Grant Agreement No.: 731677
Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

D3.6: FIRE+ Technology Assessment

Pouria Sayyad Khodashenas, Carolina Fernandez, Dani Guija, August Betzler, Marisa Catalan (i2CAT)

19/04/2018

This deliverable provides an analysis of the experimenter tools available from FIRE+ projects and targeted for the experimenters to ease the definition and set-up of their experiments, either at a specific infrastructure or for a subset of them.

Work package	WP 3
Task	T3.6
Due date	28/02/2018
Submission date	19/04/2018
Deliverable lead	i2CAT
Version	1.0
Authors	Pouria Sayyad Khodashenas, Carolina Fernandez, Dani Guija, August Betzler, Marisa Catalan (i2CAT)
Reviewers	Michael Boniface (ITInnov), Dirk Trossen (IDE)
Keywords	Tools, FIRE+, operations

DISCLAIMER

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731677.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	**R**	
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CL	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to FLAME project and Commission Services	

EXECUTIVE SUMMARY

This document aims to describe the different tools within FIRE+ that can be adopted for the Experiment Toolbox and assess the feasibility of adopting these within the FLAME platform, considering its requirements.

Co-funded by the Horizon 2020 Framework Programme of the European Union

TABLE OF CONTENTS

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF FIGURES

Co-funded by the Horizon 2020
Framework Programme of the European Union

LIST OF TABLES

Co-funded by the Horizon 2020
Framework Programme of the European Union

ABBREVIATIONS

AM	Aggregate Manager
API	Application Programming Interface
BPM	Business Process Management
CLI	Command Line Interface
EC	Experiment Controller
ED	Experiment Description
FIRE+	Future Internet Research and Experimentation
FRCP	Federated Resource Control Protocol
GENI	Global Environment for Network Innovations
GID	Global Identifier
GMOC	GENI Meta Operations Center
GUI	Graphical User Interface
HRN	Human Readable Naming
IoT	Internet of Things
LDAP	Lightweight Directory Access Protocol
MCS	Measurement Collection Server
ML	Measurement Library
OCCI	Open Cloud Computing Interface
OCF	OFELIA Control Framework
OML	OMF Monitoring Library
ORM	Object Relational Mapping
RC	Resource Controller
RSpec	Request Specification
SCS	Stitching Computation Service
SDN	Software-Define Network

Co-funded by the Horizon 2020
Framework Programme of the European Union

SFA	Slice-based Federation Architecture
SM	Slice Manager
XMPP	eXtensible Messaging and Presence Protocol

Co-funded by the Horizon 2020
Framework Programme of the European Union

1 INTRODUCTION

This document analyses FIRE+ technologies and tools according to the needs of the FLAME architecture. Technology offerings and capabilities from current Future Internet Research and Experimentation (FIRE+) are assessed for suitability as technology to support FMI ecosystem experimentation. The task uses capability analysis to evaluate each FIRE+ technology, considering both functional and non-functional requirements, pursuing two main objectives:

1. Check the existing FIRE experimentation toolkits and possibly select those suitable to be adapted on the FLAME experiment toolbox, aiming to provide FLAME experimenters with offline specification, real-time control and monitoring of the experiments carried out over the FLAME platform. By aligning the FLAME experimentation toolbox APIs with what is expected by media players, FLAME will ensure that efforts to adapt media solutions towards FLAME is done in a way that allows experimenters to exploit technology investments and keep them close to market as possible for maturity of experimental facilities.
2. Assist the integration of FLAME solution with the rest of FIRE ecosystem with the help of appropriate FIRE+ Federators and interfaces. Under this point, FLAME will pursue the possibility of interaction / collaboration with the already existing FIRE setups.

The critical analysis assesses the cost and benefits of adaptation to FLAME's needs and suggests adaptations necessary for integration.

Our assessment falls into three main categories, as listed below:

- Experimenter's interaction tools, i.e. tools to request and manage resources;
- Federation tools, i.e. tools to help interoperability between testbed / infrastructure;
- Existing FIRE+ / Fed4FIRE+ infrastructures.

Co-funded by the Horizon 2020 Framework Programme of the European Union

2 EXPERIMENTER'S INTERACTION TOOLS

This section provides an overview of some of the existing experimenter's interaction tools, resulting from the FIRE+ research programme. These tools are oriented to cover the experimenters' needs as targeted by FIRE+, i.e. request and manage resources over the experimental infrastructures. The targeted experimental infrastructures within the FIRE+ programme covers a wide range of systems including: mobile and wireless, cloud, spectrum, photonics, Internet of Things (IoT), distributed service platforms, sensors. In addition, FIRE+ allows experimenters to manage, analyse and extract value from data acquired from sensors, mobile devices and/or online interactions in social and real networks. This section only covers some of the FIRE+ tools that seems in line with the FLAME requirements.

Technology wise, experimenter's interaction tools falls into three main types: desktop clients, Command Line Interface (CLI) clients and Graphical User Interface (GUI). In addition, they support contain Application Programming Interfaces (APIs). These two are good criteria to compare one tools against another. Besides the ease of use, release format (binary, code, etc.), and adaptation by the FIRE projects are extra comparison measures that we used in our assessment.

2.1 FED4FIRE+

Being the successor of the Fed4FIRE project, Fed4FIRE+ [1] interconnects a number of experimentation facilities, each targeting a specific community within the Future Internet ecosystem, as shown in Figure 1 [2] . Through the federation of these infrastructures, cross-domain experiments become possible. It creates good basis to boost creativity and introduction of innovative services.

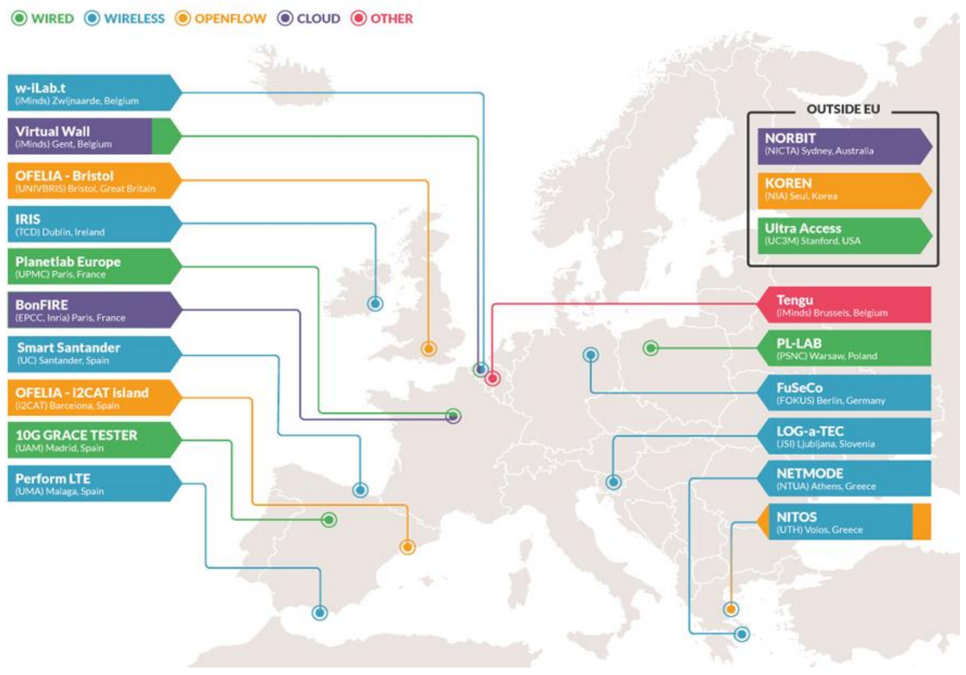

WIRED
WIRELESS
OPENFLOW
CLOUD
OTHER
w-iLab.t (iMinds) Zwijnaarde, Belgium
Virtual Wall (iMinds) Gent, Belgium
OFELIA - Bristol (UNIVBRIS) Bristol, Great Britain
IRIS (TCD) Dublin, Ireland
Planetlab Europe (UPMC) Paris, France
BonFIRE (EPCC, Inria) Paris, France
Smart Santander (UC) Santander, Spain
OFELIA - i2CAT island (i2CAT) Barcelona, Spain
10G GRACE TESTER (UAM) Madrid, Spain
Perform LTE (UMA) Malaga, Spain
OUTSIDE EU
NORBIT (NICTA) Sydney, Australia
KOREN (NIA) Seul, Korea
Ultra Access (UC3M) Stanford, USA
Tengu (iMinds) Brussels, Belgium
PL-LAB (PSNC) Warsaw, Poland
FuSeCo (FOKUS) Berlin, Germany
LOG-a-TEC (JSI) Ljubljana, Slovenia
NETMODE (NTUA) Athens, Greece
NITOS (UTH) Volos, Greece

Figure 1: Fed4Fire testbeds

Co-funded by the Horizon 2020 Framework Programme of the European Union

To improve ease of use for the experimenters, Fed4FIRE+ promotes common tools in the federation, abstracting the core testbed developments from the experimenting process. Fed4FIRE+ aims to provide open, accessible and reliable facilities supporting a wide variety of simple experimentation. Having said that, the following set of selected Fed4FIRE+ tools are reviewed. These tools aim to help end-users to create and/or define their experiments, either for specific infrastructures or for a subset of them.

2.1.1 jFed

jFed [3] is a Java-based framework for testbed federation, developed at the imec research institute and Ghent University, and was partially funded by the European Commission through the FP7 Project Fed4FIRE and the H2020 Project F-Interop. jFed is a software suite that provides different functionality, such as a desktop client to define experiments in any infrastructure federated with Fed4FIRE+ and to manage the resources, their expiration and status. It offers as well tools for testbeds providers to check the status of their infrastructure.

As shown in Figure 2, the suite is built around two main libraries. The low-level library aims to implement the client side for all the supported APIs, while as, the high-level library eases the experiment lifecycle management. On top of these libraries various components, such as thorough examination tools and user-friendly graphical experimenter GUI, were developed. In particular, jFed Experimenter GUI allows end-users to provision and manage experiments; jFed Probe assists testbed developers in testing their API implementations; jFed Automated tester performs extensive full-automated tests of the testbed APIs (it is part of the Fed4FIRE testbed monitor available on [4]).

Currently, jFed supports the following APIs:

Aggregate Manager APIs:

- **GENI Aggregate Manager v2:** allows aggregates to advertise resources, to allocate resources to slices and to manage them.
- **GENI Aggregate Manager v3:** the third version of the GENI Aggregate Manager, extending the workflow to allow for more fine-grained management of resources.
- **Open Cloud Computing Interface (OCCI):** protocol and API for all kinds of cloud-based management tasks. It is a flexible API with a strong focus on integration, portability, interoperability and innovation while still offering a high degree of extensibility.
- **ProtoGENI-extensions (emulab) to the GENI AM-APIs:** allow the entire federation to test interoperability of independent components.

User and Slice APIs:

- **ProtoGENI Slice Authority:** runs a Slice Authority XMLRPC server at each member of the federation.
- **Planetlab SFA Registry:** records and retrieves info about objects.
- **Common Federation API v1:** a set of standard APIs that any GENI-compatible Federation should or may provide. This is the API version 1 last revised on 2013.
- **Common Federation API v2:** the second version of the Common Federation API.

