Project Acronym	**Fed4FIRE**
Project Title	**Federation for FIRE**
Instrument	**Large scale integrating project (IP)**
Call identifier	**FP7-ICT-2011-8**
Project number	**318389**
Project website	**www.fed4fire.eu**

D4.5 - Report on second cycle developments of the services and applications community

Work package	WP4
Tasks	T4.2, T4.3, T4.4, T4.5
Due date	30/04/2015
Submission date	08/05/2015
Deliverable lead	Mark Sawyer (UEDIN)
Version	V1.0
Authors	Mark Sawyer, Gareth Francis, Iakovos Panourgias (UEDIN) Julien Lefeuvre, David Margery (Inria) Thomas Günther (FOKUS) Pablo Sotres. Jorge Lanza (UC) Lucia Cossio (Adele)
Reviewers	Steve Taylor (IT Innovation) Dai Davies (DANTE)

Abstract	This document describes the work done in WP4 to support cycle 2 of Fed4FIRE.
Keywords	Services, applications, experiment control, monitoring

Nature of the deliverable	R	Report	X
	P	Prototype	
	D	Demonstrator	
	O	Other	
Dissemination level	PU	Public	X
	PP	Restricted to other programme participants (including the Commission)	
	RE	Restricted to a group specified by the consortium (including the Commission)	
	CO	Confidential, only for members of the consortium (including the Commission)	

Disclaimer

The information, documentation and figures available in this deliverable, is written by the Fed4FIRE (Federation for FIRE) *– project consortium under EC co-financing contract FP7-ICT-318389 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.*

Executive Summary

The Fed4FIRE architecture addresses the needs of the experimenter communities in three key areas: experiment lifecycle, measurement and monitoring, and trustworthiness. The architecture is evolving as time progresses and the requirements of the federation (including its sustainability) become better understood. The architecture is being implemented in a series of cycles. Cycle 2 of the architecture is an evolutionary progression from cycle 1, meaning that the basic ideas have not changed. Cycle 2 consist of improvements, clarifications and additions to cycle 1

There are four service and application-focused test facilities: BonFIRE, SmartSantander, FUSECO and FIONA, the last of which joined Fed4FIRE through the Open Calls in M27. Good progress has been made in all the test facilities towards implementing the architectural features. Ideally, all test facilities would have been able to implement all features of cycle 2 of the architecture. However the technical level of difficulty associated with implementing some features differed between test facilities, and therefore a prioritised subset of the ideal cycle 2 features became the target. Furthermore, as the project has progressed the notion of different levels of federation has been recognised, and test facilities have chosen to implement appropriate features of the architecture to achieve the right level of federation. An illustration of this the approach of SmartSantander, where a service level that provides access to data retrieval functionality is available to users in addition to a 'native' experiment layer that allows low-level access to resources. The emergence of different levels of federation in Fed4FIRE is inline with the overall mission of making experimentation easy for the users.

Fed4FIRE is an evolutionary project and the process of development is ongoing, driven by the principle of offering benefits and value to experimenters. In some cases this has involved temporary work-arounds to ensure that critical functionality is available. All test facilities are making good progress towards compliance with Fed4FIRE.

Acronyms and Abbreviations

AM	Aggregate Manager
API	Application Programming Interface
CA	Certificate Authority
EC	Experiment Controller
FLS	First Level Support
FRCP	Federated Resource Control Protocol
GENI	Global Environment for Network Innovations
NEL	SmartSantander Native Experiment Layer
NEPI	Network Experiment Programming Interface
OCCI	Open Cloud Computing Interface
OMF	Control and Management Framework
OML	OML Measurement Library
OMSP	OML Measurement Stream Protocol
QoS	Quality of Service
PDP	Policy Decision Point
RSpec	Resource Specification
SEL	SmartSantander Service Experiment Layer
SFA	Slice Federation Architecture
SNAA	Sensor Network Authentication and Authorization
VM	Virtual Machine
WSN	Wireless Sensor Network
XMPP	Extensible Messaging and Presence Protocol

Table of Contents

1 Introduction

This document is the report on the implementation of cycle 2 of the Fed4FIRE architecture as defined in deliverable D2.4 in the service and application-focused test facilities BonFIRE, SmartSantander, FUSECO and FIONA, the last of which joined Fed4FIRE through the Open Calls.

Cycle 2 of the architecture is an evolutionary progression from cycle 1, meaning that the basic ideas have not changed. Cycle 2 consist of improvements, clarifications and additions to cycle 1. The implementation and deployment with the test facilities has therefore also followed an evolutionary path. As with cycle 1, test facilties have had some freedom to prioritize which features to deploy in cycle 2, based on their usefulness to the end-users (experimenters) and the cost and technical difficulty of implementation. In some cases certain features of the architecture make no sense to deploy because of the nature of the test facilities. This is to be expected in a heterogeneous collection of test facilities, and particulary in WP4 which involves service and application oriented test facilities which tend to be offer diverse features.

Due to the cyclic nature of the Fed4FIRE development, the definition of cycle 3 of the architecture by WP2 has been done in parallel with some the implementation of cycle 2 by WP4 (and WP3). This has allowed those working on implementing the cycle 2 features to do so in such as way as to be ready for the cycle 3 developments, and the readiness for implementing cycle 3 features is also covered in this document.

1.1 Experiment lifecycle management

For cycle 2, just as in cycle 1, discovery, specification and provisioning will be done based on the SFA GENI AM API v3 standard, while for advanced reservation and extended policy based authorization extensions are currently being made. In order to allow interoperability among different Aggregate Managers (AMs), GENI requires a common language for describing resources, resource requests, and reservations. GENI now uses standardized Request Specification (RSpec) documents which are XML documents following agreed schemas to represent resources., In cycle 2 each testbed can still provide its own RSpec to be used and tools have to be adapted to these RSpecs.

1.2 Measurement and monitoring

For monitoring, three types of monitoring have been identified and described. Facility monitoring is used to monitor a testbed as a whole and is e.g. interesting for First Level Support and experimenters to know if a testbed is functioning correctly or not. Infrastructure monitoring is monitoring information provided by the testbed itself to an experimenter (e.g. switch port statistics, spectrum measurements or virtualization infrastructure monitoring) which is normally not available to an experimenter. Experimenter measuring is related to monitoring activities that the experimenter can do on the resources themselves.

1.3 Trust and security

The architecture defines also multiple identity providers with a chain of trust, and a central portal accompanied by an identity provider and directories (machine and human readable) for tools, testbeds and certificates. Cycle 2 also introduces a means of integrating the trust and security framework with other parts of the architecture. This is provided by the Policy Decision Point, designed and implemented by WP7.

1.4 Federation service

Cycle 2 introduces two new federation services: SLA management and Reputation. The Reputation service allows provides experimenters with information on the performance of test facilities based on the experience of previous users. These require support from testbed monitoring, and in the case of SLA management there is also the need for a software component to be deployed in a test facility. It is optional whether test facilities support these services in Fed4FIRE.

1.5 Test Facility Status

The status of each test facility is shown in the tables below:

	WP4 Testbeds			
	BonFIRE	**SmartSantander**	**FUSECO**	**FIONA**
RESOURCE DESCRIPTION, DISCOVERY, RESERVATION AND PROVISIONING				
SFA Support				
Basic SFA AM v2/v3 functionality	In progress (v3)	v2 - SFAWrap	v3	Under study
Resource Description & Discovery through SFA	In progress	Cycle 3	OK	Under study
Ontology based Rspec	Not currently planned	Under study	Cycle 3	Under study
Future reservation support (Reservation Broker)	In progress	N/A	Under study	Under study
Resource provisioning through SFA	In progress	Cycle 3	OK	Under study
Service Layer Support (testbed specific API)				
Deploy Application service with F4F PKCS12 credentials on Service Directory	Under Study	OK	N/A	In progress
Resource Description & Discovery support	OK via BonFIRE API	Partial / In progress	N/A	Under study
Resource reservation support	OK via BonFIRE API	N/A	N/A	Under study
Resource provision support	OK via BonFIRE API	Partial / In progress	N/A	Under study
EXPERIMENT CONTROL				
OMFv6 (FRCP)				
FRCP compatibility (either AMQP or XMPP)	OK	On the roadmap	Partial / In progress	Under study
Secure FRCP (using the F4F PDP component)	Partial / In progress	On the roadmap	Partial / In progress	Under study
SSH / OpenFlow based control				
SSH login (direct access)	OK	N/A	N/A	N/A
SSH login (through gateway)	OK	N/A	N/A	N/A

	WP4 Testbeds			
	BonFIRE	**SmartSantander**	**FUSECO**	**FIONA**
Service Layer Support (testbed specific API)				
Experiment control support with F4F PKCS12 credentials	N/A	Partial / In progress	N/A	N/A
F4F Service Orchestration Tool support (compatible M2M service description)	Under study	Cycle 3	N/A	N/A
DOCUMENTATION (linked in F4F doc site, but maintained by each testbed)				
Generic documentation				
Testbed description	OK	OK	OK	OK
Basic experiment showing the testbed (described as a tutorial)	OK	OK	Partial / In progress	Partial / In progress
SFA / OMFv6 related documentation				
RSpec models description	Cycle 3	Cycle 3	OK	Under study
OMFv6 RC parameters and definitions documentation	OK	On the roadmap	Under study	Under study
Service Layer related documentation				
Application Service usage (specific API documentation, extended service documentation, ...)	OK	OK	N/A	OK

