginLab), Susana Gonzalez Zarzosa, Joaquin Iranzo (ATOS), Daniel Nehls (TUB), Bernd Bochow (Fraunhofer)
V1.0	21.02.2014	Final revised version	Federico Facca, Silvio Cretti(CREATE-NET)

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	

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

EXECUTIVE SUMMARY

Deliverable D2.2 is 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 design documents.

This design document focuses on the **federation concepts and architecture**, on how to **manage the federation** when established, on **procedures** to create and maintain the federation, and it documents **APIs and other interfaces** utilized by the tools and portals that provide access to the federation for infrastructure owners and users.

This document targets various XIFI stakeholders, aiming in particular to support developers, infrastructure owners, operators and end users to understand the XIFI federation methodology, to learn about the services enabled through infrastructure federation, and to gain specific knowledge in utilizing the federated XIFI infrastructures.

In particular, this document emphasizes on the **Federation Management**, on **Federated Monitoring** and on Federated **Identity Management** in a federated infrastructures environment. Following an integrating approach, this document builds upon the approaches and findings of other XIFI work packages documented in several other deliverable documents, complementing these by a specific view on federation aspects. Whenever suitable, information already provided by related documents but required herein is referenced or briefly summarized rather than duplicated.

It is shown that a “thin” federation layer is required to manage and maintain the federation and to assist infrastructures in their interaction with the federation. The federation layer presented here aims to enable infrastructures to join the federation and to commit local services as well as to access, interface and monitor these services and their supporting infrastructures. The underlying functionality required to support and implement the federation layer then is realized by enhancing existing services and components through certain enabling interfaces and functions described in this document.

Thus, this document provides an overview on the procedures and capabilities required in the XIFI context to federate infrastructures. It elaborates further on some of the main components already in a stable state of development and discusses their interaction in the federation. Finally, it documents the APIs of software components delivered that implement the federation.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application Programming Interface
GE	Generic Enabler
DEM	Datacenter & Enablers Monitoring
IaaS	Infrastructure as a Service
IMA	Infrastructure Management Authority
NAM	Network Active Monitoring
NMA	Node Management Authority
NPM	Network Passive Monitoring
PaaS	Platform as a Service
RMA	Region Management Authority
SaaS	Software as a Service
SCIM	System for Cross Domain Identity Management
SE	Specific Enabler
SLA	Service Level Agreement
SME	Small or Medium Enterprise
UC	Use Case (Trial)
UFI	Universal Federation Identifier
UUID	Universally Unique Identifier
VM	Virtual Machine
XIMM	XIFI Infrastructure Monitoring Middleware

1 INTRODUCTION

XIFI creates a community cloud for European FI-PPP developers. Being part of the ambitious FI-PPP programme [1] by the European Commission's Framework Programme 7 that aims at exploring the potential of a common platform for Future Internet technologies and to establish new business ecosystems, XIFI will provide a platform and marketplace for

- hosting web-based services offered by FI-PPP, namely the Generic Enablers developed by FI-WARE [2] and the Specific Enablers developed by particular Use Case Trials, and for
- advanced Future Internet infrastructures providing capabilities and data to leverage applications developed by early adopters of FI-PPP technologies.

Loosely speaking, XIFI itself is an adopter of FI-PPP technologies in the sense that it creates and maintains an infrastructure of federated hosts across Europe (i.e. the XIFI Nodes). It deploys a common set of Generic Enablers for distributed data centres, potentially integrating advanced infrastructures (e.g. sensor and actor networks, local wireless access technologies). Then, the XIFI infrastructure is offered to developers (whether they are part of large trials or will be part of Phase III SMEs and web entrepreneurs) enabling them to validate Future Internet technologies through their applications.

XIFI aims to provide and operate a federated Future Internet infrastructure to overcome potential fragmentation and diversification of infrastructures, technologies and services offered to developers of Future Internet technologies in Europe. The federation is formed by integrating heterogeneous infrastructures throughout Europe and is intended to establish a sustainable marketplace for different stakeholders being able to cope also with large trial deployments.

This document gives a concise overview of APIs and tools provided to create and maintain the federation of XIFI infrastructures, in that complementing and documenting the software implementation of the first prototype available for deployment. The prototype consists of various software module contributions provided by all technical work packages of the XIFI project. While contributions developed with a clear focus on infrastructure federation (cf. also [6]) are documented herein, supporting modules, functions and tools that complement or enable federation without being developed with a main focus on federation have been referenced or briefly summarized, which seemed more preferable than to duplicate information.

This document aims to support various XIFI stakeholders [18], in particular developers, infrastructure owners, operators and end users, to understand the XIFI federation methodology, to learn about the services enabled through infrastructure federation, and to gain specific knowledge in utilizing the federated XIFI infrastructures.

1.1 Document structure

The following sections will focus on the federation aspects of the XIFI infrastructure and of the XIFI nodes. In particular, they will deal with tools required to establish and maintain the federation, the required services, and the extensions to Future Internet technologies and interfaces needed (i.e. the Generic Enablers and the APIs).

More in detail, this document consists of four main sections.

First, the section on **federation concepts and architecture** links with XIFI work packagesWP1through WP5 and outlines the concepts of federating XIFI nodes with regards to the core services infrastructure federation, identity federation and infrastructure monitoring. For the time being and in the scope of this document, the term "federation" shall refer to any scenario that comprises collaboration or cooperation involving more than two tiers. In practice, we refer to a federated use case relevant in the scope of this document when more than two XIFI nodes (e.g. a master node and two or more slave nodes [3]) are involved in a certain use case, procedure or proof-of-concept that requires

coordination among entities, roles or stakeholders. Aiming to establish terminology and framework definitions for further developing detailed procedures, suitable tools and API enhancements, this section elaborates on the various roles considered, and on the functional enhancements required within the framework to enable XIFI nodes to federate. This section concludes by giving a brief overview of roles, functions, tools, interfaces and data structures comprising the framework further detailed and specified in subsequent sections.

Next, the section on **procedures** aims to formalize the federation procedures and to prepare the ground for defining the corresponding interfaces and data structures. It grounds on the roles discussed in the previous section to identify the stakeholders and entities involved in the core services and aims to enable subsequent specification of enhancements to existing interfaces, newly defined interfaces, APIs, data structures and message flows. The section addresses, for example, how to join a federation, how to acquire infrastructure and service monitoring data, or how to manage identities for authentication and authorization purposes in a federation. These procedures are discussed with a clear focus on federation or with regards to the enhancement of existing (non-federated) procedures. This list will be complemented and extended by future documents.

Then, the section dedicated to the task of **managing the federation** summarizes the concepts, tools and functional solutions chosen to operate and maintain the federation. This section aims to complement other XIFI work packages' efforts in tool development with special focus on enabling procedures, tools and interfaces: It grounds on the previous section and considers the baseline procedures discussed thereunder the objective of utilizing tools and interfaces to the federation layer for the purpose of managing federated infrastructures operations and service provisioning. Hence, this section describes a particular use case of the federation layer that, for example, enables exposing XIFI infrastructure nodes' services and resources, manages networking resources between nodes, handles service level agreements and related performance monitoring, or enables service and infrastructure maintenance, testing, fault detection and fail-over.

It is worth to recall that the tools required to realize this functionality are dished out in more detail within other XIFI work packages, as they are also part of the user portal functionality or of the XIFI infrastructure node set-up, operations and maintenance, for example.

Finally, the section providing **API and other interface specifications** aims to summarize the services and methods of the interfaces to the federation layer in a unified way. For the time being and in the scope of this document the specification is mostly informal. A more comprehensive documentation and specification is included with the software components and modules delivered.

1.2 Document scope

This document summarizes a set of tools, interfaces and functional modules complementing each other to implement formation, management and user access to the XIFI federation, including topics in the scope for operations, management and monitoring of the federation, as well as SLA management and service provisioning by the federation. Consequently, this document has a specific objective on the integration of components from different sources for the purposes of federation. The document's scope thus is best outlined by providing an overview of the components in scope, their objectives and purposes in the scope of federating the XIFI infrastructures.

- The **Federation Manager** provides the capability to query a federated infrastructure's properties such as a certain federated node's role, location, identifier(s), and contact information. It is the primary entry to the federation and is utilized by the user and management portals and by the components responsible for infrastructure and service monitoring, identity management and SLA management as well.
- The **Federation Monitoring** component implements a distributed service that facilitates multiple infrastructure nodes' local capabilities for active and passive monitoring and their computing and storage capacities to provide infrastructure and service monitoring through a single API. It federates the nodes through the use of common access methods, data storage

capacities and data models, also supporting some heterogeneity with regards to probing and processing of monitoring data.

- The **Federation Identity Management** is considered here with regards to the binding of identities, roles and regions. It considers replication and distribution of credentials as well as authentication based on federation and infrastructure node based roles including replication aspects. While the federation layer described above is focused on identifying, locating and describing infrastructure nodes, federated identity management focuses on identifying users, associating them with roles in a particular (spatial and temporal) context and associating them with proper credentials. It is providing the means for authorizing access to the federation or to individual federated infrastructure nodes. Thus, federated identity management is one of the core components of along with federated monitoring and the federation layer API.

A number of components is still under development and could not be addressed in depth by this document. In particular the following components are around the corner, further complementing the components summarized above.

- The **Help Desk Integration** with the federation layer is driven by the need for reliable operation of infrastructure nodes, supporting incident handling or degradation by directing issue reports (either automated or originated by federation users) to responsible roles. It is considered in scope, since it must support identifying issues potentially affecting infrastructure nodes as well as their network connections. It is closely linked with federation monitoring, SLA management, identity management, and network management since it has to track issues regarding their origins and point of impact. It also relates to dashboard and portal design of the federation and to the provisioning of suitable test tools creating meaningful outcomes for federated use cases, such as distributed UC trials experiencing connectivity problems.

- A number of **Test Tools** is currently evolving, mainly driven by the need to detect and handle connectivity problems, interface compliance issues and potential fraudulence or malicious use issues. Future documents will summarize these and put them in the federated use context.

- The **Federation Network Management** is currently evolving but has not yet been considered in the scope of this document (WP3 will deliver the component at M12). Considering that networking resources are at the same level of relevance as infrastructure node resources, the network management completes the federation core components (federation layer API, monitoring, identity management) and will need integration with all other components (help desk, user and management dashboards, SLA management, and test tools).

The **Federation Manager**, its user interface and its core service are developments made within the XIFI WP2. Monitoring concepts and adaptors are developed in the scope of WP3, providing the various modules required implementing the **Federation Monitoring** in the scope of WP2. The SLA management framework has been developed in the scope of XIFI WP4 and links with WP2 through its dependencies with Federation Monitoring. The **Federation Identity Management** is based on developments made in the context of XIFI WP2. The WP responsible for a certain component or test case develops specific Test Tools in general. XIFI WP3 and WP5 currently address the Federation Network Management concepts and tools.

2 FEDERATION CONCEPTS AND ARCHITECTURE

This section emphasizes on the basic concepts of the XIFI federation. The underlying architecture has been developed in the scope of XIFI WP1 based on the deployment of FI-WARE GEs [2]. In particular, the federation architecture is outlined in [5], while this section provides a complementary and federation-specific perspective on concepts, roles and architecture which are relevant in the context and scope of this document.

2.1 Actors, roles and stakeholders

In the federation of infrastructure nodes, the XIFI stakeholders (cf. [18]) can be considered as the actors playing different roles, depending on their specific task, authorization, context and objective. The following focuses on technical roles actively participating in utilizing and maintaining the federation.

The **Federator** is a role with full access to the federated XIFI infrastructure. Generally speaking, within a federation model the federator (also known as federation manager) is the individual or the component in charge of the control and the management of the federation. More in detail, within the XIFI federation model, the federator will manage the overall provisioning and deployment of resources made available, by providing a single entry point to all those federation participants having a compliant software and hardware architecture. The federation manager has read-only access to current and past monitoring data regarding all users, physical and virtual machines (VMs), services, network elements and interfaces on all federated infrastructure nodes, as well as regarding the interaction of virtual machines. Access to federated infrastructure nodes is based on the node's Region ID.

The **Infrastructure Owners** (also known as node *providers*) have full access to their local infrastructure nodes and network between virtual machines of any tenant and physical machines. This role is responsible for exposing the offerings of an infrastructure node including GEs to the federation and for the policy of utilization of their resources and components. The infrastructure owner has full access to current and past monitoring data regarding all users, physical and virtual machines, services, network elements and interfaces and the interaction of virtual machines of his infrastructure node. **Operators** are particular roles that work with infrastructure owners to keep the federated infrastructure working and available.

Technology Providers would deploy Generic Enablers and Specific Enablers in the cloud services offered by XIFI federated infrastructure and eventually connect them to underlying FI facilities.

Developers would have access to their resources on the XIFI federated infrastructure. They have read-only access to current and past monitoring data regarding their virtual machines, services and interaction between virtual machines and full access to their virtual machines and services. The **Developers** role includes the ones that will be involved in FI-PPP Phase II and III through SMEs and web entrepreneurs [3]. A Developer can request resources, can run processes and can deploy software on allocated resources only.

End User role refers to users having access to a web service exposed by the Portal or by an application registered on the Portal. This role is not directly related to the XIFI federation, since a user will not need to recognize that he is interacting with the federation. When he starts interacting with the federation, then that user will be playing one of the aforementioned roles.

Additional roles not mentioned above have to be considered further in order to discuss the challenges of federated identity management in the following sections:

Identity Providers can make assertions on other entities in relation to authorization aspects.

Infrastructure Management Authorities (IMA) (also known as Node Management Authorities, NMAs) are responsible for infrastructure components or for resources of an infrastructure node. Moreover, they are responsible for providing policies on resource allocation.

2.2 Infrastructure and Identity Federation

There are two levels of Federation to manage

- Infrastructure Federation
- Identity Federation

The **Infrastructure Federation** allows a "transparent" access to the resources and services provided by all nodes (in this context called Region) of the. As detailed in [3] and [5], the infrastructure federation is composed by two or more master-nodes and many slave nodes.

The **Identity Federation** is achieved when different identities from different domain belonging to the same user are federated so as to be equivalent each other. In order to implement this identity federation layer, protocols like SAML [39] can be used. Another aspect that should be taken into consideration, as far as the Identity Management is concerned, is the High Availability. XIFI should avoid any Single Point of Failure in the deployment of the Identity Manager so as to avoid that a crash of a node (for example a master node) can cause the impossibility to use all the services of XIFI (because the Identity Manager is unable to provide authentication and authorization functionalities). For this reason the XIFI deployment architecture foresees a "redundant" installation of the Identity Manager on each node of the XIFI federation.

All this concepts related to the Identity Management (federation and high availability) are detailed and explained in the following section.

2.3 Federated Identity Management

Federated Identity Management aims at providing a federated management of identities for members of XIFI federation. This component is distributed on all the nodes of the federation and stores identities of authorized users. The Federated Identity Management component extends the FI-WARE IdM GE to support XIFI requirements. The following figure highlights this component in the context of the XIFI architecture: the Federated Identity Management component is composed by the blocks in yellow.

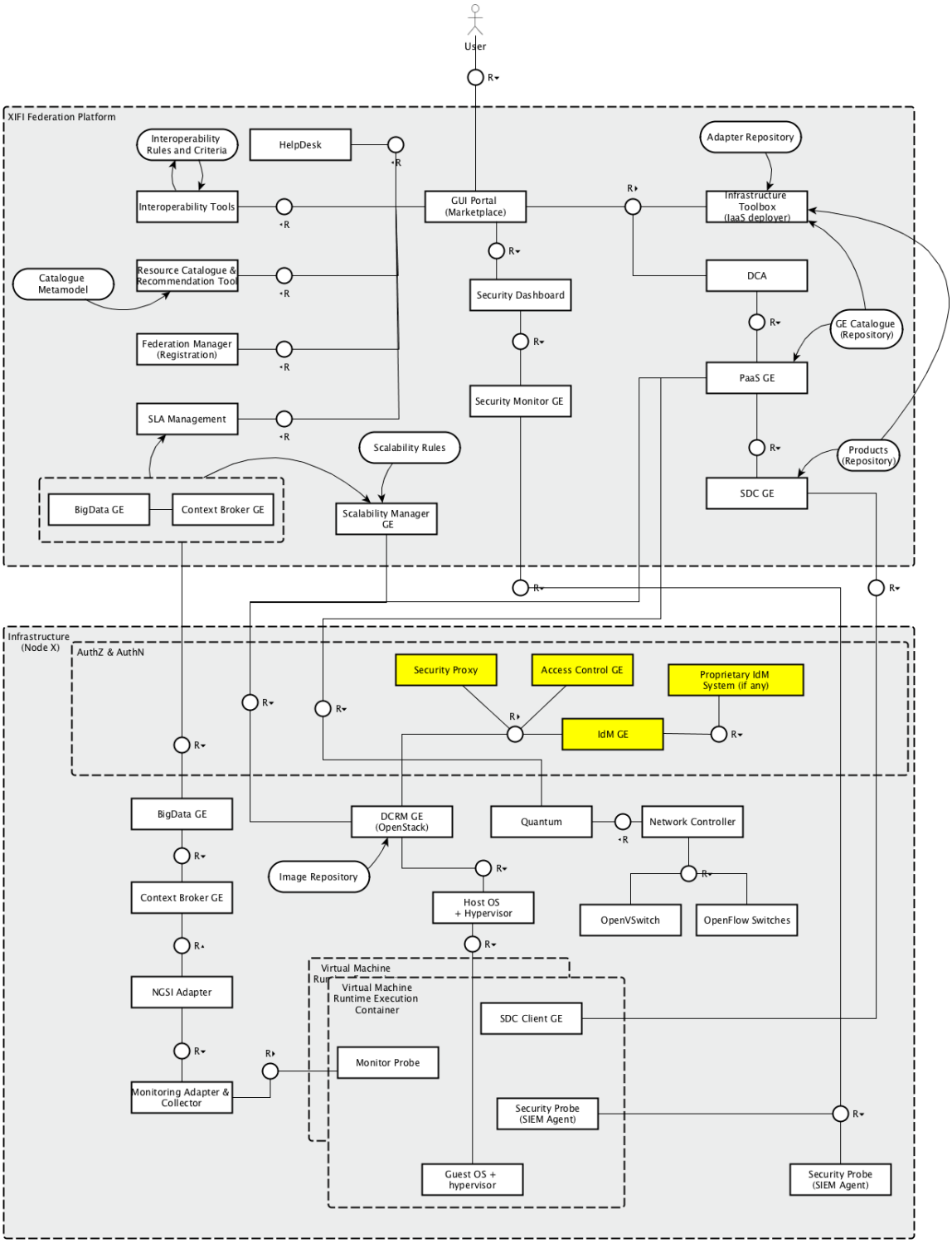


Figure 1: Federated Identity Management Context

Reference Scenarios	UC1: Joining the Federation & UC2: Setup and use of development environment
Reference Stakeholders	• Federator: allows for management of federated identities • Infrastructure owner & operator: allows to share the identities of

	the owned infrastructure with the federation • Developer: allows to authenticate to federation services in a distributed fashion
Type of ownership	Deployment and Extension
Original tool	Based on KeyRock FI-WARE IdM GEi [28]
Planned OS license	GNU Affero General Public License 3.0[29] as the original component
Reference OS community	FI-WARE

Table 1: Federated Identity Management Reference

2.3.1 Consist of

- FI-WARE IdM GE
- HA extensions
- SAML Extensions

2.3.2 Depends on

- Federation Manager

2.3.3 Component responsible

Developer	Email	Company
Carolina García	carolgarcia@dit.upm.es	UPM-DIT
N/A	N/A	ENG
Federico Facca	federico.facca@create-net.org	CREATE-NET

Table 2: Component Responsible

2.3.4 Motivation

XIFI federation is composed of different nodes (the members of the federation) and services that by nature may have their own administrative domain and existing users base. In the specific case of XIFI, an administrative domain represents one of the different organizations that are part of XIFI federation and hence in charge of one of the nodes (or regions to use a more common cloud terminology) part of the XIFI federation. Users of XIFI federation should be able to access the different services (and nodes hosting them) in a transparent way regardless their access point, so as to simplify their authentication process across the different domains. For example, a user that access resources on node A may as well access - with the same credentials - resources on node B.

