



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

D5-4– Detailed specifications for third cycle ready

Work package	WP5
Task	T5.1 – T5.6
Due date	27/02/2015
Submission date	26/03/2015 (final internally reviewed and corrected version)
Deliverable lead	Mikhail Smirnov (Fraunhofer)
Version	01
Authors	Mikhail Smirnov (Fraunhofer) Florian Schreiner (Fraunhofer) Alexander Willner (TUB) Chrysa Papagianni (NTUA) Aris Leivadreas (NTUA) Tsiropoulou Eirini (NTUA) Donatos Stavropoulos (UTH) Elena Garrido Ostermann (Atos) Javier García Lloreda (Atos) Carlos Bermudo (i2CAT) Thierry Parmentelat (INRIA) Loïc Baron (UPMC) Brecht Vermeulen (iMinds)
Reviewers	Tim Wauters (iMinds) Steve Taylor (IT Innovation)

Abstract	This deliverable specifies the development of the common federation tools for experiment lifecycle management in the third 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 to be developed in cycle 3 of the Fed4FIRE project.

In Fed4FIRE, WP5 “Experiment Workflow and 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 facilities.

This deliverable reports the result of the cycle 3 specification. As a fundamental basis, the specifications of the third development cycle in WP5 takes into account earlier specified requirements together with new ones, identified through the feedback from experimenters, performance analysis and the work towards federation sustainability. The WP5 work is now extended also to support policies that are going to be designed by the Federation Board and maintained by the Federator in a form of Operational Level Agreements

Major implementations specified in this deliverable include:

1. Operational Level Agreement as a placeholder of all federation-wide policies designed by the Federation Board, maintained by the Federator and enforced by all facilities together with an approach for pragmatical KPI adaptations in case of a SLA violation;
2. To automate the resource description and discovery in collaboration with the international Open-Multinet (OMN) Forum an upper OMN ontology has been defined that is split into a hierarchy of a number of different ontologies; the needed software including a translation service is specified and its performance was evaluated;
3. Six enhancements to reservation broker were specified together with the specification of a reservation plugin (Scxhederal); a taxonomy of reservable resources was mapped to the federated facilities;
4. Resource provisioning specification is enhanced with the orchestration function via MySlice, jFed and YourEPM (new in cycle 3) tools, accordingly the SLA management process is fully described including the SLA Dashboard tool specification and usage examples; moreover resource provisioning service is fully defined with all adopted policies;
5. Experiment control in cycle 3 is specified with the two complementary approaches – tools and usage with the description of common features for usages within learning and production phases;
6. A new feature of Fed4FIRE portal in cycle 3 – management of “projects” – is specified in interaction with reservation broker and with experiment control.

Acronyms and Abbreviations

AA	Authorization and Authentication
AM	Aggregate Manager
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AR	Action Result
BPM	Business Process Modelling
CAMAD	Computer Aided Modeling and Design of Communication Links and Networks
CLI	Command Line Interface
CI	Configurable Item (also: Continuous Integration [methodology])
CMS	Content Management System
CRUD	Create-Read-Update-Delete
DAG	Directed Acyclic Graph
DM	Data Model
DOM	Document Object Model
EC	Experiment Controller
FB	Federation Board
Fed4FIRE	Federation for Future Internet Research and Experimentation Facilities
FCI	Federation Computing Interface
FIRE	Future Internet Research and Experimentation
FitSM	Federated IT Service Management [methodology]
FRCP	Federated Resource Control Protocol
FUSECO	Future Seamless Communication [facility]
GENI	Global Environment for Networking Innovation
GUI	Graphical User Interface

JAXB	Java Architecture for XML Binding
KPI	Key Performance Indicator
KPI-M	Measured KPI
KPI-T	Target KPI
KPI-U	Uniform KPI
LCA	Least Commonly Agreed [policy]
LGPL	Lesser General Public License (GNU)
LTE	Long Term Evolution [architecture]
MAS	Management, Abstraction and Semantics
NDL	Network Description Language
NEPI	Network Experiment Programming Interface
NITOS	Network Implementation Testbed using Open Source platforms
NICTA	National ICT Australia
OASIS	Organization for the Advancement of Structured Information Standards
OCCI	Open Cloud Computing Interface
OCF	OFELIA Control Framework
OGF	Open GRID Forum
OF	OpenFlow
OFELIA	OpenFlow in Europe: Linking Infrastructure and Applications
OLA	Operational Level Agreement
OM	Object Model
OMA	Open Mobile Alliance
OMF	cOntrol and Management Framework
OML	ORBIT Measurement Library
OMN	Open MultiNet [Forum]
OMSP	OML Measurement Stream Protocol