	WP4 Testbeds			
	BonFIRE	**SmartSantander**	**FUSECO**	**FIONA**
MEASUREMENT AND MONITORING				
Facility Monitoring – First Level Support				
Provide SFA compatible endpoint with GetVersion status	In Progress	OK	OK	Under study
Provide Red/Amber/Green OML stream	OK	OK	OK	Under study
Infrastructure Monitoring				
Provide OML stream for Federation Services (preconfigured)	Partial / In progress	Cycle 3	OK	Under study
Provide OML stream for Experimenters (dynamic)	Cycle 3	Partial / In progress	OK (On-demand)	Under study
Support SFA based infrastructure monitoring (RSpecs & AM compatibility)	Cycle 3	Under study	Under study	Under study
Experiment Measurement				
Support OML based experiment measurements (dynamic)	Cycle 3	Partial / In progress	OK (On-demand)	Under study
OML Extensions				
Compatibility with secure OML streams	Under study	Under study	Cycle 3	Under study
Compatibility with Ontology based OML streams	Under study	Under study	Cycle 3	Under study

	WP4 Testbeds			
	BonFIRE	**SmartSantander**	**FUSECO**	**FIONA**
TRUST AND SECURITY				
Identity Provider				
Deploy an IdP compliant with SFA and X.509 certificates	Not currently planned	Cycle 3	OK	Under study
Authentication				
SFA - Deploy an SFA AM accepting main F4F IdPs	Partial / In progress	OK	OK	Under study
Service Layer - Accept PKCS12 credentials signed by F4F IdPs	Not currently planned	OK	N/A	Partial / In progress
Automatically download root certificates from certificate directory	Partial / In progress	OK	OK	Under study
Authorization and access control				
SFA Only - Deploy credentials at provisioning stage (eg: SSH keys if direct resource access is provided)	OK	Cycle 3		N/A
SFA & OMFv6 - Configure PDP component from SFA AM to control FRCP interactions	Partial / In progress	On the roadmap	Partial / In progress	N/A
Service Layer - Authorization based on F4F credentials	Not currently planned	Partial / In progress	N/A	N/A
SLA Management Services				
Deploy SLA management module	Not currently planned	Not currently planned	OK	Under study
Reputation System				
Provide compatibility with reputation system	OK	Not currently planned	OK	Under study

	WP4 Testbeds			
	BonFIRE	**SmartSantander**	**FUSECO**	**FIONA**
CONNECTIVITY				
Testbed Connectivity				
SFA - Public IPv4 or IPv6 for SFA AM	OK	OK	IPv4 OK; IPv6 under study	Under study
Service Layer - Public IPv4 or IPv6 for service endpoint (API server, web interface URL, ...)	OK	OK	N/A	OK
FRCP - IPv4 or IPv6 to AMQP/XMPP endpoint (OMFv6 RCs interaction)	OK	On the roadmap (AMQP)	Under study	Under study
SSH - IPv4 or IPv6 for SSH login (direct access)	OK	N/A	IPv4 OK; IPv6 under study	N/A
SSH - IPv4 or IPv6 for SSH login (through gateway)	OK	N/A	IPv4 OK; IPv6 under study	N/A
Testbed Interconnectivity				
Layer 2 connectivity between testbeds	OK	N/A	Under study	Under study
F4F Service Orchestration Tool support	Under study	Cycle 3	Under study	Under study

2 BonFIRE

2.1 Facility description

BonFIRE is itself a federation of test facilities at different locations run by the BonFIRE Foundation. It provides cloud-computing facilities (virtual machines) at EPCC and Inria, together with the network emulation capability from the Virtual Wall run by iMinds.

BonFIRE consists of three systems: an integration system for component development; a qualification system for package testing; and a production system. Experimenters only access the production system.

2.2 Cycle 2: Overview

Efforts in cycle 2 have concentrated on the following areas:

- Providing the SFA interface to BonFIRE to bring it into line with the other Fed4FIRE test facilities.
- Enhancing the infrastructure monitoring to support federation services
- Integration of the trust and security frameworks with OMF control framework by deploying the Policy Decision Point (PDP).
- Providing IPv6 capability and layer 2 connectivity
- Supporting users with tutorials and improving documentation.

2.3 Cycle 2: Experiment lifecycle management

2.3.1 Description, Discovery and Provisioning

For describing, discovering and provisioning resources the chosen Fed4FIRE mechanism is the Slice Federation Architecture (SFA) standard (specifically GENI AM APi v3), together with ontology based resource specifications (Rspecs) which underpin SFA.

One of the reasons for this choice was that at the project outset, many test facilities were already based on SFA or had the adoption of SFA on their roadmap. Hence adopting SFA represented the most logical path overall to a federation-wide standard. However this decision required those test facilities that were not already SFA compliant to map the concepts from their resource management systems and to generate software to implement the mappings. BonFIRE is one such facility. Interaction with BonFIRE by experimenters is through the BonFIRE API which is based on the Open Cloud Computing Interface (OCCI)[8] standard.

It was clear early in the project that mapping from OCCI to SFA was not a straightforward process anuired careful analysis of the BonFIRE concepts.

A first basic implementation of an SFA gateway to BonFIRE based on SFAWrap[9] was developed. A python module to implement the basic calls was written, but a complete integration of this module in a working SFAWrap implementation was not achieved. The core difficulties were:

1. Documentation for SFAWrap is scattered on the websites of the different projects that have been funding its development, with the documentation for plugin writers being very scarce.
2. SFAWrap mixes the functionalities of an aggregate manager and those of a clearing house (the planetlab term for slice authorities. It therefore became clear that the amount of adaptations to SFAWrap required to obtain a full implementation was really high and that the resulting code would difficult to maintain.

As a result, other SFA reference implementations were investigated in order to determine which one would facilitate access to its core structure in order to minimize the amount of adaptations required.

In particular, discussions with members of GENI[10] who developed GRAM[11], an SFA gateway to Openstack[12] using GCF[13] as SFA reference implementation, provided some interesting examples that could be built upon.

While GRAM in itself could not be easily used for BonFIRE, as it expects shell access to the testbed frontend, while BonFIRE requires communication through its API, it provided a sensible example to get access to GCF core components and data structures.

Hence two tools were developed, GCF-Fed4FIRE, a GCF fork that enhances its pluggability, and GCF-BonFIRE a plugin for GCF-Fed4FIRE implementing an SFA gateway to BonFIRE. The python code produced for the SFAWrap effort was successfully leveraged to help implement GCF-BonFIRE.

The effort was divided between GCF-Fed4FIRE and GCF-BonFIRE so that GCF-Fed4FIRE could be reused by other Fed4FIRE partners (at the time of this writing at least two other Fed4FIRE testbeds are considering using GCF-Fed4FIRE as as base for their aggregate manager implementation). This also enables modifications made to GCF to be contributed back to GCF.

GCF-BonFIRE implements an SFA GENI AM APi v3 gateway to BonFIRE based on GCF-Fed4FIRE. Focusing on GENI AM APi v3 calls (GetVersion, ListResources, Allocate, Renew, Delete, Provision, Describe) allows BonFIRE future reservation (introduced in BonFIRE v4.1) to be used through SFA.

At the end of cycle 2, an implementation that is demo capable is available. Using that version of the code, it is possible to reserve resources for a VM using the Allocate call, renew that reservation using the Renew() call, delete it using the Delete call and realize the reservation by starting a VM on BonFIRE using the Provision call. That version is to be pushed online during M31 of the project.

In cycle 3, GCF-BonFIRE will aim at finalising the RSpecs expected and produced by the cycle 2 version, extending the cases covered by it and integrating specifications and features provided by other workpackages into that code base.

2.3.2 Experiment Control

Fed4FIRE adopted OMF as the federation wide experiment control, measurement and management solution. OMF is based on an Experiment Controller, one or more Resource Controllers and a communications server which facilitates the exchange of messages between the various Controllers.

OMF supports two kinds of communication servers: XMPP and AMQP.

An XMPP server is used to enable communication between the OMF Experiment Controller and the OMF Resource Controller. An XMPP server provides a standardized method of sending and receiving

messages. Furthermore, the XMPP protocol allows a Publish/Subscribe (PubSub) method of communication.

An Advanced Message Queuing Protocol (AMQP) server can also be used to enable communication between the various OMF resources. The AMQP protocol is an open standard protocol which facilitates message passing for middleware applications. It can support both point-to-point and publish-and-subscribe routing. It also supports message orientation, queuing, reliability and security features.

For OMF, Experiment and Resource Controllers subscribe to various topics and are able to exchange messages. Finally, the communication server (XMPP or AMQP) allows broadcasting messages to all Resource Controllers.