One of the simplest solutions to provide federated access to resources across different administrative domains is the use of Federated Identity Management solutions.

In the case of XIFI, the IdM solution needs to be backward compatible with the IdM deployed in FI-Lab to allow for transparent adoption of existing user base of the FI-Lab node. In FI-Lab the GEi used is Keyrock, designed with the purpose to integrate Cloud Hosting (through the keystone-proxy) and

OAuth-based Apps (including as well FI-Lab portal services). Moreover, the Keyrock IdM GEi, is the only IdM GE implementation that is released with an Open Source license, in line with the openness principle of XIFI. Thus the decision taken in XIFI was to extend it, in two directions:

- Support High Availability deployment of the IdM as way to:
  - provide a distributed IdM solution for the XIFI federation (where all nodes manages all the identities in synchronized way);
  - avoid the IdM to become a single point of failure for the federation and the services hosted in the nodes, in case of non-availability of the main IdM;
- Support Identity Federation standards and in particular SAML 2.0, supporting both Service Provider and Identity Provider roles. This is in line with the original specification of the IdM GE that includes support for SAML 2.0.

Moreover, the Identity Management solution adopted in XIFI should also support for accessing federated resources. Thus the keystone-proxy, adopted in FI-Lab to integrate the Keyrock IdM GEi into the DCRM GE and to expose to the DCRM GE the service catalogue of a single DCRM node, should be extended to load the federation service catalogue from the Federation Manager.

2.3.5 User stories backlog

User story id	User story name	Actors	Description
1	Support for High Availability	Infrastructure Owner	The IDM should not be a single point of failure and each region, as regard locally deployed services should rely on a local IdM as master. A developer should be able to access services on a node even though the master node is offline.
2	Integration with Federation Manager	Federator	The IdM should rely on the service registry provided by the Federation Manager to authenticate the users access to OpenStack Resources.
3	Support for SAML 2.0 Identity Provider	Developer	The IdM should implement the server side of SAML authentication and allow IdM to act as an IdP (Identity Provider) using the SAML v2.0 protocol. Thus it should provide a mean for managing authentication requests and confirmation responses for SPs (Service Providers).
4	Support for SAML 2.0 Service Provider	Developer / Infrastructure Owner	The IdM should implement the client side of SAML authentication and allow IdM to act as an SP (Service Provider) using the SAML v2.0 protocol. Thus it should allow a user registered on another IdM system to authenticate locally if his IdM system is trusted by the local IdM.

Table 3: User stories

2.3.6 State of the art

The IdM GE is based on the identification of who can access the resources or services of Company/Enterprise through use of the cloud and virtualization technologies. As the XIFI federation based on the re-use of the technologies developed in the FI-PPP, in this context the identity management is provided by the FI-WARE Identity Management Generic Enabler (IdM GE, FI-WARE Catalogue, “Identity Management – KeyRock” [28]). The IdM GE is involved in many aspects of the XIFI federation: networks, devices, services, authorization and access in a single sign-on (SSO)

fashion to external services to the federation. In order to use the FI-WARE IdM GE in many aspects of the XIFI federation context (networks, devices, services ...) is necessary:

1) to identify an entity for any components acting within the federation as a Universal Federation Identifier (UFI) that uses:

- UUID (Universally Unique Identifier) which is 128 bits long and can guarantee uniqueness across space and time (IETF, RFC 4122, "A Universally Unique Identifier (UUID) URN Namespace", July 2005, [2]);
- ITER (composed by ID, EntityTypeID and RegionID) and can actually be a primary key for any components in the XIFI federation if it will be identified all possible type of entities in the XIFI federation.

A system of exchange Private/Public key and LifeTime information could be used to identify the resource granted and eventually violations within the XIFI federation. In this case could be user to assign security Key pairs (Private/Public keys) an Authority or more Authorities (if exist).

2)to identify a System for Cross Region Identity Management based on the SCIM Core scheme [3] as depicted in the figure below and based on E-R module. Two types of federation "Resources" should be considered in the XIFI context:

- a collection of resources that can compose another resource;
- a resource that can be part of another resource.

An agent could be composed of multiple processes running on different nodes for this reason two type of processes could be identified: one for the Resource and one for the Node. The model used considers an Agent that runs on a resource composed by a set of virtual resources (Virtual Machine, Virtual Network, Virtual Storage) belonging to a Node of a Region managed by a Master Node where an agent must know which resource is using. These resources could be managed through the composing relationship that is normally instantiated when a resource is not related to another resource. This relationship could be defined as follows:

- Responsibility: it links the Resource to the Region.
- Belonging: it links Resource to the Node.
- Membership: it links Agent to the Group.
- RunningOnNode: it links Agent to the Node (an example of this type of agent is a main software component for the Node).
- RunningOnResource: it links the Agent to the Resource (an example of this type of resource can be a virtual machine).

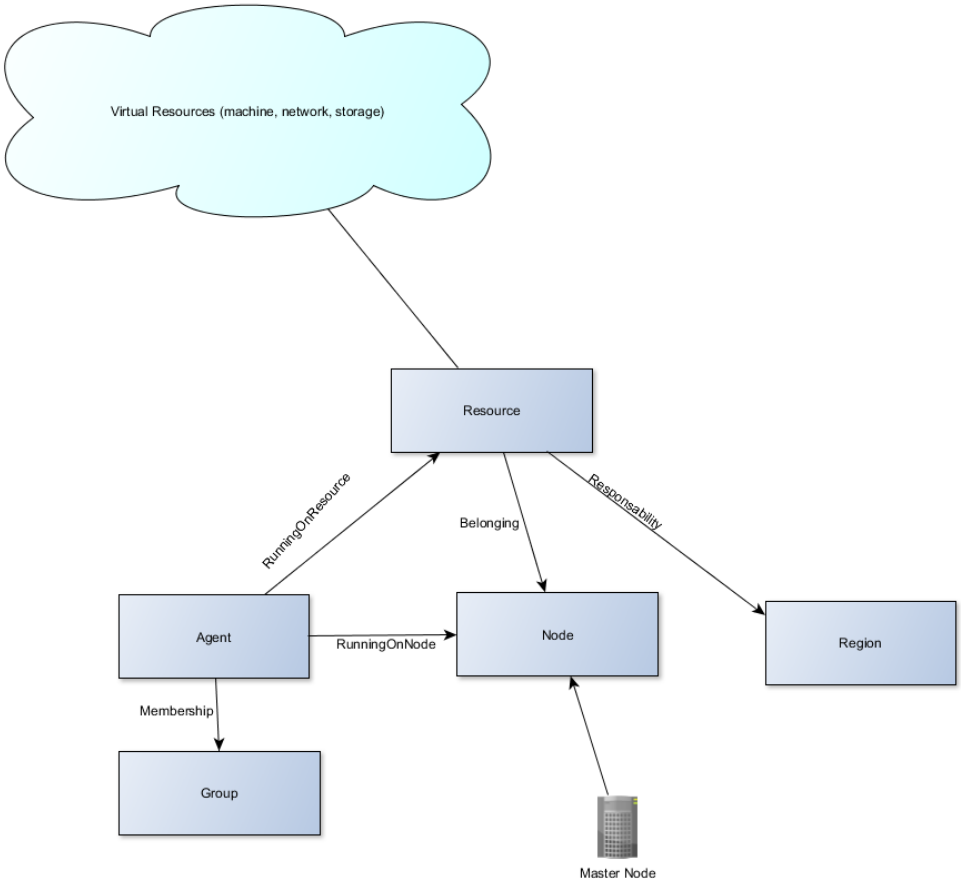


Figure 2: Composing relationship

2.3.7 Architecture design

The Federation Identity Manager component is an extension of the FI-WARE IdM GEi to:

- Support SAML 2.0 as Service Provider and Identity Provider;
- Integrate with the Federation Manager service catalogue;
- Support High Availability deployment of the IdM;

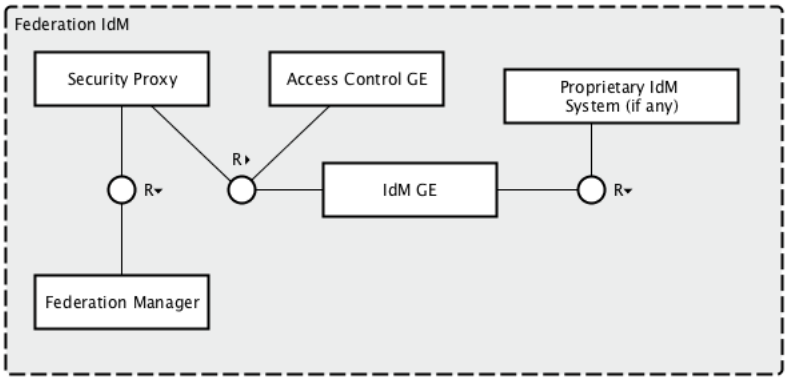


Figure 3: Federated IdM

The inclusion of SAML 2.0 support, allows the IdM for Identity Federation, and in particular to allow users to authenticate through other Identity Provider implementing the SAML 2.0 protocol such as OpenAM.

The Security Proxy, implemented through FI-WARE Keystone Proxy, a service that provide proxy authentication to DCRM GE services toward the IdM GE. In our architecture the service catalogue is retrieved from the Federation Manager.

The High Availability support is implemented through proper deployment architecture that allow multiple IdM instances to share the same database cluster - to make high available the IdM backend - and the same virtual IP - to make high available the IdM frontend. The database cluster, compatibly with the Keyrock IdM GEi architecture is based on Galera, a wsrep API implementation for MySQL. The Virtual IP service is implemented using keepalived a load balancing and high-availability IP service for Linux.

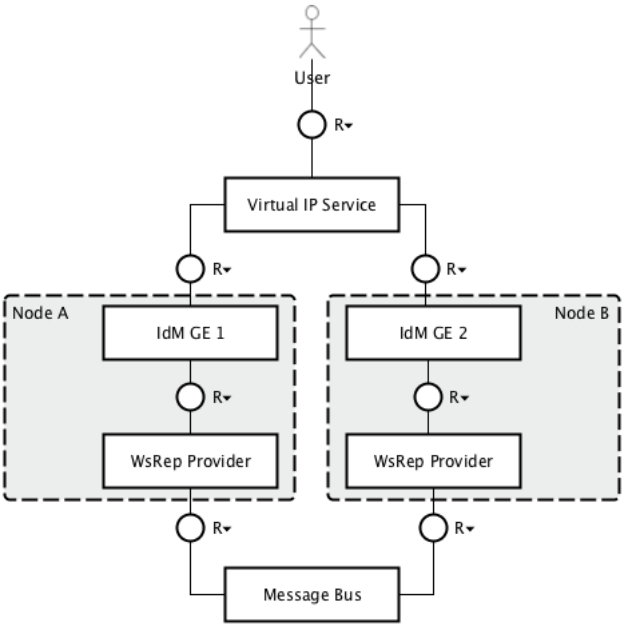


Figure 4: Federated IdM Architecture

2.3.8 Release plan

Version Id	Milestone	User Stories
1.0	30-12-2013	1
2.0	28-02-2014	2
3.0	30-06-2014	3,4

Table 4: Federated IdM Release Plan

2.3.9 Installation Manual

For the installation documentation you can refer to FI-WARE documentation [37].

The following steps are required to use the multi-region modality in high availability:

- Deploy a number of instances of Ubuntu 12.04 as the number of nodes taking part to the high availability set-up of the IdM GEs. You need at least two instances (on a real server or VM). The instances should be able to communicate each other; the easiest way is to place them in the same subnet. e.g. 10.20.0.100 and 10.20.0.101. The first node will act initially as master node, even though we will set-up a multi-master configuration.
- Install and configure MySQL Galera [40] on each instance.
- Install and configure the IdM GE on each instance.
- Install and configure keepalived on each instance.

2.3.10 Install and Configure MySQL Galera

Common settings on all instances

On every instance execute the following steps.

1. Update your packet registry:

```
$ sudo apt-get update
```

2. Execute the following to install the necessary support libraries:

```
$ sudo apt-get install libaio1 libssl0.9.8
```

3. Download Galera wsrep provider:

```
$ wget https://launchpad.net/galera/2.x/23.2.4/+download/galera-
23.2.4-amd64.deb
```

4. Install the Galera WSREP provider for MySQL:

```
$ sudo dpkg -i galera-23.2.4-amd64.deb
```

5. Install mysql-client:

```
$ sudo apt-get install mysql-client
```

6. Download MySQL server with wsrep patch:

```
$ wget https://launchpad.net/codership-mysql/5.5/5.5.28-
23.7/+download/mysql-server-wsrep-5.5.28-23.7-amd64.deb
```

7. Install the custom wsrep patched MySQL package:

```
$ sudo dpkg -i mysql-server-wsrep-5.5.28-23.7-amd64.deb
```

8. Restart MySQL:

```
$ sudo /etc/init.d/mysql restart
```

9. Connect to MySQLand remove all users without a user name:

```
$ mysql -u root -p

Enter password:

Welcome to the MySQL monitor.  Commands end with ; or \g.

...

mysql> SET wsrep_on=OFF; DELETE FROM mysql.user WHERE user='';

Query OK, 1 rows affected (0.00 sec)


mysql>
```

10. Still connected to the mysql server, add the user that will be used for global cluster variable read / write:

```
mysql> SET wsrep_on=OFF; GRANT ALL ON *.* TO wsrep_sst@'%'
```

```
IDENTIFIED BY 'wspass';
Query OK, 1 rows affected (0.00 sec)

mysql>
```

11. Make sure system-wide my.cnf contains "!includedir /etc/mysql/conf.d/" line.

Galera setup for each node

Node 1

1. Modify the bind_address=127.0.0.1 setting in /etc/mysql/my.cnf to ensure MySQL binds to the proper IP address on your system. In our case, to 10.20.0.100 and 10.20.0.101 on each of the two nodes.
2. When a new node joins the cluster it'll have to receive a state snapshot from one of the peers. This requires a privileged MySQL account with access from the rest of the cluster. Edit /etc/mysql/conf.d/wsrep.cnf and set mysql login/password pair for SST. wsrep_sst_auth is the user you added in the installation step 10 above, for example:

```
wsrep_sst_auth=wsrep_sst:wspass
```

3. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_provider option by specifying a path to provider library, in our case:

```
wsrep_provider=/usr/lib/galera/libgalera_smm.so
```

4. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_cluster_address. This is the first node in a cluster, so use:

```
wsrep_cluster_address="gcomm://"
```

5. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_node_address=10.20.0.100:

```
wsrep_node_address=10.20.0.100
```

6. Restart the MySQL node:

```
$ sudo /etc/init.d/mysql restart
```

Node 2

1. Modify the bind_address=127.0.0.1 setting in /etc/mysql/my.cnf to ensure MySQL binds to the proper IP address on your system. In our case, to 10.20.0.101 for the first node.
2. When a new node joins the cluster it'll have to receive a state snapshot from one of the peers. This requires a privileged MySQL account with access from the rest of the cluster. Edit /etc/mysql/conf.d/wsrep.cnf and set mysql login/password pair for SST. wsrep_sst_auth is the user you added in installation step 10 above, for example:

```
wsrep_sst_auth=wsrep_sst:wspass
```

3. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_provider option by specifying a path to provider library, in our case:

```
wsrep_provider is /usr/lib/galera/libgalera_smm.so
```

4. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_cluster_address. This is the an additional node in a cluster, so use:

```
wsrep_cluster_address="gcomm://10.20.0.100,10.20.0.101"
```

5. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_node_address=10.20.0.101:

```
wsrep_node_address=10.20.0.101
```

6. Restart the MySQL node:

```
$ sudo /etc/init.d/mysql restart
```

Node 1

1. Edit /etc/mysql/conf.d/wsrep.cnf and set wsrep_cluster_address to support multi-master mode.

```
wsrep_cluster_address="gcomm://10.20.0.100,10.20.0.101"
```

2. Restart the MySQL node:

```
$ sudo /etc/init.d/mysql restart
```

2.3.11 Install and Configure the IdM GE

For the installation documentation you can refer to FI-WARE documentation [37], except the MySQL installation steps.