Other APIs

- **ProtoGENI Clearinghouse [5]** the clearinghouse is the central location for looking up information about component managers, users, slices, etc. Plays a central role in trust formation and the dissemination of certificate revocation lists.

Co-funded by the Horizon 2020
Framework Programme of the European Union

- **Stitching Computation Service (SCS):** provides GENI network stitching path information between GENI sites. GENI network stitching primarily refers to the interconnection of VLAN-based Ethernets via the GENI Aggregate Manager API.

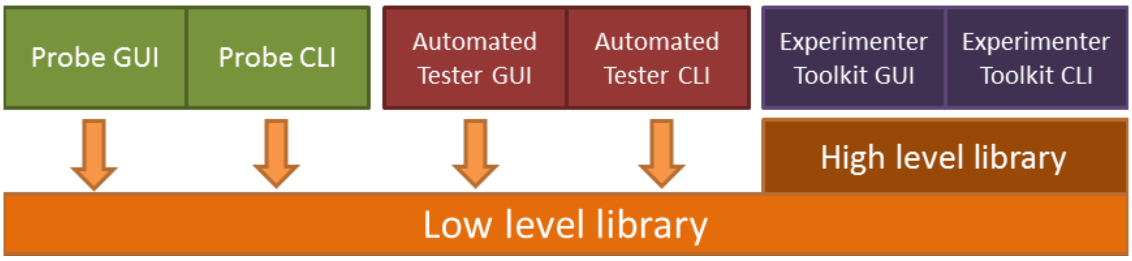


Figure 2: jFed suite (Ref:[6])

2.1.2 SFI

SFI [7] is a Slice-based Federation Architecture (SFA)-compatible command-line client. SFA aims to provide a minimal interface to enable the federation of testbeds, each with its own technology and under different administrations. This allows researchers to combine resources available in different testbeds, increasing the scale and diversity of their experiments. According to the SFA mind-set, the resources managed on a testbed can be either in the physical or virtual substrates. It also includes the share of resources assigned to a researcher, which usually correspond to virtualized versions of the former substrate. SFA abstracts those resources in:

- Components, which represent the minimal aggregation of physical resources that can be managed;
- Slivers, which are the portion of such resources let to the researchers; and
- Slices, which are collections of slivers assigned to researchers to perform an experiment. SFA defines three stages in a slice life-cycle: (i) register: at this point the slice exists only in name; (ii) instantiate: the slice is instantiated in the required components, being granted of a set of resources; and (iii) activate: the slice becomes active and runs code on behalf of the researcher.

SFA architecture is based in three main building blocks: (i) the registry (R), (ii) the Aggregate Manager (AM) and (iii) the Slice Manager (SM), as shown in Figure 3. The roles of the aggregate manager and the slice manager are to manage their respective resources. In current SFA implementations every module provides an XMLRPC over HTTPS service implementing the required interfaces: both the aggregate manager and the slice manager export the slice interface (manage slices: resources, create, delete, start, stop), whereas the registry exports the registry interface (manage records: add, update, remove, show, list).

Co-funded by the Horizon 2020
Framework Programme of the European Union

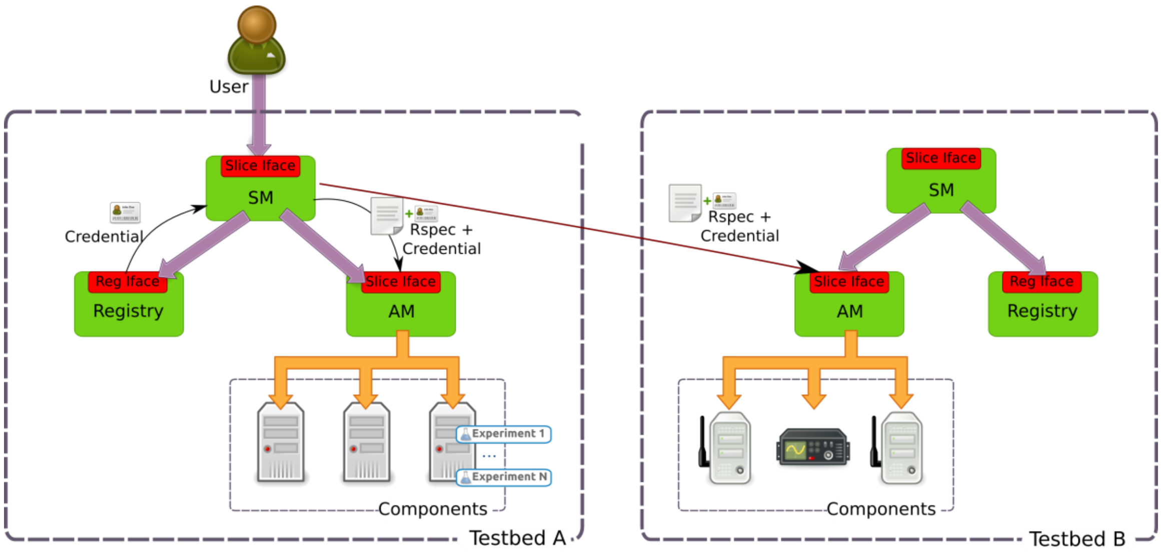


Figure 3: SFA architecture (Ref:[8])

As can be seen, SFA behaves in a hierarchical manner.

- SM: provides the slice interface to end users. SM behaves as a proxy, so whenever it receives a request, SM will send it to a corresponding aggregate manager, either one from the already managed testbed or one from a federated testbed.
- Registry: stores the mapping between the Human Readable Naming (HRN) of resources and their corresponding Global Identifier (GID). In this sense, the registry interface enables the creation, modification and deletion of entities and resources of the system. Moreover, the registry is responsible for issuing the related certificates and credentials.
- AM: performs the actual slice instantiations and other control operations. It directly interacts with the testbed components, ensuring system stability and slice provisioning.

Following the SFA architecture, SFI has been implemented in python as part of the PlanetLab [9] , an initiative that supports the development of new network services. It provides the functionality to create, update and display a slice. SFI also supports resource discovery, reservation and provisioning. It can also be used to release resources, and to start and stop a slice.

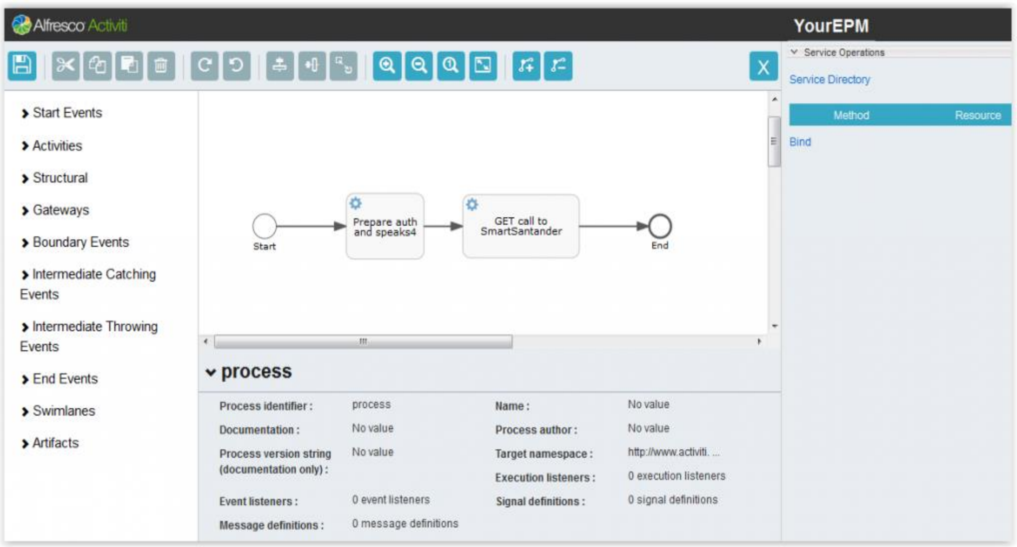
2.1.3 YourEPM

YourEPM [10] is a tool focused on the orchestration of Fed4FIRE Application Services. It helps experimenters to define and execute complex experiments involving application services over the federated testbed by using an underlying engine, so called Activiti BPM engine. Activiti [11] is a light-weight workflow and Business Process Management (BPM) platform designed and developed for business people, developers and system admins. Its core is a super-fast and rock-solid BPMN 2 process engine for Java. It's open-source and distributed under the Apache license. Activiti runs in any Java application, on a server, on a cluster or in the cloud.

YourEPM extends Activiti and provides integration with Fed4FIRE Speaks-for credentials, authentication via GENI Authorization Tool, multi-tenant functionality, integration with Fed4FIRE Application Services via Service Directory and RESTful invocation of external services. Figure 4 shows the Activiti modeller view while creating a new diagram or editing an existing one. On the left side of the editor there are the different items that can be chosen to build the process definition, grouped per

Co-funded by the Horizon 2020 Framework Programme of the European Union

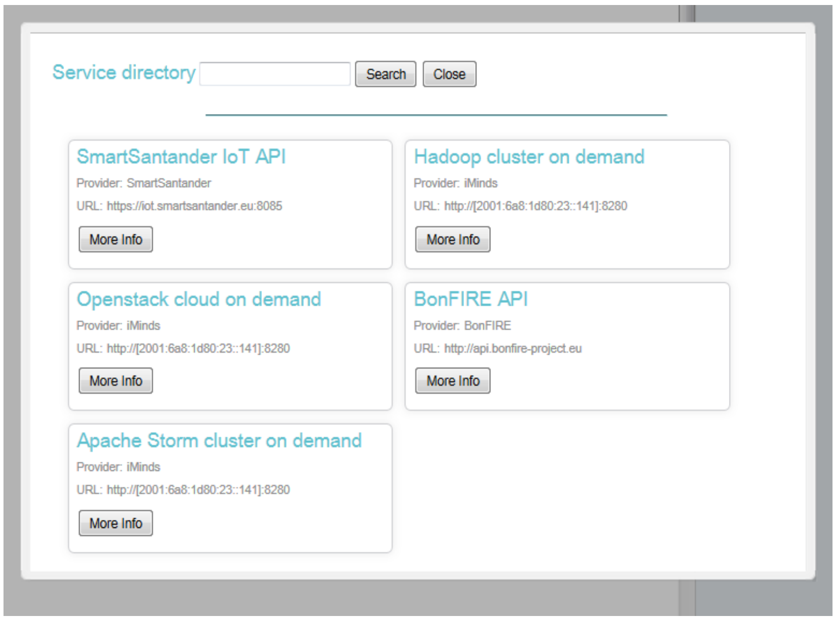
type: Start Events, Activities, Structural, Gateways, Boundary Events, Intermediate Catching Events, Intermediate Throwing Events, Swimlanes and Artifacts. On the right side of the editor the service directory information together with any information of bound services is shown. Elements can be added to the editor with a simple drag and drop action. Fed4FIRE offers a number of REST services that can be orchestrated by YourEPM, shown in Figure 5. Those services can be accessed from the YourEPM process editor by clicking on the Service Directory link.



process

Process identifier :	process	Name :	No value
Documentation :	No value	Process author :	No value
Process version string (documentation only) :	No value	Target namespace :	http://www.activiti. ...
		Execution listeners :	0 execution listeners
Event listeners :	0 event listeners	Signal definitions :	0 signal definitions
Message definitions :	0 message definitions		

Figure 4: Activiti modeller view (Ref: [10])



SmartSantander IoT API
Provider: SmartSantander
URL: https://iot.smartsantander.eu:8085

Hadoop cluster on demand
Provider: iMinds
URL: http://[2001:6a8:1d80:23::141]:8280

Openstack cloud on demand
Provider: iMinds
URL: http://[2001:6a8:1d80:23::141]:8280

BonFIRE API
Provider: BonFIRE
URL: http://api.bonfire-project.eu

Apache Storm cluster on demand
Provider: iMinds
URL: http://[2001:6a8:1d80:23::141]:8280

Figure 5: Fed4FIRE APIs (Ref: [10])

Once defined and deployed, the process is then started, detailing which step of the workflow is currently being executed. Information about the experimenters to which a task is associated, the

Co-funded by the Horizon 2020
Framework Programme of the European Union

execution environment variables, partial outputs and options selected in the manual dialogs can be seen during the experiment execution, Figure 6.