On BonFIRE, the OMF Resource Controllers (and Experiment Controller if the user decides to use BonFIRE as the hosting infrastructure) are installed and configured in Virtual Machine images. These images are deployed (instantiated) by Fed4FIRE users. Once an OMF enabled Virtual Machine gets deployed; the OMF RC automatically subscribes to the BonFIRE XMPP server. The BonFIRE XMPP server is able to route and receive messages to/from the central Fed4FIRE XMPP server. Once the OMF RC establishes a connection with the BonFIRE XMPP server; it waits for instructions from the experimenters OMF EC.

The OMF software stack allows experimenters to create simplistic scripts (i.e. running a single application and waiting until the application ends) or complex scripts (i.e. running several applications based on external or internal events). Furthermore, if the application is OMF-enabled, the OMF Resource Controller can modify parameters of the application.

In order to support OMF, BonFIRE kept the existing two XMPP servers running, one for development and the other for production. The XMPP server was integrated into Fed4FIRE. Thus, a Fed4FIRE user could access resources registered on the BonFIRE XMPP server.

In order to support OMF applications, BonFIRE created OMF/OML-enabled OS images, allowing Virtual Machine instances to be provisioned which include all the required OMF libraries and executables. The Virtual Machines are pre-configured to contact the BonFIRE XMPP production server and are ready to receive commands. These images are currently available on the BonFIRE production system and are actively used by Fed4FIRE users.

2.4 Cycle 2: Measurement and monitoring

2.4.1 Facility Monitoring

BonFIRE implemented the Facility Monitoring Service (FLS) and has been pushing availability metrics for critical Physical Hosts and services to the central Fed4FIRE OML database.

BonFIRE uses a Nagios based infrastructure monitoring and alerting solution. The Nagios solution provides a BonFIRE-wide up-to-date status of the infrastructure. It can also pinpoint errors in specific services (for instance SSH or HTTP). BonFIRE created a middleware service which translates the Nagios output to an OML stream of data. It is managed using puppet, supervision being assured by a combination of Monit, Nagios and Zabbix. An automated alert is send to the BonFIRE administrators if the service terminates due to an internal error or if the service fails to start. Up-to-date values are

sent to the centralized Fed4FIRE OML database. A service queries the database and updates the FLS monitoring page.

2.4.2 Infrastructure Monitoring

For this work, extension to the BonFIRE API were developed so that the BonFIRE API is able to publish to privileged users the list of experiments and reservations using each node. This required considerable work to make all the components aware of physical nodes, whereas in previous versions of the API, this knowledge was delegated to the local testbeds, except for the name of nodes for which monitoring was available at testbeds. These were listed in a configuration file so the BonFIRE API was able to publish them so users could request monitoring metrics from the testbeds (with no knowledge of slices, slivers, experiments or reservations)

Fed4FIRE adopted OML as the federation wide instrumentation and monitoring solution. OML can be used to instrument and collect monitoring information from applications and from infrastructure servers. BonFIRE exposes several infrastructure metrics to the experimenter. The monitoring data of the physical machines that run experimenters virtual machines include: CPU load, total and free memory, free swap memory, the number of VMs on a physical node, disk IO, disk read/write, incoming and outgoing traffic on interfaces, energy usage and CO2 estimation, etc.

BonFIRE provides metrics to Central Federation Services and to experimenters using BonFIRE hosted resources.

2.4.2.1 Central Federation Services

BonFIRE provides the following metrics per Physical Host:

Category	Values
Availability	Physical Host ID (STRING), Availability (1 or 0), last_check (TIME/DATE)
CPU	Physical Host ID (STRING), Total (INT), FREE (INT), AVAILABLE (INT), last_check (TIME/DATE)
MEM	Physical Host ID (STRING), Total (INT), FREE (INT), AVAILABLE (INT), last_check (TIME/DATE)
Running VMs	Physical Host ID (STRING), RVM (INT), last_check (TIME/DATE)
Disk IOPS	Physical Host ID (STRING), DISKIOPS (INT), last_check (TIME/DATE)

BonFIRE also provides the following metrics per site:

Category	Values
Storage	Site ID (STRING), Storage (INT), last_check (TIME/DATE)

These metrics are stored in a centralized Fed4FIRE OML database and can be used by any Federation Service. The metrics can be queried based on the unique Physical Host ID (or Site ID for the Storage metric).

In order to collect these infrastructure metrics, BonFIRE developed and installed a centralized service. Two instances of the service exist; one collects data from the development/integration server of BonFIRE and stores data in a BonFIRE controlled OML DB and a production version which collects data from the production servers and pushes the metrics to the Fed4FIRE central DB. The first instance is used for development and the second instance is the live service.

The "federationServicesMonitoring" services are hosted in BonFIRE service VMs and are managed using Puppet, supervision being assured by a combination of Monit, Nagios and Zabbix. Thus, the services are guaranteed to be up and running and an automated alert is send to the BonFIRE administrators if the services terminates due to an internal error or if the services fails to start.

The service is written in Ruby. A configuration file is used to control which BonFIRE sites are measured and the interval between measurements. The service can be started and stopped using common Linux initialization scripts. The service runs in an infinite loop; querying the various BonFIRE Physical Hosts. Once the relevant metrics have been collected; the service pushes the data to the central Fed4FIRE OML database.

2.4.2.2 Experimenter Infrastructure Monitoring

The experimenter infrastructure monitoring service was completed in Cycle 1. The experimenters can collect Physical Host metrics (for the Physical Hosts which host one or more of their VMs) and store them locally.

This functionality is implemented using a service which runs locally on each VM. An enhancement to the service (implemented in Cycle 2) allows the experimenter to set a remote location for the OML database can be specified. The remote location can be specified using a configuration file or by starting the service with an extra command line argument. This functionality allows the experimenter to store metrics in a location that he controls.

2.4.3 Experiment Monitoring

BonFIRE OMF/OML-enabled images also provide OML-instrumented applications, enabling the user to collect "classical" metrics (cpu usage, ram consumption etc.). As for Infrastructure Metrics, no collection of metrics is set up by default; the user may requests the specific metrics required.

Monitoring the user's own application is a task for the experimenter. The application source code must be modified in order to include Measurement Points (MPs). An application which includes MPs and is compiled and linked with the OML libraries is an OML-instrumented application. Further details of how to do this can be found at these locations:

- https://mytestbed.net/projects/oml/wiki/Client_Programming and
- https://mytestbed.net/projects/omlapp/wiki/OML-instrumented_Applications.

BonFIRE provides images which include the OML libraries and are ready to compile and run OML-enabled applications. Furthermore, the SQLite engine is installed and measurements can be stored locally. If the experimenter wants to store measurements remotely, OML data streams can be re-directed to other machines (either BonFIRE virtual machines or testbeds under the control of the experimenter).

2.5 Cycle 2: Trust and security

2.5.1 User registration

At the end of cycle 1 a portal which enabled automatic registration of users having certificates provided by Fed4FIRE identity providers was implemented on the BonFIRE integration system. This has been migrated to the production system.

2.5.2 PDP

BonFIRE adopted and implemented the Control and Management Framework (OMF). BonFIRE users can use OMF to control applications running on BonFIRE compute nodes. However, the OMF solution does not provide authorization and authentication mechanisms. Thus, an attacker can abuse an OMF based testbed. Fed4FIRE's solution was to design and develop Policy Decision Point (PDP). The PDP solution is discussed in D7.2 [6]. The components needed to provide a PDP service have been developed by University of Southampton IT Innovation Centre.

In order to support the PDP, BonFIRE enhanced the external Broker API (which is based on OCCI). The updated API allows an experimenter to associate a BonFIRE experiment with a particular slice ID as well. Whenever an experiment is created in BonFIRE, a message is placed on BonFIRE's internal management message queue; an update to BonFIRE's resource manager ensures that the slice ID and slice owner are included in this message, which can be read by a new "PDP MQ Listener" service.

The "PDP MQ Listener" service is hosted in BonFIRE service VMs and is managed using Puppet, supervision being assured by a combination of Monit, Nagios and Zabbix. Thus, the service is guaranteed to be up and running and an automated alert is sent to the BonFIRE administrators if the service terminates due to an internal error or if the service fails to start. The "PDP MQ Listener" middleware service creates a connection with the BonFIRE internal message queue. It detects experiment and VM creation events. When a creation event is detected, the service intercepts the message and programs the PDP accordingly. The PDP is programmed by the "PDP MQ Listener".

BonFIRE uses a modified version of the OMF Resource Controller. The modified version performs an extra check before executing any incoming FRCP message. It first checks with the PDP. The resource controller uses the FRCP protocol and asks the PDP if the incoming FRCP message should be trusted and thus allowed to execute. The PDP replies using the FRCP protocol and if it instructs the OMF Resource Controller to trust the message; the message is executed. Otherwise it gets ignored.

BonFIRE also uses a modified version of the OMF Experiment Controller. The modified version appends an encrypted version of the users slice ID and slice credentials. The slice ID is extracted at the PDP and compared against the assertion rules that have been programmed. The PDP is currently undergoing integration testing. Manually configuring BonFIRE VM images and the PDP has been successful. On startup; a manually configured modified OMF RC image is able to contact the PDP using FRCP messages. The PDP is also able to authenticate that the FRCP messages originate from a trusted OMF RC VM. Furthermore, the PDP is able to determine the relationship between the OMF RC VM, a slice and a BonFIRE experiment. We have been able to run a simple FRCP based experiment using PDP as a trusted policy decision point.

