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**

D5-3 – Report on first cycle developments regarding experiment workflow tools and lifecycle management

Work package	WP5
Task	T5.1
Due date	01/04/2014
Submission date	09/05/2014
Deliverable lead	Mikhail Smirnov (Fraunhofer)
Version	04
Authors	Mikhail Smirnov (Fraunhofer) Florian Schreiner (Fraunhofer) Lucia Guevgeozian Odizzio (INRIA) Alina Quereilhac (INRIA) Thierry Rakotoarivelo (NICTA) Alexander Willner (TUB) Danieln Nehls (TUB) Ozan Özpehlivan (TUB) Chrysa Papagianni (NTUA) Georgios Androulidakis (NTUA)

	Aris Leivadeas (NTUA) Donatos Stavropoulos (UTH) Aris Dadoukis (UTH) Wim Vandenberghe (iMinds) Loïc Baron (UPMC) Carlos Bermudo (i2CAT) Albert Vico (i2CAT)
Reviewers	Tim Wauters (iMinds) Yahya Al-Hazmi (TUB)

Abstract	This deliverable details the development of the common federation tools for experiment lifecycle management in the first development cycle
Keywords	Experiment, testbed, resource, service, process, specification

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

This project has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 318389. The information, documentation and figures available in this deliverable, are 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 report describes the experiment lifecycle management in the federated testbed environment as developed in cycle 1 of the Fed4FIRE project.
In Fed4FIRE, WP5 “Experiment Lifecycle Management” together with WP6 “Monitoring and Measurement” and WP7 “Trustworthiness”, deal with Fed4FIRE’s “federation-wide” mechanisms, i.e. unified mechanisms that are applied across heterogeneous testbeds facilities.
This deliverable reports the result of the implementation and practical evaluation efforts to realise what was defined in D5.1 - the first deliverable of WP5, describing the specifications for the first development cycle. As a fundamental basis, the specifications of the first development cycle in WP5 took into account and were aligned as much as possible with D2.1 “First Federation Architecture”, taking also into account D3.1 “Infrastructure Community Federation Requirements” as well as requirements laid out by D4.1 “First Input from Community to Architecture” and D8.1 “First Level Support”.
We are pleased that the implementations reported here have no major deviations from the D5.1 specification; moreover this implementation is forward-looking in that it is also aligned with the cycle 2 specification reported recently in D5.2.

Acronyms and Abbreviations

AA	Authorization and Authentication
AM	Aggregate Manager
API	Application Programming Interface
CLI	Command Line Interface
CI	Configurable Item
CMS	Content Management System
CRUD	Create-Read-Update-Delete
EC	Experiment Controller
Fed4FIRE	Federation for Future Internet Research and Experimentation Facilities
FCI	Federation Computing Interface
FRCP	Federated Resource Control Protocol
GUI	Graphical User Interface
OCCI	Open Cloud Computing Interface
OCF	OFELIA Control Framework
OF	OpenFlow
OFELIA	OpenFlow in Europe: Linking Infrastructure and Applications
OLA	Operational Level Agreement
OMA	Open Mobile Alliance
OMF	cOntrol and Management Framework
OML	ORBIT Measurement Library
OMSP	OML Measurement Stream Protocol
PDP	Policy Decision Point
PI	Principal Investigators
RA	Resource Adapter

REST	Representational State Transfer
RPC	Remote Procedure Call
RSpec	Resource Specification
SC	Service component
SFA	Slice-based Federation Architecture
SLA	Service Level Agreement
SSH	Secure Shell
UI	User Interface
URL	Uniform Resource Locator
VCT	Virtual Customer Testbed
VM	Virtual Machine
XMPP	eXtensible Messaging and Presence Protocol

Contents

List of Tables and Figures

1 Introduction

The Fed4FIRE project has successfully developed all tools specified in D5.1 [1] for the experiment lifecycle management. Furthermore, these tools no longer appear as a collection of disjoint or loosely coupled software components. In D5.2, following the sustainability strategy setup by WP2 [2], we have started looking at these tools as well as at the components of their respective services.

The check list of the components relevant for WP5 developments:

- Aggregate manager and Aggregate manager directory
- SSH server & client
- Resource controller
- XMPP server
- Experiment controller/ control server
- Scenario editor
- Documentation center
- Portal
- Authority directory
- Service directory
- Future reservation broker
- Stand-alone tools (jFed, NEPI)

We continue with an overview of all the requirements with their structure as defined in the project. We add to the requirements satisfaction the novel guarantees stemming from the fact that we are no longer working on loosely-coupled tools and components but on an integral service covering the entire experiment lifecycle toolkits addressed in WP5.

The rest of this report is compiled along the tasks structuring the WP5.

2 Inputs to this Deliverable

This section presents the main constrains and requirements that have been reported by previously submitted deliverables and are relevant to WP5 developments in the first cycle.
WP5 has specified the components to be implemented in cycle 1 in D5.1 [1].

2.1 Cycle 1 from the architectural viewpoint

D5.1 specified the following functionality to be developed in cycle 1:

Table 2-1 Cycle 1 functionality

Functional element of the Fed4FIRE architecture	Implementation strategy
Portal	Evolution of MySlice
Testbed directory	Extension of SFA API and MySlice Database, MySlice plugin
Tool directory	Extension of SFA API and MySlice Database, MySlice plugin, Wiki
Future reservation broker	Evolution of NITOS scheduler
Exposing testbeds through SFA	Initially evolution of SFAwrap, in the mid-term potential use of AMsoil
Experiment control	Cycle 1: FRCP and EC deployment on testbeds through OMF6 install. Cycle 2: add NEPI which interacts with the deployed FRCP layer.
Support of existing experimenter front-ends and tools	VCTTool and FCI, Flack, Omni, SFI

2.2 Deviations from specifications in D5.1

This section was designed as a placeholder for all deviations of what was developed from what was planned in D5.1 together with a short rationale for such deviation, if any. The rationale was assumed to refer to the evaluation provided in D5.1: was it a wrong or a subjective judgement? Was it due to newly available information? Were the design boundaries in D5.1 wrong?
Since not a single deviation was reported by all partners involved in WP5 we happily conclude that all evaluations deliberately specified in D5.1 are still correct at the time of this writing.
Nevertheless, this intentionally idle section is kept in the body of this report as a reminder for future work and for the subsequent releases of this document.