1. Install developers libraries:

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

2. Install rvm:

```
$ curl -L get.rvm.io | bash -s stable --auto

$ . ~/.bash_profile

$ rvm requirements
```

3. Install dependences:

```
$ sudo apt-get install build-essential openssl libreadline6
libreadline6-dev \

curl git-core zlib1g zlib1g-dev libssl-dev libyaml-dev libsqlite3-
dev sqlite3 \

libxml2-dev libxslt-dev autoconf libc6-dev ncurses-dev automake
libtool bison  \

subversion pkg-config
```

4. Install ruby v2.0.0-p247:

```
$ rvm install 2.0.0

$ rvm use 2.0.0

$ rvm install ruby-2.0.0-p247

$ rvm --default use 2.0.0-p247
```

5. Install rails -v 4.0.0:

```
gem update --system

gem install rails -v 4.0.0
```

6. Install additional libraries::

```
$ sudo apt-get install --yes libmysql-ruby libmysqlclient-dev

$ sudo apt-get install --yes imagemagick graphicsmagick
libmagickwand-dev
```

7. Install a javascript engine, the latest nodejs:

```
$ sudo apt-get install --yes python-software-properties python g++
make

$ sudo add-apt-repository --yes ppa:chris-lea/node.js

$ sudo apt-get update

$ sudo apt-get install --yes nodejs
```

8. Install a mailer:

```
$ sudo apt-get install sendmail-bin
```

9. Clone the IdM GE code:

```
$ git clone https://github.com/ging/fi-ware-idm
```

10. Install the IdM GE code:

```
cd fi-ware-idm/
bundle install
```

11. Now edit database.yml and enter the password you chose:

```
cd config
cp database.yml.example database.yml
```

12. Only on node 1, create the database:

```
rake db:create
rake db:schema:load
```

13. Start the server:

```
rvmsudo rails server -p 80
```

14. Change parameters in initializers/0fiware.rb.example:

```
cd config/inizializers
cp 0fiware.rb.example 0fiware.rb
```

2.3.12 Install and configure keepalived on each instance

1. Install keepalived on all nodes:

```
$ sudo apt-get install keepalived
```

2. On on all nodes, before proceedin with configuring keepalived itself, edit the file /etc/sysctl.conf and add this line to the end of the file:

```
net.ipv4.ip_nonlocal_bind=1
```

This option is needed for applications to be able to bind to non-local addresses (ip adresses which do not belong to an interface on the machine).

3. To apply the setting, run the following command:

```
$ sudo sysctl -p
```

4. On node-1, configure the keepalived creating the file /etc/keepalived/keepalived.conf:

```
global_defs {
lvs_id IdM01 #Unique name of this IdM
}

vrrp_script check_status {
script "wget http://127.0.0.1:80"
interval 1
weight 2
}

vrrp_instance FloatIP01 {
state MASTER
interface eth1
```

```
priority 101
virtual_router_id 51
virtual_ipaddress {
                10.20.0.99 #floating IP
        }
track_script {
check_status
        }
}
```

5. On node-2, configure the keepalived creating the file /etc/keepalived/keepalived.conf:

```
global_defs {
lvs_id IdM02 #Unique name of this IdM
}

vrrp_script check_status {
script "wget http://127.0.0.1:80"
interval 1
weight 2
}

vrrp_instance FloatIP01 {
state BACKUP
interface eth1
priority 100
virtual_router_id 51
virtual_ipaddress {
                10.20.0.99 #floating IP
        }
track_script {
check_status
        }
}
```

2.4 Federation Monitoring

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. Figure 5 highlights this component in the context of the XIFI architecture: the Federation Monitoring component is composed by the blocks in yellow.

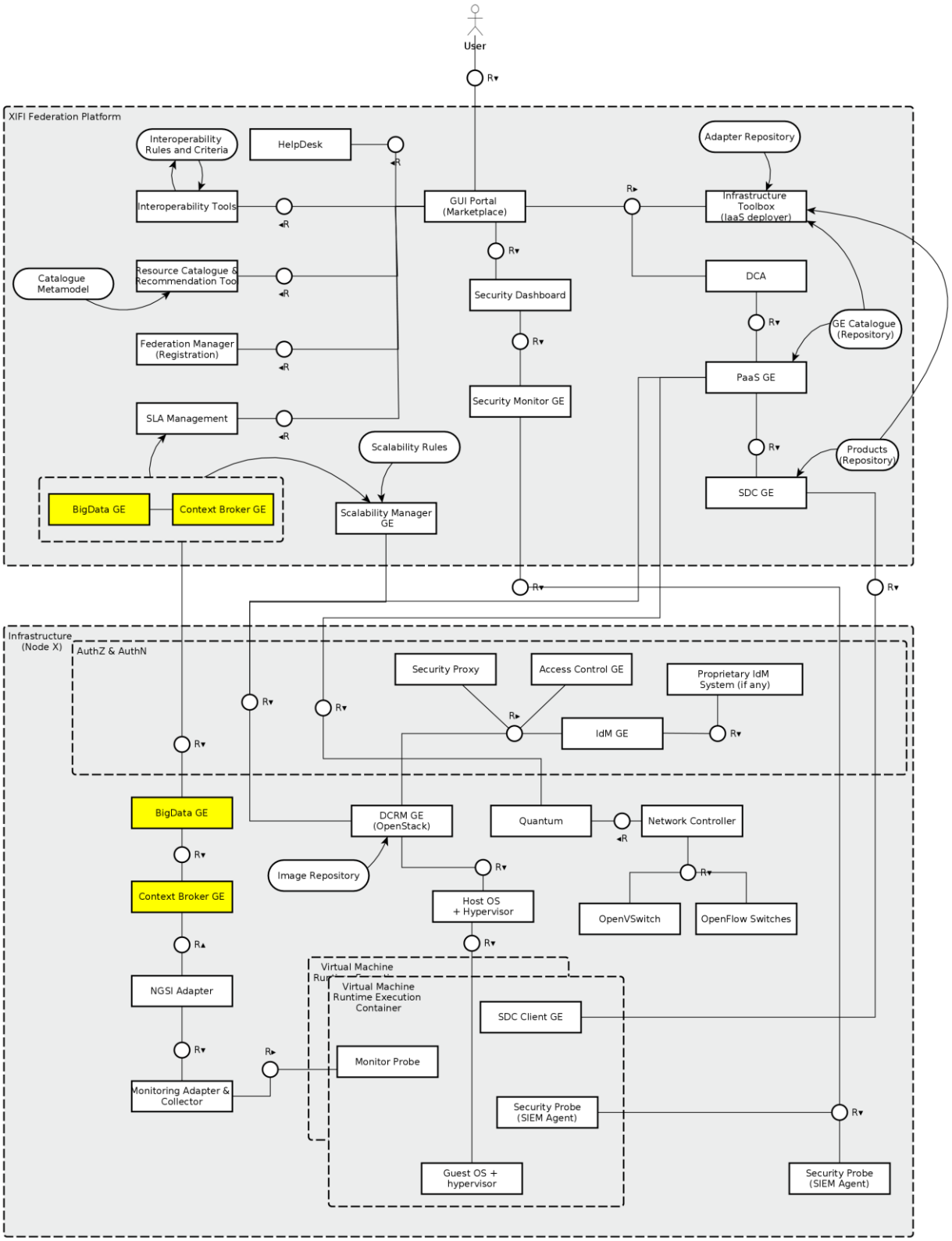


Figure 5: Federation Monitoring Architectural Context

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.
Reference OS community	None at the moment

Table 5: Federation Monitoring Reference

2.4.1 Consist of

- API specifications and implementation for accessing federation monitoring data
- Data aggregation functions implemented via MapReduce framework
- FI-WARE BigData Analysis – COSMOS (http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos)
- FI-WARE Publish/Subscribe Context Broker – Orion (http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker)
- FI-WARE ContextBroker to BigData GEs connector - ngsi2cosmos (https://github.com/telefonicaid/fiware-livedemoapp#ngsi2cosmos)
- Apache Hive (http://hive.apache.org/)

2.4.2 Depends on

- XIMM modules (Network Active Monitoring-NAM, Network Passive Monitoring-NPM, Data Center & Enablers Monitoring-DEM) – see D3.2
- FI-WARE NGSI Adapter – see D3.2

2.4.3 Component responsible

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

2.4.4 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.

2.4.5 User stories backlog

User story id	User story name	Actors	Description
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
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)
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 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
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
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
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
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
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
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
11	Provide historical	Monitoring	Federation Monitoring should be able to provide

User story id	User story name	Actors	Description
	monitoring data for each network element (at the moment only *interfaces*) of the federation	Dashboard Component	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
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
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
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.
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.
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.

Table 6: Federation Monitoring User Stories Backlog

2.4.6 State of the art

This component is based on the 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 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 datacenter. Some well-known "standards" as SNMP MIB and DMTF CIM 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.

2.4.7 Architecture design

The architectural design of this component is depicted in the following figure.

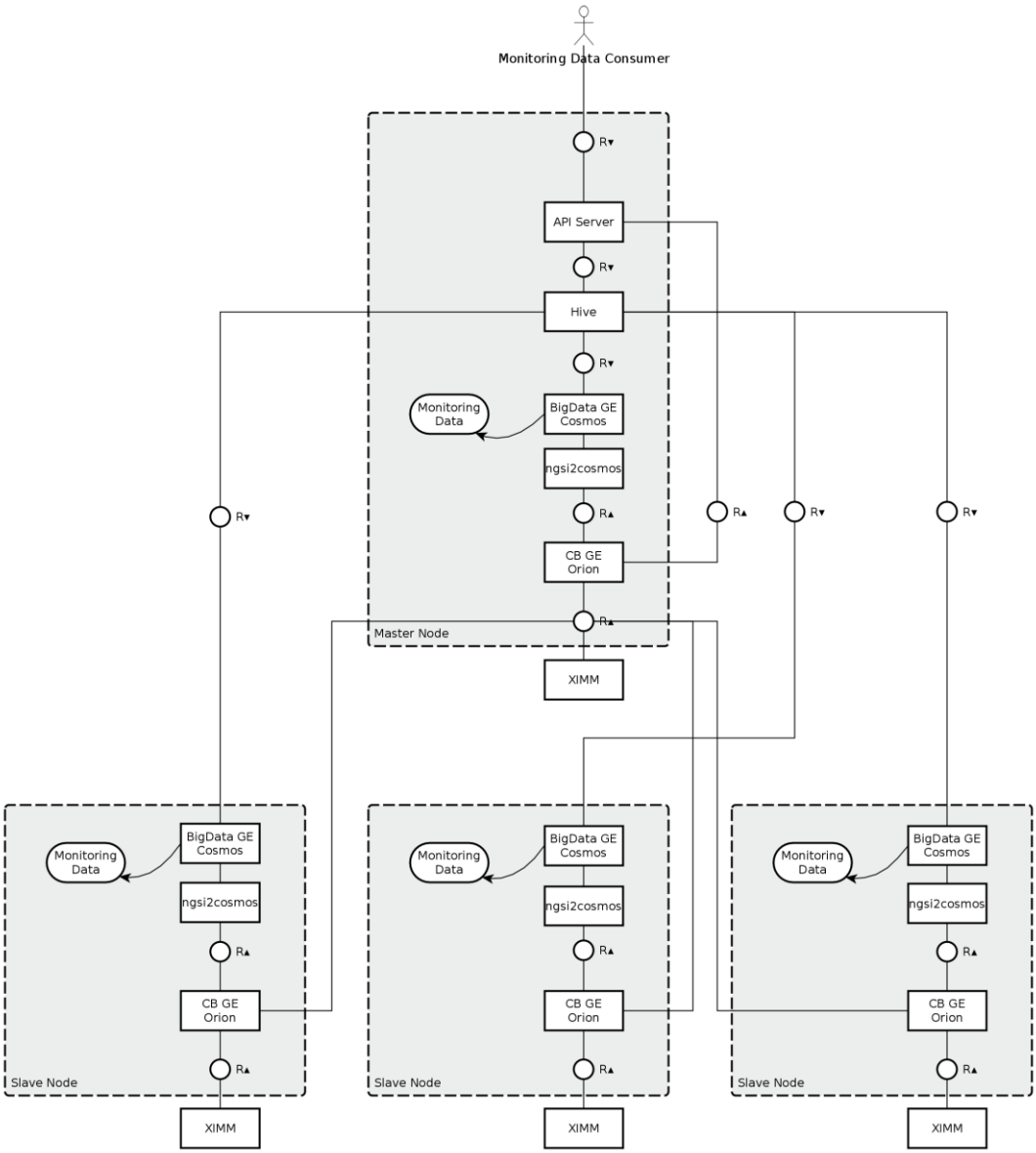


Figure 6: Federation Monitoring Architecture

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

- Context Broker GE (Orion GE)
- Big Data GE (Cosmos GE)
- Connector (ngsi2cosmos)
- Monitoring Data Storage (HDFS - Hadoop Distributed File System) where monitoring data will be stored

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

- Apache Hive: in order to query the hadoop file system monitoring data at Big Data GE

- API Server: in order to provide an implementation of the APIs for accessing monitoring data

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.

The use of Big Data GE in this architecture let us cope with huge amounts of data generated by XIMM monitoring modules. Indeed, as data size increases, the horizontal scaling allows storage, while the computational time needed to perform the data analysis keeps a linear growth, thus making such analysis feasible (e.g. within a tolerable processing time). Data warehouse tools like Apache Hive not only facilitate querying such data, but also implement some logic to perform several aggregations and data analysis on behalf of the user.

The following figure provides an overview of the monitoring data model considered. The APIs developed for accessing the monitoring data are base on this model.

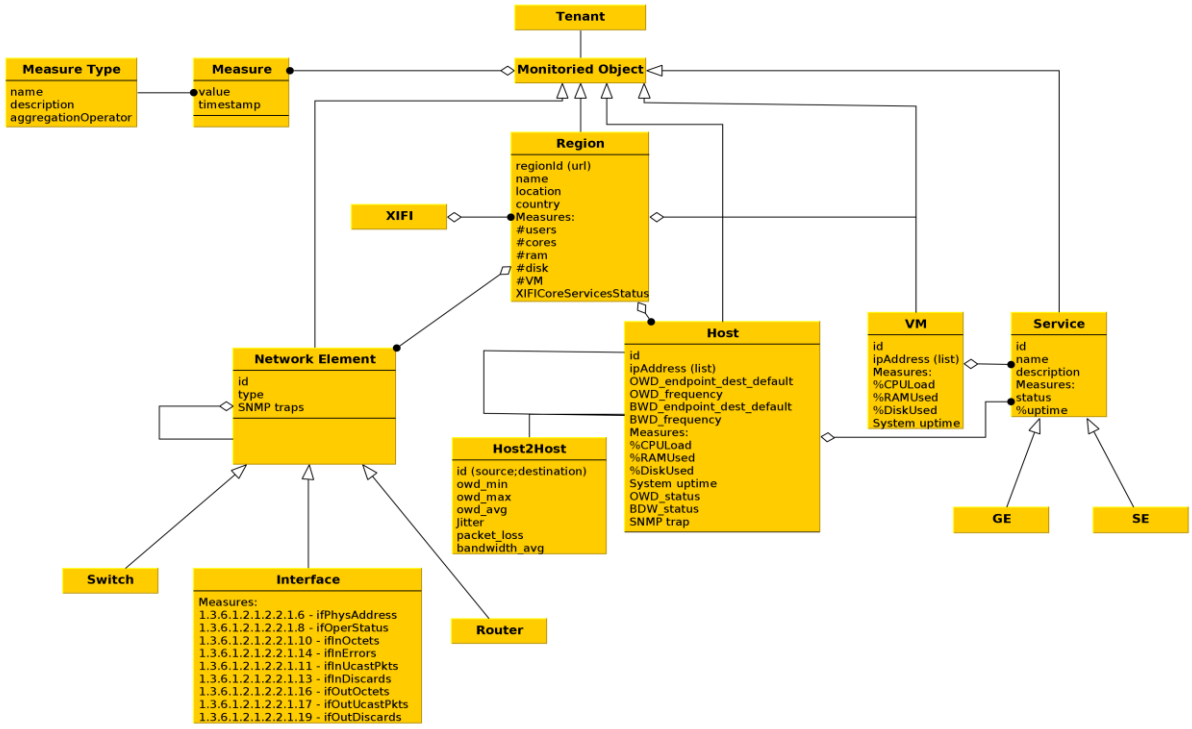


Figure 7: Federation Monitoring Data Model

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 type of operations:

- 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

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*.

2.4.8 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

Table 7: Federation Monitoring Release Plan

2.4.9 Installation Manual

Note: for M9 - Rel. 1.0 (see previous release plan) the only components that need to be install are (see below for the details):

- the Context Broker GE on each node of the federation
- Node.js on the master node
- the monitoringAPI.js on the master node

2.4.9.1 Prerequisites

- Ubuntu 12.04 as operating system
- See prerequisites for Context Broker GE (Orion - http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker) and Big Data GE (COSMOS - http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos).
- The hardware characteristics of the machine where to install a slave node is:
  - 4 cores CPU
  - 1 TB hard disk
  - 16 GB RAM
- For the master node it is suggested to increase the disk size to 2 TB.