In cycle 3 we plan to complete testing the PDP and automate the configuration and provisioning process.

2.5.2.1 PDP Certification Authority

The PDP and the modified OMF Experiment and Resource Controllers require a trusted chain of certificates in order to authenticate that received FRCP messages. In order to support the PDP, BonFIRE created a Certification Authority service. A trust relationship has been created between the PDP and the BonFIRE CA.

The PDP also requires that each BonFIRE VM has an individual public/private key pair and a certificate from a trusted CA. Furthermore, the certificate must include in the v3 extensions tag the Slice User URN. BonFIRE will implement support for an automatic creation of certificates which conform to the requirements of the PDP.

The certificates will be requested by a user agent running on each BonFIRE VM. The user agent will create the public/private key pair during the first boot of the VM. It will then create a personalized Certificate Signing Request (CSR). The request has to be personalized because the private key of the VM must be used and the Slice User URN must be included in the v3 extensions tag. The user agent will upload the completed CSR to the CA server and wait until the CA server creates the certificate.

The CA server will poll the upload CSR directory and once a new request is detected it will verify and sign it. The signing process takes less than 2 seconds; thus, we do not plan to use multiple threads for the signing server. The CA server will sign and create the certificate and copy it to the download directory. Old certificates (older than 2 hours) will be automatically deleted.

The user agent on the VM will wait for 30 seconds and then it will try to download the certificate from the CA server. If the certificate is not available; the user agent will retry after 30 seconds. It will retry for 20 times (i.e. 10 minutes). If the certificate is not available by then, the user agent will terminate and inform the user. The OMF RC will not be started.

The certification process will be documented in the BonFIRE documentation pages and users will be able to request certificates manually.

2.6 Connectivity

Native IPv6 connectivity for virtual machines is now provided by default at one BonFIRE site (EPCC). An updated version of OpenNebula with improved IPv6 support has been tested, but not yet deployed (due to the desire to deploy other, higher priority, components first). When deployed, this will provide additional network information through the OCCI interface, which will make it easier for client tools and scripts to make use of the IPv6 interface.

Layer 2 interconnection using the GÉANT AutoBAHN Bandwidth on Demand service was demonstrated between the BonFIRE site at EPCC and the Virtual Wall at iMinds. This is now available for experiments to use (albeit with an initial one-off configuration step by administrators required). The possibility of a dedicated layer 2 interconnection between a BonFIRE site and one of the OFELIA islands was investigated, but rejected due to lack of demand from potential experimenters.

2.7 Progress towards cycle 3

- Add slice ID, sliver ID and BonFIRE experiment ID in collected Infrastructure Monitoring data.
- Add support for the experimenter to dynamically configure the OML datastream endpoint for Infrastructure data.
- Add support for automatically creating certificates for the PDP.
- Make the PDP available to users.
- Prepare to deploy AMQP server.

3 SmartSantander

3.1 Facility description

The SmartSantander testbed is an experimental test facility for the research and experimentation of architectures, key enabling technologies, services and applications for the Internet of Things in the context of a city (the city of Santander located in the north of Spain). More than 20,000 IoT devices both fixed and mobile have already been deployed. From the whole SmartSantander consortium, which includes Belgrade, Guildford, Lübeck and Santander, only the Santander deployment is being used within Fed4FIRE.

The SmartSantander infrastructure enables a twofold approach in terms of experimentation: service and native experimentation.

- Service experimentation consists of running experiments and/or applications based on the data gathered by SmartSantander sensor infrastructure and stored in a shared repository. Therefore, these services will be mainly based on data retrieval from this repository. This way, third parties will be able to provide added-value functionalities based on them, hiding the complexity of the SmartSantander infrastructure and only dealing with a high-level interface. In this sense, by using the data retrieved, service experimenters could run data mining procedures in order to infer more elaborated metrics and provide these extensions to, for instance, represent diagrams, maps, etc.
- On the other hand, native experimentation requires a thorough knowledge of the SmartSantander infrastructure and how the different nodes actually work. This type of experimentation is considered a low level experimentation as it directly accesses the nodes and its hardware.

Inside SmartSantander, those two experimentation paradigms are often referred as SEL (Service Experimentation Layer) and NEL (Native Experimentation Layer).

3.2 Cycle 2: Overview

From the two experimentation modes SmartSantander provides, only SEL mode has been included as part of Fed4FIRE project. For this reason, most of the information included on this deliverable is based on work carried out on top of this SmartSantander service experimentation layer. However, NEL mode inclusion is on the roadmap for the future, and some of the Fed4FIRE enabling technologies will be included for NEL as well.

As part of the architectural work carried out on WP2 (T2.1), different federation models have been defined [13]. This has provided the different testbed facilities under the Fed4FIRE umbrella with two different approaches on how to enable federated access to their resources:

- The first one is based on the SFA/OMFv6 stack, with a security layer based on X.509 client certificates. This is the main approach used in Fed4FIRE. OMFv6, or its subjacent protocol, the Federated Resource Control Protocol (FRCP), is not strictly necessary (e.g. SSH access to the resource can be provided instead). In addition, deployment of a PDP component to validate the FRCP requests is needed to provide secure OMFv6 functionality.

- The second one, currently known as “API Only”, is based on the usage of an interoperable authentication layer (derived PKCS#12 client certificates) combined with proprietary APIs (based on standard or non-standard technologies). This approach should be used if it better fits the testbed experimentation model.

While the functionality provided by NEL experimentation is resource-oriented, thus, somehow similar to the one provided by other experimentation facilities in WP3 (in terms of resource management and experiment control), SEL functionality is quite different. SEL is data-oriented and provides an abstraction layer on top of raw sensor data which allows an experimenter (or in general, an end user) to interact with such an amount of resources transparently, without the need of having any knowledge about how the underlying infrastructure is organized. Even though some of the generic federation enabling technologies used in Fed4FIRE, such as SFA and OML, can be used to provide access to SmartSantander sensor data, they can’t yet offer the flexibility this kind of experimentation needs by themselves (in terms of abstraction and QoE). Therefore during cycle 2, and after a deep analysis of the different possible Fed4FIRE federation strategies, SmartSantander decided to dive into the “API Only” model in order to federate its SEL experimentation layer. Indeed, as it will be further described, SmartSantander roadmap on SEL also includes the usage of the SFA/OML stack as a part of the federation architecture.

It is important to highlight that this approach is not completely new. During Fed4FIRE cycle 1, a proprietary REST interface was developed and deployed to fully support OC1 experiments. As it was intended as a temporary solution, this interface only provided a very basic set of functionalities. However, experiences and feedback coming both from Fed4FIRE OC1 experimenters together with some other external experiments demonstrated that despite the fact it did covered the needs, this interface had to be enhanced to provide an adequate QoE.

As a result, work during Fed4FIRE cycle 2 has been mainly focused on providing an authenticated and flexible interface to interact with SmartSantander service experimentation layer in a generic way. The implications of this task included the unification of the different information models present throughout the different infrastructures deployed in SmartSantander, the creation of some new components inside the SmartSantander architecture and the redesign of several pre-existent SmartSantander modules to better fit the new additions. A complete production-ready deployment of all the envisioned functionalities has not yet been achieved during cycle 2. Work on this task will continue on cycle 3, together with some new functionalities derived from this new cycle requirements.

The following sections deal with progress in SmartSantander on each of the three major areas on which Fed4FIRE work has been traditionally differentiated: experiment lifecycle management, experiment measurement and monitoring, and trust and security.

3.3 Cycle 2: Experiment lifecycle management

3.3.1 Description, Discovery and Provisioning

During cycle 1, basic SFA AM functionality was deployed inside SmartSantander. It only supports a subset of the GENI AM API v2 standard and it does did not provide access to perform resource discovery or provisioning. This has not changed during cycle 2. Instead, work during this cycle on this task has focused on the service layer support.

In order to demonstrate Fed4FIRE service directory functionality, SmartSantander deployed an application service on it [15]. This application service, known as SmartSantander IoT API, is just a proof of concept for resource discovery and does not include the complete experimentation functionality SmartSantander SEL layer can provide. It currently supports authentication and authorization based on PKCS12 client certificates derived from the X.509 ones used on the SFA AM interface. In addition, an associated online documentation site [16] was set up to allow experimenters to interact with this platform in an easy and intuitive way. Finally, this service is described using Swagger v1.2 specification [17] to automate M2M discovery.

Following a similar concept, UC has been working on the redefinition and implementation of some of the SmartSantander core components and interfaces aiming at improving experimenter's usability and QoE.