2.3 Meeting the Requirements

2.3.1 General Requirements

In order to ensure a sustainable experiment lifecycle management as a service, we repeat the general requirements below so that they can be applied to any newly defined lifecycle management service, regardless of the fact that this service was or was not yet on the agenda when cycle 1 specification was defined., The answers for the general requirements are extended as compared to D5.1 with the service-related guarantees.

Among the requirements, those of major concern are as follows:

- **Scalability:**
    - Question: How can the architecture cope with a large number and a wide range of testbeds, resources, experimenters, experiments and tools?
    - *Answer: As tools can speak directly to the testbeds through SFA in a peer-to-peer paradigm, this is inherently scalable. The same applies for multiple identity providers with a chain of trust model. For deployment of very large experiments over multiple testbeds, experimenter tools can directly talk to all testbeds or through broker services that can orchestrate this (making the experimenter tool simpler).*
    - *Service answer: since service-orientation is hierarchical with service components (even with configurable items underpinning these service components that are able to operate either within the service or as a standalone tool), this makes the entire system inherently scalable. accordingly, in future workWe shall proceed with the definition of scalability metrics, which, obviously, shall contribute to sustainability metrics within the dynamics of the federation.*

- **Support:**
    - Question: How easily can components/testbeds/software be upgraded?
    - *Answer: For this, the APIs should be versioned and tools and testbeds should support 2 or 3 versions at the same time, so that all components can be gradually upgraded.*
    - *Service answer: again the service hierarchy with the preserved interfaces facilitates various levels of easy upgrades.*
    - Question: How can different versions of protocols be supported? (e.g. upgrade of RSpec)
    - *Answer: With versions.*
    - *Service answer: new versions of service must be backwards compatible with older versions until a certain threshold, which, when met, triggers a new SLA definition.*

- **Experimenter ease of use:**
    - Requirement: The final goal is to make it easier for experimenters to use all kinds of testbeds and tools. If an experimenter wants to access resources on multiple testbeds, this should be possible from a single experimenter tool environment.
    - *Answer: It is possible, but Fed4FIRE should also aim to keep such tools up-to-date during the lifetime of the project and set up a body which can further define the APIs, also after the project.*

- *Service answer: when fully defined and implemented the experiment lifecycle management service shall make almost all configurations invisible that are redundant from the purpose of the experiment.*

2.3.2 Sustainability Requirements

At the beginning of cycle 1 sustainability requirements were understood mainly as the easiness of the following actions during a sustainable flow of experiments:

1. Easy to add / drop a testbed
2. Easy to add / drop a tool
3. Easy to add / drop an experimenter

In order to provide sustainable lifecycle management services, despite the fact that tools and service components are added to or dropped from the federation framework occasionally, we aim for a service and process orientation during the design and implementation, using the FitSM (http://www.fedsm.eu/fitsm) approaches and templates.
Such lifecycle management services include:

- Resource Description and Discovery;
- Resource Reservation;
- Resource Provisioning ;
- Experiment Control;
- User interface / portal.

This approach will further guarantee the automation of handling of all kinds of possible incidents as well as continuous improvements of services and service components within the WP5 Service hierarchy (Figure 2-1), as defined in D5.2 [6].

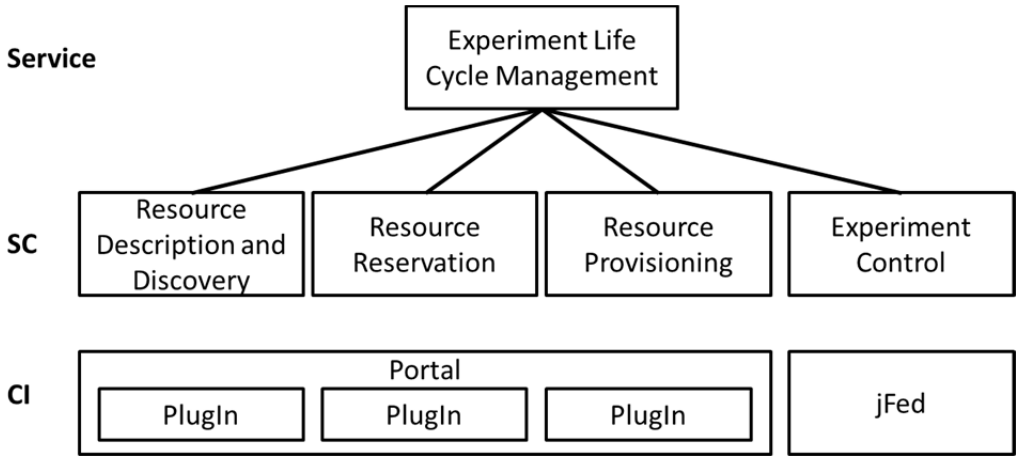

Figure 2-1 WP5 Service Hierarchy
(Legend: SC – Service Component, CI – Configurable Item)

2.3.3 Infrastructure community requirements

The Remark field in the Table 2-2, which refers to all components, is the accurate statement of the actual development after cycle 1. (Note that non-requirements and those not planned for cycle 1 are removed) Requirements with an orange background were planned for a partial satisfaction in cycle 1.

Table 2-2 Cycle 1 requirements are satisfied

Federation aspect	Req. id	Req. statement	Remark
Resource discovery	I.1.101	Node capabilities	In cycle 1, the basic node capabilities can be retrieved in the same way across the facilities. Not all facilities provide all capabilities, but provide the most tangible for their infrastructures. In cycle 2 and 3 this will be further refined to an ontology model of the resources.
Resource discovery	I.1.105	Discovery through federation-wide APIs	SFA is adopted for this on all testbeds.
Resource discovery	I.1.106	Intra-infrastructure topology information	Provided as is by the facilities, but will be refined further in cycle 2 and 3 with the ontology model
Resource discovery	I.1.109	Query search	Based on the resource discovery of SFA, this is possible. Brokers (with the same interface) can help in doing this on multiple testbeds at once.
Resource discovery	I.1.110	Catalogue search	Tackled by the portal which lines up all testbeds, and user tools which listen up all resources based on the discovery phase.
Resource requirements	I.1.201	Manually extract requirements from discovery query results	When the query in the discovery phase returns a certain list of resources, it is possible for the experimenter to select the resources he/she would like to include in the experiment. This is supported in relation with a specific resource ID (e.g., *I want this specific node at this specific Wi-Fi testbed)*.
Resource reservation	I.1.301	Hard resource reservation	SFA is extended to cope with future reservations. Testbeds implement hard reservation (not an architecture requirement).
Resource reservation	I.1.304	Automated reservations handling	SFA is extended in this way, brokers can handle this in a multi-testbed way.