2.4.9.2 Installation Procedure

On each node of the federation:

1. Install and run the Context Broker GE (Orion) following the instruction provided here: http://catalogue.fi-ware.eu/enablers/publishsubscribe-context-broker-orion-context-broker
2. register/federate the Context Broker present on slave nodes with the master node Context Broker (see Context Broker Federation - https://forge.fi-

ware.eu/plugins/mediawiki/wiki/fiware/index.php/Publish/Subscribe_Broker_-_Orion_Context_Broker_-_User_and_Programmers_Guide#Context_Broker_Federation)

3. Install and run the Big Data GE (COSMOS) following the instruction provided here: http://catalogue.fi-ware.eu/enablers/bigdata-analysis-cosmos. Note that:
    a. The size of the HDFS (Hadoop Distributed File System) should be determined based on (more detailed information will be provided in the next release - R2.0):
        i. the number of Monitored Objects
        ii. the number of measures for each Monitored Object
        iii. the interval among two data sample collected
        iv. the data retention
4. install the ngsi2cosmos connector. Download it from here (https://github.com/telefonicaid/fiware-livedemoapp/tree/master/package/ngsi2cosmos) and documentation can be found here (https://github.com/telefonicaid/fiware-livedemoapp#ngsi2cosmos).

Only on the master nodes:

1. install Apache Hive and connect it to each Big Data GE (COSMOS) present on the nodes
2. install Node.js
3. download monitoringAPI.js from https://xifisvn.res.eng.it/wp2/software/trunk/Components/FederationMonitoring/API/ and put it in a directory of your choice (suggestion /usr/local/monitoringAPI)
4. 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).
5. the MapReduce functions are collected in the file mapreduce.tar.gz. Download mapreduce.tar.gz from https://xifisvn.res.eng.it/wp2/software/trunk/Components/FederationMonitoring/MapReduce and uncompress it in a directory of your choice (suggestion /usr/local/mapreduce). Then configure BigData (COSMOS) GE to get MapReduce functions from the previous directory.

2.5 Federated SLA Management

In the XIFI federated environment, the component SLA Manager described in detail (i.e. functionalities, technical description, GUI and API) in D4.1 [10] and is complemented in D4.2 [11], is responsible for providing the mechanisms for the service level agreements management through all the SLA lifecycle. To understand how this SLA management is implemented in the federation, itis necessary to identify the processes associated to the service lifecycle which are mainly composed by the discovery, deployment and managing. Hence this section will complement this, focusing more on the management of the service from the point of view of the service level agreement process.

There are three main actors that will interact in the process. They are part of the federated environment and they are described in D1.3 [5] (cf. also section 2.1):

- **User**: He wants to discover the services, register on them in order to use them (indicating what he wants to monitor), and manage his services.
- **Federation Member (infrastructure)**: is a service provider in our federated environment. They have to configure, deploy and manage the services which have been published.
- **Federator**: He is responsible for providing all the necessary processes and tools in order to cover the deploying and managing of the services.

The component has been based on the WS-Agreement specification[16], so the definition of the metadata, the structure of the documents and the components will be based on the standard specifications.

We can consider the following steps to describe the process:

- The available service should be registered and published in the federated environment. The service provider is responsible for registering the service (manual or automatically) and, if it necessary, to associate the SLA Template (Agreement Template), where it is described the metrics which can be monitored for this exposed service.

  The template details the metrics and the operations that the service provider can monitor through the federation. These definitions are provided by the grammar definition or the specific operation implementation which should be defined by the Federator Members and the Federator actor. The service provider has the decision to associate a template to this service in order to create an agreement for this services and user. Thus, the service provider can create this association when the service is being created or later.

  This template is defined following the WS-Agreement specification [16] and the API to create templates is described in section 5.

  The following modules are involved:

  - Resource Catalogue & Recommendation tool: It is responsible for managing the offering for this service provider and it helps the user to discover the services.
  - Federation Manager: It is responsible for managing and providing the information of the Federation Member.
  - Federation Monitor: In this phase, it provides the available metrics for this Infrastructure in order to be included in the template.
  - SLA Manager: It is the main component to manage the creation, store and gather the template.

- After the user has discovered the service, the system shows an agreement form, when he is registering in this service (if there is one), in order to be filled by the user. Hence, he will be able to introduce the boundaries of the metrics that he wants to apply for this service. In this first release, a real negotiation between user and the infrastructure does not exist, so the created agreements will be more what the user wants to monitor. The SLA Manager will follow up these metrics and raise a violation notification, if they are not fulfilled.

  As it is not the intention to include a real negotiation, the template agreement (offered by the service provider) and the final agreement document (determined by the user) will be almost the same. The difference it is just that the user has introduced the boundaries of the guarantee terms to define the agreement.

- The API description to create, store and obtain the agreement is detailed in D4.2[11].

  On the other hand, these components have been designed to be modular and flexible following the standard in order to increase the functionalities in the next releases.

  Involved modules:

  - Resource Catalogue & Recommendation: It integrates the agreement form which the user has to fill with the boundaries.
  - SLA Manager: It is responsible to manage the agreement phase and start the enforcement of the agreement's guarantee terms.

- Once defined the agreement, we have the associated guarantee terms (composed by the metrics the value and the operation to be applied) and the SLA Manager component can collect the metric data through the Federation Monitoring, in order to detect when a violation occurs. This violation may be displayed by the user or the infrastructure owner. The user wants to see the quality of service that he has defined previously, and the Infrastructure Owner wants to see the status of their service agreements. From the point of view of the Infrastructure it is not a real agreement, so the violation either. This data is only informative and they have

not a contractual value, hence it is not required actions to avoid them since the values have been introduced by the user unilaterally.

Involved modules:

- SLA Manager: It is responsible for managing the violation in the enforcement phase.
- Federation Monitoring: It is responsible for collecting all the metric information, and providing them to the SLA Manager which raises a violation when the guarantee terms are not fulfilled.

The following Figure 8 shows the detailed scheme of the XIFI Reference Architecture and the location of the SLA Manager (see yellow box).

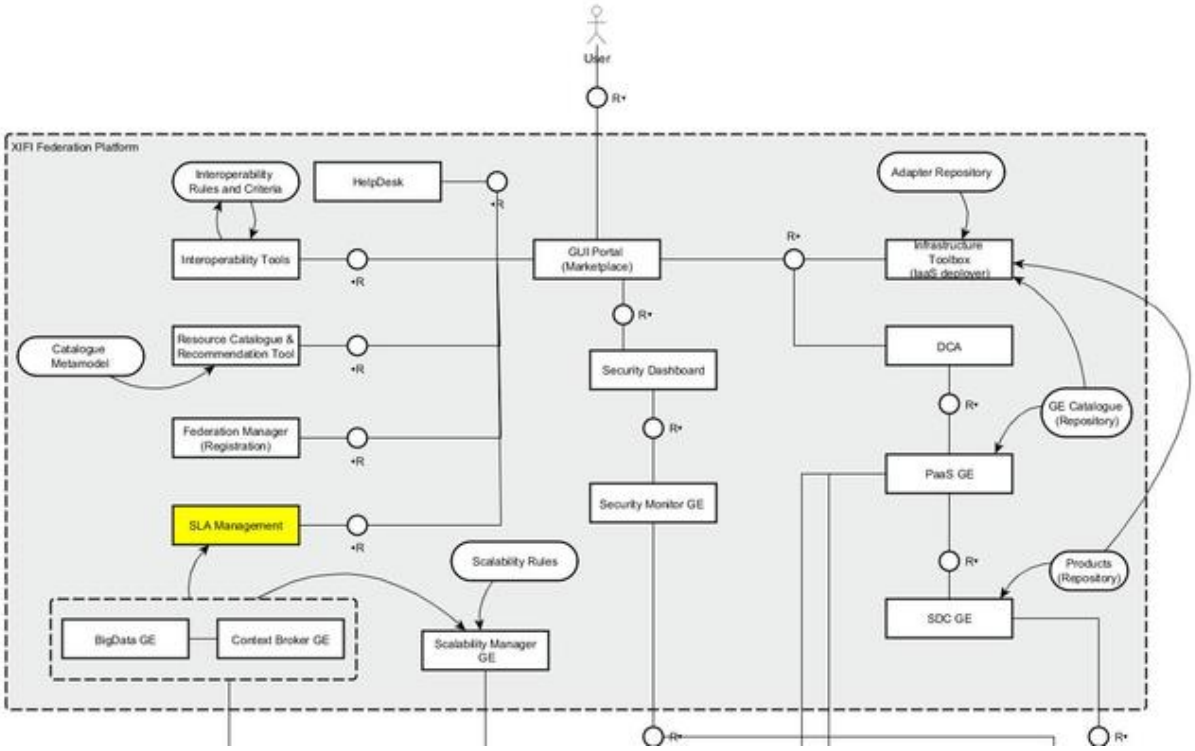
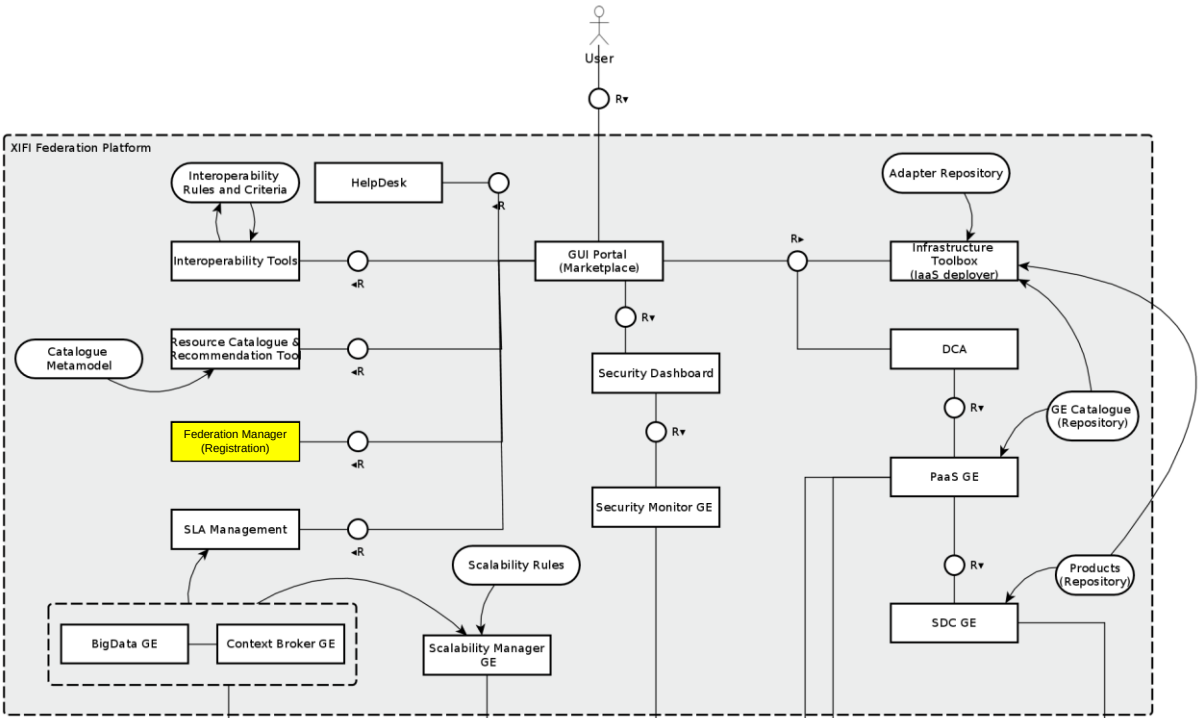


Figure 8: Federation SLA Manager in the XIFI Reference Architecture

N.B. This component is fully described in D4.2[11], thus here we provided only additional information on stakeholders and federated SLA management lifecycle as described above.

2.6 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. It consists of two parts, the Federation Manager GUI and the Federation Manager Core.



Reference Scenarios	UC-1 Joining the Federation
Reference Stakeholders	Infrastructure owners Federator
Type of ownership	New component
Original tool	None
Planned OS license	Apache License Version 2.0.
Reference OS community	None

2.6.1 Consists of

- Federation Manager Core
- Federation Manager GUI

2.6.2 Depends on

- Federation IDM
- Quick Online Test
- Infrastructure Toolbox

2.6.3 Component Responsible

Developer	Email	Company
Daniel Nehls	daniel.nehls@tu-berlin.de	Technische Universität Berlin

Alexander Willner	alexander.willner@tu-berlin.de	Technische Universität Berlin

2.6.4 Motivation

The XIFI architecture follows a centralized approach in providing the federation. 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.

2.6.5 User stories backlog

User story id	User story name	Actors	Description
1	Infrastructure Owner registers	Infrastructure Owner	The IO registers at the GUI
2	Infrastructure Owner commits Quick Online Test	Infrastructure Owner	The IO uses the GUI to commit the results of the Quick Online Test
3	List Infrastructure Join Requests	Federator	The Federation Administrator can list all Infrastructures and their state in the "join the federation" process
4	Download of ITToolbox	Infrastructure Owner	The IO can download the ITToolbox using the GUI
5	Results of Installation Tests	Federator	The Federation Admin can view the results of the installation tests performed by the ITToolbox
6	Registration of Infrastructure Federation Services	Infrastructure Owner	The Infrastructure Owner provides endpoint information for its infrastructure's services
7	Remote Testing	Federator	Federation Admin can trigger remote tests on the new infrastructure and review results
8	REST Interface	Federator	Federation Admin can use the Federation Manager's REST interface
9	Authentication with IDM	Infrastructure Owner, Federator	Federation Admin and Infrastructure Owner authenticate via KeyRock IDM

2.6.6 State of the art

Federation of infrastructures can be achieved by manifold architecture models (see D1.1), 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 shall touch on testbed registry mechanisms in three important federation projects.

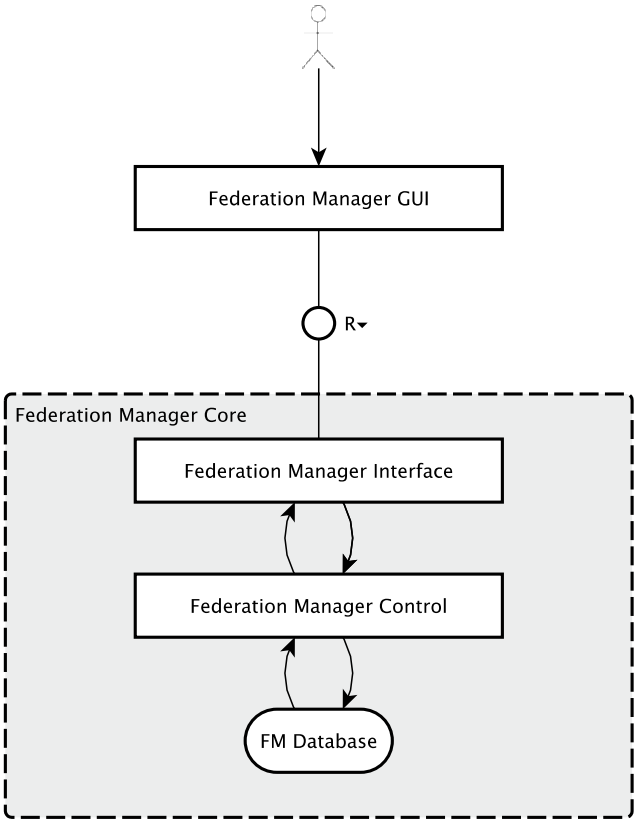
- **Planetlab[54]** is listed exemplarily for projects run by GENI [55]. Planetlab aims to be an „open platform for developing, deploying, and accessing planetary-scale services“ [54]. It unites independent testbeds which agreed on common APIs for federation based on the Slice-based Facility Architecture (SFA) [56]

Access to the federation registry component is provided by the GENI portal [57], which builds an user interface on top of the GENI Clearing House API [58].
The Clearinghouse provides a collection of services: Federation Services, Authorization Services and Accountability Services. For this analysis only a subset of the Federation Services is relevant, namely the Service Registry.
The Service Registry provides 'yellow pages' of URL's pointing to the aggregate managers [59] of participating testbeds.

- **Fed4FIRE[60]** is a project funded by the European Union's FP7, aiming at building a federation of infrastructures launched in the context of FIRE [61]. Fed4FIRE uses the same approach as GENI using SFA as common interface for the infrastructures. Similar to GENI the infrastructures are mostly independent with own groups of users etc. The Fed4FIRE portal takes the registry component part. It makes use of the MySlice [62] software on top of SFA to list testbeds and provision resources.
- **BonFIRE[63]** is another EU funded project in the context of FIRE and the one comparable the most with the aims of XIFI but is focused on experimentation mainly. BonFIRE provides a "multi-site cloud-based" platform "on top of six infrastructure testbeds" [64].
The registry is represented by the BonFIRE portal and a REST API, based on OCCI to list locations (the BonFIRE term for nodes/regions) and provision resources on these.

2.6.7 Architecture Design

As stated above the Federation Manager Component consists of two parts.



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.

2.6.8 Release plan

Version Id	Milestone	User Stories
1.0	M9	1, 2, 3
1.1	M10	8
1.2	M12	4,5
1.3	M14	6,7
1.4	M15	9

2.6.9 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.