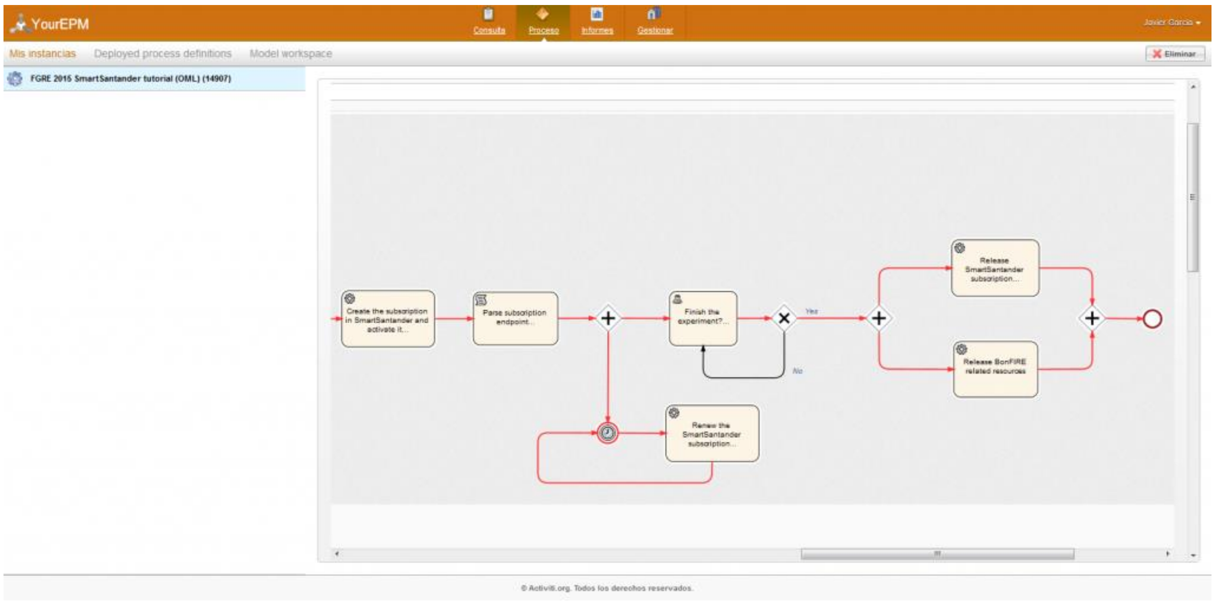

YourEPM
Consulta
Proceso
Informes
Gestionar
Javier García
Mis instancias
Deployed process definitions
Model workspace
Eliminar
FGRE 2015 SmartSantander tutorial (OML) (14907)
Create the subscription in SmartSantander and activate it...
Parse subscription endpoint...
Finish the experiment?...
Yes
No
Release SmartSantander subscription...
Release BonFIRE related resources
Renew the SmartSantander subscription...
© Activiti.org. Todos los derechos reservados.

Figure 6: YourEPM execution environment (Ref: [10])

2.2 ONELAB TOOLS

OneLab [12] is a consortium formed in 2014 by research and academic institutions. Their members provide different testbeds and tools to allow experimenters to easily conduct experiments. The most prominent tool is MySlice, a graphical GUI.

2.2.1 MySlice

MySlice [13] is a free OS resource management tool for testbeds, mainly developed in python. In principle, it is possible to tailor down MySlice according to the testbed needs. It is a web client enabling users to access a Federation of testbeds (e.g. Fed4FIRE federation and its facilities) using the SFA APIs, see Figure 7. Fed4Fire Portal is powered by MySlice software [14] . It acts as an experimentation tool for resource discovery, reservation and provisioning as well as bridging to experiment control tools. MySlice provides information to the experimenter in several forms. The testbed directory is a catalogue that provides a high-level overview of all federated testbeds and their capabilities. The portal also provides pointers for the experimenters to the project website and to the First Level Support systems (e.g. Trouble Ticket System), if they need help to register or use the portal or encounter problems while setting up experiments.

The portal supports an easy registration of new experimenters. That is experimenters could register themselves and access the federated testbeds without waiting too much time. Note however that the testbeds will always determine whether the user can actually access them according to their access policies.

Co-funded by the Horizon 2020
Framework Programme of the European Union

Moreover, MySlice performs the role of a client tool. On behalf of the experimenter, the portal will forward queries to federated testbeds using SFA delegation mechanism. Using the portal, the experimenter will be able to search, browse and reserve resources across federated testbeds. The portal will also act as a bridge to experiment control tools.

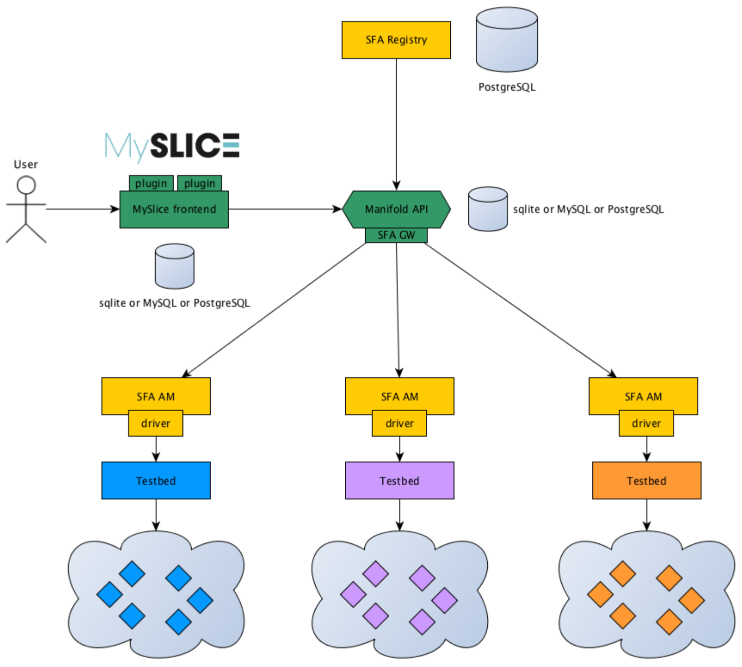


Figure 7: MySlice architecture (Ref:[15])

2.2.2 NEPI

NEPI [16] , the Network Experimentation Programming Interface, is a set of python libraries (in particular 2) that provides a life-cycle management tool for network experimentations. It is a tool to run and orchestrate network experiments, involving several tens of target hosts. In this sense, NEPI provides a single tool to design, deploy, control network experiments, and gather the experiment results. NEPI is specially conceived to function with arbitrary experimentation platforms, i.e. it supports researchers with a single tool to work with network simulators, emulators, or physical testbeds, or even a mixture of them.

NEPI includes a high-level interface to describe experiments [17] that is independent from experimentation platforms. Although, the high-level experiment descriptor captures platform specific configurations. It can be represented and stored in XML format to be later re-produced and modified according to experimentation needs. A global experiment controller that is platform independent orchestrates experiment execution. It interacts with different platform-dependent testbed controllers to carry out the experiment execution. This structure forms a control hierarchy that is able to adapt to platform specific requirements while providing an integrated control scheme.

The initial focus of NEPI is to support design, and control through the Federated Resource Control Protocol (FRCP). It uses an abstraction model composed of boxes and connectors to construct the experiment design. Each supported experimentation platform defines a set of boxes, representing the conceptual constructive blocks of an experiment. These boxes interact with one another through named ports called connectors. Each connector has a specific function. An experiment is described by a graph with boxes as vertex. Those boxes / vertexes along with the connections between them will

Co-funded by the Horizon 2020
Framework Programme of the European Union

define the experiment topology, both at a physical (infrastructure) and application (services) levels. Boxes also have a set of attributes that allow defining the experiment configuration, and traces that allow defining experiment results to be collected.

2.2.3 OMF

OMF [18] is a framework developed in Ruby language and based on XMPP (eXtensible Messaging and Presence Protocol) with the focus on controlling and managing network devices. The OMF suite also provides OML [19] (OMF Monitoring Library), which allows instrumentation of applications for collecting measurements. Actually, OMF depends on a suite of software components to provide services such as dynamic IP addressing, remote booting, remote OS imaging, data archiving, etc. OMF supports a large number of different wired and wireless resources, such as PC-like devices, Android phones, routers, sensors, among others.

The OMF architecture consists of three logical planes: control, measurement, and management. This architecture is not completely clear in version OMF 5.4, but it is the approach defined for the version 6. The control plane includes the OMF tools that a researcher uses to describe his/her experiments, and the OMF entities responsible for orchestrating them. The measurement plane includes the OMF tools to instrument an experiment, and the corresponding OMF entities to collect and store measured data. The management plane includes the OMF functions and entities to provision and configure the resources, which are provided by the testbed facilities and used by the experiments. Figure 8 presents a system view of an OMF deployment. The main OMF elements are detailed below.

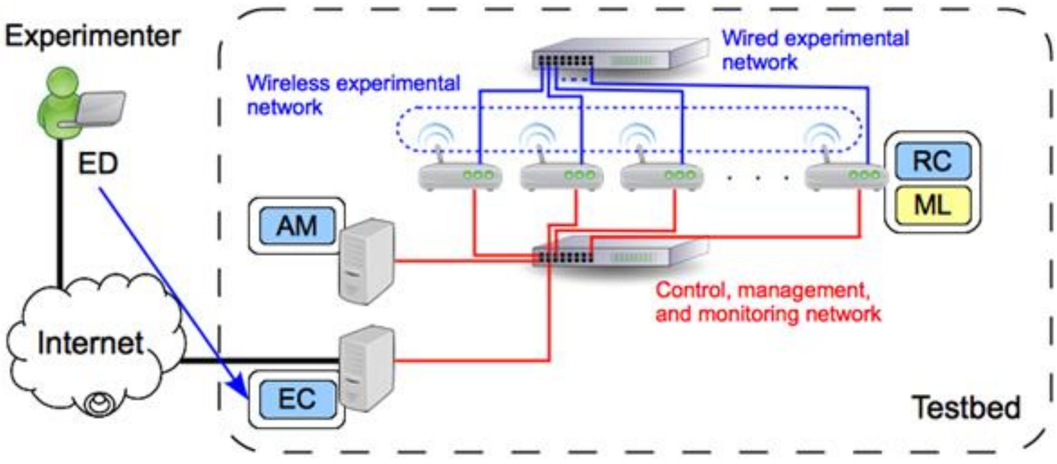

Experimenter
ED
Internet
Wireless experimental network
Wired experimental network
AM
RC
ML
Control, management, and monitoring network
EC
Testbed

Figure 8: System view of OMF (Ref:[20])

A domain specific language (the OMF Experiment Description Language – OEDL) allows an experimenter to write an Experiment Description (ED), which details the resource requirements, their initial configuration, and a state machine describing the time/event-triggered actions required to realize the experiment.