Resource reservation	I.1.305	Reservation information	SFA is extended in this way
Resource provisioning	I.1.401	Provisioning API	Done by SFA
Experiment control	I.1.501	SSH access	Architecture makes it possible to distribute SSH public keys as part of the SFA API ("geni_users" option in the provision call).
Experiment control	I.1.502	Scripted control engine	Architecture facilitates the use of experiment control engines
Inter-connectivity	I.4.005	IPv6 support	In cycle 1, some testbed resources are reachable through IPv6. The architecture can cope with this, e.g. through DNS names which resolve to IPv6

2.3.4 Service Community Requirements

The Remark field in the Table 2-3, which refers to all components, is the accurate statement of the actual development after cycle 1. (Note that non-requirements and those not planned for cycle 1 are removed) Requirements with an orange background were planned for a partial satisfaction in cycle 1.

Table 2-3 Service community requirements are satisfied

Field	Req. id	Req. statement	Remark
Resource Provisioning	ST.1.007	Experiment Deployment	Architecture makes it possible to deploy such tools.
Resource Requirements	ST.1.013	Resource description	Identical to I.1.101
Resource Reservation	ST.1.017	Testbed reservation information and methods according to experimenter profiles/policies	Similar to I.3.201
Resource Reservation	ST.1.020	Scheduling parallel processing	The architecture in cycle 1 can cope with this, but interconnectivity is not supported yet in a structured way.
Experiment Control	ST.1.029	Multiple testbeds available	The architecture in cycle 1 can cope with this, but interconnectivity is not supported

			yet in a structured way. Some testbeds can probably use public IP addresses (IPv4 or IPv6) which can be used for inter testbed communication over the public internet.
Authorization	ST.3.002	Single sign on for all testbeds required for experiment	This is tackled by using certificates signed by identity providers. Once you upload this certificates in the experimenter tool and provide a passphrase, you can use it on all testbeds.

3 Implementation of experiment life-cycle management (cycle 1)

This section is structured along with the organization of WP5 in its tasks.

3.1 Resource description and discovery (Task 5.2)

3.1.1 Semantic Aspects

D5.2 provides an extensive motivation and description of a semantic resource directory offered as a service, being at the same time a service component within the entire WP5 service. This approach is in line with the latest trends to link data in the web. Communities in the fields of federated cloud computing (e.g. IEEE P2302) and Internet of Things (e.g. OneM2M) are heading in the same direction to semantically describe resources based on RDF and OWL. A related example is the myExperiment **Error! Reference source not found.**[5] semantic workflow directory at http://rdf.myexperiment.org. More public SPARQL endpoints are listed at http://www.w3.org/wiki/SparqlEndpoints.

This defined semantic resource directory was not planned from cycle 1 implementation, though we are pleased to report here that its integration in the Fed4FIRE architecture is feasible without any changes to the former and was described in D5.2 for the main cases :

- Semantic Resource Directory: Discovery, Reservation and Provisioning
- Semantic Resource Directory: Monitoring and Measurements
- Semantic Resource Directory: Experiment Control

which proves that the architectural choices made are viable.

Next to that T5.2 has already performed some preparatory steps in cycle 1 towards the adoption of a semantic resource description in the later cycles of the project. It has collected the different attributes that all testbeds want to expose to the outside world about their resources. These attributes are related to different kinds of resources, such as nodes (embedded PC, server, etc.), virtual machines, specific resources (software defined radio modules, network sniffers, etc.), links, and the testbeds by themselves, etc. Other attributes have to do with more abstract concepts such as reservations or wireless spectrum. This collection activity resulted in an Excel sheet containing over 300 different attributes. This is valuable input to create a first ontology-based information model. Building further on these results, T5.2 has also defined some possible ontology starting points (first classifications of the object space) in the context of the Fed4FIRE federation. It has successfully explored that INDL and NDL-OWL could be used as a starting point when trying to semantically model some of the Fed4FIRE testbeds (iMinds Virtual Wall and w-iLab.t).

Next to these activities that investigate the information model, using the experimenter tool jFed, T5.2 has also explored the possibilities of abstracting the underlying technical details of resources by classifying them in high-level categories (generic node, wireless node, physical node, OpenVZ VM, XEN VM, Virtual Machine) and presenting them to the experimenter as such. Although this representation is currently hard-coded in the tool, this gives some first hands on experience with the user experience that ontologies could bring to experimenter tools.

3.1.2 Novel resources: discovery of application services and the documentation center

This functionality was not planned in the cycle 1 specification however in the cause of the project it was identified that it is reasonable to extend the concept of a resource to cover also application level

services, perhaps developed outside of the project but available through the project partner facilities.
In D5.2 [6] (Section 3.1.2.1) sufficient motivation for this new functionality is provided together with details on:

- authorisation and authentication of application level services,
- quality control of [application] service directory entities,
- application service representation with the details on the service directory API.

Another novel component within the resources is the documentation center. The documentation center defined by WP2 in D2.4 can be seen as a superset of the components of the cycle 1 architecture that were then called the human-readable testbed directory and the tools directory. The documentation center is expected to contain both types of information (catalogue of Fed4FIRE testbeds, and information about how to use Fed4FIRE tools on them). The HTML version of the documentation center can be accessed on http://doc.fed4fire.eu. It is driven by the Sphinx Python Documentation Generator software that can be downloaded from http://sphinx-doc.org. This is a tool that makes it easy to create intelligent and attractive documentation. A screenshot of homepage of the documentation center is given in Figure 3-1. Note that this is currently still a work in progress, but it has been already populated adequately to support our first wave of open call experimenters.