Resource description, discovery and provisioning functionality via SmartSantander IoT API have been heavily enhanced during cycle 2. However, data orientation of the SmartSantander SEL experimentation, blurs the boundaries between resource discovery, resource provisioning and experiment control stages. In this sense, isolating particular functionalities that are only related to one of these experiment steps is not easy. Anyway, even though they are also part of experiment control, we can highlight:

- Resource discovery does not necessarily need to be based on the resource URN. Combination of geographic restrictions together with other information such as type of sensors can also be used without the need of knowing the exact resource on which data has been generated.
- A user can “provision” its experiment by creating a subscription to a set of sensor measurements (or observations) which will provide no immediate sensed information back but this information will start coming afterwards, when the specified criteria are met. In order to create a subscription the experimenter will have to include parameters regarding how and where he wants to be notified.

A more detailed description of the modifications carried out on top of SmartSantander is included on next section. Although a production ready version was initially planned by the end of cycle 2, it won’t be released until cycle 3 due to the amount of effort needed to its implementation.

3.3.2 Experiment Control

Experimentation on top of SmartSantander service layer consists in the retrieval of real time measurements generated by the sensors deployed across the city of Santander. These nodes sense different parameters such as traffic intensity, parking occupancy, temperature or pollutants, for

example. Experimenters will use this data as an input for their developments to offer value added services on top of a smart city.

During cycle 1, access to this data was provided via a proprietary REST interface deployed on top of the SmartSantander internal data repository. This was set up as a temporary solution while SFA/OML/OMFv6-compliant solution was developed. However, the experience gained thanks to this solution suggested that, for the kind of experimentation SmartSantander SEL provides, the provision of a traditional service-oriented stack allows a level of abstraction and QoE which would be highly valued by the users and experimenters.

In order to be able to fulfil the requirements and to address the feedback the users provided during Fed4FIRE OC1 together with some external experiments performed during the same period, several important changes had to be made inside SmartSantander core and federation architectures. The implementation of all those changes together with the design and implementation of the new components deployed have concentrated most of the effort during this cycle 2.

In the next subsections we detailed the different modifications and additions that have been carried out on top of the SmartSantander platform. All of them together form what is known as SmartSantander v2. The simplified architecture of this platform core (only including SEL related components) is depicted in Figure 1.

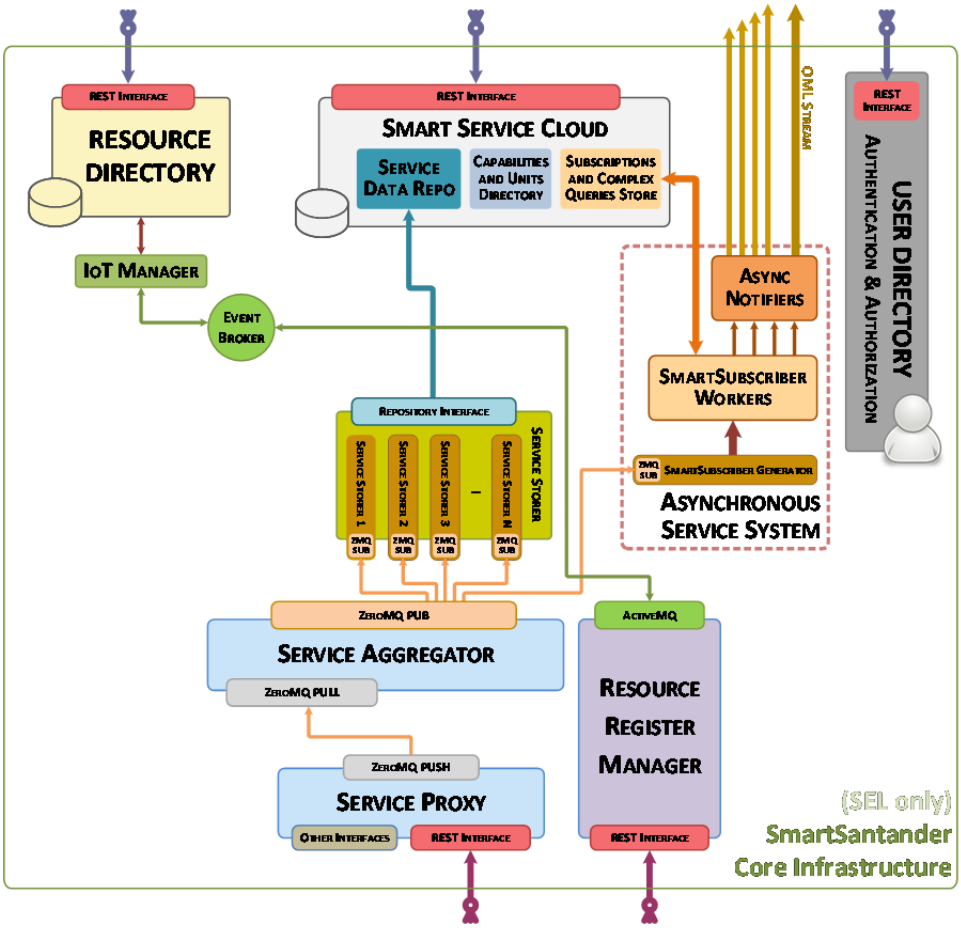

Figure 1: SmartSantander v2 SEL related components

3.3.2.1 SmartSantander sensor infrastructure unification

SmartSantander provides access to a heterogeneous sensor network. During the project duration there were three main deployed infrastructures:

- The fixed and mobile WSN oriented nodes, which conformed an IEEE 802.15.4-based meshed network.
- The augmented reality infrastructure, including all the QR and NFC passive tags deployed across the city and used for smart-shopping, smart-transportation and smart-tourism application domains.
- The participatory sensing application, enabling citizens' around the city to send sensor measurements and to generate city-related claims and events through their smartphones.

Integration of the data coming from these three infrastructures was done at the repository level, but different data models were used inside the SmartSantander core architecture until they reached that level. In practice, this implied having components in the architecture in charge of dealing with all these different data sources. Indeed, the way of accessing data was different depending on the kind of data the user wanted to access. During some of the experiments carried out on top of SmartSantander the users wanted to have a transparent access to data coming from all these three infrastructures. This was not possible with the design available at that moment without modifying each of the components related to them. As that solution did not seem to scale well, we decided to begin the data integration process at a lower level and maintain a single core infrastructure to manage it.

The first step to accomplish this integration has been the definition of a common data model able to support all the existent measurements, but generic enough to be able to support new data sources in the future. The work done on this topic led to the usage of two different basic data models: "measurements" and "observations". A measurement is the minimum quantum of information and presents the information captured by one sensor at a given place and moment. An observation is an extension of the measurement concept, and includes a set of one or more sensor measurements generated by a single IoT device at the same moment in time. It is important to highlight that a single IoT device can be equipped with many sensors. Thus, an observation can have one to multiple measurements. Both of them are formatted using standard JSON format. The decisions taken on data models are also reinforced by the chosen technology for the new Service Data Repository, which will be discussed later..

Any sensor data has to be transformed into an observation before reaching the SmartSantander core infrastructure. This converted service information is injected through a component known as Service Proxy, which is in charge of the authentication and authorization. It also validates the measurements on each observation based on the capabilities with which that resource was registered. Finally, when needed, it fills missing information (such as timestamp or location of data generated by fixed nodes).

The resulting valid observations are pushed towards the Service Aggregator component. This component is the central broker of service-level information inside SmartSantander. Any internal components willing to use service information have to subscribe to the Service Aggregator.

Summarizing, work done on this area during Fed4FIRE cycle 2 includes:

- Analysis of previous SmartSantander data models and definition of a common data model able to cover the existing experimenters' requirements. Standard JSON schemas to validate them have been generated. This has been fully completed during cycle 2.
- Design and development of the Service Proxy component. Basic functionality has been completed, including authentication and basic authorization. However, further integration with the advanced access control component (part of the User Directory) will be done once deployed during Cycle 3. In addition, SmartSantander Resource Directory will be enhanced during Cycle 3 to support ontologies, so some minor integration work is also expected.
- Modifications of the Service Aggregator component in order to support the new defined data model. This component has been fully completed during cycle 2.
- Development of the corresponding adaptation layers for the different SmartSantander deployments. This task has started but it's not yet completed. During cycle 2 only fixed and mobile WSN oriented nodes have been ported. Work on the other deployments have started but was not completed during cycle 2.

3.3.2.2 SEL Synchronous components (for historical access)

SmartSantander service data storage has been traditionally implemented on top of two different clouds: SmartSantander internal mirror one and Telefonica IDAS platform (now part of FIWARE). As the former data store does not impose any restriction on its usage policy, it was the one chosen to provide access to service experimentation layer using Fed4FIRE interfaces. However, the design of this platform was originally made just as a backup of the IDAS one, so query speed and scalability were not taken into account during its deployment.

One of the most repeated requirement received from experimenters was to lower the latency on historical data access, which was too high.

In order to satisfy it, a whole redesign of how the SmartSantander service data repository is organized has been done. Previous version was based on different SQL databases, each of them serving as storage of information according to the different existent data models. In SmartSantander v2, a NoSQL database has been chosen for holding all the observations, which is also in line with the usage of JSON format to structure them. Indeed the different data models are heavily influenced by how we access information from this repository, which is controlled by the IoT API described on section 3.3.2.4.

To inject the observations published by the Service Aggregator into the Service Data Repo, some modifications were needed on top of another component, namely the Service Storer. This module is subscribed to every service observation in the SmartSantander core and is in charge of optimizing each one before storing them on the Service Data Repo.