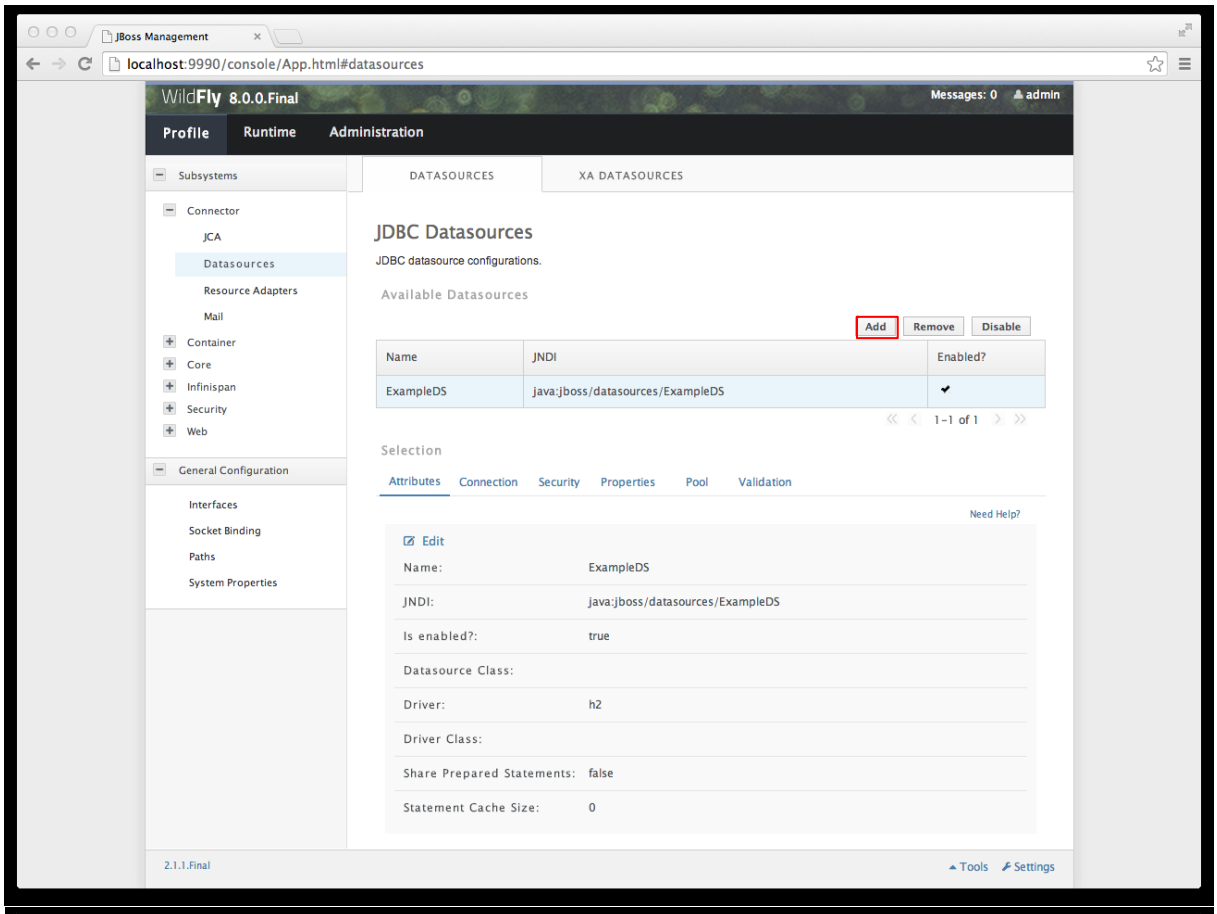
2.6.9.1 Prerequisites

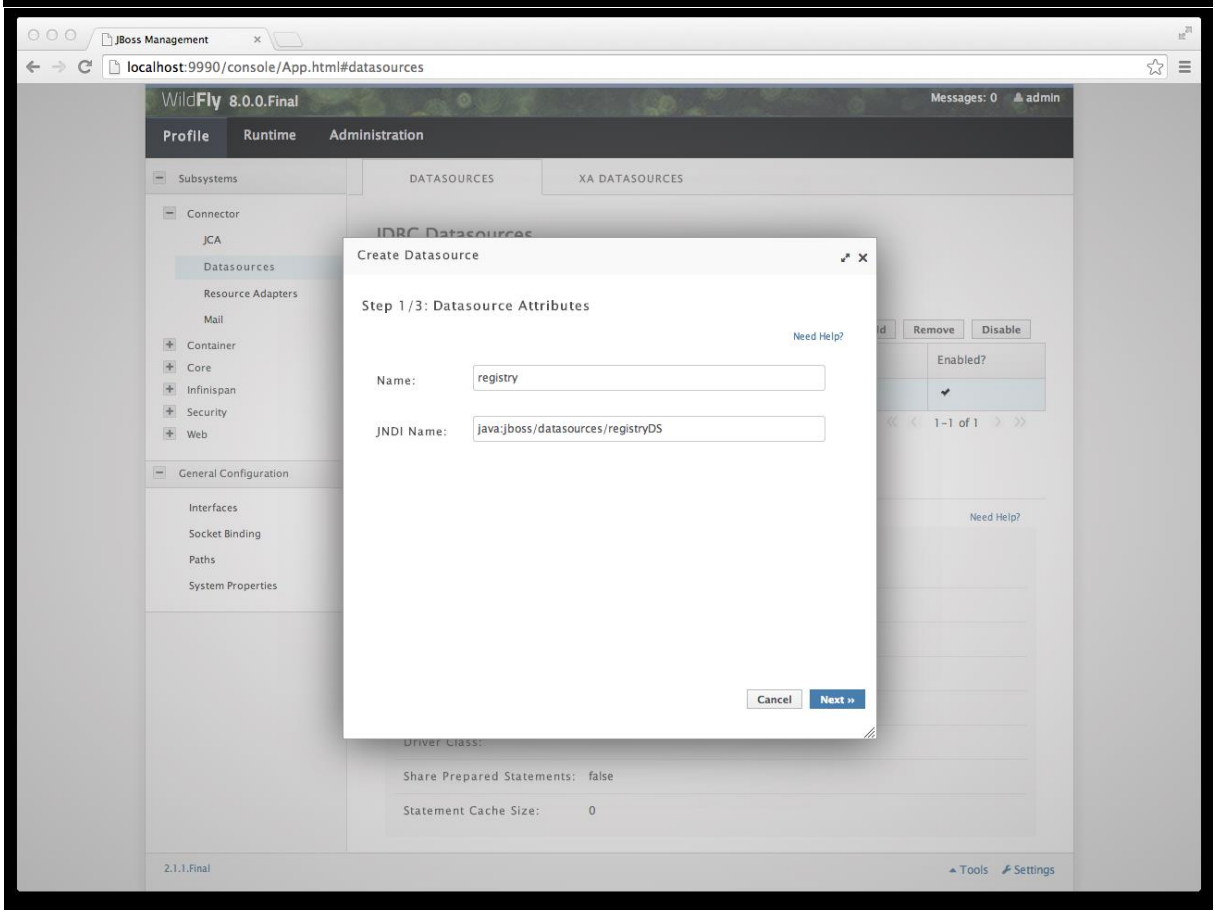
- Java 7
- Maven
- WildFly 8 AS [67]
- Federation Manager source code

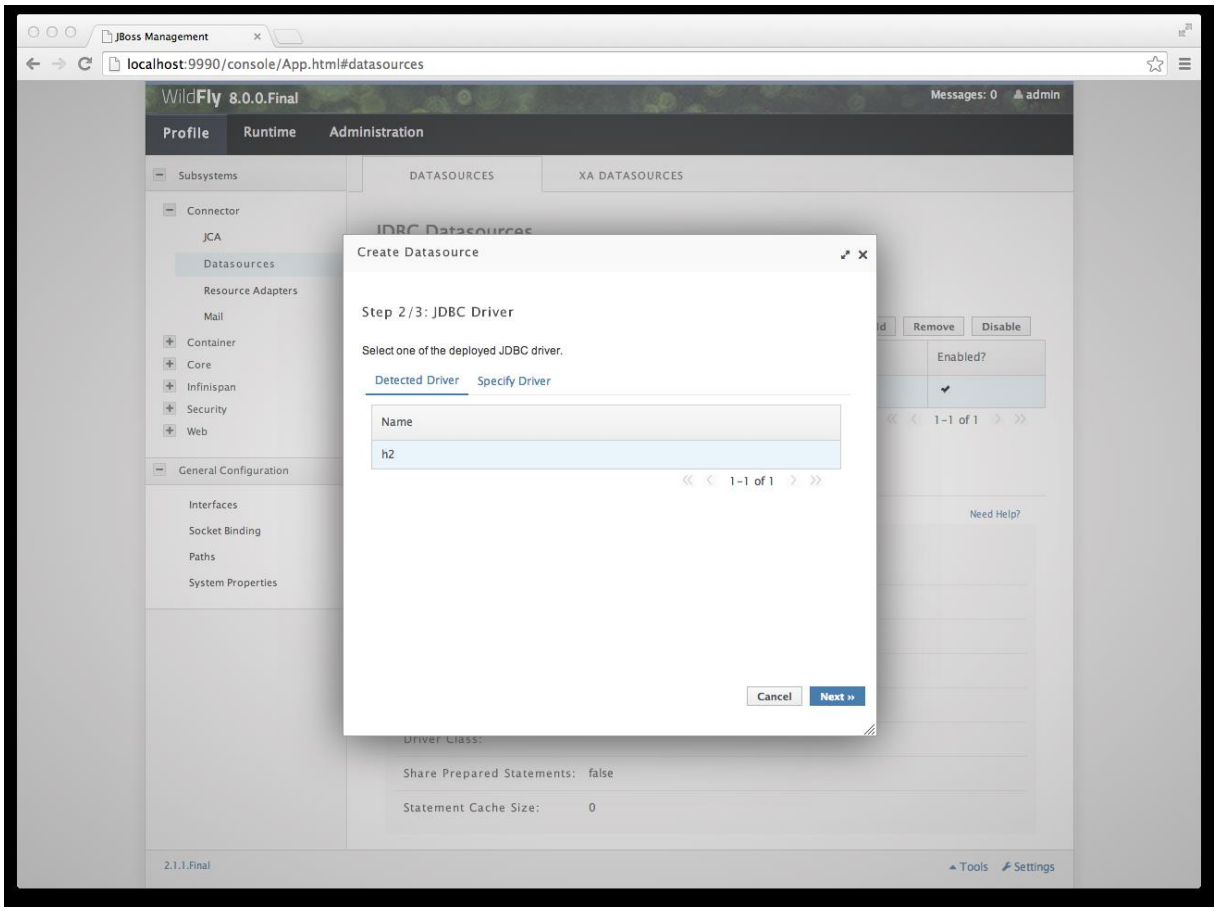
2.6.9.2 Installation procedure

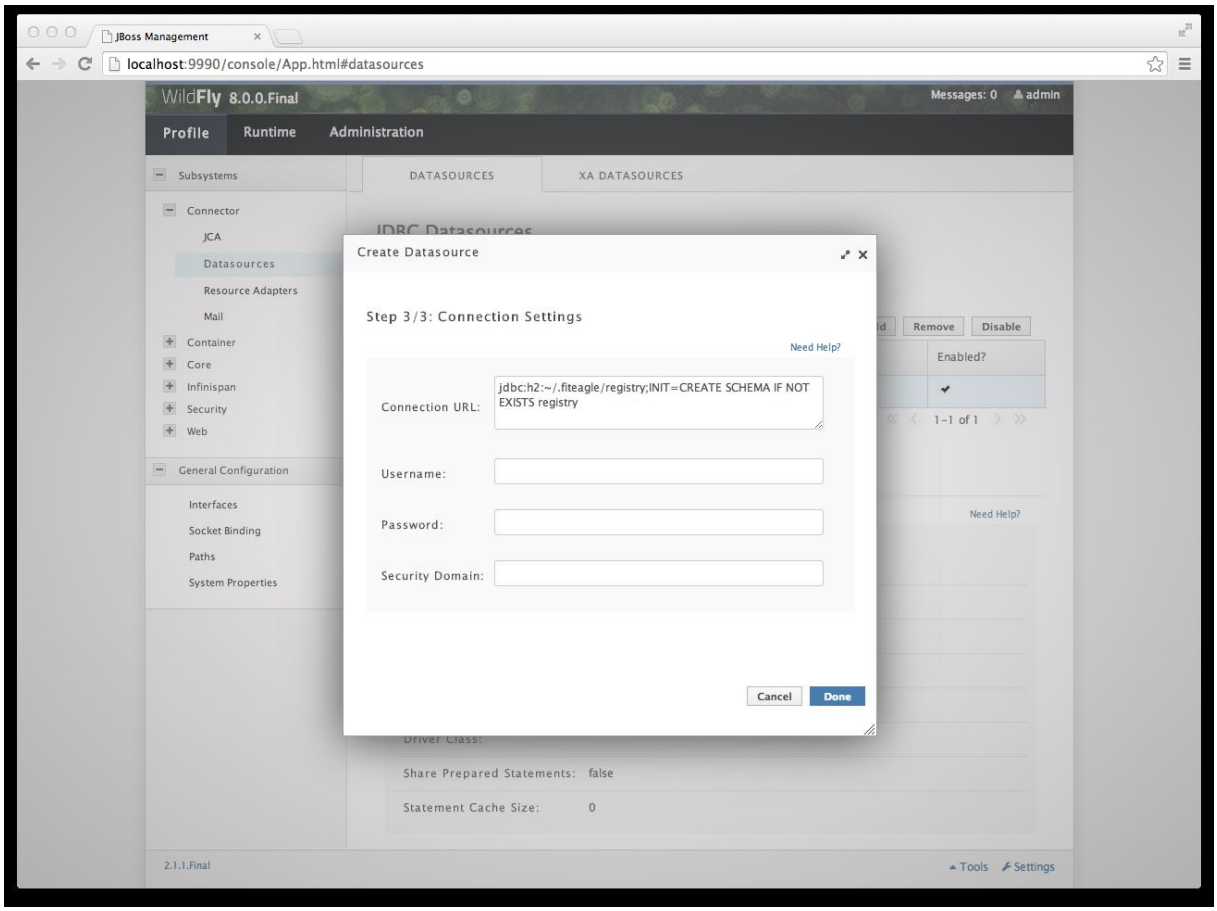
- Before running WildFly for the first time an adminstrator user has to be created:
  .<wildfly_installation_folder>/bin/add-user.sh

  and follow the instructions prompted.
- run WildFly AS:
  .<wildfly_installation_folder>/bin/standalone.sh
- log into WildFly AS at *localhost:9990* with username and password provided during the step above
- under the *Profile* tab **add** a new datasource:


JBoss Management
localhost:9990/console/App.html#datasources
WildFly 8.0.0.Final
Messages: 0
admin
Profile
Runtime
Administration
Subsystems
Connector
JCA
Datasources
Resource Adapters
Mail
Container
Core
Infinispan
Security
Web
General Configuration
Interfaces
Socket Binding
Paths
System Properties
DATASOURCES
XA DATASOURCES
JDBC Datasources
JDBC datasource configurations.
Available Datasources
Add
Remove
Disable
Name
JNDI
Enabled?
ExampleDS
java:jboss/datasources/ExampleDS
1-1 of 1
Selection
Attributes
Connection
Security
Properties
Pool
Validation
Need Help?
Edit
Name:
ExampleDS
JNDI:
java:jboss/datasources/ExampleDS
Is enabled?:
true
Datasource Class:
Driver:
h2
Driver Class:
Share Prepared Statements:
false
Statement Cache Size:
0
2.1.1.Final
Tools
Settings


JBoss Management
localhost:9990/console/App.html#datasources
WildFly 8.0.0.Final
Messages: 0
admin
Profile
Runtime
Administration
Subsystems
Connector
JCA
Datasources
Resource Adapters
Mail
Container
Core
Infinispan
Security
Web
General Configuration
Interfaces
Socket Binding
Paths
System Properties
DATASOURCES
XA DATASOURCES
Create Datasource
Step 1/3: Datasource Attributes
Need Help?
Name:
registry
JNDI Name:
java:jboss/datasources/registryDS
Cancel
Next »
Share Prepared Statements:
false
Statement Cache Size:
0
2.1.1.Final
Tools
Settings


JBoss Management
localhost:9990/console/App.html#datasources
WildFly 8.0.0.Final
Messages: 0
admin
Profile
Runtime
Administration
Subsystems
Connector
JCA
Datasources
Resource Adapters
Mail
Container
Core
Infinispan
Security
Web
General Configuration
Interfaces
Socket Binding
Paths
System Properties
DATASOURCES
XA DATASOURCES
Create Datasource
Step 2/3: JDBC Driver
Select one of the deployed JDBC driver.
Detected Driver
Specify Driver
Name
h2
1-1 of 1
Cancel
Next »
Remove
Disable
Enabled?
Need Help?
Share Prepared Statements: false
Statement Cache Size: 0
2.1.1.Final
Tools
Settings


JBoss Management
localhost:9990/console/App.html#datasources
WildFly 8.0.0.Final
Messages: 0
admin
Profile
Runtime
Administration
Subsystems
Connector
JCA
Datasources
Resource Adapters
Mail
Container
Core
Infinispan
Security
Web
General Configuration
Interfaces
Socket Binding
Paths
System Properties
DATASOURCES
XA DATASOURCES
Create Datasource
Step 3/3: Connection Settings
Need Help?
Connection URL:
jdbc:h2:~/.fiteagle/registry;INIT=CREATE SCHEMA IF NOT EXISTS registry
Username:
Password:
Security Domain:
Cancel
Done
Remove
Disable
Enabled?
Need Help?
Share Prepared Statements: false
Statement Cache Size: 0
2.1.1.Final
Tools
Settings

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)
- Run
  mvnwildfly:deploy

  to compile and deploy the Federation Manager Webarchive

3 PROCEDURESTO ACCESS AND OPERATE SERVICES OF THE XIFI FEDERATION

This section focuses on procedures to access and operate services of the XIFI federation. It is strongly related to operating XIFI infrastructure nodes (cf. [12]) and complements the infrastructure owner perspective by a user perspective. Since partially depending on practical experience, this set of procedures will be expanded and elaborated further in upcoming documents.

3.1 Accessing the federation

In the following, the process of accessing the federation that was described already in [6] in a high level manner will be exemplified.

After registration at the Federation Manager GUI, the infrastructure owner has to fill in a compliance survey (cf. Figure 9, Figure 10) based on the Quick Online Test component ([27], please note that this is currently non-public but is referenced here for the sake of completeness). In this survey the infrastructure owner declares legal and technical compliance to the terms of the XIFI federation and provides contact information (e.g. management and support contacts).

When the infrastructure owner has completed the survey, the new request is shown to the federation administrator. The federation administrator now can view the details provided and approve the request.

This approval activates the next step for the infrastructure owner (cf. Figure 11, Figure 12): the download of the infrastructure toolbox (ITBox) which is described in detail in [9].