ORCA	Open Resource Control Architecture
OWL	Web Ontology Language
PDP	Policy Decision Point
PLE	PlanetLab Europe [facility]
PI	Principal Investigators
PyPElib	Python Policy Engine library
RA	Resource Adapter
RAML	RESTful API Modeling Language
RC	Root Cause
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RPC	Remote Procedure Call
RSpec	Resource Specification
SAWSDL	Semantic Annotations for WSDL and XML Schema
SC	Service component
SFA	Slice-based Federation Architecture
SLA	Service Level Agreement
SM	Semantic Model
SON	Self Organised Network
SPARQL	Protocol And RDF Query Language
SSH	Secure Shell
TDD	Test Driven Development
TOSCA	Topology and Orchestration Specification for Cloud Applications
UI	User Interface
URL	Uniform Resource Locator

VCT	Virtual Customer Testbed
VM	Virtual Machine
WADL	Web Application Description Language
XML	eXtensible Markup Language
XMPP	eXtensible Messaging and Presence Protocol
YourEPM	Your Experiment Process Manager

Contents

1	Introduction.....	10
2	Inputs to this Deliverable	11
2.1	Cycle 3 from the architectural viewpoint (Task 5.1)	11
2.2	OLA: operational definition	13
3	Specification of experiment life-cycle management (cycle 3)	16
3.1	Resource description and discovery (Task 5.2)	16
3.1.1	Introduction.....	16
3.1.2	Related Work on Information Modeling	16
3.1.3	Ontology Specification	18
3.1.4	Software Specification.....	19
3.1.5	Evaluation	22
3.1.6	Summary and Outlook for Information Modeling in Cycle 3	24
3.2	Central Reservation Broker Specification (Task 5.3)	25
3.2.1	Resource Reservation Overview.....	25
3.2.2	Reservation Broker in Cycle 2.....	25
3.2.3	Reservation Broker in Cycle 3.....	27
3.2.4	Reservation Plugin.....	29
3.2.5	Reservable Resources.....	31
3.3	Resource Provisioning (Task 5.4).....	35
3.3.1	Resource provisioning overview	35
3.3.2	Provisioning in Cycle 3.....	35
3.3.3	Policies.....	49
3.4	Experiment control (Task 5.5)	52
3.4.1	Approach	52
3.4.2	Current position of the various tools involved.....	52
3.4.3	Usages	54
3.4.4	Specifications.....	55
3.4.5	Summary and Conclusion	57
3.5	User Interface / Portal (Task 5.6)	58
3.5.1	Fed4FIRE Portal	58
3.5.2	Authority Portal.....	60
4	Conclusion and Future Plans	62
	References.....	63
5	Annex 1: Extending FitSM with Service Policies.....	64
6	Annex 2: Resource Provisioning Service.....	66
7	Annex 3: jFed functionality.....	68

1 Introduction

The Fed4FIRE project has successfully developed all tools specified in D5.2 for the experiment workflow and 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, 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 / AMQP server
- Experiment controller/ control server
- Scenario editor
- Documentation centre
- Portal
- Authority directory
- Service directory
- Future reservation broker
- Stand-alone tools (jFed, NEPI)

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; major highlights are below.

1. Task 5.1: Operational Level Agreement as a placeholder of all federation-wide policies designed by the Federation Board, maintained by the Federator and enforced by all facilities together with an approach for pragmatistical KPI adaptations in case of a SLA violation;
2. Task 5.2: to automate the resource description and discovery in collaboration with the international Open-Multinet (OMN) Forum an upper OMN ontology has been defined that is split into a hierarchy of a number of different ontologies; the needed software including a translation service is specified and its performance was evaluated;
3. Task 5.3: six enhancements to reservation broker were specified together with the specification of a reservation plugin (Scxhedular); a taxonomy of reservable resources was mapped to the federated facilities;
4. Task 5.4: resource provisioning specification is enhanced with the orchestration function via MySlice, jFed and YourEPM (new in cycle 3) tools, accordingly the SLA management process is fully described including the SLA Dashboard tool specification and usage examples; moreover resource provisioning service is fully defined with all adopted policies;
5. Task 5.5: experiment control in cycle 3 is specified with the two complementary approaches – tools and usage with the description of common features for usages within learning and production phases;
6. Task 5.6: a new feature of Fed4FIRE portal in cycle 3 – management of “projects” – is specified in interaction with reservation broker and with experiment control