Summarizing, work done on this area during Fed4FIRE cycle 2 includes:

- Redesign of the SmartSantander Service Data Repository. In particular, during cycle 2 a TokuMX instance (a MongoDB fork) has been deployed and configured as data store for service related information. This repository, named in Figure 1 as Service Data Repo, is not

yet available on production, but it is already up and running on development deployment and will be migrated for the SmartSantander v2 release.

- Modifications to the Service Storer component in order to support the new defined data model together with the new storage backend. This component has been fully completed during cycle 2.

3.3.2.3 SEL Asynchronous components (for real time notifications)

Some users and experimenters have required the possibility of just subscribing to the measurements (or observations) matching a set of filters and receive them as notifications. This is a very interesting use case which will allow experimenters not having to periodically poll to check if there is something new. As a side effect, it will also reduce the congestion on the IoT API interface and will lower the total number of requests to the Service Data Repository.

Subscription mechanisms have been designed and implemented from scratch during Fed4FIRE cycle 2.. After analysing the access requirements imposed by the different users (also including non-Fed4FIRE ones) two main aspects have been concluded:

- Some users prefer to work with independent measurements, while other wish to work with the complete observations containing related measurements.
- The filtering categories are essentially three:
  - » A user might want to filter measurements (or observations) by what those measurements are, including its type, unit or value.
  - » A user might want to filter measurements (or observations) attending to where are they generated, either by area or by the resource originating it.
  - » Although it is not very common, a user might want to filter measurements (or observations) looking at their timestamp. In this case, date or periods can be used, and a period is subtracted from the actual time.

Criteria can be combined to form a complex subscription. Indeed, a similar schema is also used on synchronous access to allow reusable complex queries.

The subscription information is currently injected into the system by using the corresponding operation of the IoT API interface. These subscriptions will also be filtered by the authorization component, which will decide whether or not a particular user have the rights to create a particular subscription. Once a subscription is created, it will be valid until it expires (if expiration is no extended by the user) or if notifications can't be sent because the user is not reachable.

Experimenters must use the target field to indicate how and where to receive the notifications. All the subscriptions are stored in a repository, which is based on a combination of MongoDB and Redis databases. Active subscriptions are then run against new observations to decide if a notification has to be sent. There are different "Async Notifiers" for each of the supported notification technologies. The details of the Asynchronous Service System architecture are included on Figure 2.

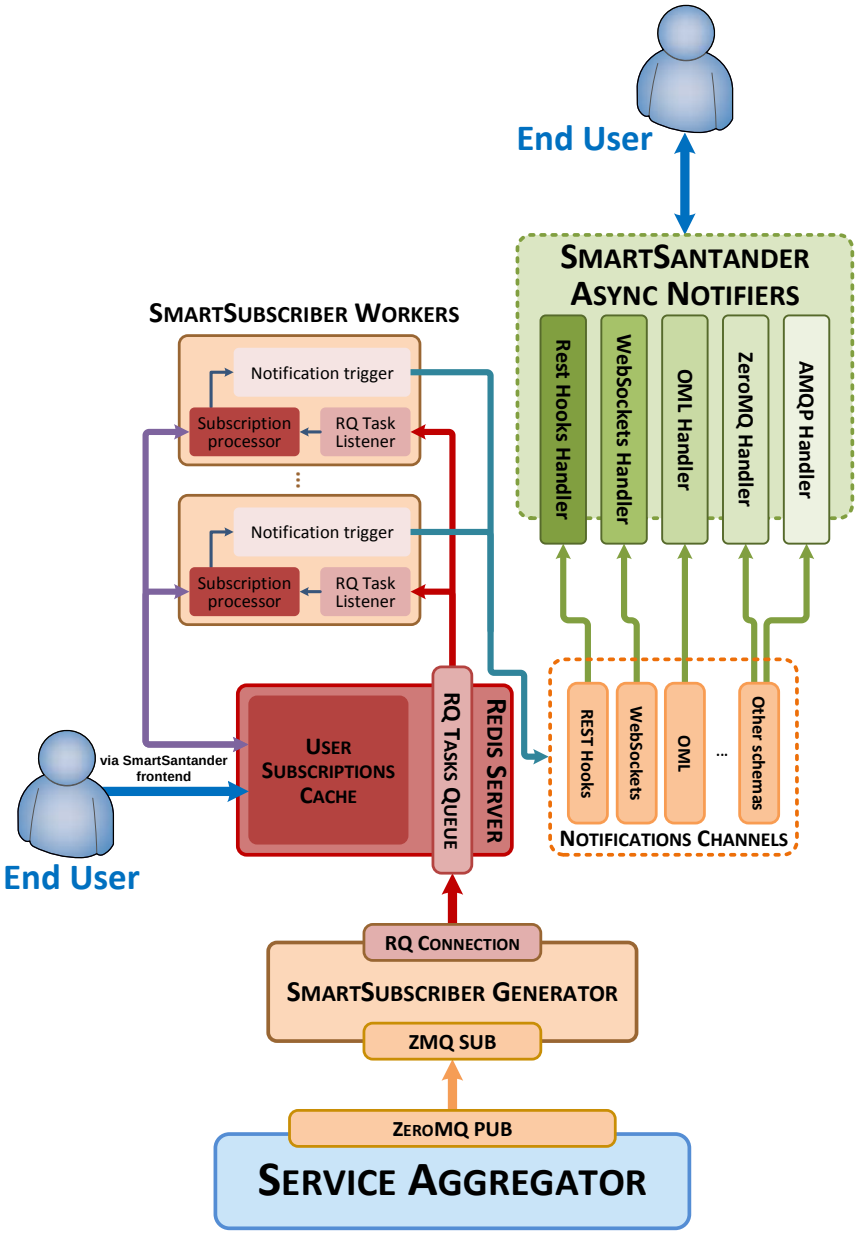

Figure 2. SmartSantander Asynchronous Service System

Summarizing, work done on this area during Fed4FIRE cycle 2 includes:

- Analysis of the different usage models for consuming SmartSantander service data in order to create a generic subscription model able to cover the required usage modes. All needed subscription models and criteria have been fully designed during this cycle starting from scratch. In addition, the corresponding standard JSON schemas to validate those subscriptions have been generated.
- Deployment of the necessary backend infrastructure, based on MongoDB and Redis, to support subscription persistence and evaluation. It is currently deployed on development infrastructure.

- Design and development of the SmartSubscriber Generator module. This component have been already finished at the beginning of cycle 3.
- Design and development of the SmartSubscriber Worker modules. Basic functionality has been already achieved at the beginning of cycle 3, but integration and testing with the authorization system will be done in the future.
- Regarding Async Notifiers, RestHooks, Websockets and OML components implementation has already been started. However, they are not yet finished and development will continue during cycle 3.

3.3.2.4 SmartSantander IoT API design and definition

During Fed4FIRE cycle 2, we have focused on increasing usability and QoE of the system. Before cycle 1, SmartSantander provided proprietary interfaces tailored to the experimenter needs. Experience obtained during this period recommended the provision of a generic authenticated REST API as the more suitable solution to perform the kind of experimentation SmartSantander SEL provides.

In this sense, after deploying a preliminary proof of concept to demonstrate its functionality, one of the first tasks in this Fed4FIRE cycle 2 was to define a REST API to fully cover the whole SEL experimentation lifecycle. All "REST interface" blocks on **Error! Reference source not found.** are part of the common SmartSantander IoT API. Security aspects have also been considered, and access is based on two different schemas: X.509 client certificates derived from Fed4FIRE credentials and a per-experimenter API key approach. In Figure 3, a summarized version of this REST API is presented.

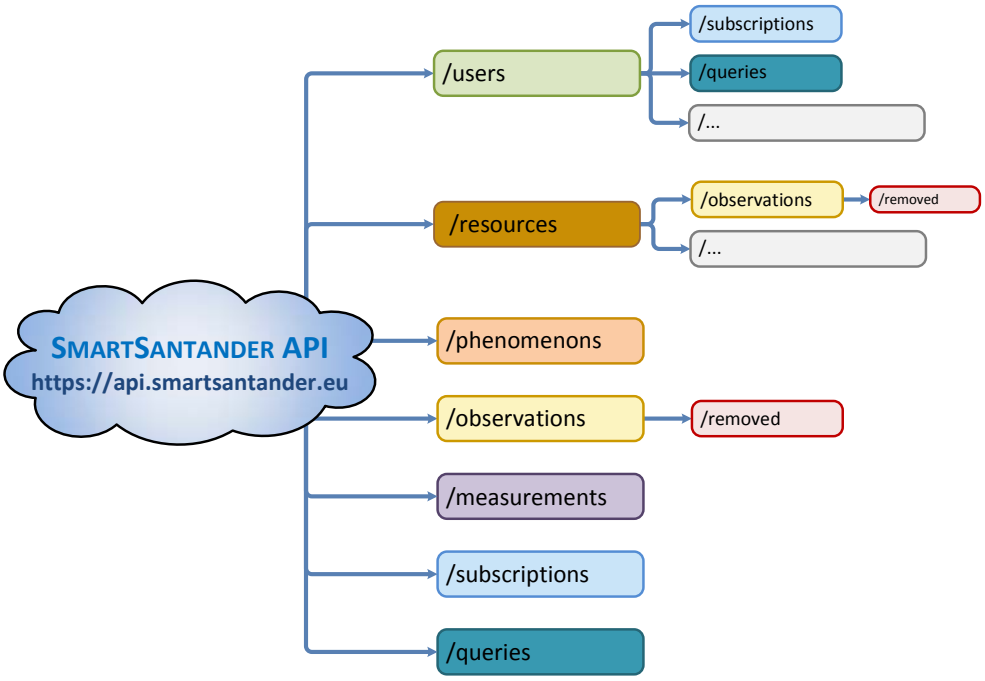

Figure 3. SmartSantander IoT API high level description

The first iteration of its implementation has been completed during cycle 2. However, as it is a component which depends on the existence of the whole ecosystem, further development iterations and integrations will be carried out during cycle 3.