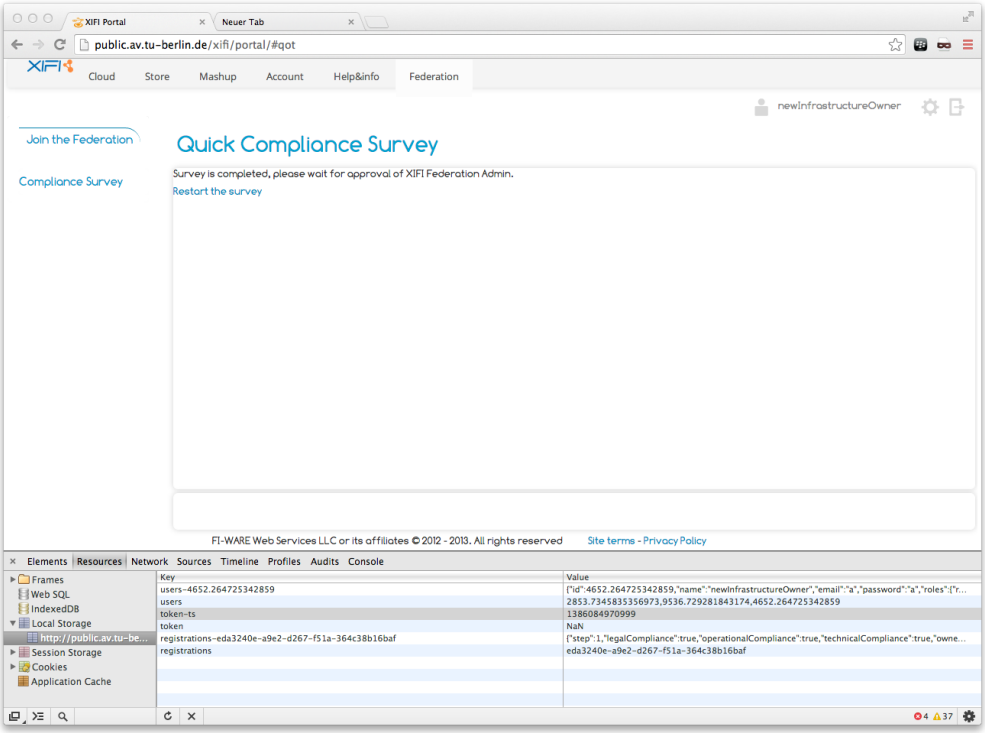


Figure 9: Federation Manager GUI - Infrastructure Owner Compliance Survey

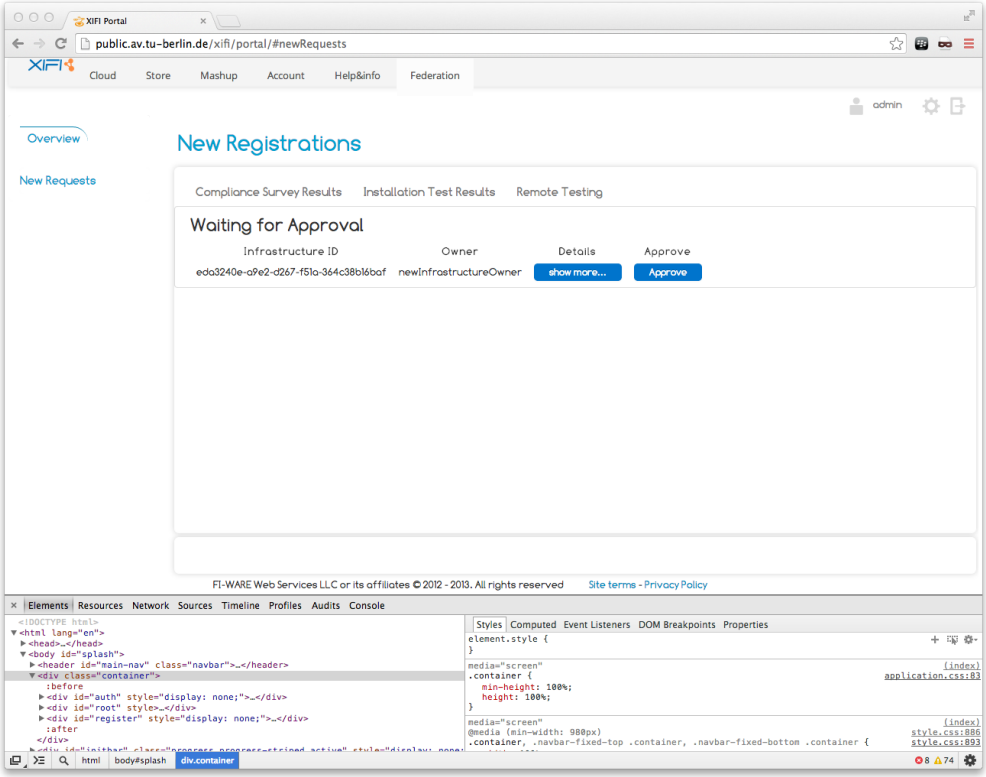


Figure 10: Federation Manager GUI - Federation Admin New Requests

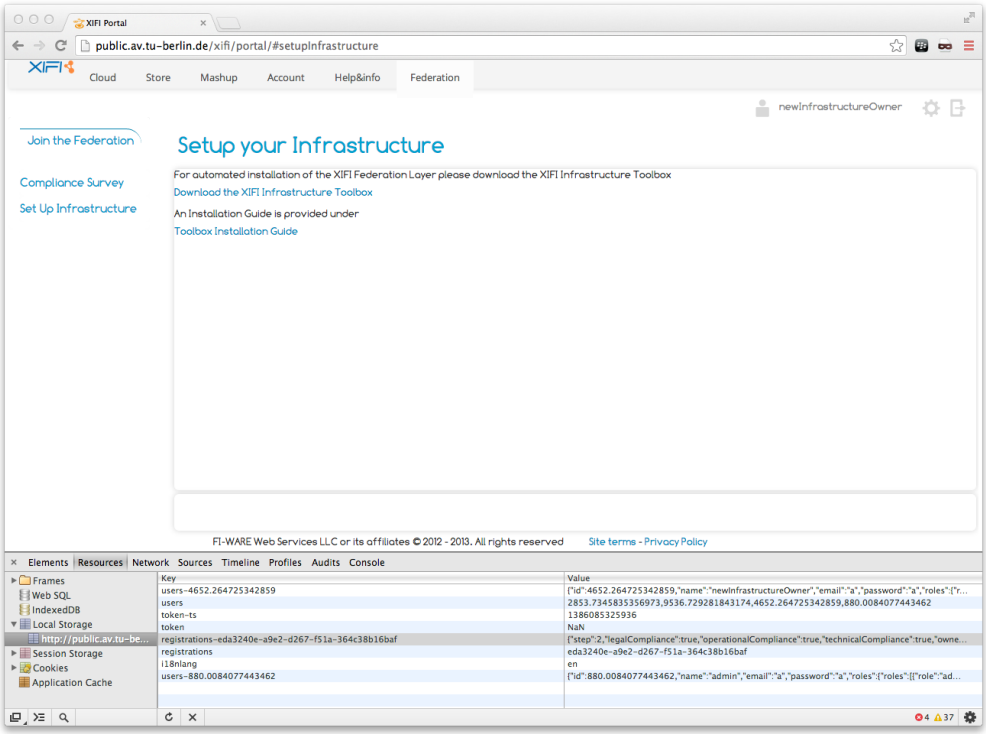


Figure 11: Federation Manager GUI - Download of the Infrastructure Toolbox

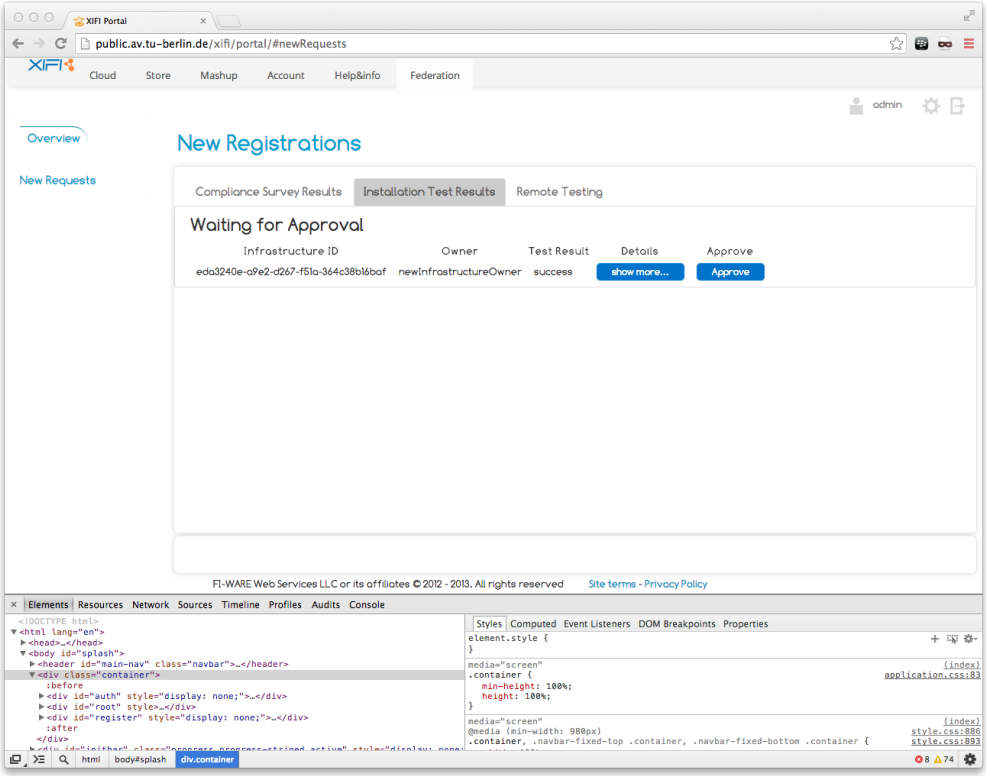


Figure 12: Federation Manager GUI - Infrastructure Toolbox Test Results

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 on going and will be further detailed in upcoming documents.

3.2 Locating and allocating resources

Prior to deploying Generic Enablers, the verification of the available resources of a tenant, as well as the Region of XIFI federation that these resources belong to is mandatory. These tasks present challenges in a federated environment, such as XIFI.

Typical challenges include:

- The specification of the required resources for the deployment of one or more GE(s). This, though not federation-specific, is a presupposition for formulating the query on the identification and location of the resources.
- The capability of asking a master (or entry or proxy) federation entity about the availability of resources in the federation and in the constituent clouds. For security reasons (on behalf of the Telco infrastructure) we do not foresee further elaboration of the available resources at federation level or per constituent cloud.
- The capability of selecting (based on proprietary application - specific criteria) the constituent cloud(s) that can accommodate the GE(s). While resources are considered equivalent in the constituent clouds, criteria for selection can be the geographic location, the availability of resources or factors related to policies (such as security or billing policies).
- The capability of allocating (in the sense of committing) the needed resources in the selected cloud(s). The allocation of resources is tightly related to the deployment of the GE(s).

The related procedure may pertain not only to the clean deployment of a GE but also to re-

deployment, e.g. due to migration purposes (the migration itself is beyond the context of this section).

While these are the main steps of the procedure as perceived by the application / service provider, who is interested in deploying the GE(s) in the federation, the procedure has further (background) challenges for the federation infrastructure. Specifically:

- The master (or entry or proxy) federation entity should have prompt access to updated and valid information, regarding the availability of resources in the constituent clouds. It is not necessary (or even desirable for scalability as well as "separation of concerns" reasons) to manage the full set of information; rather it may collaborate with the master (or proxy /entry) entities of the constituent clouds.
- In the hierarchical tree-like structure, the master (entry / proxy) nodes of the constituent clouds should have consistent information on the available resources, updated in real time. These nodes may keep a registry of the resources allocated for the deployed GEs.

3.3 Obtaining use case results

All systems need to obtain detailed information about the resources and services that have been exposed. The federation will offer to the user a homogenized way of showing all the monitored data, identifying the origin (resource or service) where it was generated in order to be able to take decision on them and the adopted federation type (which is defined in [5]). This visibility is important not only for the user, but also for the infrastructure owners and the Federator.

We could differentiate the following interested actors:

- The Application Developers, who want detailed information of the resource and service usage which have been registered and whether they have filled the conditions of the quality that they desire.
- The Infrastructure Owners and Operators, who are also interested on knowing the status of their resources and if they fulfil the quality parameters offered.
- The Federation Manager, who is interested on knowing the status of the services that are deployed and published in the federated environment in order to manage them.

In an isolated system, the compilation of this data is somehow straightforward, since the service provider decides and controls what type of data and how to recover them. Then, he can unilaterally select the best option, without having to reach consensus. On the contrary, this task is more complicated in a federated environment, because each service provider has its own tools and data to monitor.

To standardize the different environments and to converge to a common data structure, it is necessary to introduce adapters on different nodes, as described in [7][9], enabling the federated environment to group and collect this information through federation monitoring (cf. section 4.1).

The Federation Monitoring component will provide all the required information to be shown in a unified way, both the user and the service provider. The main components, which will use this information, are (cf. [11]):

- Infographics and Status Pages will use this information to show the status of the federated environment to the user, in a unique entry point through the portal.
- Resource Catalogue will use this information to show to the user the status of the infrastructure and the services exposed by the federation. The objective of this component is not to show the details, but rather to provide useful information to the user when he is discovering the exposed service of the federation.
- Monitoring Dashboard is responsible to show all the collected data of the different services and infrastructures. Its objective is to show the details of the actual status and the history of all

the available metrics.

- SLA Manager is responsible to manage the quality of the service that the user wants to maintain. Hence, it should collect all the defined metrics though the federation monitoring in order to identify eventual violations of the services the user requested.

4 MANAGING THE FEDERATION

This section provides an initial overview of tools and solutions required to operate and maintain the federation. The section is meant to provide hints toWP5 (focusing on infrastructure node maintenance) from a federation tool and service components perspective, complementing the procedures described in the previous section. In line with the current status of the activities in the project, the section focuses on monitoring aspects, and its integration with the Help Desk.

4.1 Monitoring

In principle, monitoring is a fundamental part of every infrastructure in order to expose monitoring services and measurements, either to the infrastructure owners (willing to monitor their physical/virtual resources to ensure their operational health and availability), either to the users (willing to monitor their allocated resources to evaluate performance and validate SLAs).

Up to now, the state-of-the-art monitoring solutions on cloud environments mainly target homogeneous, single-entity administered cloud infrastructures. Nevertheless, the trend (followed also by FI-PPP) is moving towards federated service cloud architectures. However, due to the high heterogeneity of such infrastructures in terms of resources, tools, procedures, etc., the concept of federation monitoring should tackle a number of challenges.

In particular, taking into consideration the architecture diagram depicted in [3], the requirements set by the Phase II Use Case projects, as described in [4], the results given in [6] and the initial analysis arising from the work performed in WP3 [7] and WP4 [10], the XIFI federated service cloud monitoring solution must address the following issues:

1. Adaptation of already deployed monitoring tools with XIFI solution
2. Integration of adapters with FI-WARE Monitoring GE
3. Federation of monitoring data
4. Authentication/authorization for accessing service monitoring data
5. Integration of service monitoring with XIFI Marketplace

4.2 Monitoring Adapters

The adaptation of the deployed monitoring tools will collect measurement data, process it and pass this information to the monitoring GE in the federation layer. This data should be exposed through an outward facing API.

Monitoring tools deployed to the infrastructure nodes as components or self-contained monitoring sub-systems are:

- NGSI Adapter
- NAM - Network Active Monitoring
- NPM - Network Passive Monitoring
- DEM - Datacenter & Enablers Monitoring
- Openstack Data Collector Adapter

The adaptation of the deployed monitoring components would be implemented as such. All of the monitoring modules would have to communicate with an "adaptation layer" from which the data would be processed and requests could be handled and responded to from the components that made up the federation monitoring GE as shown in Figure 13. The tool that will act as this adaptation layer is the XIMM. A summary also exists in the XIFI handbook [6].

This section will cover what these adaptation components of the XIMM do, rather than how they work.

4.2.1 NGSI Adapter

The NGSI adapter is the component that takes in the raw monitoring data collected by the monitoring component present in the XIMM. This collected monitoring data is standardised and published to the Federation Monitoring GE's context broker. The standardised format is NGSI format. The full details on this software component, including the summary, consists and depends on, motivation, user stories, installation guide can be found in [8].

4.2.2 Network Active Monitoring Adapter

The adaptation of the NAM module in the XIMM would need to collect the monitoring data, standardise it and publish the monitoring data from the active monitoring component, which will be exposed to the monitoring GE via the adaptation interface. This type of monitoring will establish an end-to-end communication connection. The NAM will be deal with several main metrics. These include bandwidth(throughput), latency(delay) and error rate between the end-to-end connections being monitored. NAM is based on the process of packet injection into active networks and determining the performance of these test packets as they traverse the network. The nature of the traffic packets (volume, size of the packets, type and other metrics) are fully modifiable and exchangeable. And to this extend allows the module to provide any test necessary. Using this module requirements contained in terms and conditions such as Quality of Service (QoS) and Service level agreements (SLA's) can be ensured. This module is a two-peer transaction component and therefore cannot succeed without correct configuration of both endpoints. The full details on this software component, including the summary, consists and depends on, motivation, user stories, installation guide can be found in [8].

4.2.3 Network Passive Monitoring Adapter

The adaptation of the NPM module in the XIMM would need to perform the same necessary steps (collect, standardize and publish) in order to facilitate the provision of passive monitoring data in the XIMM, which will also be exposed to the federation monitoring GE through the aforementioned adaptation interface. The NPM is valuable part of the XIMM and important in the task of troubleshooting as the packets that are monitored are real network traffic on the network.(Not the inject test packets present in the NAM). The use of this module does not alter the packets travelling across the network but rather monitors the on-going connections. This monitoring traffic is monitored periodically via external measurement tools and the data store in data storage. Due to the nature of this passive monitoring module it does not instantiate an end-to-end communication point but rather monitor specific points on the network and works independently. The full details on this software component, including the summary, consists and depends on, motivation, user stories, installation guide can be found in[8].