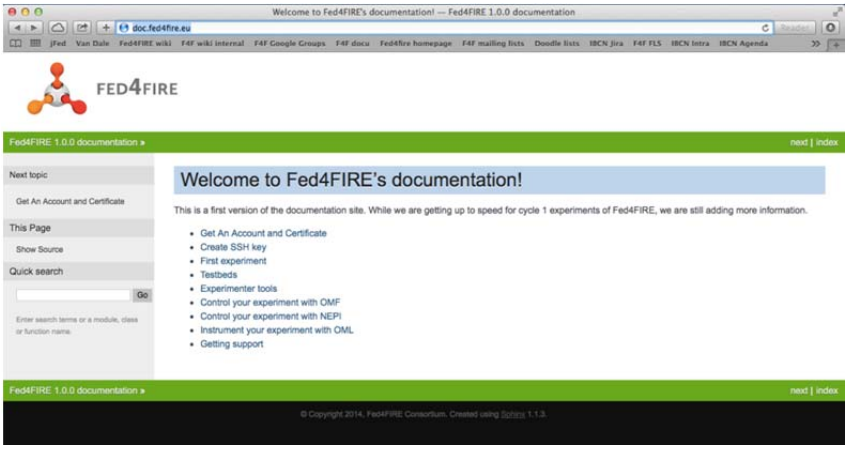

Figure 3-1: Homepage of the documentation center

3.3 Resource reservation (Task 5.3)

During the first cycle of development of the Reservation Broker, all involved parties worked towards a fully functional brokerage system that is described at the predecessor of this document [1]. Having that in mind, a Reservation Broker was deployed locally in a testbed as a first step towards deploying a Central Reservation Broker that will be in charge of keeping an inventory of all the available resources advertised by all federated testbeds and serving reservation requests regarding those resources.

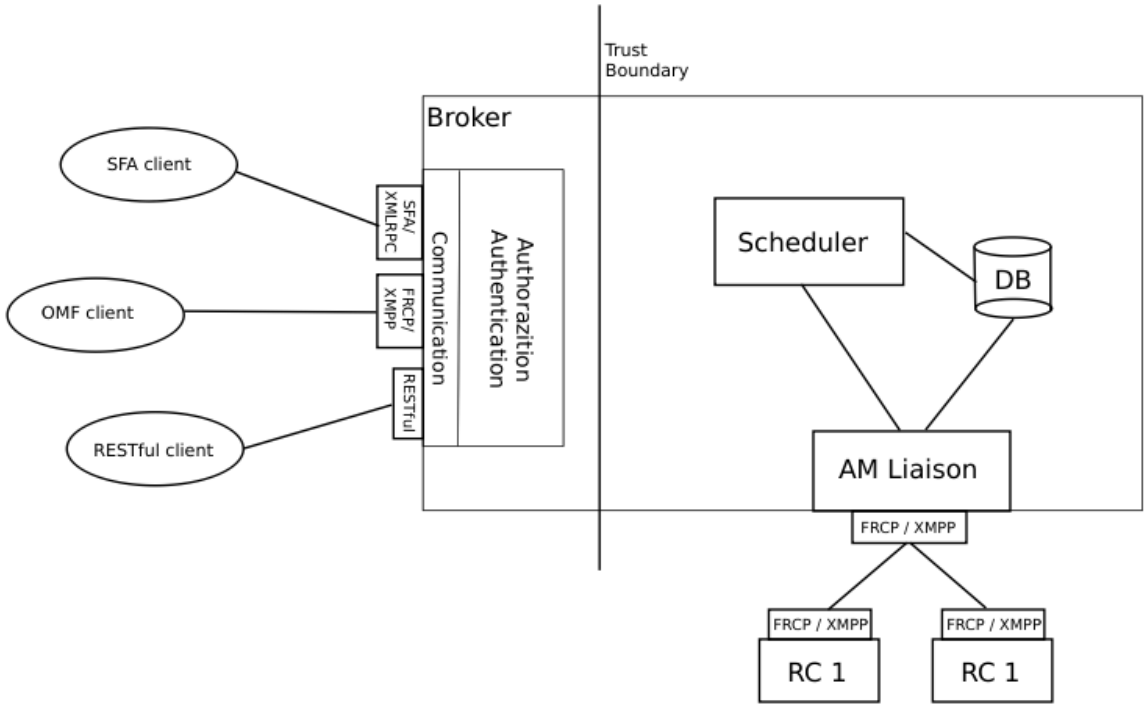

Figure 3-2 Reservation Broker Architecture
(Legend: RC – Resource Controller)

Figure 3-2 depicts the main components of the Reservation Broker and their interactions. In the following, we outline the modules that constitute the Reservation Broker.

- **Communication Interfaces**: the Reservation Broker can communicate with other components of the Fed4FIRE ecosystem via:
  - XML-RPC / SFA interface that supports SFA clients.
  - XMPP / FRCP interface that supports FRCP messages to control resources at the testbed infrastructure level through their respective resource controllers (RC)
  - RESTful interface that supports 3rd party clients.
- **Authentication / Authorization:** is the policy enforcement point of the Reservation Broker.
- **Database (DB):** contains all the required information of the available resources.
- **Scheduler:** handles requests for resource reservation.
- **AM Liaison: is** responsible for contacting FRCP resources.

SFA is one of the most important messaging protocols that are supported in the Fed4FIRE architecture; in this context an XML-RPC interface that serves SFA messages was implemented.