The experimenter submits the ED to an Experiment Controller (EC), which is the control entity orchestrating the experiment on behalf of the experimenter. The EC issues requests on the management plane to configure the resources as specified in the ED. Once the experiment prerequisites are met, the EC sends directives to the Resource Controller (RC) associated with each resource. Actually, the RC also does the configuration of the resources. Following the ED's state machine, the EC sends directives to the RCs, which are the control entities responsible for executing actions on the resources.

Co-funded by the Horizon 2020 Framework Programme of the European Union

The measurement plane consists of two parts, namely the Measurement Collection Server (MCS) and the Measurement Library (ML). OML allows recording of any relevant metrics and instrumented applications. The resulting streams are forwarded to the MCS in either real-time or batch mode to minimize interference with the experiment itself. The MCS stores them in a SQL database created for each experiment instance. The user can directly run SQL queries against it or retrieve a data dump from it. Standing queries can feed events back into the EC to support steerable experiments.

The batch mode is monitoring part of the disconnected operation mode, which also involves specific configuration in OMF elements. The capability of performing experiments with partial disconnection of mobile devices is an important feature, which allows tests with smart phones, cars, DTN devices, etc.

Another important feature is the capability to visualize the measurements at real-time on the experimenter's Web browser. This is useful for the beginners, but it can also help the experts to easily verify if the expected results are being generated at very begin of an experiment.

OMF combines a set of management services into an AM. Thus, the AM is a collection of services, which can be deployed across multiple servers for performance and redundancy reasons. AM accepts request from the ECs or the testbed operator and sends corresponding commands to the RC running on each resource. MCS may be treated as one of the services of the AM.

2.3 GENI TOOLS

The Global Environment for Network Innovations (GENI) [21] is a virtual laboratory of networking and distributed systems research and education. Similar to Fed4Fire testbed, GENI is also composed of various technologies and facilities as shown in Figure 9. It is well suited for exploring networks at scale, thereby promoting innovations in network science, security, services and applications. GENI allows experimenters to:

- Obtain compute resources from various locations;
- Connect compute resources using Layer 2 connections and create topologies best suited to their experiments;
- Install custom software (even custom operating systems) on the available compute resources;
- Control the traffic flows on switches involved in their experiment;
- Run their own Layer 3 and above protocols by installing protocol software in their own premises and providing appropriate flow control rules for switches.

GENI suits mostly experiments that require:

Co-funded by the Horizon 2020 Framework Programme of the European Union

- A large-scale experiment infrastructure: GENI can potentially provide more resources than what is typically found in a single laboratory. GENI gives access to hundreds of widely distributed resources, including compute resources (e.g. virtual machines and "bare-machines") and network resources (e.g. links, switches and WiMax base stations).

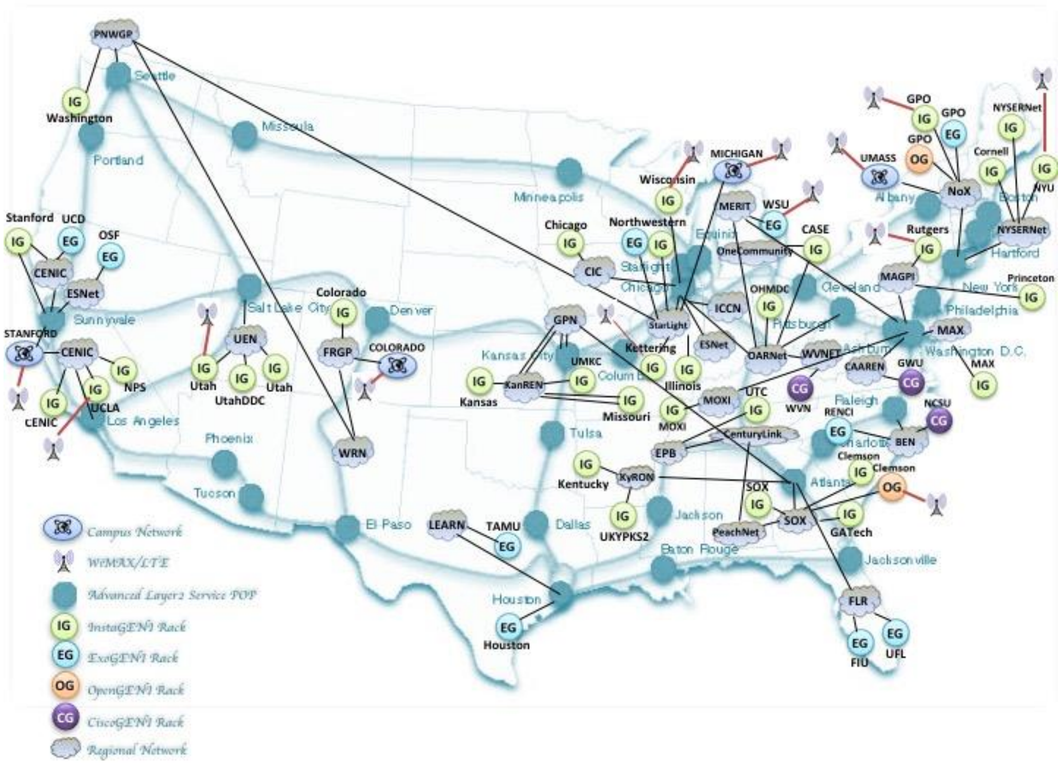


Figure 9: GENI testbed (Ref: [21])

- Non-IP connectivity across resources: GENI allows setting up Layer 2 connections between compute resources while running Layer 3 and above protocols on the experimenter premises controlling connectivity between resources.
- Deep programmability: GENI allows to program not only the end hosts of an experimental network but also the switches in the core of the network. This enables experimentations with novel network layer protocols and/or with novel IP-routing algorithms.
- Reproducibility: certain GENI resources can be access exclusively, including CPU resources and network resources. This guarantees control over the experimentation environment and hence the ability to repeat experiments under identical or very similar conditions.
- Instrumentation and measurement tools: GENI has instrumentation and measurement systems that provide probes for active and passive measurements, measurement data storage and tools for visualizing and analysing measurement data.

In the following subsections, some experimentation toolsets developed over GENI are presented.

Co-funded by the Horizon 2020
Framework Programme of the European Union

2.3.1 Flack

Flack [22] is a visual tool used to manage computing and networking resources (allocating, provisioning, configuring, etc) that are located at the different sites that belong to the GENI network. Because of its low barrier to entry, Flack is the preferred tool for getting developers up and running in a short period. After correctly authenticating an experimenter on Flack, it will attempt to retrieve resources from the various authorities, and display them over the North America map, as shown Figure 10:

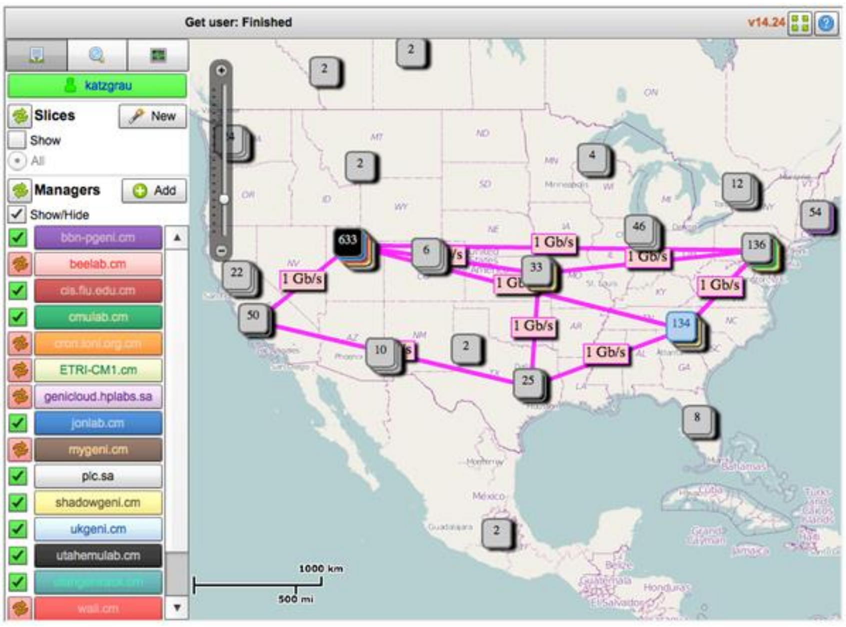


Figure 10: Flack overview (Ref: [23])

Flack allows creating resource slices (each appropriate to an experiment) out of all retrieved resources. The process is done in an easy drag and drop fashion. After creating the slice, Flack provides enough means to configure compute resources as well as network resources. Figure 11 shows an exemplary slice created with the help of Flack over GENI.

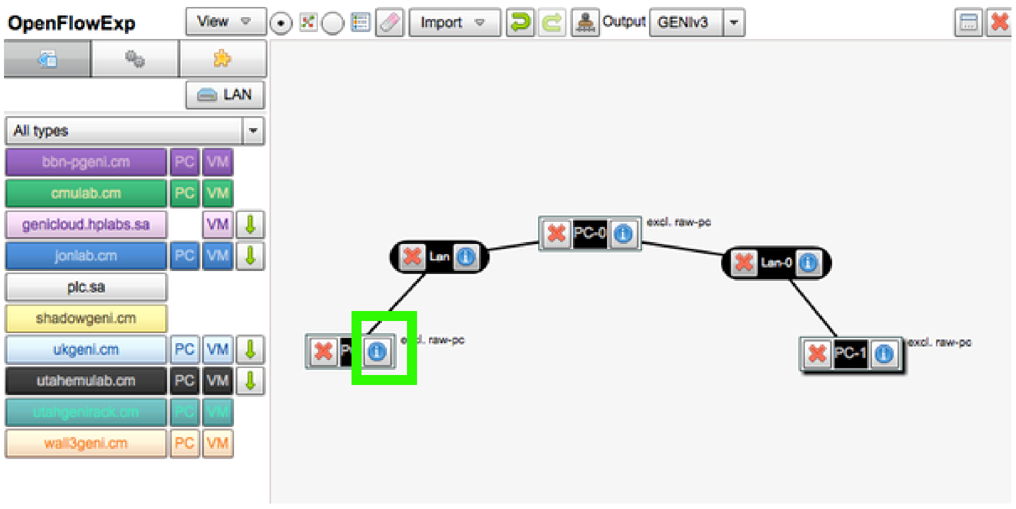
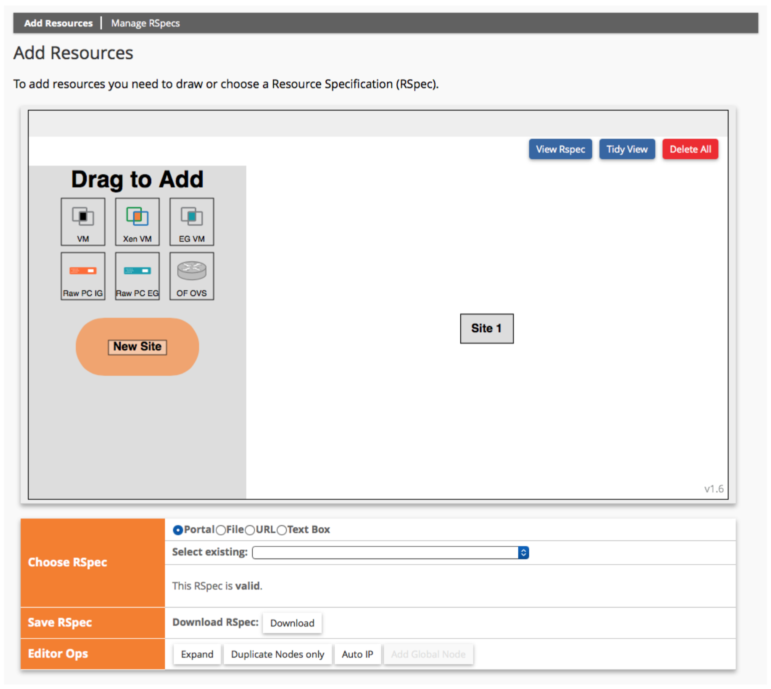