2 Inputs to this Deliverable

Additionally to the development plans that were first outlined in D5.1 [1], and then précised and enhanced in D5.2 [2] this deliverable addresses the following inputs that are considered relevant for experiment workflow and life-cycle management in a federated environment. First, the feedback collected from several experimental projects selected after Fed4FIRE open calls; this feedback was processed by Task2.3 and served as an input to our work. Second, several architectural improvements that were analysed and recommended by Task2.1 did affect the detailed makeup of WP5 “services”. Third, in preparation for sustainability phase the project decided to specify precisely the Federation Board, its organisation, operation and the outcome. This work has started three months ago with the creation of a new Task2.4, which produced the first release of the FB specification [3] taken into account in this document.

2.1 Cycle 3 from the architectural viewpoint (Task 5.1)

In cycle 1 the main work of WP5 was to collect information about tools and systems available for experiment workflow and life-cycle management in a federated environment, evaluate those and to perform a gap analysis that steered the further development. During this work multiple requirements were taken into account, structured and reported in D5.1 [1], while there implementation was outlined in D4.3 [6].

In cycle 2 this work continued but the mainstream effort was to integrate the selected and developed tools and platforms into the Fed4FIRE experiment workflow system following the FitSM methodology [7] adopted by the Fed4FIRE project. The outcome of this work reported in D5.3 [2] is service orientation adopted by all tasks of WP5 and strongly related to the work of WP6 and WP7 in supporting SLA’s and reputation service within the Fed4FIRE ecosystem.

The WP5 specification work in cycle 3 reported in this document is the logical consequence of the service orientation and follows the adoption of particular governance structure being worked out by the Task2.4. Figure 1 presents a concept map of the emerging governance structure of Fed4FIRE; this structure should be considered as work in progress though it already gives a useful orientation. In short, the WP5 has enhanced the FitSM service descriptions with the four types of policies that are to be designed and managed by the FB, maintained by the Federator and enforced by the testbeds. These types are:

- access policy: e.g. Open Access policy - a user with Fed4FIRE credentials can access services and resources of the federation:
- usage policy: e.g. Least Commonly Agreed (LCA) policy – an Open Access policy is constrained at each testbed by the predefined experiment type specified in the testbed tutorial;
- dependency policy: e.g. Concurrency Policy - constraints for concurrent usage of multiple resources;
- Pricing policy. E.g. flat rate, usage based, utility based, reputation based, etc.

In cycle 3 a novel architectural concept of OLA becomes important for WP5 specification work because it directly follows the service orientation. For a sustainable federation FitSM recommends to care about offered services in the following way:

1. each service is to be described by a service portfolio entry (a template with the three sub-records shown in Annex 1: Extending FitSM with Service Policies);

2. each service is to be protected by an SLA (assuming that even best-effort service offering must have an SLA that specifies e.g. planned service outages and service maintenance schedule);
3. service evolution proceeds in the direction of enhancing service capabilities (increasing maturity levels) reflected by the corresponding modifications of service portfolio records.