Alongside with the development procedure of this interface, the XML-based resource specification language RSpec (GENI v3) has been extended with reservation information. An example of RSpecs that contain reservation information can be seen in Figure 3-3:

```
<?xml version="1.0"?>
<rspec xmlns="http://www.geni.net/resources/rspec/3" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:ol="http://nitlab.inf.uth.gr/schema/sfa/rspec/1" xmlns:omf="http://schema.mytestbed.net/sfa/rspec/1"
  type="manifest" xsi:schemaLocation="http://www.geni.net/resources/rspec/3
  http://www.geni.net/resources/rspec/3/manifest.xsd http://nitlab.inf.uth.gr/schema/sfa/rspec/1
  http://nitlab.inf.uth.gr/schema/sfa/rspec/1/request-reservation.xsd" generated="2013-12-09T17:15:19+02:00">
 <ol:lease id="9e2893b9-af8e-4d60-9b5d-4d1836475a39" client_id="l1" valid_from="2013-01-08T21:00:00+02:00"
    valid_until="2013-01-08T22:00:00+02:00"/>
  <node client_id="my_node" component_id="urn:publicid:IDN+omf:xxx+node+node1"
      component_manager_id="urn:publicid:IDN+omf:xxx+authority+am" component_name="node1" exclusive="true">
   <available now="true"/>
   <interface component_id="urn:publicid:IDN+omf:xxx+interface+node1:if0" component_name="node1:if0">
    <ip address="10.0.1.1" ip_type="ipv4" netmask="255.255.255.0"/>
   </interface>
  <ol:lease_ref id_ref="9e2893b9-af8e-4d60-9b5d-4d1836475a39"/>
 </node>
</rspec>
```

Figure 3-3 Broker Reservation RSpec

SFA though, is not the only interface of the Reservation Broker developed in cycle 1, an XMPP interface that supports all FRCP messages (the messaging protocol of OMF) was also deployed together with a RESTful interface that can support any 3rd party RESTful clients. Those three interfaces are more than sufficient for the Fed4FIRE ecosystem.

Towards the direction of providing information for exclusive (reservable) resources to the user, a "Scheduler" plugin was designed and developed which is integrated in the new Django version of MySlice. Through this plugin, an experimenter is able to discover and select exclusive resources exposed by the federated testbeds and include them in his/her experiment. The plugin can depict reservable resources from different testbeds, with diverse granularities and various reservation policies. Once the experimenter selects the desired date, he/she is able to see the availability of each of the resources during the selected day and select according to the policies applied by each facility. A screenshot of the Reservation plugin is depicted in Figure 3-4:

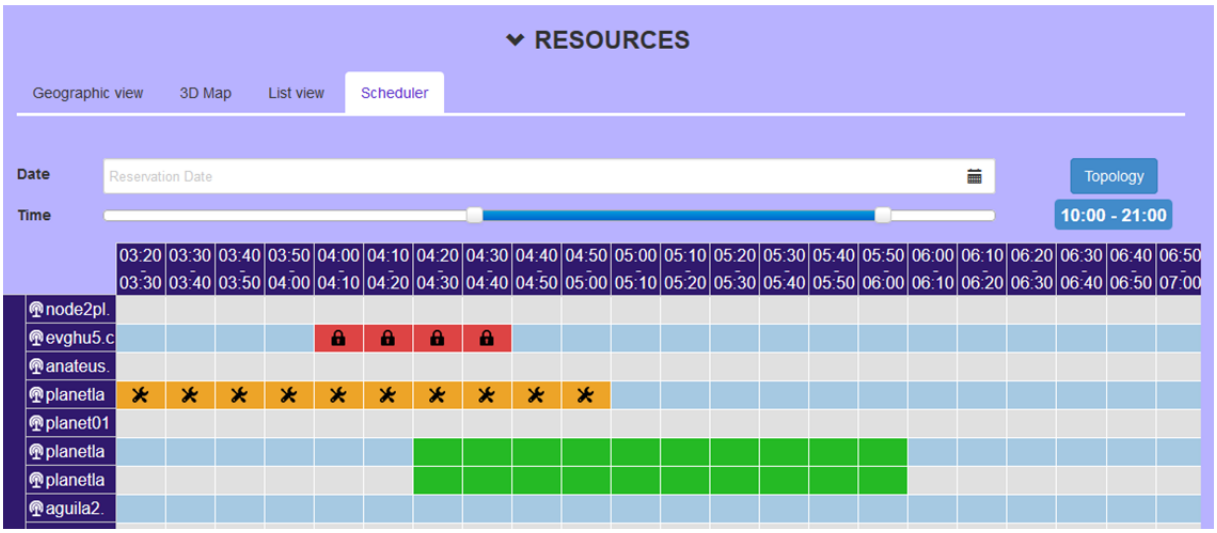

Figure 3-4 Scheduler plugin

Another important aspect of the Reservation Broker is its database scheme, which was enriched in terms of extensibility of the model in a way that it can be extended according to the needs of the testbed operator and its resources. This highly dynamic database model is the core of the Reservation Broker and will always be evolving during all the cycles of development of Fed4FIRE.

To summarize, in the first development cycle, the main goal for the Reservation Broker was to become a functional part of the federated testbeds by achieving two objectives, i) a local reservation mechanism fully interoperable with OMF, and 2) a structure to expose this mechanism through various interfaces, with emphasis given to SFA. In the second development cycle, the efforts will be focused in all the necessary modifications. This allows creating a central instance of the Reservation Broker that will act as the brokerage system in the Fed4FIRE architecture.

3.4 Resource Provisioning (Task 5.4)

As commented in [1], resource provisioning in Fed4FIRE will be carried out by each federated testbed by being exposed through SFA. GENI SFA v2 was the chosen API aimed to perform this task during cycle 1. GENI v2 was the most extended version and some of the testbeds already have it in place. As there are many implementations of SFA interface, it was left to each testbed to choose the tool to be used to perform this task and to deploy it, although there were two recommended tools: SFAWrap and AMSoil.

Next is a short description of the efforts performed during Cycle 1 by some testbeds to be able to provision their resources through SFA.

OFELIA testbed exposes its resources through two AMs: one for virtualised resources (VT AM) and another for OpenFlow resources (OptIn AM). Those AMs were not SFA-compliant, so an adaptation was required and performed. Of the suggested tools in [1], SFAWrap was the first option, as it provided GENI v2 interface. SFAWrap proved to be not fully compatible with OFELIA AMs, so an adaptation of its interfaces was performed to provide these functionalities to the VT and OptIn AMs. During cycle 1, the OpenFlow technology used in OptIn AM proved to be deprecated, so in the OFELIA project, an update of this AM was performed to use FOAM [3]. As FOAM uses GENI v3, while it was being developed, and foreseeing future compatibilities, the adaptation of the SFAWrap to the OptIn AM was done using GENI v3, so when finally OptIn was replaced by FOAM, so there was no need to do any adaptation regarding the interface. At the same time, in the OFELIA project, an

adaptation of the VT AM to the AMSoil framework was being performed which will support GENIv3, so for the same reason, until the AMSoil version of VT AM was available, the SFAWrap adaptation was done using GENI v3.