Figure 11: Flack overview (Ref: [23])

Co-funded by the Horizon 2020 Framework Programme of the European Union

2.3.2 Jacks

Jacks [24] is a tool integrated with the GENI portal, which allows drawing topologies of GENI resources and then reserving them, as shown in Figure 12.



Add Resources

To add resources you need to draw or choose a Resource Specification (RSpec).

This RSpec is **valid**.

Figure 12: Jacks interface (Ref: [25])

The following steps explain how to reserve two virtual machines using Jacks:

- Use the Add Resources button on the upper left-hand corner of the Slice page.
- Click the black VM box and drag it onto the canvas. This icon represents a generic *default-vm*. Repeat the action for the other VM too.
- To connect the two VMs, first click near one of the VM boxes on the canvas, then click and drag towards the other VM. Release it when reach the other VM. The created line, as shown in Figure 13, between VMs represents the desired connection between VMs.
- It is possible to edit the name of the VMs, assign IP addresses to the interfaces, etc. When the nodes are ready, they will turn to green.

Co-funded by the Horizon 2020
Framework Programme of the European Union

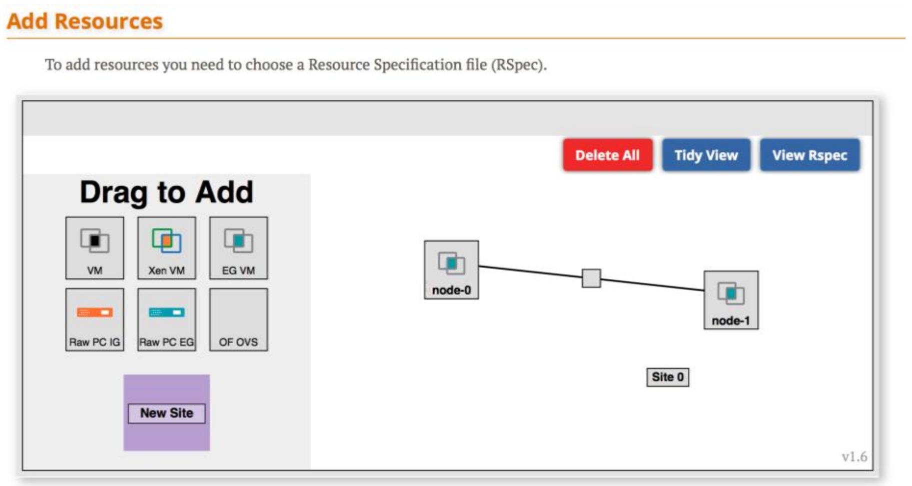


Figure 13: Two VMs connected by a link using Jacks (Ref: [25])

2.3.3 OMNI

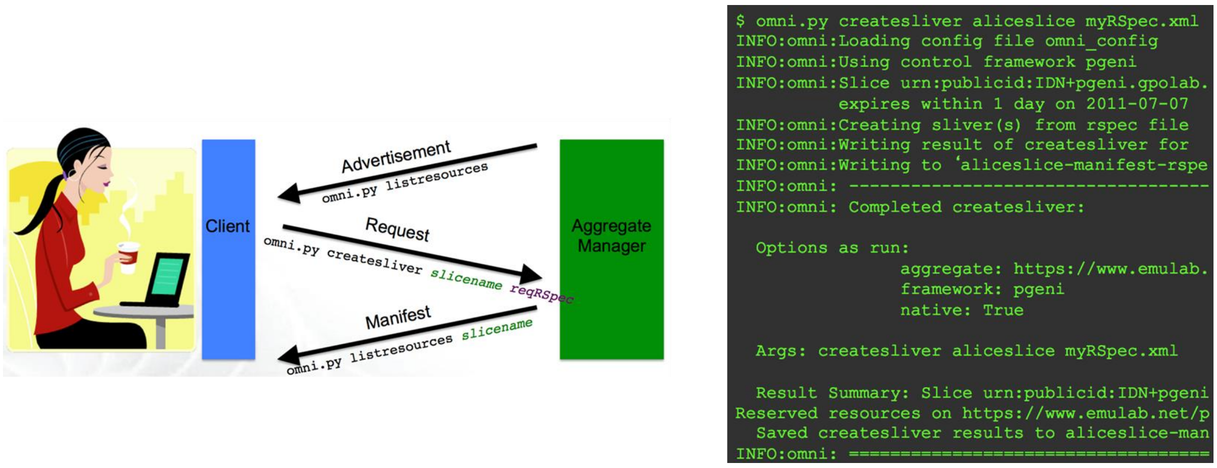
OMNI [26] is a GENI API-compatible command line tool (Figure 14) for reserving resources at GENI aggregate managers, using user accounts typically issued by a GENI-based clearinghouse.

An aggregate manager is a software server that provides resources to clients based on the GENI aggregate manager API [27] . In particular, GENI Aggregate Manager API allows to advertise resources and to allocate resources to Slices in the form of Slivers. A clearinghouse [5] , a collection of related services supporting federation among experimenters, aggregates and the GENI Meta Operations Center (GMOC), provides users with GENI accounts (credentials). The Omni client communicates with a user's clearinghouse to create slices, enumerate available GENI aggregates, and access user and slice credentials, which OMNI uses when reserving resources from the available aggregates. The OMNI client is distributed either as part of the GENI Tools software package or alone as Mac or Windows binaries. OMNI can also be used as a library in custom Python scripts. In the following, an overview of the actions required by an experimenter to run a GENI experiment using resources from multiple resource aggregates.

- Get credentials from one Clearinghouse; this allows accessing multiple aggregates.
- Install and configure OMNI tools
- Use the OMNI tools to create a slice
- Request resources from one or more aggregates to add to the slice and of course, run the experiment.

Figure 15 shows the OMNI workflow in a visual way.

Co-funded by the Horizon 2020
Framework Programme of the European Union



```
$ omni.py createsliver aliceslice myRSpec.xml
INFO:omni:Loading config file omni_config
INFO:omni:Using control framework pgeni
INFO:omni:Slice urn:publicid:IDN+pgeni.gpolab.
          expires within 1 day on 2011-07-07
INFO:omni:Creating sliver(s) from rspec file
INFO:omni:Writing result of createsliver for
INFO:omni:Writing to 'aliceslice-manifest-rspe
INFO:omni: ------------------------------------
INFO:omni: Completed createsliver:

  Options as run:
                aggregate: https://www.emulab.
                framework: pgeni
                native: True

  Args: createsliver aliceslice myRSpec.xml

  Result Summary: Slice urn:publicid:IDN+pgeni
Reserved resources on https://www.emulab.net/p
  Saved createsliver results to aliceslice-man
INFO:omni: ===================================
```

Figure 14: OMNI CLI (Ref: [26])

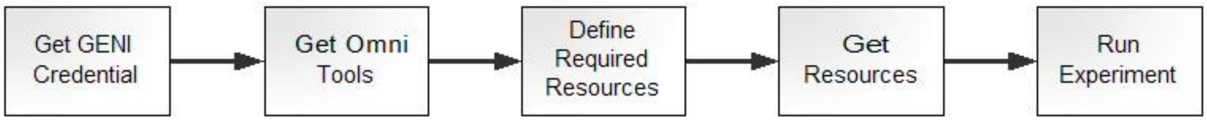


Figure 15: Omni workflow (Ref: [26])

2.4 OFELIA AND FIBRE TOOLS

The OFELIA project [28] constructed an OpenFlow-based research facility interconnecting different countries throughout Europe. Along with the testbed setup and configuration, the OFELIA Control Framework (OCF) is its major outcome.

In a similar line, the FIBRE [29] testbed departs from OFELIA tools to extend and operate a research facility, funded by the 2010 Brazil-EU Coordinated Call in ICT [30] . It is a large-scale Virtual Laboratory for students and researchers to test new applications and network architecture models, as shown in Figure 16. The FIBRE infrastructure consists of a federation of local testbeds, also called “experimentation nodes” or simply “islands”. Each “island” have a set of network devices to support experiments in both fixed and wireless technologies. An overlay network on the RNP backbone [31] , comprised of two network separate layers, connects them: a control plane and an experiment plane. This network is named “FIBREnet”.

FIBRE pursues the following objectives:

- Build a Future Internet testbed in Brazil, federated with other worldwide testbed initiatives;
- Provide a large-scale platform for promoting the Future Internet research in Brazil and region;
- Encourage educators to use the testbed for network classes to foster a new generation of researchers.