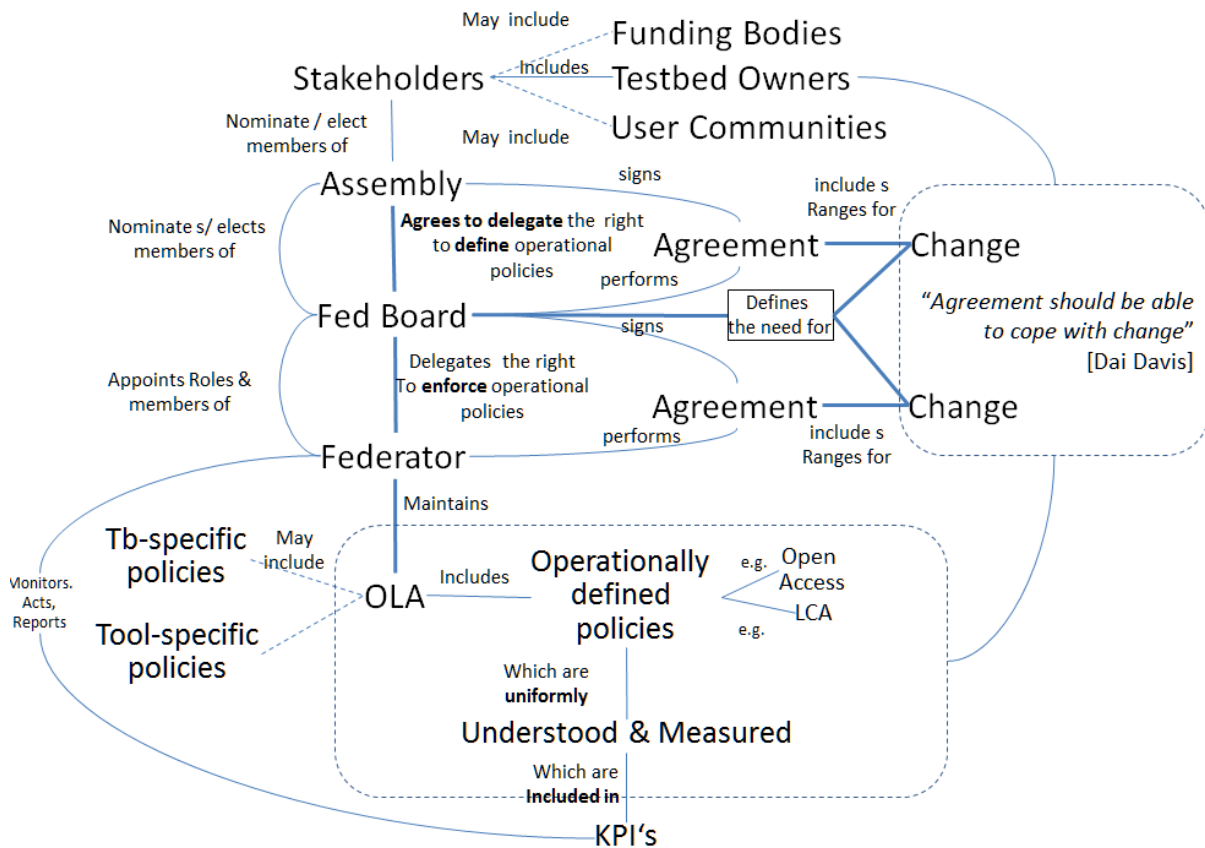


Figure 1: Fed4FIRE Governance structure (work in progress)

The service protection by an SLA means that service scope and service levels defined in an SLA are being monitored by a provider, can be observed by a consumer, and in case of [predicted] violation an action is taken by a provider to reduce the harm. SLA violation is such degradation of prescribed KPI's that their values are not within the prescribed ranges. SLA's per se are not in the focus of this document and should be detailed elsewhere, while in this document we shall concentrate on

- methods to predict SLA violations, and
- actions to be taken to avoid SLA violations.

Obviously the above two items are largely defining the OLA design, because methods for SLA violation predictions must inevitably be based on KPI monitoring and the actions to be taken in response to the predictions must be applied to root causes of the KPI's violations. Accordingly, the KPI set must be defined and clustered into two groups:

1. Monitored KPI (KPI-M) set that is being defined by the offered service portfolio, and
2. Target KPI (KPI-T) set to be acted upon.

The source of KPI-T definition as shown in Figure 1 is within the Fed4FIRE operation and governance structure, while the source of KPI-M definition is within the experiment workflow and life-cycle management. As Figure 2 shows the experimenter expects an SLA per workflow, while practical support of a workflow is to be implemented via the OLA maintained by the Federator and enforced by the testbeds and by their interconnections in case of concurrent usage.