At this moment, the OFELIA AMs exposed are OptIn and VT AM with the SFAWrap adaptation providing GENI v3 API. It is expected that during cycle 2, OptIn AM will be replaced by FOAM and VT AM by a new version based in AMSoil. As all the AMs are GENI v3 compliant, no major issues are expected regarding resource provisioning.

NITOS testbed has also implemented the SFA API through the adoption of the SFAWrap during cycle 1 and exposed its resources. A NITOS driver for the SFAWrap (which has been included in the SFAWrap repository as part its sample drivers) was implemented to support SFA GENI v2 interface. This driver exposes the testbed functionalities, specifically the OMF 5.4 AM services related to resource discovery and provisioning. These modifications were related with other states of the experiment lifecycle, e.g. the SFA RSpecs modifications allow supporting reservations through the NITOS Scheduler. Aiming for the cycle 2, NITOS expects to support SFA through the OMF 6's native SFA interface and upgrade its current GENI AM v2 interface to v3.

For cycle 1, in the Smart Santander testbed only service layer experimentation is provided, so the nature of the resources exposed made the provisioning stage unnecessary. Anyway, an SFA AM based on SFAWrap that provides access to the testbed by using messages defined by the GENI AM API v2 was deployed. Although only a subset of the different methods that the API specifies has been deployed (testbed description is fully implemented and resource discovery is being developed). More details about this development can be found in section 3.3.1 of [4].

BonFIRE also chose SFAWrap to provide SFA GENI v3 compliance to the testbed. But as the BonFIRE API is based on the Open Cloud Computing Interface (OCCI) [7], it required to map the concepts from their resource management systems and to generate software to implement the mappings so SFA compliance is a goal for cycle 2, although work has already started. Until then, Fed4FIRE experimenters must provision BonFIRE resources by accessing the testbed's OCCI-based mechanisms.

As described in [4], the FUSECO Playground adopted the GENI RSpec v3 and the SFA AM v3 API. This was realized by using the FITeagle framework and further extensions have been made in this context. Namely the "SFA getVersion method call response struct" has been extended to offer information about the testbed and related tools.

To assist the adoption of the SFA API by the Fed4FIRE testbeds, T5.4 has developed and released a first version of jFed, a framework developed in Java to support testbed federation according to the SFA architecture. It can be used for both the manual (GUI based) and automatic nightly testing (command line based) of all API calls of the GENI SFA AM API versions 2 and 3, of the Emulab and PlanetLab Europe SFA Slice manager APIs, and of the GENI Clearing House. This tool is the key for ensuring that all Fed4FIRE testbeds will be fully compliant with the chosen SFA API to support resource provisioning. In practice jFed has also revealed itself as a valuable teaching tool, bringing people new to the SFA world quickly up to speed with the technical details of these APIs. This will lower the threshold for supporting resource provision through Fed4FIRE for any new facilities that could join the federation in the future. More information about this tool can be found at http://jfed.iminds.be. A screenshot of the corresponding functionality to manually test compliance with Fed4FIRE's adopted API for resource provisioning is given in Figure 3-5.

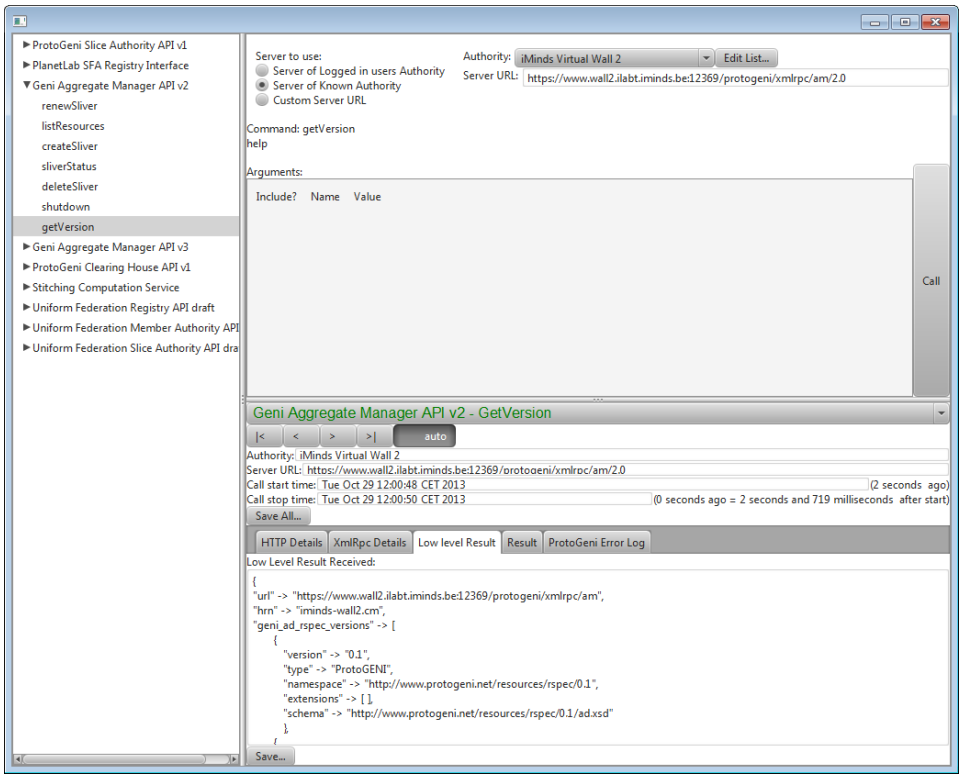

Figure 3-5 jFed probe for manual compliance testing and API learning

During cycle 1, T5.4 has also contributed to the process of harmonizing the small differences that today can be observed in the different implemented flavours of the SFA API. This activity is also related to WP8 of Fed4FIRE, and tries in the first place to harmonize the SFA Aggregate Manager API. This work is being done together with the relevant partners from GENI (US) and the rest of FIRE, and is expected to result in the first Common AM API, which will follow up the current GENI AM API v3. After this, T5.4 is considering to continue this harmonisation process on the SFA slice and member authority APIs. In this context iMinds has already written down some ideas and shared them with the community when contributing to a first draft of a possible SFA Clearing House API. This however needs to be further discussed within the Fed4FIRE consortium.