4.2.4 Datacentre & Enablers Monitoring Adapter

The adaptation of the DEM (Data centres & enablers) module in the XIMM would need to perform the same necessary steps (collect standardize and publish) in order to facilitate the provision of physical hardware monitoring data in the XIMM. The DEM is responsible for gathering data for physical hardware as well as VM's and the services deployed within the datacentres of the XIFI federation. The collected monitoring data gathered from this process is also required in order to evaluate the performance of the GE's deployed in XIFI federation. Several monitoring tools are implemented to

this task such as Nagios, OpenNMS and Zabbix. This hardware and service monitoring data will also be exposed via the aforementioned adaptation interface to the federation monitoring GE. The full details on this software component, including the summary, consists and depends on, motivation, user stories, installation guide can be found in[8].

4.2.5 Openstack Data Collector Adapter

The software component can collect the capacity information of an OpenStack installation running on an infrastructure by accessing the Nova API and keystone interfaces. Such parameters collected include the number of cores, amount of Ram and disk space available to use. As well as VM,s deployed the resource allocated and the number of users on the installation. The full details on this software component, including the summary, consists and depends on, motivation, user stories, installation guide can be found in [8].

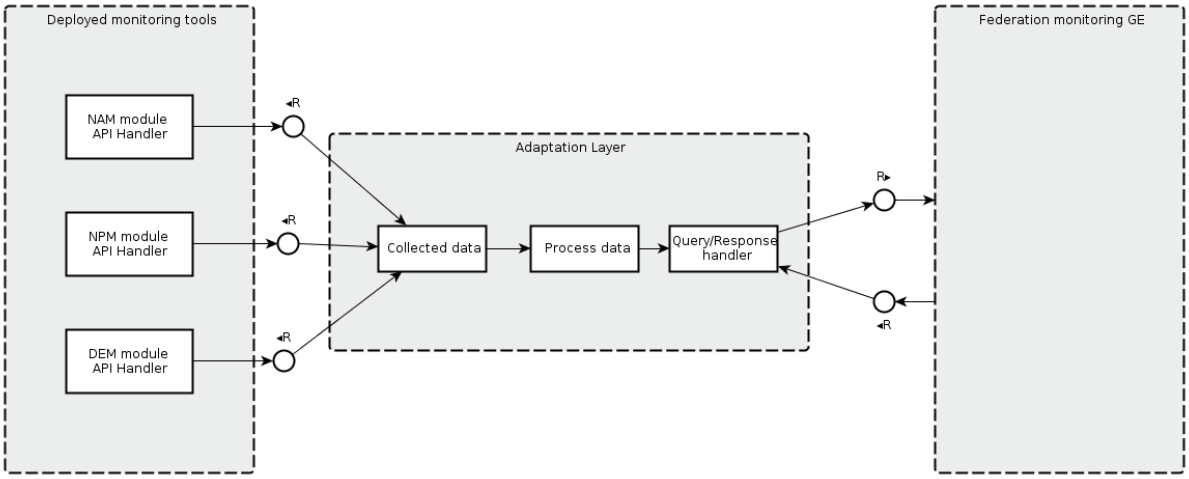


Figure 13: Federation Monitoring Adaptation Layer

4.3 Integration of Monitoring and Help Desk

A Helpdesk provides to end user (developer, service provider or in general a user of the federation) the means for requesting support, issuing a new ticket and so on. A view, integrating the monitoring of an infrastructure is one of the services that can be provided within a Helpdesk. It is this service that this chapter is aimed to respond in order to provide it to XIFI.

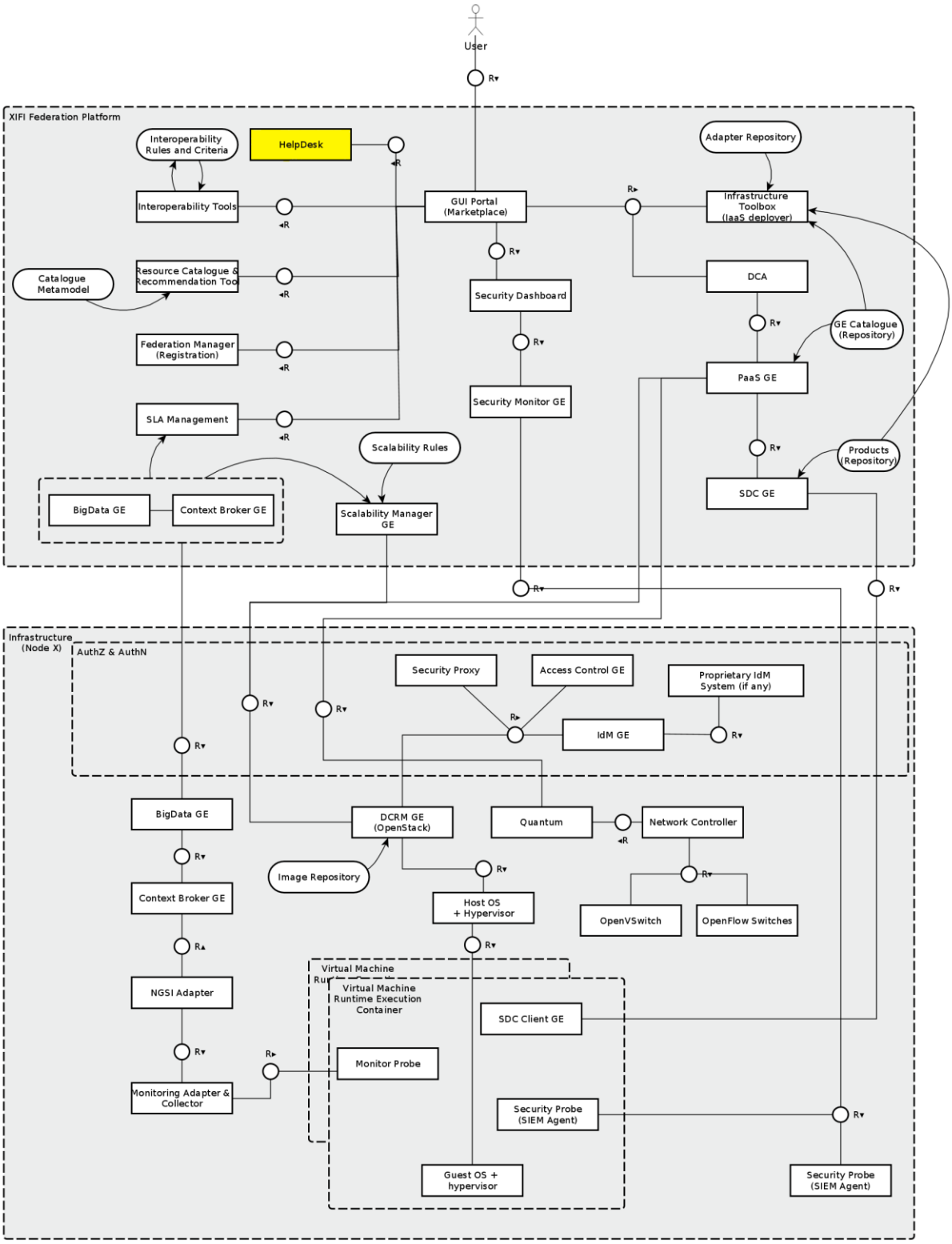


Figure 14: Integration of Monitoring and Help Desk Architectural Context

Reference Scenarios	UC-5 - Network and Data Centre operations.

Reference Stakeholders	This component is aimed to provide monitoring data useful to Infrastructure Operators that use the Help Desk facility.
Type of ownership	Deployment.
Original tool	Nagios for Jira
Planned OS license	The original tool, required in Jira, is not Open Source.
Reference OS community	None

4.3.1 Consist of

- Nagios for Jira

4.3.2 Depends on

- Jira ticketing tool, developed by Atlassian Software Systems [23]
- Nagios monitoring tool, installed on the different nodes

4.3.3 Component responsible

Developer	Email	Company
N/A	N/A	ILB

The responsibility is only for deployment.

4.3.4 Motivation

Jira is a ticketing tool used for bug tracking, issue tracking and also for project management. It is a commercial software whose price depends on the number of users, but it can be obtained for free by Atlassian in some specific cases (e.g. for open source projects).

Moreover, Jira is the ticketing tool that was already adopted by other projects like FI-LAB/FI-WARE. XIFI is going to share with them the tool and add supplementary licenses if needed.

The Jira based Helpdesk is extensible and many extensions or plugins are developed for it. In the XIFI context it is under discussion to support the extension called “Nagios for JIRA”[24]that integrates Nagios status information in a Jira instance.

4.3.5 User stories backlog

User story id	User story name	Actors	Description
1	Access node status information for the single nodes from the help desk	Infrastructure Operator	The infrastructure operators supporting FI-Lab users (as part of the XIFI federation) should be able to access from the help desk tool the status information of their node.

Table 8: Integration of Monitoring and Help Desk User Stories Backlog

4.3.6 State of the art

On the time being, "Nagios for JIRA" is the only extension that was found, integrating external monitoring system based on Nagios into JIRA. In front of this fact, a brief state of the art analysis was performed to explain the selection. It was a "leave or seize" choice.

4.3.7 Architecture design

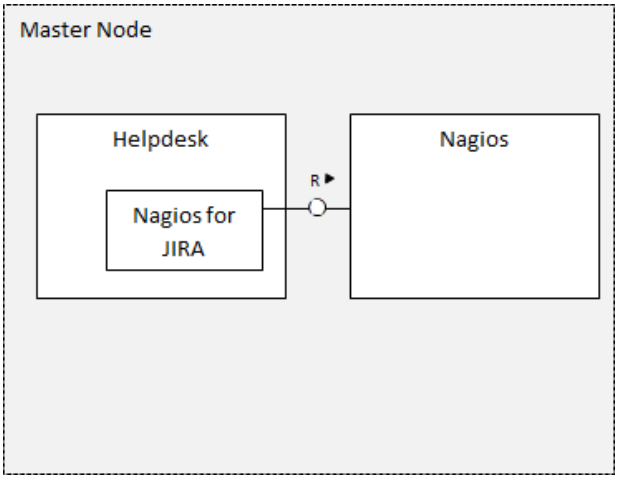


Figure 15: Integration of Monitoring and Help Desk Architecture

4.3.8 Release plan

This extension is not developed within XFI consortium and is not responsible of its release plan. The actual release of the extension is supposed to fulfil our minimum requirements.

Version Id	date	User Story
1.1.3	02/08/2013	1

Table 9: Nagios for JIRA Release

4.3.9 Installation Manual

4.3.9.1 Prerequisites

- Having a Jira version in between 6.0 and 6.1.7

4.3.9.2 Installation Procedure

The procedure described below is extract from Atlassian website [24]

1 Log into your JIRA instance as an admin.

2 Click the admin dropdown and choose Atlassian Marketplace.

The Manage add-ons screen loads.

3 Click Find new add-ons from the left-hand side of the page.

4 Locate Nagios for JIRA via search.

The appropriate add-on version appears in the search results.

5 Click Try free to begin a new trial or Buy now to purchase a license for Nagios for JIRA.

You're prompted to log into MyAtlassian. Nagios for JIRA begins to download.

6 Enter your information and click Generate license when redirected to MyAtlassian.

7 Click Apply license.

8 After the configuration of a Nagios Monitoring server, “Nagios for JIRA” will raise alarms as shown on the screenshots in Figure 16.

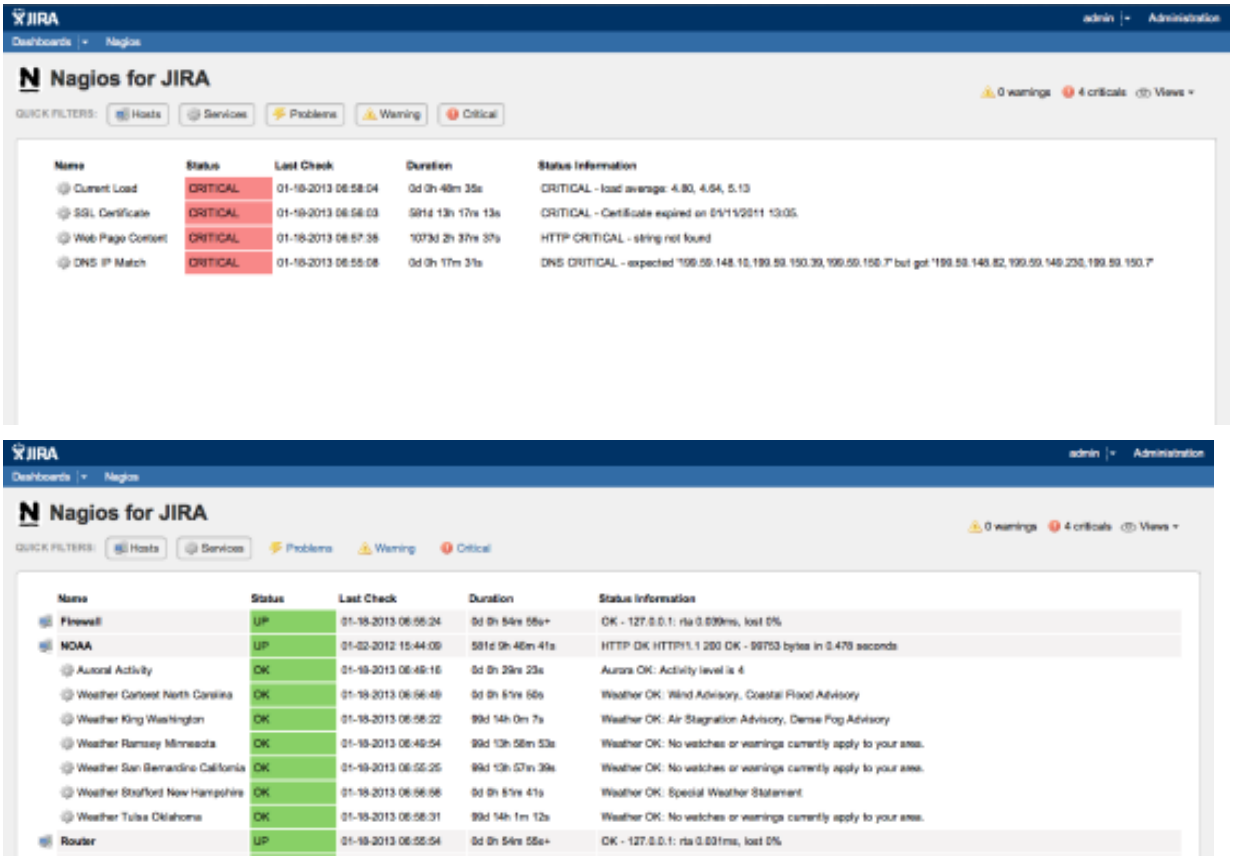


Figure 16: Sample screenshots showing the integration of XIFI infrastructure monitoring and helpdesk utilizing Nagios for JIRA

4.4 Identity management and Security monitoring

The monitoring of the security is one of the key factors to ensure the operation and availability of the federation, helping with a quick detection of security incidents, vulnerabilities and potential attacks. With this purpose and as it is depicted in the architecture diagram of the federation included in the D1.3, Security Probes need to be distributed and installed on the different nodes integrated in the XIFI Federation. Depending on the devices installed on the infrastructure (such as web servers, firewalls, databases, virtualization servers, etc.) one or more Security Probes can be required to collect the logs generated by the different data sources and to transform them into normalized security events, which will be sent to the Master node for being analysed. One essential data source in the nodes is the own log produced by the Access Control layer installed which includes information about the authentication and authorization management for each node. In case of existing proprietary security system already installed on the infrastructure, it is necessary to analyse each case to determine if the events generated by that system can be directly sent to the Security Probes (e.g. through a syslog service) or a plugin needs to be created for this integration.

On the other hand, in the XIFI Master node the Security Monitoring GE will be the responsible for the analysis of the events received from the Security Probes distributed through the federation and to provide the Security Dashboard with the relevant security information to be shown to the user.

Depending on the user, security events and alarms concerning the entire XIFI Federation will be shown to the Federation Manager or only the ones coming from a node in the case of the Infrastructure Owner Administrator.

In the XIFI Master node, the Federation Manager is the one in charge of configuring in the Security Monitoring GE:

- Data sources: although there are a set of data sources and event types already included by default in the Service Level SIEM component (such as snort events), new ones can be defined when required.
- Sensors: the server needs to have configured where the Security Probes are installed so it can control that only receives events from authorized and active nodes.
- Users: each Infrastructure Owner administrator needs to have its user configured in the Service Level SIEM with the allowed subset of assets (hosts/networks included in its infrastructure). Only the Federation Manager has permissions to define security directives that involve events generated in all the nodes of the Federation and visualize what happens in all the Federation.
- Security Directives: the Federation Manager must configure the correlation rules, policies and actions to be applied on the events coming from the different nodes in the Federation.

Once an Infrastructure Owner administrator has the Security Probes installed and configured in its node, and the Federation Manager has configured its user associated in the Security Monitoring GE, he/she can:

- Configure Security Directives involving events generated within its own infrastructure
- Visualize in the Security Dashboard all the security events generated by the Security Probes installed on its infrastructure and the alarms generated by the security directives configured.

5 INTERFACE AND API SPECIFICATIONS

This section provides an informal specification of the federation APIs.

5.1 Federation Manager API

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 [17].

5.1.1 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.