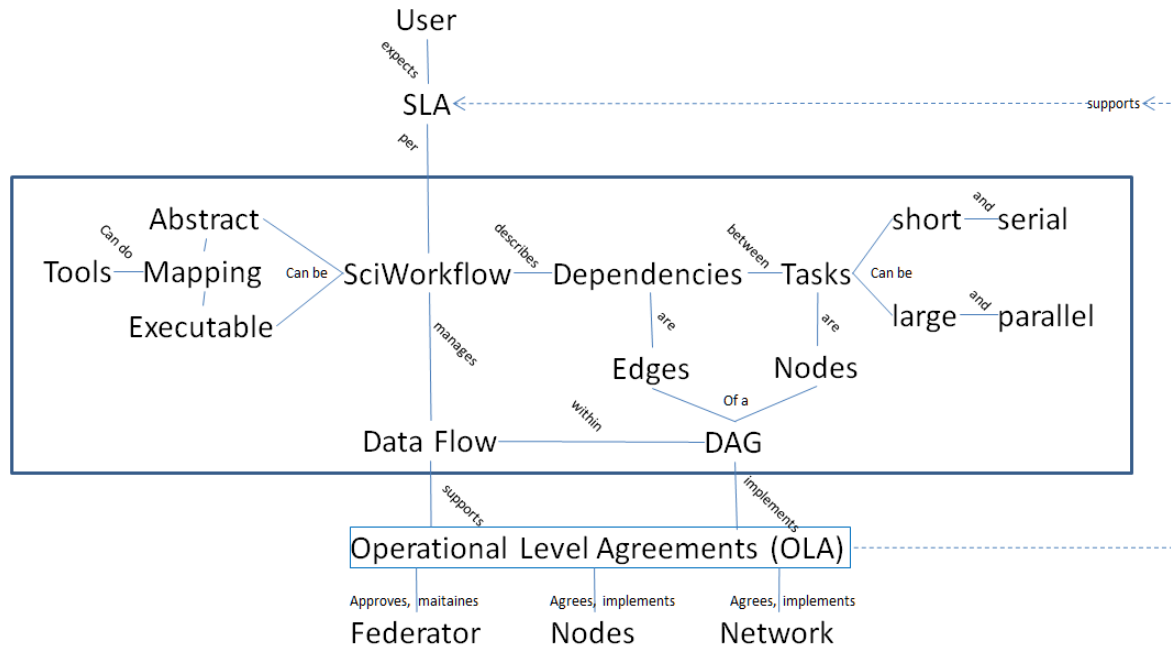


Figure 2 OLA from the experimenter viewpoint

Next, we define the OLA operationally as a placeholder of all federation policies to be enforced.

2.2 OLA: operational definition

We concentrate now on a pragmatic approach to implement OLA. This approach is rooted in the envisaged three-tier governance structure of a sustainable federation shown in Figure 1, where the Federation Board is empowered by the Assembly with the right to define and maintain Federation-Wide Policies within the prescribed range of agreements. The set of federation policies constitutes federation OLA and is maintained by the Federator, enforced by testbeds. As any other business agreement an OLA, a set of federation policies must be based on operational definitions.

The meaning of operational definition is best explained by quoting the [8] as a “translation of a concept into measurement of some kind.”

Directly following these explanations we suggest that federation policies are based on KPI’s that is uniform federation-wide, meaning that their semantics and the process of monitoring are the same throughout the federation. It is reasonable to suggest that the set of these uniform KPI’s (KPI-U) is based on utility metrics. However we need to recognize that typically policies are associated with services, no surprise because they are strongly connected to respective SLA’s.

Indeed, in a service oriented world an OLA can be seen as a placeholder for policies. A federation may use different methods to keep a record of all policies that are agreed upon and that are maintained, that is managed, enforced, modified, etc. in full accordance with the policy life cycle. However keeping policies aligned with an OLA helps federation to migrate from pure access control

to a process control, which in turn facilitates another migration path, namely from point correctness to process correctness as required by the sustainability of a federation.

Annex 1: Extending FitSM with Service Policies demonstrates pragmatically with an example of hypothetical cloud service how to associate policies to services and how to differentiate them from those policies that constitute OLA. The starting point will be to look at every service description and extend it with relevant policy fields, which then will be collected and harmonized before being associated with OLA.

Obviously the four types of policies introduced above are service specific. For SLA conformance they will need to be monitored per service specific KPI-M, and, as it is well known, the decision on target KPI-T for service improvements is generally hard because it must involve the Root Cause Analysis (RCA), which in a federation is a complex task. Nevertheless, the pragmatic approach outlined in Figure 3 can be proposed.