3.4 Cycle 2: Measurement and monitoring

3.4.1 Facility Monitoring

This was already implemented during cycle 1 so no further work has been carried out on this topic during cycle 2.

3.4.2 Infrastructure Monitoring

During Fed4FIRE cycle 2, an OML component has been introduced as part of SmartSantander asynchronous architecture. It will be in charge of application monitoring data. However, most of the metrics that are calculated in SmartSantander SEL layer are derived from the collected sensor observations, so they do not suit with Fed4FIRE infrastructure monitoring concept. In practice, infrastructure monitoring mechanisms and experiment monitoring are equivalent. Due to the inherent characteristics of SmartSantander SEL experimentation, the system has been designed to address experiment monitoring requirements. Dynamic infrastructure monitoring will come as a by-product. Both of them imply the creation of subscriptions through the IoT API, although the criteria used for each of them and the post-processing to be performed will probably differ.

While system design was done during cycle 2 implementation phase has already started and development and integration will continue during cycle 3.

3.4.3 Experiment Monitoring

Traditional, OML-based, Fed4FIRE experiment monitoring is not applicable to SmartSantander SEL. This experimentation does not offer the experimenter a way of running applications, so OML-instrumented applications are not applicable in any way.

However, as explained in previous section, OML has been introduced as one of the different compatible technologies of the Asynchronous Service System. In this sense, one Async Notifier module will be in charge of OML based notifications. This way, it is possible to access an OML instrumented application that the users can configure via subscription definitions.

The status of this task has been commented in previous section: it has already been started but will be finished during cycle 3.

3.5 Cycle 2: Trust and security

3.5.1 User registration

Due to the resource sharing policy enabled by SmartSantander SEL, access control was not a priority before Fed4FIRE. Indeed, security during cycle 1 was based on network restricted access and protected access to the API documentation.

Trust and security aspects has been included in SmartSantander IoT API during cycle 2. As it can be seen on **Error! Reference source not found.**, authentication can be done based on the Fed4FIRE PKCS#12 derived client certificate. This way, SmartSantander IoT API can authenticate different Fed4FIRE involved identity provider users and authorization can be performed based on the certificate fields.

In addition, during cycle 2 SmartSantander SEL redesign process, security aspects have been introduced as one of the basic pillars on the new SmartSantander v2 architecture. In this sense, a new component, known as User Directory, has been defined. It will be in charge of managing user credentials and its associated policies. Even though XACML format was considered, those policies will be defined by using a proprietary JSON based scheme which fits better the requirements.

Although its definition and design has been done during cycle 2, implementation has not yet started in cycle 2. It will be carried out during cycle 3.

3.5.2 PDP

No further work has been done on Fed4FIRE PDP component. However, work on user access control is depicted in previous section.

3.6 Connectivity

SmartSantander work on connectivity is focused on inter-testbed service orchestration. This task is based on the ATOS' YourEPM tool integration. During cycle 2 there has been several discussions on this topic, but no further developments have been yet done.

3.7 Progress towards cycle 3

SmartSantander plans for cycle 3 include three main working areas:

- Finish all the pending work on SmartSantander v2 redesign to enhance IoT API usability and to better align its SEL layer with how experimentation is provided under the Fed4FIRE umbrella. Work on this tasks has been already covered on the previous sections, so no further explanations is included.

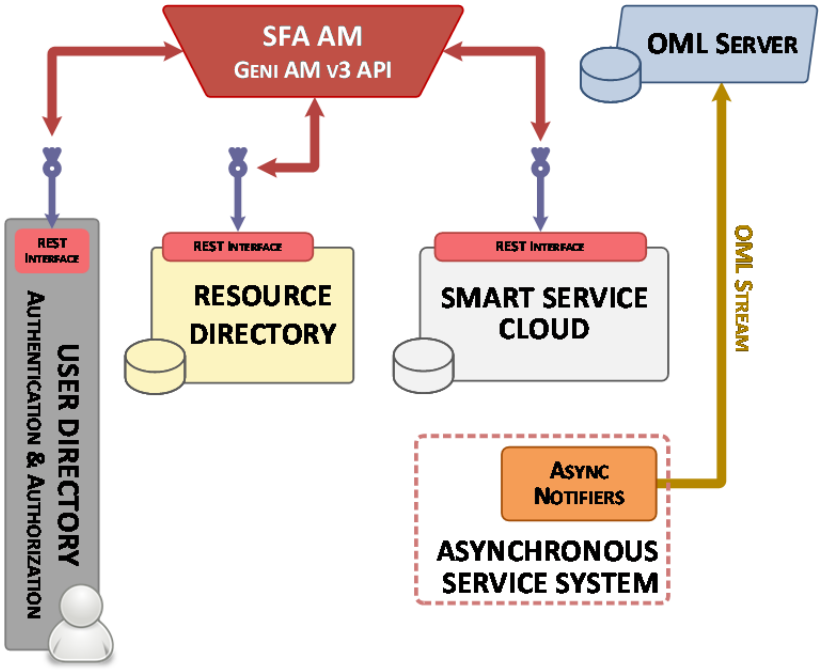

Figure 4. SmartSantander SFA based SEL experimentation

- Provide an SFA/OML based experimentation layer on top of SmartSantander SEL experimentation. The provided functionality using SFA AM interface will not be as rich as the one based on IoT API, but it will empower SmartSantander integration with the rest of Fed4FIRE involved facilities. A new SFA AM v3 component will be deployed during cycle 3, probably based on GCF flavor. Work carried out in Bonfire facility will be used as a basis for its development. In addition, basic RSpec definition versus an ontology based version is still under study. Details on how SFA AM is planned to be deployed can be seen on Figure 4.

  Usage of basic RSpec definition versus an ontology based version is still under study. The same is also applicable for secure OML extensions.
- Integrate SmartSantader IoT API together with Fed4FIRE service orchestration tool, which will be based on ATOS' YourEPM platform. Swagger v2 has been already decided as the service description language to be used for REST APIs.

4 FUSECO Playground

4.1 Facility description

FUSECO Playground is a comprehensive testbed which provides multi radio access technologies ranging from Wi-Fi 802.11 ac, to 2G and 3G as well as LTE. The objective here is to allow proof of concept or experiments in a small-scale mobile operator network. It is limited in terms of scalability, because of available femto cells and Wi-Fi access points for experimentation. Additionally it offers OpenEPC (see http://www.openepc.com) as a Service, which is required to run end-to-end experiments in a mobile operator network. The EPC clients are running on small size PCs which are equipped with USB modems to connect to the radio network. Furthermore physical servers are attached to the operator backhaul network, where the experimenter can run applications that are required for its experiments.

The Cloud infrastructure, which is based on OpenStack, allows the experimenter to launch virtual instance for their experiments. On top of OpenStack the experimenter can run a virtualized OpenIMS core and virtualized IMS clients. Recently the OpenMTC platform was added as a service to the FUSECO Playground. OpenMTC is a prototype implementation of a Machine-to-Machine (M2M) middleware aiming to provide a standard-compliant platform for M2M services, to develop innovative M2M and Internet of the Things (IoT) applications.

4.2 Cycle 2: Overview

In Cycle 2 the FUSECO Playground needs to adapt to the requirements that were coming from the experimenters of the open call phase. It was requested to provide more physical nodes which are able to run applications for the experiments. Furthermore precise time protocol (PTP) was deployed in the Playground to support more accurate measurements within experiments. The physical nodes of the wireless testbed are fully integrated in jFed, which allows the experimenter to provision and access one or multiple resources. The PDP and SLA engine Fed4FIRE software components that have been developed in Cycle 2 are integrated and tested in the FUSECO Playground.

4.3 Cycle 2: Experiment lifecycle management

4.3.1 Description, Discovery and Provisioning

The FUSECO Playground acted as a test environment for the development and testing of a semantic based SFA AM v3 implementation (FITeagle). It is planned to further support this development and to provide FUSECO Playground resources as Open-Multinet encoded graphs.