Co-funded by the Horizon 2020 Framework Programme of the European Union

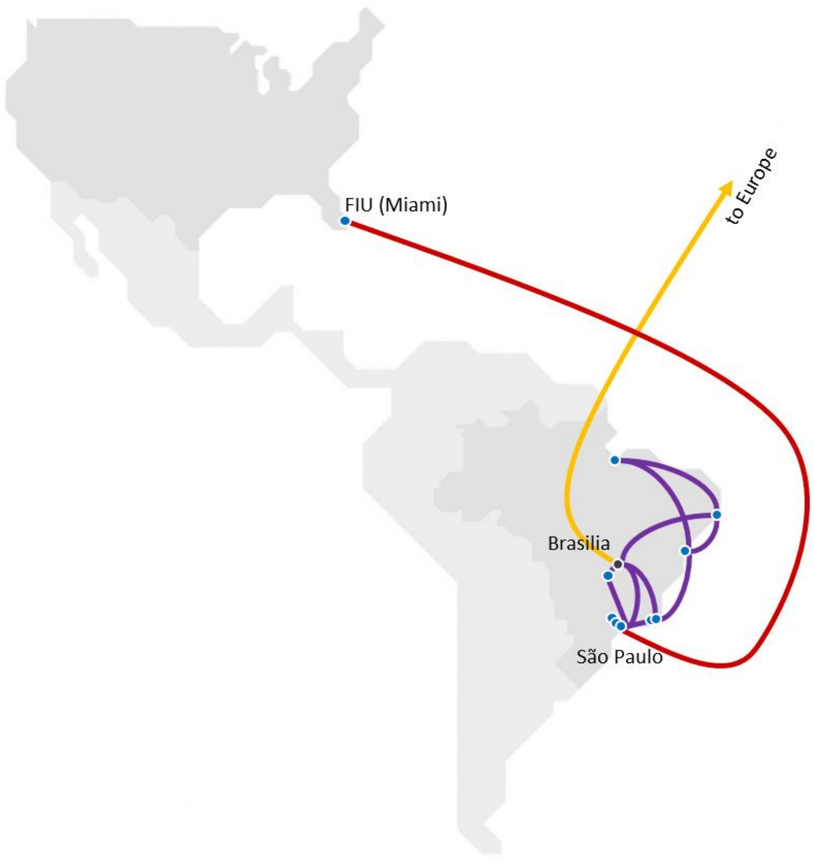


Figure 16: FIBRE architecture (Ref: [32])

2.4.1 OCF

The OCF is a tool developed in the framework of the OFELIA project, funded by the European Union through its 7th Framework Programme (FP7). It is an open-source control framework that followed the evolution of Expedient and Opt-in manager [33] , and the introduction of the VT-Manager, all publicly available (Github) and intended to be easy to use, both for the infrastructure manager (to configure for managing part of the stack) and for the experimenter.

Expedient (Figure 17) consists of several subsystems: an Object Relational Mapping (ORM) database, a base platform subsystem, and an extendible plugin system with two built-in plug-ins, the so-called connectors and user clients. These plugins cooperate to provide the user interface for creating and managing slices across multiple island providers. In OFELIA, there was significant work done to improve, in particular the WebUI, that was the most used mean by the experimenters.

Co-funded by the Horizon 2020
Framework Programme of the European Union

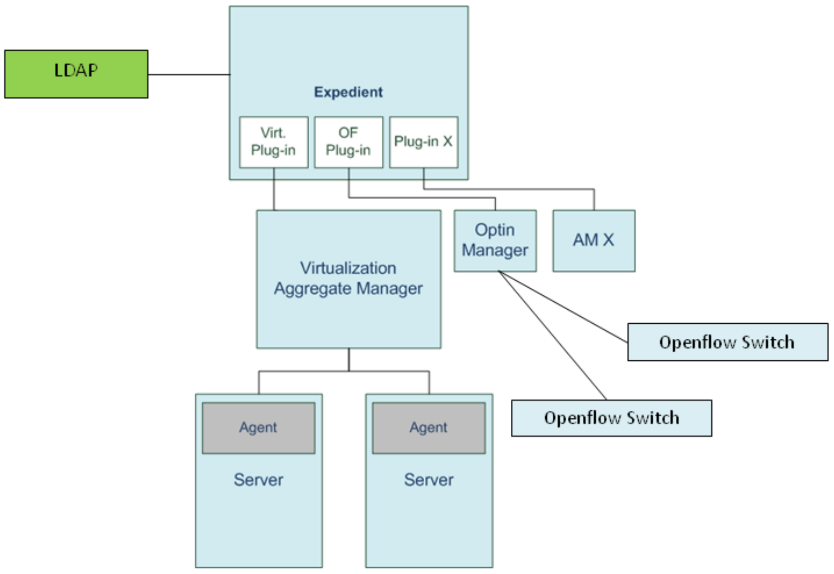


Figure 17: Expedient architecture (Ref: [34])

To slice the OpenFlow resources to support multiple experiments and users, it makes use of 4 layers: at the bottom there are OpenFlow switches. On top of them, the Flowvisor, the Opt-In Manager, and finally the OpenFlow Expedient connector. The Flowvisor is the most necessary part to isolate and support the multiple experiments' slices. At an abstract level, Flowvisor is a transparent proxy between an island with OpenFlow switches and the OpenFlow controller used in a certain slice. The Flowvisor(s) present in each island (or provider) monitors OpenFlow protocol messages from and to the controllers, ensuring that each slice defined by the FlowSpace (a set of header values defined to isolate experiments) operates on traffic within that space. The OCF, in addition to what was explained so far, makes use of an additional resource of a virtualization service, to provide virtual machines to the users as end-hosts or in order to, allocate local OpenFlow controllers to test the experiments. The OCF system uses Lightweight Directory Access Protocol (LDAP) [35] as the main software authenticator and provides the Virtualization Aggregate Manager based on Xen machines, which can allocate on-demand virtual machines.

2.5 EXPERIMENTATION TOOLS ASSESSMENT

In this section, we compare the above-mentioned tool against some identified criteria. It will help us to build an overview of the capabilities of tools and check their suitability for FLAME.

Co-funded by the Horizon 2020 Framework Programme of the European Union

Table 1: Experimenter's interaction tool assessment

Type	Tool	API/s supported	Ease of use	Usage in projects
Desktop/GUI	jFed	SFA, GENIv3	🟩	Fed4FIRE, GENI, …
Desktop/CLI	jFed	SFA, GENIv3	🟩	Fed4FIRE, GENI, …
	SFI	SFA	🟨	Fed4FIRE, …
	OMNI	SFA, GENIv3	🟧	Fed4FIRE, GENI, …
	NEPI	NEPI	⬜	Fed4FIRE, …
Web/GUI	OFELIA	SFA, XMLRPC	🟩	OFELIA, FIBRE
	Flack	SFA, GENIv3	⬜	GENI, …
	MySlice		🟨	Fed4FIRE, FIBRE
	YourEPM	OCCI, REST (BPM workflows)	⬜	Fedf4FIRE
Web/REST	NEPI	NEPI	⬜	Fed4FIRE, …
	OMF6	OMF	⬜	Fed4FIRE
	OFELIA	SFA, GENIv3	🟩	OFELIA, FIBRE, FELIX, Fed4FIRE

* Perceived ease of use (from more easy to more difficult) – gray: unknown → 🟩🟨🟧⬜

The assessment of interaction tools shows that technologies have emerged as part of the FIRE and FIRE+ programme supporting the generalised needs of experimentation for the access, control and monitoring of resources. The issue is that the tools overlap significantly with the existing APIs offered by platform and infrastructure technologies in ways that are not aligned with production environments. FLAME's approach, as defined in the architecture (D3.4) and roadmap (D3.5) is to provide a service-based API using existing standards and specifications implemented within technologies and where possible align these with the needs of media service developers. The API can be used for experimentation but it is also the same API to be used for testing and production. This approach reduces the cost of experimentation and the transfer of results to production, both factors are critical for adoption of FLAME during and after the project. Similarly for measurement, FLAME uses industry standard time-series database technologies within CLMC rather than experimental technologies such as OMF.

Co-funded by the Horizon 2020
Framework Programme of the European Union

3 FEDERATION TOOLS

To ensure that the possibility of federation with other testbeds, in the following we review some recommendations originated within the framework of the above-mentioned initiatives. We classify them in two levels, i.e. basic and advanced federation recommendation levels. At the end of this section we provide an assessment of some tools and solutions which would help the testbed federation.

3.1 BASIC FEDERATION GUIDELINES

To guarantee federation with other FIRE infrastructures, two important points should be taken into account: at the physical level, L2 connectivity is essential or even preferable); however, it is also recommended to have the possibility of VPN connectivity between testbeds. At the logical level, it is suggested to use GENIv3 API, which allows utilizing the same clients for experimenters. To ease this action, correct documentation is very important. It helps building a comprehensive infrastructure topology, 1st-step guide, operation examples, contact, etc.

To access resources, it is suggested to establish user access with VPN for resources within the experimenter's slice and having Public IP access to expose operation APIs. As a common practice, it is also suggested to use the following tool to help user's operation on resources: Desktop (jFed), CLI (omni), API (XML-RPC). Moreover, a GUI might be designed for listing, allocating, provisioning and managing resources. It is suggested to keep an updated repository of images/firmware on virtual resources. This will help experimenters to plan and execute better the experiments. Moreover, to automate configurations at bootstrapping time it is helpful to permit user contextualisation (i.e., inserting specific user configuration in the resources).

It is assumed that the virtualised network resources are requested and defined explicitly (with low-level details) by the experimenter. Multi-tenancy is guaranteed with the help of slice isolation (VLAN, VxLAN, VPN, etc.). Multiple parameters are supported when allocating/provisioning resources, e.g. IT: VM Flavour (OS, RAM, CPU, Storage), containers; network: interfaces, BW enforcement, QoS, private/public IPs. Moreover, it is important to guarantee a level of resilient/backup for links in slices (both management and data plane).

Monitoring depends on two important factors

- Slices: how many are requested, how many are active, how many are expired or soon-to-expire, the status and failures on resources within the slice.
- Available resources: how many resources are in-use now, total servers/VMs/devices the experimenter can request, etc.

3.2 ADVANCED FEDERATION GUIDELINES

The experimenter's experience, as well as federation activity, can be improved with the following suggestions:

Monitoring

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Data usage on user's virtual resources (IT)
- Data usage on user's virtual resources (network)

User's access to resources

- NAT or public IP (resources) (would need firewall and IP or MAC filtering)
- Dedicated access to HW for limited periods of time

Slicing

- Automatic configuration of network (VMs, edge and core network)
- Virtualised network defined and requested internally; based on the computing nodes requested
- Resource scheduling

Services

- Pool of services (VNFs) to deploy on location chosen by user

3.3 RECOMMENDED TECHNOLOGIES

The process for federating a FIRE-enabled testbed is built upon a number of technologies and tools. The following describes those commonly used.

3.3.1 SFA

Slice-based Federation Architecture (SFA) [36] provides a minimal interface to enable the federation of testbeds with different technologies and belonging to different administrators, while granting the control of the resources to their owners. This allows researchers to combine resources available in different testbeds, increasing the scale and diversity of their experiments. SFA is based on a set of high-level concepts that define the actors and the resources that interact on the testbed, as well as defining an architecture with its interfaces and main datatypes to facilitate the federation of testbeds.

On SFA, three principals or actors are considered on the architecture:

- The member authority is responsible of managing the information about federation members and assert attributes about particular members. It provides services to, for instance, consult the status of the revocation list or to verify certificates.
- The slice authority is responsible of managing slices and generate credentials for members with respect to slices; i.e., it names and registers slices while giving users access and control to them.
- Users, who are people playing one or more roles in the facility.

This architecture can be implemented with some variations. Taking one research project (CONFINE [8]) as example to illustrate how SFA is adopted, this project defines a data model that introduces the "groups" instead of the authorities and brings a more general approach on the role of the "resource authorities".