3.5 Experiment control (Task 5.5)

This section is structured in two parts: we first report NEPI developments that are followed by the OMF report for cycle 1.

3.5.1 NEPI report for cycle 1

During cycle 1 we fully re-implemented NEPI in order to support new important requirements that arose from our participation in the Fed4FIRE project.

Among those requirements we can enumerate the ability to dynamically provision experiment resources during the experiment run time (i.e. add new resources to the experiment at any point of time), the interactive experiment execution (i.e. NEPI can be used as an interactive experiment management tool), and the description of experiment workflows (i.e. allow to include execution/deployment dependencies across resources). This re-implementation of NEPI gave place to an improved, more extensible and user friendly framework which was officially released as NEPI 3.0 in December 2013. This new version of NEPI was shipped with support for new testbeds,

including any testbeds supporting SSH key authentication and OMF (cOntrol and Management Framework) enabled testbeds. Support for OMF technology is a very important feature since it is a key part of the federation control and management framework proposed in the Fed4FIRE project.
The version of OMF currently supported by NEPI is 5.4. OMF version 6, which is the new mainstream release, will be supported during cycle 2. In order to comply with the requirements of Fed4FIRE federation architecture, work is undertaken in NEPI to fully support SFA (Slice Federation Architecture), in its latest version (APIs and RSpecs), for resource discovery and provisioning across federated testbeds. Additional improvements to the NEPI framework during cycle 1 include out-of-the box support for Future Internet technologies such as Content Centric Networking and OpenFlow for certain testbeds.
Finally, during cycle 1 a new NEPI web site was released, including an improved look and feel (http://nepi.inria.fr), user manual and code reference pages, detailed experiment examples and an issue tracking page.

3.5.2 OMF report for cycle 1

The OMF Experiment Control Framework is progressing along the four following points.

1) Integration of previous OMF 5.4 features into OMF 6

The latest version 6 of OMF has been a complete re-design from the previous version 5.4. This re-design was made necessary as we adopted a new model for resource handling, and the new FRCP protocol to exchange control requests and information between controllers and resources. To focus on the timely delivery of the first stable releases of OMF 6, we deliberately left aside some of the less used OMF 5.4 features for later implementation.
The gradual implementation of new features in OMF 6 can be exemplified by the support for custom user-defined events. In the previous 5.4 version, a user was able to define a custom event based on the state of specific resources or certain measurements that they are collecting. While some user-defined event support currently exists in OMF 6, it does not yet support this scenario, and we will extend it to do so. Another example of such a feature is the support for loading different parts of an experiment from different standalone files, e.g. one file containing the main experiment description together with multiple other files with application definitions. These files could be local to the experimenter's platform or available remotely via URI addresses.

2) Support long-running experiments

Current OMF-based experiments are executed in the context of an Experiment Controller (EC) instance. Thus when the EC instance is terminated, so is the experiment. In many experimental scenarios, the experimenter should be able to terminate the EC process used to launch the experiment, while still having the involved remote resources continue running whatever tasks required by the experiment. At a later time, the experimenter should be able to start a new EC instance, "re-attach" it to the running experiment, and issue further request/control commands to the remote resources.
The implementation of such a long-running experiment feature requires many separate smaller features reported in D5.2 (cycle 2 specifications).

3) Improved scalability

The communication between the OMF entities has so far been using a XMPP-based publish-and-subscribe system. The publish-and-subscribe paradigm offers many interesting properties such as being asynchronous, supporting any-to-any messaging, scaling to a large number of entities. However, following recent performance tests we identified some limits of the current XMPP-based solution that OMF is using, as well as some internal communication-related inefficiencies within the EC. These issues effectively limit the number of resources and the number of experiments that could be supported by the current OMF entities. This is clearly an issue in the context of Fed4FIRE, since one of the goals of federating all these different facilities is to enable large-scale experiments.
We have started addressing these inefficiencies and testing a new publish-and-subscribe scheme based on Advanced Message Queuing Protocol. While the initial results are promising, i.e. we manage to support more resources and experiments, our current investigations are not yet completed. We plan to integrate the results of this work in a production-ready release of OMF 6 and update its documentation accordingly.

4) Support new resources
The list of resources supported by the OMF Experiment Control tools has been growing steadily (e.g. PC-based, KVM-based virtual machines, software application, Android phones). We plan to add some new resources to this list, such as GENI Racks, Amazon Cloud-based virtual machines.

3.6 User Interface / Portal (Task 5.6)

3.6.1 Cycle 1 Portal achievements

The Fed4Fire portal has been developed in two phases in cycle 1. The first phase focused on the development and the deployment of MySlice v1 as a client tool enabling users to browse resources of the testbeds participating in the Fed4FIRE. FUSECO playground, Netmode, NITOS, PlanetLab Europe, Virtual Wall and w-iLab.t were the first testbeds to be available through the portal as presented during the first review of the project in July 2013 (M10).

The web frontend of the Fed4FIRE portal has been quickly developed inheriting some code from OpenLab project using PHP and JavaScript languages; whereas the server side of the portal relies on Manifold, which is written in Python. An XMLRPC API is responsible of the communication between the web frontend and the Manifold backend. Since Manifold and SFA Wrapper have been developed in Python, a decision was made to use Python to further develop the web frontend instead of PHP during the phase 2 of the first cycle. This implementation choice is based on the benefit to easily import classes from Manifold and SFA Wrapper into the web frontend. However, the plugins developed mainly in JavaScript in the first version have been easily imported into version 2 of MySlice. This new version is using the Django framework. The Fed4FIRE portal based on MySlice v2 has been deployed in a test environment during the Fed4FIRE plug fest in November 2013 (M14). The release of a production version primarily intended to Open Call Experimenters has been deployed in January 2014 (M16) as planned in the DoW (MS52). The following functionalities have been developed during the first cycle, as described in D5.1 (3.2 Portal): registration, authentication, authorization, testbed resources (browse, filter, and reserve), plugins, API.