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"
   }
]
```

Add a service [POST]

Request (application/json)

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

Response 201 (application/hal+json)

```
{
   "service": {
      "id": "--service-id--",
```

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

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

Get service instance [GET]

Response 200 (application/hal+json)

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

Updated a specified service [PATCH]

Request (application/json)

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

Response 201 (application/hal+json)

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

Delete a service [DELETE]

Response 204

5.1.3 Endpoints Catalogue [/v3/endpoints]

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

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--"
   }
]
```

Add a 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--"
}
```

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

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--"
}
```

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--"
}
```

Delete an endpoint [DELETE]

Response 204

5.1.5 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.

Lists regions [GET]

Parameters

```
+ country (optional, string)
+ page (optional, string)
+ per_page (optional, string)
```

Response 200 (application/hal+json)

```
[
   {
      "id": "--region-id--",
      "country": "isocode",
      "latitude": "latitude",
      "longitude": "longitude",
      "admin": "admin username",
      ...
   },
   {
      "id": "--region-id--",
      "country": "isocode",
      "latitude": "latitude",
      "longitude": "longitude",
      "admin": "admin username",
      ...
   }
]
```

Add a region [POST]

Request (application/json)

```
{
   "country": "isocode",
   "latitude": "latitude",
   "longitude": "longitude",
   "admin": "admin username",
   ...
}
```

Response 201 (application/hal+json)

```
{
   "id": "--region-id--",
   "country": "isocode",
   "latitude": "latitude",
   "longitude": "longitude",
   "admin": "admin username",
   ...
}
```

5.1.6 Regions Catalogue [/v3/regions/{region_id}]

Get region instance [GET]

Response 200 (application/hal+json)

```
{
   "id": "--region-id--",
   "country": "isocode",
   "latitude": "latitude",
   "longitude": "longitude",
   "admin": "admin username",
   ...
}
```

Update a region [PATCH]

Request (application/json)

```
{
   "id": "--region-id--",
   "country": "isocode",
   "latitude": "latitude",
   "longitude": "longitude",
   "admin": "admin username",
   ...
}
```

Response 201 (application/hal+json)

```
{
   "id": "--region-id--",
   "country": "isocode",
   "latitude": "latitude",
   "longitude": "longitude",
   "admin": "admin username",
   ...
}
```

Delete an endpoint [DELETE]

Response 204

5.1.7 Regions Catalogue [/v3/regions/{region_id}/status]

Get region status [GET]

Response 200 (application/hal+json)

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

Update a region [PATCH]

Request (application/json)

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

Response 201 (application/hal+json)

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

```
}
```

5.1.8 Regions Catalogue [/v3/regions/{region_id}/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"
   }
]
```

Add a region contact [POST]

Request (application/json)

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

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"
         }
   }
}
```

5.1.9 Regions Catalogue [/v3/regions/{region_id}/contacts/{contact_id}]

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"
         }
   }
}
```

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

Delete a contact [DELETE]

Response 204

5.2 Federation Monitoring API

The following listings provide the APIs defined by the Federation Monitoring component that are callable by the users of this component. The syntax used to define the APIs is the one proposed by apiary [26].

5.2.1 Monitoring API Root [/]

Retrieve Entry Point [GET]

Model (application/hal+json)

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

5.2.2 Group Region

Region related APIs

5.2.2.1 Regions [/monitoring/regions]

Retrieve all regions

Model (application/hal+json)

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

List All Regions [GET]

Response 200 [Regions][]

5.2.2.2 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)

```
{
   "_links": {
      "self": { "href": "/monitoring/regions/Trento" },
      "hosts": { "href": "/monitoring/regions/Trento/hosts" }
   },
   "id": "Trento",
   "name": "Trento",
   "country": "Italy",
   "latitude": "xyz",
   "longitude": "xyz",
   "nb_users": "10",
   "nb_cores": "100",
   "nb_ram": "1000",
   "nb_disk": "10000",
   "nb_vm": "100000",
   "measures": [
      {
         "timestamp" : "2013-12-20 12.00",
         "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"
         },
         "percUptime": {
            "value": "123",
            "description": "average of the percUptime for all the hosts"
         },
         "xifiCoreServicesStatus": {
            "value": "green/orange/red",
            "description": "status of the xifi core services - aggregated
               value for all the services of the region"
         }
      }
   ]
}
```

Retrieve a Region [GET]

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

Response 200 [Region][]

5.2.3 Group Host

Host related APIs.

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

Retrieve all hosts in a given region.

Parameters
`regionid (string)`

Model (application/hal+json)

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

List All Hosts for a given Region [GET]

Response 200 [Hosts][]

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

Retrieve a host. 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)` `hostid (string)`

Model (application/hal+json)

```
{
   "_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"
       }
    ]
}
```


Retrieve an host [GET]

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

Response 200 [Host][]

5.2.4 Group VM

VM related APIs

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

Retrieve all VMs on a given region.

Parameters
`regionid (string)`

Model (application/hal+json)

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

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

Response 200 [VMs][]

5.2.4.2 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 monitored interval are provided. Measures are aggregated on an hourly basis.

Parameters

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

Model (application/hal+json)

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

Retrieve a vm [GET]

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

Response 200 [VM][]

5.2.5 Group Service

Service related APIs

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

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

Model (application/hal+json)

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

List all Services running on a given Host [GET]

Response 200 [Services4Host][]

5.2.5.2 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)

```
{
   "_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": "up",
            "description": "status of the service"
         },
         "percUpTime": {
            "value": "10",
            "description": "it is the percentage of upTime"
         }
      }
   ]
}
```

Retrieve a service belonging to an host [GET]

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

Response 200 [Service4Host][]

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

Retrieve all Services running on a given VM.

Parameters
regionid (string) vmid (string)

Model (application/hal+json)

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

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

List all Services running on a given VM [GET]

Response 200 [Services4VM][]

5.2.5.4 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)
```

Model (application/hal+json)

```
{
   "_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": "up",
            "description": "status of the service"
         },
         "percUpTime": {
            "value": "10",
            "description": "it is the percentage of upTime"
         }
      }
   ]
}
```

Retrieve a service belonging to an vm [GET]

Parameters

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

Response 200 [Service4VM][]

5.2.6 Group Network Element

Network Element related APIs.

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

Retrieve all NEs monitored in a given region

Parameters

```
regionid (string)
```

Model (application/hal+json)

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

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

Response 200 [NEs][]

5.2.6.2 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)

```
{
   "_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"
          },
          "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"
       }
    ]
}
```


Retrieve a network element [GET]

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

Response 200 [NE][]

5.2.7 Group Host2Host

Host related APIs.

5.2.7.1 Host2Hosts [/monitoring/host2hosts]

Retrieve all host2hosts.

Model (application/hal+json)

```
{
   "_links": {
      "self": { "href": "/monitoring/host2hosts" }
   },
   "host2hosts": [
      {
         "_links" : {
            "self": { "href": "/monitoring/host2hosts/12345" }
         },
         "id": "12345;6789"
      }
   ]
}
```

List All Host2Hosts [GET]

Response 200 [Host2Hosts][]

5.2.7.2 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)

```
{
   "_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"
         },
         "owd_avg": {
```

```
            "value": "123",
            "description": "desc"
         },
         "jitter": {
            "value": "123",
            "description": "desc"
         },
         "packet_loss": {
            "value": "123",
            "description": "desc"
         },
         "bandwidth_avg": {
            "value": "123",
            "description": "desc"
         }
      }
   ]
}
```


Retrieve an host2host [GET]

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

Response 200 [Host2Host][]

5.3 Federated IdM API

Native APIs and Protocols supported by the Keyrock IdM GEi are described in the KeyRock - User and Programmers Guide[28].

In particular Keyrock IdM GEi implements the OAuth protocol as detailed in the IdM Open Specification [38].

Additional from release 1.2, XIFI will add to the Keyrock IdM GEi the support for Web Browser SSO Profile of the SAML2.0 [39] protocol. To comply with the FI-WARE Open Specification, the SAML 2.0 protocol will be implemented in compliancy with the IdM Open Specification [38].

5.4 SLA Management Interfaces

The SLA Manager provides a REST interface to interact with the different entities. The detailed API for the SLA Manager is given in [11].

6 CONCLUSIONS

This document focused on the **federation concepts and architecture**, on how to **manage the federation** when established, on **procedures** to create and maintain the federation, and documents **APIs and other interfaces** utilized by the tools and portals that provide access to the federation for infrastructure owners and users.

This document targets various XIFI stakeholders, aiming in particular to support developers, infrastructure owners, operators and end users to understand the XIFI federation methodology, to learn about the services enabled through infrastructure federation, and to gain specific knowledge in utilizing the federated XIFI infrastructures.

In particular, this document emphasizes on the **Federation Management**, on **Federated Monitoring**, on **SLA Management**, and on **Identity Management** in a federated infrastructures environment.

Following an integrating approach, this document builds upon the approaches and findings of other XIFI work packages documented in several other deliverable documents, complementing these by a specific view on federation aspects. Whenever suitable, information already provided by related documents but required herein is referenced or briefly summarized rather than duplicated.

This document shows that a "thin" federation layer is required to manage and maintain the federation and to assist infrastructures in their interaction with the federation. The federation layer presented aims to enable infrastructures to join the federation and to commit local services as well as to access, interface and monitor these services and their supporting infrastructures. The underlying functionality required to support and implement the federation layer then is realized by enhancing existing services and components through certain enabling interfaces and functions described in this document.

From this document we conclude that federation APIs and tools cannot easily be addressed without relating them to each other and to the underlying services and platform capabilities. This is due to the strong dependencies between APIs and tools often used jointly for a given purpose (e.g. under a common dashboard function). Consequently, understanding dependencies is essential for understanding the federation methodology and operations. A detailed and distinct documentation of components, their APIs and supporting tools is mandatory but obviously not sufficient. A comprising documentation also should add on aspects of interaction and best practice for use of APIs and tools.

As an accompanying document to software contributions that may provide comprehensive documentation on their own, some of these dependency aspects have been addressed herein as a complementary contribution. It is suggested to keep this approach also for upcoming APIs and tools deliverables.

The best practice concept has not yet been applied in the scope of this document. It has been recognized while preparing this document that the availability of "best practice documents" directly aiming at the main objectives of federation (e.g. at federation management, federation access control, help desk support …) may have the potential to maximize the benefit for stakeholders identified as the target audience.. It is a suggestion and for further discussion if such best practice contribution could be added as a dedicated section, for example, to upcoming deliverable documents.

REFERENCES

[1] The Future Internet Public-Private Partnership Programme, http://www.fi-ppp.eu, accessed 31. January 2014

[2] FI-WARE, http://www.fi-ware.eu, accessed 31. January 2014

[3] The XIFI Consortium, Deliverable D1.1 XIFI core concepts, requirements and architecture draft – v1.1", July 2013

[4] The XIFI Consortium, Deliverable D1.2, "Analysis of UC, Infrastructures and Enablers v1 – v1.1", October 2013

[5] The XIFI Consortium, Deliverable D1.3," Federated Platform Architecture Version 1", February 2014

[6] The XIFI Consortium, Deliverable D2.1,"XIFI Technical Handbook Version 1", February 2014

[7] The XIFI Consortium, Deliverable D3.1,"XIFI infrastructure adaptation components API open specification – v1.1", October 2013

[8] The XIFI Consortium, Deliverable D3.2," Infrastructures monitoring and interoperability adaptation components toolkit and API", February 2014

[9] The XIFI Consortium, Deliverable D3.3," Infrastructures management toolkit API", January 2014

[10] The XIFI Consortium, Deliverable D4.1," Service & Tools specification – v1.0", October 2013

[11] The XIFI Consortium, Deliverable D4.2,"Baseline Tools", January 2014

[12] The XIFI Consortium, Deliverable D5.1," Procedures and Protocols for XIFI Federation – v1.1", October 2013

[13] IETF, "System for Cross-Domain Identity Management: Core Schema", http://tools.ietf.org/html/draft-ietf-scim-core-schema-02, accessed 30. January 2014

[14] IETF, RFC 4122, "A Universally Unique IDentifier (UUID) URN Namespace", July 2005, http://www.ietf.org/rfc/rfc4122.txt, accessed 30. January 2014

[15] The Open Group, "Distributed Computing Environment (DCE) – Remote Procedure Call – Universally Unique Identifier",http://pubs.opengroup.org/onlinepubs/009629399/apdxa.htm, 1997, accessed 30. January 2014

[16] The Open Grid Forum (OGF), "WS-Agreement Negotiation Version 1.0", http://www.ogf.org/Public_Comment_Docs/Documents/2011-03/WS-Agreement-Negotiation+v1.0.pdf, Jan 2011, accessed 30. January 2014

[17] "OpenStack Identity Service API v3", http://api.openstack.org/api-ref-identity.html, accessed 30. January 2014

[18] XIFI Consortium, "XIFI Stakeholders", https://www.fi-xifi.eu/about-xifi/stakeholders.html, accessed 31. January 2014

[19] IETF, SCIM, "SAML 2.0 Binding for SCIM", http://www.simplecloud.info/specs/draft-scim-saml2-binding-02.html, draft-scim-saml2-binding-01, accessed 31. January 2014

[20] FI-WARE Catalogue, "Identity Management – KeyRock",http://catalogue.fi-ware.eu/enablers/identity-management-keyrock/documentation, accessed 31. January 2014

[21] OASIS, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf, March 2005, accessed 31. January 2014

[22] IETF, RFC 6749, "The OAuth 2.0 Authorization Framework", http://tools.ietf.org/html/rfc6749, Oct 2012, accessed 31.January 2014

[23] Atlassian Software Systems, JIRA tracker, https://www.atlassian.com/en/software/jira, accessed 31. January 2014

[24] Atlassian Software Systems, Nagios for JIRA, https://marketplace.atlassian.com/plugins/net.collabsoft.plugins.jira.nagios-for-jira, accessed 31. January 2014

[25] Atlassian Software Systems, Marketplace, https://marketplace.atlassian.com/vendors/1210574, accessed 31. January 2014

[26] Apiary Ltd., http://apiary.io/how-it-works, accessed 31. January 2014

[27] Quick Online Test (non-public), http://wiki.fi-xifi.eu/Xifi:Wp4:Quick_Online_Test

[28] KeyRock Fiware IdM GEi, http://catalogue.fi-ware.org/enablers/identity-management-keyrock.

[29] GNU Affero General Public Licens 3.0, http://www.gnu.org/licenses/agpl-3.0.html.

[30] IETF, RFC 4122, "A Universally Unique Identifier (UUID) URN Namespace", July 2005,http://tools.ietf.org/html/rfc4122, http://www.ietf.org/rfc/rfc4122.txt.

[31] "System for Cross-Domain Identity Management: Core Schema", http://tools.ietf.org/html/draft-ietf-scim-core-schema-02.

[32] OpenAM project, http://openam.forgerock.org/.

[33] FI-WARE KeyStone Proxy, https://github.com/ging/fi-ware-keystone-proxy.

[34] Galera, http://www.codership.com/products/mysql_galera.

[35] Wrep API, https://launchpad.net/wsrep.

[36] Keepalived, http://www.keepalived.org/.

[37] KeyRock, user and programming Guide, https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/Identity_Management_-_KeyRock_-_User_and_Programmers_Guide.

[38] IdM OpenSpecification, https://forge.fi-ware.eu/plugins/mediawiki/wiki/fiware/index.php/FIWARE.OpenSpecification.Security.IdentityManagement#OAuth.

[39] SAML2.0, http://en.wikipedia.org/wiki/SAML_2.0.

[40] MySQL Galera, http://www.codership.com/content/using-galera-cluster .

[41] Modacloud-Analysis of the state of the art and scope definition:http://www.modaclouds.eu/publications/public-deliverables/

[42] WSLA: http://www.research.ibm.com/people/a/akeller/Data/WSLASpecV1-20030128.pdf

[43] WS-Agreement Structure: http://www.mcs.anl.gov/~keahey/Meetings/GRAAP/WS-Agreement%20Structure.pdf

[44] S. Becker, "Quality of service modeling language," in Dependability Metrics, ser. Lecture Notes in Computer Science, I. Eusgeld, F. Freiling, and R. Reussner, Eds. Springer Berlin Heidelberg, 2008, vol. 4909, pp. 43–47. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68947-8

[45] D. D. Lamanna, J. Skene, and W. Emmerich, "Slang: A language for defining service level agreements," in Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems, ser. FTDCS '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 100–. [Online]. Available: http://dl.acm.org/citation.cfm?id=795675.797134

[46] A. Paschke, "Rbsla a declarative rule-based service level agreement language based on ruleml,"

in Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents,Web Technologies and Internet Commerce, International Conference on, vol. 2, nov. 2005, pp. 308 –314.

[47] SLA@SOI: http://sla-at-soi.eu

[48] OPTIMIS: http://www.optimis-project.eu

[49] J. L. Prieto and F. J. Nieto, "A bidirectional trust model for service and infrastructure providers in clouds."

[50] Cloud4SOA: http://www.cloud4soa.eu/

[51] Contrail: http://contrail-project.eu/

[52] Mosaic: http://www.mosaic-fp7.eu/

[53] RuleML: http://wiki.ruleml.org/index.php/RuleML_Home

[54] Planetlab, http://www.planet-lab.org/, accessed 19. February 2014

[55] GENI, www.geni.net, accessed 19. February 2014

[56] Slice-based Federation Architecture, http://svn.planet-lab.org/attachment/wiki/WikiStart/sfa.pdf, accessed 19. February 2014

[57] GENI-Portal, https://portal.geni.net, accessed 19. February 2014

[58] GENI Clearinghouse, http://groups.geni.net/geni/wiki/GeniClearinghouse, accessed19. February 2014

[59] GENI Aggregate Manager V3, http://groups.geni.net/geni/wiki/GAPI_AM_API_V3, accessed19. February 2014

[60] Fed4FIRE, http://www.fed4fire.eu, accessed19. February 2014

[61] FIRE, http://www.ict-fire.eu/, accessed19. February 2014

[62] MySlice, http://myslice.info/home, accessed19. February 2014

[63] BonFIRE, http://www.bonfire-project.eu, accessed19. February 2014

[64] About BonFIRE, http://doc.bonfire-project.eu/R4.0.5/overview/about-BonFIRE.html, accessed 19. February 2014

[65] BonFIRE REST API, http://doc.bonfire-project.eu/R4.0.5/reference/bonfire-api-spec.html, accessed 19. February 2014

[66] OCCI, http://occi-wg.org/ ,accessed 19. February 2014

[67] WildFly Application Server, http://www.wildfly.org, accessed 19. February 2014