The resources managed on a testbed are not only the physical substrate, but also the share of resources assigned to a researcher, which usually correspond to virtualized versions of the former substrate. SFA abstracts those resources in:

- Components, which represent the minimal aggregation of physical resources that can be managed;

Co-funded by the Horizon 2020 Framework Programme of the European Union

- Slivers, which are the portion of such resources let to the researchers; and
- Slices, which are collections of slivers assigned to researchers to perform an experiment. Slices are the primary abstraction for accounting and accountability. SFA defines three stages in a slice life cycle: (i) register: at this point, the slice exists only in name; (ii) instantiate: the slice is instantiated in the required components, being granted of a set of resources; and (iii) activate: the slice becomes active and runs code on behalf of the researcher.

The software modules that manage those components are the aggregate manager for the components (or component manager if it manages a single component) and the slice manager for slices and slivers.

The case in CONFINE is similar. Slivers and slices are used, where slices have an equivalent set of states: register, deploy and start, and the only components that CONFINE considers are research devices (nodes). Regarding the management, it is all done in a centralized way by the controller. SFA defines the interaction between actors and resources by means of a set of APIs, however, before getting into detail; some data types need to be explained, to understand them. The main data types in SFA are:

- Global identifier or GID is an identifier assigned to components, slices, services and every principal (actor) participating in the system. Specifically, a GID is a certificate that binds together a public key, a UUID and a period during while the GID is valid. Even though now CONFINE controller does not consider GIDs, it has been enriched with an extension that will allow the assignation of GIDs to the entities in the controller.
- Resource specification or RSpec is used to describe the resources on the system. Therefore, for a component it will describe the set of resources it possesses and constraints and dependencies on the allocation of those resources; whereas for slices it will describe the set of slivers it consists of and their characteristics. The RSpec is an XML dependent of the testbed, since every testbed will have its own type of resources and requirements. The RSpec for CONFINE testbeds has not been defined yet, although resource descriptions (JSON-based) in its REST API may be regarded as equivalent.
- Ticket: a promise signed by an aggregate manager, giving an entity the right to allocate the resources that are being granted.
- Credential, on the other hand, is a grant of a set of rights and privileges associated with a particular principal.

The interaction of all these components is shown in Figure 18.

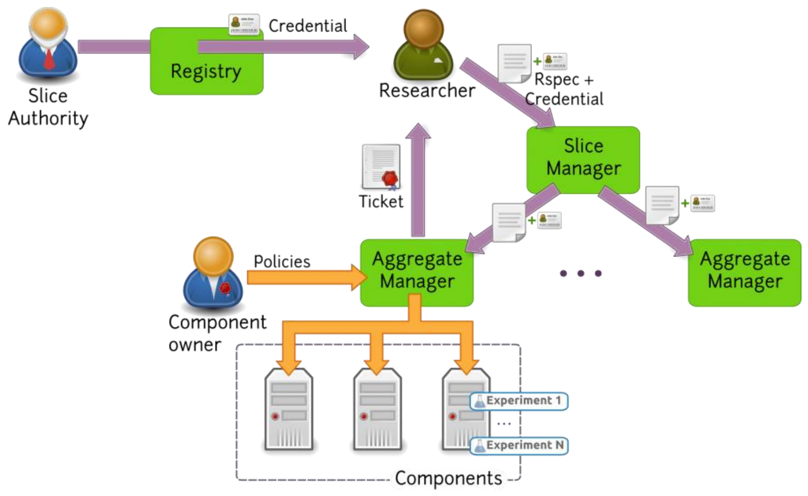


Figure 18: Slice Federation Architecture (Ref: [8])

Co-funded by the Horizon 2020 Framework Programme of the European Union

3.3.2 Resource Specification (RSpec) Documents

In order to allow interoperability among different 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. The schemas support Aggregate or resource specific extensions. In GENI there are three different types of RSpec (Figure 19), each used to describe resources when communicating with an AM. The communication with an AM, is based on the common GENI AM API [37] that requires AMs to communicate using RSpec data types [38] .

- Advertisement RSpec: This document is returned by an AM that describes the resources that the AM has.
- Request RSpec: This document is sent by the user to AM to describe the resources that needed to be reserve.
- Manifest RSpec: This is the document returned by an AM that describes the resources that a user has reserved at an AM.

In addition to the three types of RSpecs, there are also different versions of RSpecs. The current version is GENI v3. Depending on which tool is used to reserve resources, the user should be able to get a list of supported RSpecs from each AM through the GetVersion API call.

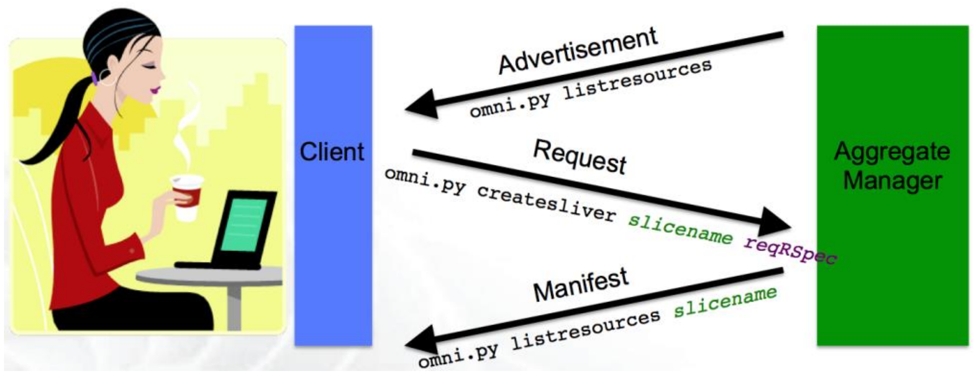


Figure 19: Request Specification (RSpec) documents (Ref: [39])

3.3.3 Open Cloud Computing Interface

OCCI [40] is a RESTful Protocol and API for all kinds of management tasks. OCCI was originally initiated to create a remote management API for IaaS model based Services, allowing for the development of interoperable tools for common tasks including deployment, autonomic scaling and monitoring. It has since evolved into a flexible API with a strong focus on interoperability while still offering a high degree of extensibility. The current release of the Open Cloud Computing Interface is suitable to serve many other models in addition to IaaS, including PaaS and SaaS.

In order to be modular and extensible, the current OCCI specification is released as a suite of complimentary documents, which together form the complete specification. The documents are divided into three categories consisting of the OCCI Core, the OCCI Renderings and the OCCI Extensions.

Co-funded by the Horizon 2020 Framework Programme of the European Union

- The OCCI Core specification consist of a single document defining the OCCI Core Model. The OCCI Core Model can be interacted with renderings (including associated behaviors) and expanded through extensions.
- The OCCI Rendering specifications consist of multiple documents each describing a particular rendering of the OCCI Core Model. Multiple renderings can interact with the same instance of the OCCI Core Model and will automatically support any additions to the model, which follow the extension rules defined in OCCI Core.
- The OCCI Extension specifications consist of multiple documents each describing a particular extension of the OCCI Core Model. The extension documents describe additions to the OCCI Core Model defined within the OCCI specification suite.

The Open Cloud Computing Interface is a boundary protocol and API that acts as a service front-end to a provider's internal management framework. Figure 20 shows OCCI's place in a service provider's architecture. Service consumers can be both end-users and other system instances. OCCI is suitable for both cases. The key feature is that OCCI can be used as a management API for all kinds of resources while at the same time maintaining a high level of interoperability.

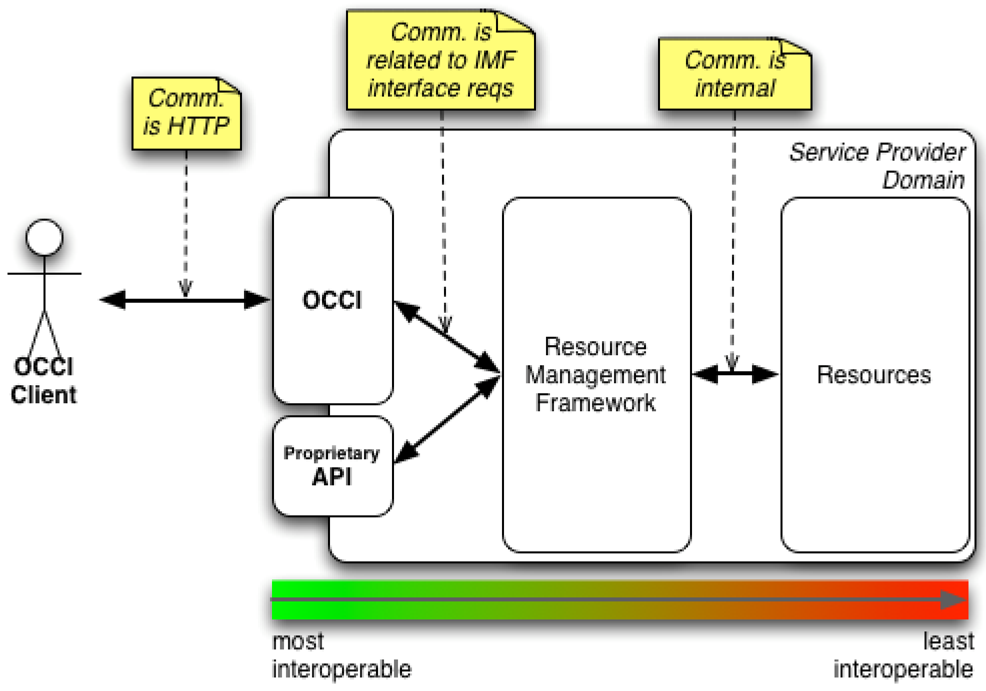


Figure 20: Open Cloud Computing Interface (Ref: [41])

3.3.4 Open-Multinet

The Open-Multinet3 upper ontology [42] is the product of a research over new models to describe resources and experiments in a way that interoperability between federated infrastructures is supported. It was considered that a tree-based representation for resources, as used in the definition of RSpecs and extensions, aggravates such interoperability. The main contribution is a semantic description of resources that can be used in Future Internet testbeds, but also on related fields like Intercloud, IoT or Software-Define Networks (SDNs).

Co-funded by the Horizon 2020 Framework Programme of the European Union

It includes a formal information model [43] for federated infrastructures (OpenMultinet, OMN) as well as a Java library (omnlib) to translate between such ontology, GENI v3 RSpecs and extensions, OASIS TOSCA, and the IETF NFV models. Specifically, this translation tool converts stateless GENI RSpec XML documents into RDF and back using the OMN ontology.

3.3.5 XML-RPC

XML-RPC [44] is a specification and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet. Its remote procedure calls using HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted processed and returned. A set of compatible XML-RPC implementations span all operating systems, programming languages, dynamic and static environments, open source and commercial, for Perl, Python, Java, Frontier, C/C++, Lisp, PHP, Microsoft .NET, Rebol, Real Basic, Tcl, Delphi, WebObjects and Zope, and more are coming all the time.

In Fed4FIRE, the federation AM API is provided via XML-RPC over an SSL connection. Aggregate Managers shall require client-side certificates to authenticate users, accepting only certificates that comply with the adopted GENI certificates standards. This API therefore assumes that users have already been authenticated, and that the aggregate manager has available the client certificate to identify the user. Figure 21 highlights the role of XMLRPC in federation tools.