T5.6 has also developed a mainly drag-and-drop based graphical user interface (GUI) in which experimenters can select their desired resources, and easily design the topology according to which they should be interconnected. The GUI is designed in such a way that as much underlying technical details are abstracted from the experimenter as possible, without sacrificing the possibility for experienced experimenters to control every aspect of the experiment. At the moment, the focus of the developments of this GUI lies on the discovery and provisioning of resources, defining topologies, and supporting very easy SSH login on the nodes. This functionality is part of the jFed framework, more specific the jFed UI component. It will be seamlessly integrated in the portal using Java Web Start Technology. In the next phase of the project, additional functionalities will be explored, such as experiment control and experiment measuring and monitoring. A screenshot of this GUI is given in Figure 3-6.

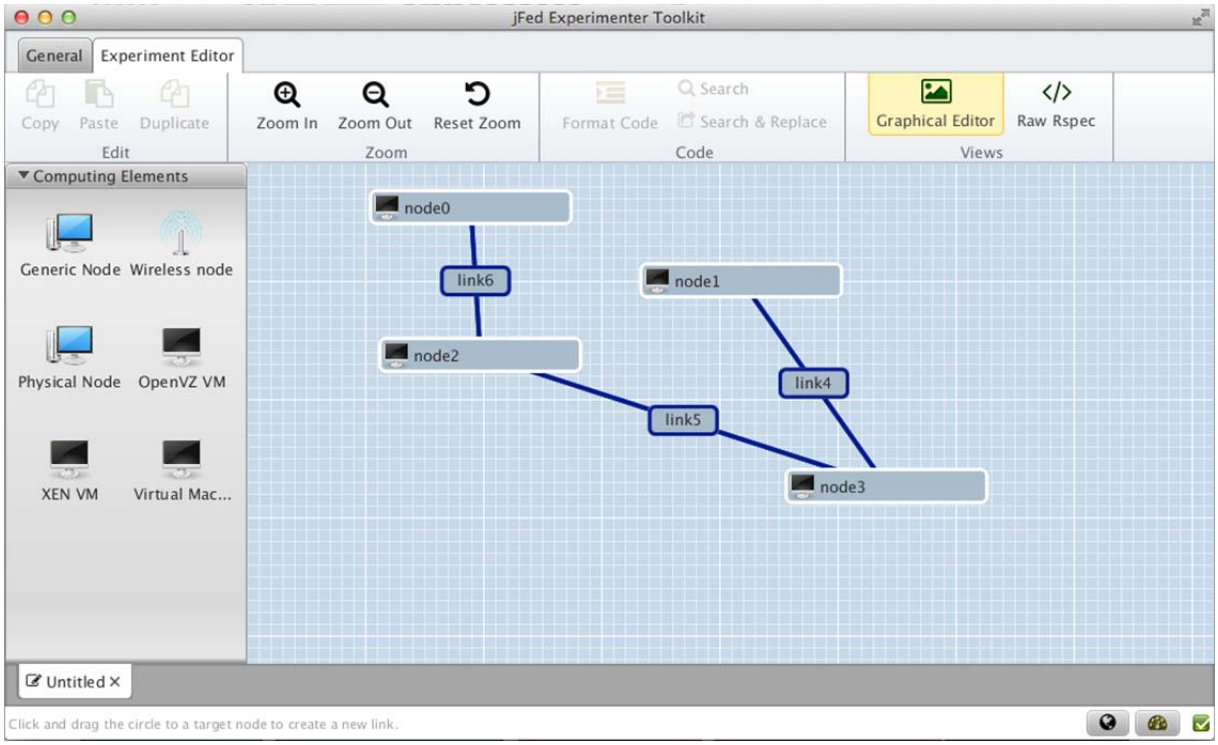

Figure 3-6 screenshot of the jFed graphical user interface

4 Conclusion and Future Plans

The experiment life cycle management is specified as a service in conformance with the emerging industry standard for service management in federated IT infrastructures. This strategy is in line with the plans for cycle 1 implementation, meaning that all developed tools appear now either as service components or as configurable items underpinning these service components.

In this document we report the developments made after the initial specification (D5.1) of tools for experiment lifecycle management.

This development occurred within the project environment concurrently with the

- development of cycle 2 architecture,
- discussions on the Big Issues;
- joint work with GENI,
- sustainability of Fed4FIRE infrastructure supported by FitSM for service- and process orientation, and lastly, by
- discussions of interworking with other federations (e.g. XiFi project).

Although several minor corrections to these specifications have occurred following these collaborations, in general no deviations are to be reported from the original specification and evaluation.

However, it should be noted that the major difference of cycle 2 evaluation from that of cycle 1 consists in the change of operational paradigm. The cycle 2 paradigm is that of an operational federation of facilities, in which experiment life cycle management service (yet under completion) is not only going to be demanded but also tested by the real-life requirements defined by the experiments from the first Open Call and from the SME Call.

Since it is hard to underestimate the importance of these real-life requirements - hence these experiments are the so called “early adopters” - for the service defined within WP5, the future work must proceed not only along the fine-tuning and tailoring of service components developed but also must deepen the evaluation of lessons learnt in the course of this operations and drive future developments.

References

[1] Fed4FIRE. (2013). *Detailed specifications for first cycle ready (Deliverable 5.1).* EU: Fed4FIRE Consortium.

[2] Fed4FIRE. (2014). *Second Federation Architecture (Deliverable 2.4 version 8).* EU: Fed4FIRE Consortium.

[3] FOAM: http://groups.geni.net/geni/wiki/OpenFlow/FOAM

[4] Fed4FIRE (2014) Report on first cycle developments of the services and applications community (Deliverable 4.3). EU: Fed4FIRE Consortium.

[5] D. Newman, S. Bechhofer, and D. D. Roure, "myExperiment: An ontology for e-Research," 2009.

[6] Fed4FIRE. (2014). Detailed specifications regarding experiment workflow tools and lifecycle management for the second cycle (Deliverable 5.2 version 12). EU: Fed4FIRE Consortium.

[7] OCCI Open Cloud Computing Interface: http://occi-wg.org/