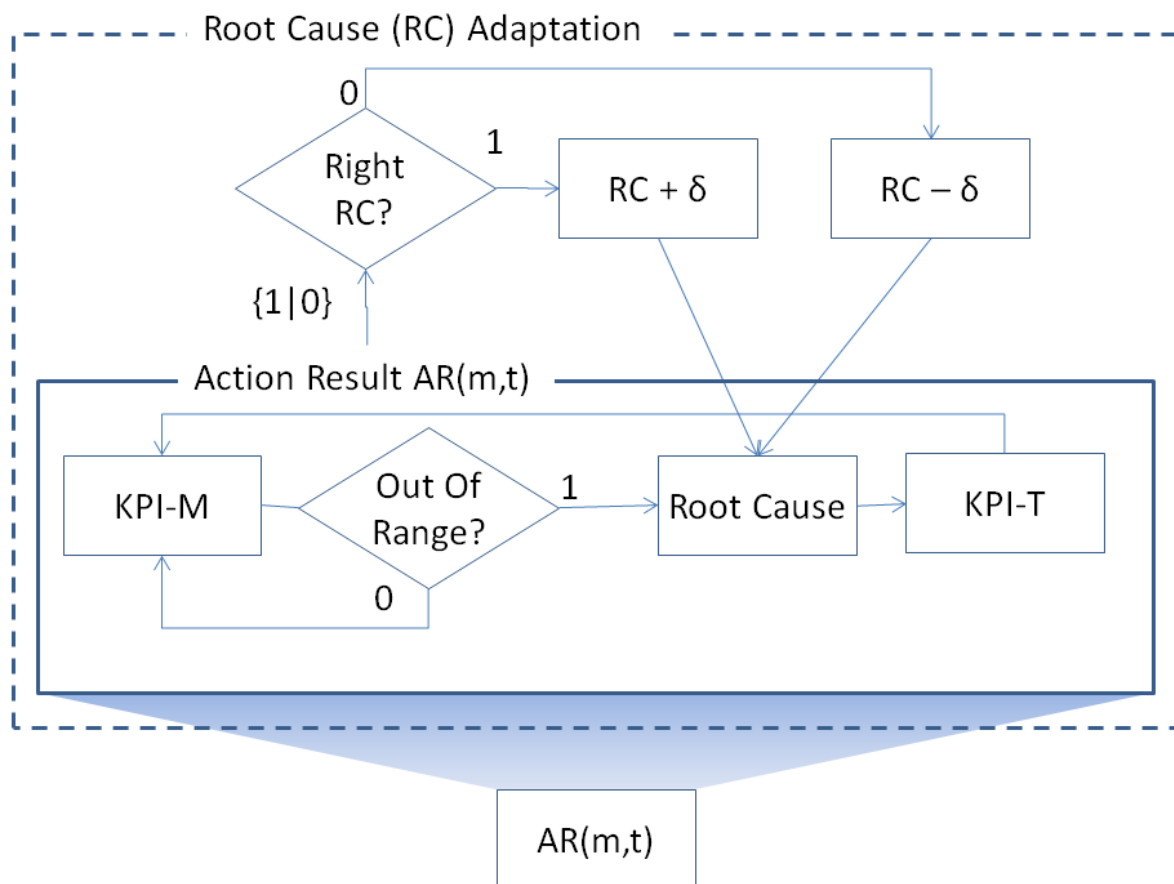


Figure 3 Pragmatic Root Cause Analysis

We explain the approach starting with the separation of two concerns:

- (1) Action on KPI-T in case of KPI-M violation (under assumption that RC is known and valid);
- (2) RC adaptation.

This is demonstrated in Figure 3, where internal module denoted as $AR(m,t)$ – action result on t -th KPI-T after violation of the m -th KPI-M under and assumption that RCA is done right. The outer module provides RCA adaptations but not after each KPI-M violation (which would be service specific KPI-M) however based on smaller set of indicators, namely on violations of KPI-U i.e. those that serve

the basis of federation-wide policies. KPI's denoted as KPI-M - constitute a set of all KPI's that are present in all SLA's offered by a federation; KPI-U are those KPI's that constitute the OLA; the AR(u) is the result of action performed under the violation of KPI-M, the mapping $AR(u) \rightarrow AR(m,t)$ is the one that makes the RCA relatively easy.

The sets of KPI-M pertaining per each service are not disjoint and/or independent for all offered services, thus it is always possible to define generic (unified) KPI's based on service specific ones, for example as it was done for LTE SON in [9] based on utility concept. Thus obtained KPI-U's constitute federation's OLA and enjoy uniform monitoring procedures. In essence, KPI-U's are all being targets for actions under violation of any of KPI-U thresholds, because the federation utility is then the only "monitored KPI". This does not mean that – much simplified in this case – RCA is not needed; however it translates into a set of parameters, on which the KPI-U depend.