4.3.2 Experiment Control

A number of OMF RCs have been deployed for SSH nodes and they are interconnected via XMPP (OpenFIRE). Further, an AMQP server has been started (WildFly) in order to pave the way the next iteration of the AMQP based PDP and RCs

4.4 Cycle 2: Measurement and monitoring

4.4.1 Facility Monitoring

Facility Monitoring is already supported since cycle 1.

4.4.2 Infrastructure Monitoring

Infrastructure monitoring of federation services is up and running. The Zabbix monitoring tool is used to monitor the infrastructure resources. Infrastructure monitoring information for SLA and reputation services are provide for the EPC and Cloud infrastructure resources and collected in an OML collection resource provided for the federation services.

4.4.3 Experiment Monitoring

Monitoring experiment resources can be provided on-demand for experimenters and can also be provided as OML streams if required.

4.5 Cycle 2: Trust and security

4.5.1 User registration

Experimenters can register on the F4F portal or directly on any F4F facility that provides an SFA and X.509 certificate compliant identity provider. With this credentials the experimenter is able to access the resources provided by the F4F federation. The FUSECO Playground AM automatically updates the root certificates of existing CAs which are trusted within the federation.

4.5.2 PDP

The PDP server is already installed on a VM. However, the SFA interface of FITeagle needs to be extended to call the PDP and provide the user slice credentials as well as resource lifetime that are then used by the PDP server to authorize later resource control interactions. Further, it is currently on halt as FUSECO PG experimenters get SSH access and therefore there is currently no need to deploy FRCP solution that requires the PDP.

4.5.3 SLA management module

The SLA management module is running. SLA profiles have been defined but not yet deployed. Once these are deployed and integrated with the experiment tools like the portal, SLA will be ready.

4.6 Connectivity

All internal and external IP networks of the FUSECO Playground are configured to support dual stack. Currently all hosts that have IPv6 enabled configure their IPv6 addresses through SLAAC. If IPv6 access to the federation services or resources would be requested FUSECO Playground firewall rules would be configured on demand at the central firewall. Neutron of the current OpenStack release Juno doesn't fully support IPv6 at the moment. Therefore we cannot provide IPv6 access to the virtual machines that are running in the cloud infrastructure. It will be fully supported as soon as OpenStack IPv6 is available.
Currently there is no demand from the experimenters to have Layer 2 connection from FUSECO Playground to other F4F testbed facilities.

4.7 Progress towards cycle 3

4.7.1 Description, Discovery and Provisioning

It is planned to deploy the ontology enabled version of FITeagle and to provide FP resources as Open-Multinet graphs.

4.7.2 Experiment Control

On a best-effort basis further ECs will be deployed and the experimental FRCP support of FITeagle will be enabled (when available), that allows control of resources based on the same semantic description used for SFA (and therefore allow a proper handover between both protocols). Under further investigation is the deployment of the LTErf OMF RC to configure specific LTE parameters.

4.7.3 Infrastructure Monitoring

RSpecs will be extended to advertise the infrastructure monitoring capabilities as well as deployment of FITeagle-based resource adapters that support infrastructure monitoring based on a semantic resource description. Resource specific OML wrappers will be provided in order to provide infrastructure monitoring for experimenters.

5 FIONA

5.1 Facility description

FIONA is a cloud platform for creating, improving and using virtual robots, which represent a new way of interacting with technology that includes access to information, web-based services and also actions over intelligent environments.

FIONA allows users to create and use intelligent virtual robots and/or to do them more interactive, more striking and smarter. Developers, researchers, companies, geeks and designers from around the world contribute to the platform uploading their killer feature, a specific behaviour, an amazing character or the best of their knowledge in a topic, which are encapsulated as "Sparks" and that can be combined to create different behaviours and personalities for the characters to enable diverse skills and capabilities.

While the end-user of the platform is able to run it on any device, FIONA's cloud infrastructure runs the whole intelligence of the agent, and also is the platform to share the experience gained by the individual agents.

FIONA provides the mechanisms to encapsulate code of HMI technologies. It provides the required infrastructure to grab remote audio, video and text (input sensors) from a remote location and provides it to different modules (ASR, face recognition, etc.) to process an output (3D character video, TTS) which is streamed to the user.

There are three environments into Fiona:

- Sparklink: to create your Spark and upload it to Fiona. At the Sparklink people can combine different Sparks to create their avatar. It's a drag and drop thing.
- SparkStore: It's an online store where users share, either for free or charging for it, the Sparks they created or buy new ones from others. The main difference from other online stores is that users only share Fiona modules.
- Sparkrender: This is the final stage. Once the avatar is complete, users can post it on their website through the Sparkrender.

5.2 Cycle 2: Overview

FIONA joined Fed4FIRE project on December 1st 2014, and hence this report covers developments over its first five months in the project. Since then, Adele Robots has developed the integration with Fed4FIRE certificates that will be on production environment in the coming weeks.

Adele Robots has also been working on tutorials describing the testbed.

5.3 Cycle 2: Experiment lifecycle management

5.3.1 Description, Discovery and Provisioning

Regarding SFA Support, the following functionalities are under study: AM v2/v3 functionality, resource description and discovery through SFA, ontology based rspec, future reservation support (through AM) and resource provisioning through SFA.

And regarding Service Layer Support, the deployment of the application service with F4F PKCS12 credentials on Service Directory is ready, pending to be uploaded to the production environment. The following functionalities are under study: resource description and discovery support, resource reservation support and resource provision support.

5.3.2 Experiment Control

OMFv6-compliant solution is under study. SSH / OpenFlow based control and Service Layer Support are not applicable in FIONA testbed.

5.4 Cycle 2: Measurement and monitoring

5.4.1 Facility Monitoring

The facility monitoring implementation is under study.

5.4.2 Infrastructure Monitoring

The infrastructure monitoring implementation is under study.

5.4.3 Experiment Monitoring

The experiment monitoring implementation is under study.

5.5 Cycle 2: Trust and security

5.5.1 User registration

Trust and security aspects are being developed in FIONA. Authentication can be done based on the Fed4FIRE PKCS#12 derived client certificate. FIONA can authenticate different Fed4FIRE involved identity provider users and authorization can be performed based on the certificate fields.

5.5.2 PDP

No further has been done on Fed4FIRE PDP component.

5.6 Connectivity

FIONA's connectivity is based on ATOS' YourEPM tool integration. Adele Robots has been in several meetings with ATOS regarding this issue and the development is under study.

5.7 Progress towards cycle 3

For cycle 3 Adele Robots plans to:

- Finish the integration of the user registration with Fed4FIRE certificates in the production environment.
- Finish the tutorials of basic experiment showing the testbed.
- Integrate FIONA with Fed4FIRE service orchestration tool, which will be based on ATOS' YourEPM platform.

Depending on the results obtained in the tasks under study, we plan to choose to develop some of the following functionalities:

- SFA Support.
- Service Layer Support.
- OMF v6
- Monitoring
- Security

6 Conclusions

Progress in the service and application test facilities of WP4 has proceeded well in the reporting period. The emergence in the architecture of different levels of federation has enabled each test facility to implement the features of the architecture that provide access at the most appropriate levels to meet the needs of experimenters.

The overlapping development cycles in the project has allowed the test facilities to carry out developments in each cycle with a view to how they will tackle the developments required by the following cycle.

The expansion of the user base through further experiments has given the test facilities an opportunity to interact with real users. The response from users to Fed4FIRE has largely been positive from the experimenters.

In cycle 3 the test facilities will be a further consolidate the architectural features defined by WP2, and aim to move towards a position from which the federation will be sustainable.

References

[1] Fed4FIRE Architecture Workpackage D2.1 “First federation architecture”

[2] Fed4FIRE Experiment Lifecycle Management Workpackage D5.1 “Detailed specifications for first cycle ready”

[3] Fed4FIRE Measuring and Monitoring Workpackage D6.1 “Detailed specifications for first cycle ready”

[4] Fed4FIRE Trustworthiness Workpackage D7.1 “Detailed specifications for first cycle ready”

[5] Fed4FIRE Service and Applications WorkPackage “MS4.1 First design specifications for facilities”

[6] Fed4FIRE Trustworthiness Workpackage D7.2 “Detailed specifications for second cycle ready”

[7] Control and Management Framework http://mytestbed.net/projects/omf/

[8] Open Cloud Computing Interface http://www.occi-wg.org

[9] SFAWrap, http://sfawrap.info

[10] GENI Aggregate Manager API Version 2, http://groups.geni.net/geni/wiki/GAPI_AM_API_V2

[11] GRAM GENI Rack Aggregate Manager, http://groups.geni.net/geni/wiki/GENIRacksHome/OpenGENIRacks http://www.opengeni.net/opengeni-software/

[12] OpenStack, http://www.openstack.org

[13] GCF reference implementation of the GENI Aggregate Manager API, http://trac.gpolab.bbn.com/gcf

[14] Fed4FIRE Architecture WorkPackage D2.7 “Third federation architecture”

[15] Fed4FIRE Service Directory, https://portal.fed4fire.eu/portal/servicedirectory

[16] SmartSantander IoT API hub, https://iot.smartsantander.eu:8085

[17] Swagger description language specification, https://github.com/swagger-api/swagger-spec

[18] FI-Lab, The Open Innovation Lab, http://lab.fi-ware.org