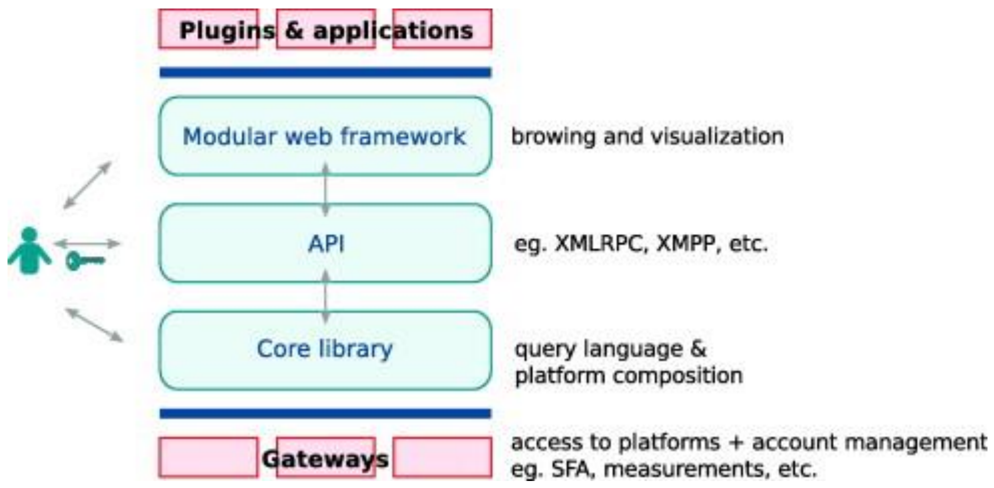


Figure 21: XMLRPC in Federation (Ref: [45])

3.3.6 GENI API

The GENI API is the de-facto standard interface used to expose resources, and thus to federate resources and testbeds. It builds on SFA, XML-RPC, RSpec and comes in two versions (v2, v3). The offerings of this interface are two-fold:

1) Expose resources of a given infrastructure. The API relies in a number of methods and workflows to be followed in order to allocate, provision, manage and terminate resources. To support the former, the GENI organisation and Open-Multinet provide materials to build upon them, like [46] or [47] .

Co-funded by the Horizon 2020
Framework Programme of the European Union

2) Allow experimenters to contact the aforementioned API and to define their experiments. To do so, the experiment is defined through data models that are based on RSpecs. This applies to resources like virtualised and physical computing nodes, slices within OpenFlow-capable switches, stitching paths across infrastructures, wireless nodes, and the like.

Along with this API, GENI provides a number of related standards and tools. One of such standards is the common federation API [48] , which should be met by any GENI-compliant testbed and helps providing and manage identity for any user of the GENI API [49] .

3.4 FEDERATION TOOLS ASSESSMENT

In this section, we provide an assessment of some tools/solutions, which will help the testbeds federation. Similar methodology as Section 2.5 has been utilized here.

Table 2: Federation tools assessment

Type	Framework	API/s supported	Ease of use	Usage in projects
CLI	SFA (SFA/GENI provisioning)	SFA	🟨	Fed4FIRE, OFELIA, FIBRE, FELIX, …
	geni-tools (SFA/GENI provisioning), OpenGENI	SFA, GENIv3	🟧	Fed4FIRE, GENI, …
	NEPI (other provisioning)	SSH	▨	Fed4FIRE, …
	OMF6 (other provisioning)	OMF	▨	Fed4FIRE
	FlowVisor (network hypervisor)	OpenFlow 1.0, XMPLRC	🟧	Fed4FIRE, OFELIA, FIBRE, FELIX, …
REST, GUI, CLI	ONOS (Network controller OS)	OpenFlow 1.0, 1.3, netconf, …	🟩	CORD, GÉANT4/JRAs, Huawei/FusionSphere, …
	ODL (Network controllers frameworks)	OpenFlow 1.0, 1.3, netconf, …	🟩	TeNOR, CHARISMA, TID/OpenMano, …
	CORD (DC (SDN&NFV) provisioning framework)	OpenFlow 1.0, 1.3, netconf, TOSCA, …	▨	Fed4FIRE
	TeNOR (SDN&NFV orchestration)	OpenFlow 1.3, REST, …	🟩	T-NOVA, CHARISMA, SESAME, …
	OpenStack (resource provisioning framework)	REST, …	🟩	T-NOVA, CHARISMA, SESAME, GÉANT4/JRA2, …
GUI	OFELIA (SFA/GENI provisioning)	SFA	🟩	Fed4FIRE, OFELIA, FIBRE, GÉANT3/GTS …
	MySlice (SFA/GENI provisioning)	SFA, GENIv3	🟨	Fed4FIRE, FIBRE, …
	YourEPM (application/service orchestration)	OCCI, REST (BPM workflows)	▨	Fed4FIRE

* Perceived ease of use (from more easy to more difficult) – gray: unknown → 🟩🟨🟧▨

Federation between facilities must be driven by experimenter requirements, as the cost of federation between different domains of control is high, even if interoperability can be achieved. Examining the FLAME validation experiment scenarios, it is clear that the site specific and localized infrastructure focus means that federation between infrastructures is not needed to achieve the goals of experiments and trials. Therefore the benefits of aligning with a federation API is low in relation to the cost of implementation, especially when that API is so to experimentation and not media service developers. FLAME is interested in infrastructure abstraction and the infrastructure APIs based on OpenStack and

Co-funded by the Horizon 2020
Framework Programme of the European Union

OpenFlow allow for interoperability and vertical federation between heterogeneous deployments. However, federation tools offered by FIRE+ are not needed to achieve this goal.

Co-funded by the Horizon 2020
Framework Programme of the European Union

4 CONCLUSION

According to the objectives of T2.4: Analysis, Selection and Adaptation of FIRE+ Technologies and Tools, this document provides an assessment of existing experimenters' interaction and federation tools available in the framework of FIRE+ facilities. Depending on the FLAME requirements in future, the current assessment might help the selection of one set of tools as bases for the FLAME experimenters' interaction and federation tools. Such selection is subject to the actual needs and requirements of the FLAME project.

The analysis did not show a real contribution to the adoption of any of the tools studied so far, although some ideas can be used.

Co-funded by the Horizon 2020
Framework Programme of the European Union

5 REFERENCES

[1] Fed4FIRE+ site, https://www.fed4fire.eu/
[2] Fed4FIRE+ testbeds, https://www.fed4fire.eu/testbeds/
[3] jFed tool, http://jfed.ilabt.imec.be/
[4] Fed4FIRE+ testbeds monitoring overview, https://fedmon.fed4fire.eu/overview
[5] GENI Clearinghouse, http://groups.geni.net/geni/wiki/GeniClearinghouse
[6] jFed tool features, https://jfed.ilabt.imec.be/features/
[7] SFI tool, https://old.fed4fire.eu/sfi/
[8] Slice-based Federation Architecture, https://wiki.confine-project.eu/sfa:arch
[9] PlanetLab site, https://www.planet-lab.org/
[10] YourEPM site, https://www.fed4fire.eu/tools/yourepm/
[11] Activiti BPM site, http://activiti.org/
[12] OneLab site, https://onelab.eu/
[13] MySlice site, https://www.myslice.info
[14] Fed4FIRE+ GUI portal, https://portal.fed4fire.eu/
[15] MySlice development site, http://trac.myslice.info/
[16] NEPI site, http://nepi.inria.fr
[17] NEPI tutorials, https://r2lab.inria.fr/tutorial.md
[18] T. Rakotoarivelo, M. Ott, G. Jourjon, Iv. Seskar, "OMF: a control and management framework for networking testbeds", ACM SIGOPS Operating Systems Review archive Volume 43, Issue 4, 2010.
[19] OML, https://wiki.confine-project.eu/oml:start
[20] OCF and OMF, https://fibre.org.br/infrastructure/experimentation-environments/
[21] GENI site, http://www.geni.net/about-geni/what-is-geni/
[22] Flack site, http://www.protogeni.net/wiki/Flack
[23] Flack description, http://geni-app-developer-documentation.readthedocs.io/en/latest/tools/flack.html
[24] Jacks: getting started, http://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials/jacks/GettingStarted_PartI/Procedure
[25] Jack: getting started (part I), http://groups.geni.net/geni/wiki/GENIExperimenter/Tutorials/jacks/GettingStarted_PartI/Procedure/DesignSetup
[26] OMNI site, https://github.com/GENI-NSF/geni-tools/wiki/Omni
[27] GENI Aggregate Manager API, http://groups.geni.net/geni/wiki/GAPI_AM_API
[28] FP7-OFELIA site, www.fp7-ofelia.eu
[29] FIBRE site, http://fibre.org.br/
[30] https://ec.europa.eu/research/participants/portal/desktop/en/opportunities/fp7/calls/fp7-ict-2011-eu-brazil.html
[31] FIBRE connectivity, https://www.rnp.br/en/services/connectivity/ipe-network
[32] FIBRE site, http://fibre.org.br/about/what-is-fibre/
[33] Jad Naous, Path-Policy Compliant Networking and Platform for Heterogeneous IaaS Management, PhD Thesis, Stanford, March 2011 [White, 2010] White, J. (2010). A Tutorial Introduction to OML. GEC9. Available at http://omf.mytestbed.net/projects/oml
[34] Public Deliverable 5.1 of OFELIA Project. Available at https://fibre.org.br/wp-content/uploads/2015/06/OFELIAD5-1-final.pdf
[35] Arkills, B (2003). LDAP Directories Explained: An Introduction and Analysis. Addison-Wesley Professional. ISBN 0-201-78792-X.

Co-funded by the Horizon 2020 Framework Programme of the European Union

[36] SFA specifications, http://git.planet-lab.org/?p=sfa.git;a=blob;f=docs/sfa.pdf;h=3edd2a9259e0f67ba23ac9e1f524af8b383963bb;hb=9f731109bf3be4c02475c6f56d5e4319281e3e76
[37] GENI Aggregate Manager APIv2, http://groups.geni.net/geni/wiki/GAPI_AM_API_V2#APIMethods
[38] GENI Aggregate Manager APIv3, http://groups.geni.net/geni/wiki/GAPI_AM_API_V3/CommonConcepts#RSpecdatatype
[39] GENI Aggregate Manager APIv3 (RSpec definition), http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs#GENIv3
[40] OCCI site, http://occi-wg.org/
[41] OCCI description, http://occi-wg.org/about/
[42] A. Willner, C. Papagianni, M. Giatili, P. Grosso, M. Morsey, Y. Al-Hazmi, I. Baldin; "The Open-Multinet Upper Ontology ". Available at http://eudl.eu/pdf/10.4108/icst.tridentcom.2015.259750
[43] The Open-Multinet Specification, http://w3c.github.io/omn/
[44] XML-RPC site, http://xmlrpc.scripting.com/
[45] Jordan Augé, et. al, "Tools to foster a global federation of testbeds," Computer Networks, Vol 63, 22 April 2014, Pages 205-220.
[46] GENI tools, https://github.com/GENI-NSF/geni-tools
[47] GENI Aggregate Manager APIv3 (Docker-based), https://github.com/open-multinet/docker-am
[48] GENI's Common Federation API v2, https://geni-nsf.github.io/CommonFederationAPI/CommonFederationAPIv2.html
[49] Certificate-based Clearinghouse, https://github.com/EICT/C-BAS

Co-funded by the Horizon 2020
Framework Programme of the European Union