3 Specification of experiment life-cycle management (cycle 3)

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 Introduction

The task considers the description and discovery of the two types of resources offered by Fed4FIRE for experimentation: those being objects and those being services. For the former the task specifies on operational ontology specification, while for the latter – on the specification of an SLA-dashboard tool, which is described in the resource provisioning section.

The main advantage of the ontology based resource description is shown in Figure 4: it allows automating a consistent object modelling and is further elaborated below. The advantage of SLA-based service discovery was motivated previously.

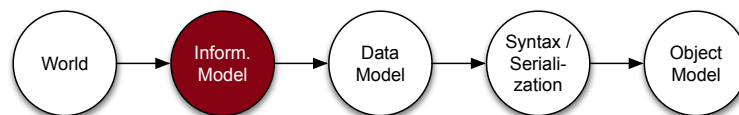


Figure 4 The relationship between different levels of abstraction

However, as stated in D5.2, a “semantic resource directory” is not an envisioned logical component in this context, i.e. there is not a single centralized federator component, as it is not defined in D2.4 as such.

Currently in Fed4FIRE, resources are described based on RSpecs, which have a testbed specific character. In order to support sustainable and open standards, the documentation is put online and open for improvements and changes through the open github model where everyone can comment or issue pull requests: <https://github.com/open-multinet/federation-am-api> (RSpecs are described in `rspec.adoc`). A compiled, more user-firnedly version can be found at <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/rspec.html>.

In Fed4FIRE, for new testbeds, we try to have similar RSpecs for the similar types of resources (e.g. virtual machines, wireless nodes, openflow switches), which works reasonably well for current production tools.

3.1.2 Related Work on Information Modeling

In order to place the contribution in context and to identify the gap the work is intended to fill, we provide a short literature survey of related work in the field of federated resource management with a focus on information modeling.

XML-based RSpecs are the accepted standard for describing the resource life cycle in GENI. These use a set of base XML-schemas and a number of extensions defined for specific purposes (e.g. stitching, OpenFlow and others). The original idea of RSpecs relied on structure-implied semantics in order to express facts about the resources. The location of the particular element or attribute within the Document Object Model (DOM) dictated its meaning. The extension mechanism was introduced in order to support the wider range of resources being incorporated into GENI. An extension is accomplished by using the XML `<any>` tag, which allows a chunk of XML conforming to the extension schema to appear virtually anywhere in the RSpec document. The upside of this approach is the infinite extensibility of RSpecs. The downside is the loss of structure-implied semantics for the

extensions, since there is no mechanism for dictating which extension is allowed to appear in which part of the DOM. Syntax checking cannot be applied, since an RSpec document with a misplaced extension is typically syntactically correct. Instead, this leaves the checking process to the code parsing. This solution does not scale in the long run and forces constraints on extensions to be expressed in procedural code, rather than declaratively, like the rest of the schema.

To overcome these issues and to allow mutual understanding and minimum interoperation, a formal canonical reference model has to be introduced. The current RSpec approach is to specify a loosely defined tree-based Data Model (DM) that is serialized in XML. After translation to an Object Model (OM), the contained information is validated by functional code. Encoding the information about provided, requested, controlled and monitored resources in a Semantic Model (SM) would allow us to exploit the advantages sketched in the introduction in a declarative manner. For these purposes, particular developments from within the Semantic Web community could be adopted, namely the Resource Description Framework (RDF), the Resource Description Framework Schema (RDFS), the Web Ontology Language (OWL), and the SPARQL - Protocol And RDF Query Language (SPARQL).

A number of fields of application have already adopted similar mechanisms. For example, search engines companies Bing, Google, and Yahoo! have collaborated to provide a vocabulary called Schema.org. It provides a shared collection of thematic schemas used to annotate websites in a common way to allow search engines to recognize, evaluate and display their semantics. Further, within the federated cloud context, the Open-Source API and Platform for Multiple Clouds (mOSAIC) ontology has been defined and further been adopted by the IEEE Standard for Intercloud Interoperability and Federation (P2302). Other examples include the Machine-To-Machine Communication (M2M) community, which is developing SMs within the OneM2M Working Group Management, Abstraction and Semantics (MAS); and the semantic web services that have been developed under the Semantic Annotations for WSDL and XML Schema (SAWSDL) umbrella. In fields related to federated testbeds and e-infrastructures, a variety of existing work has been defined, including NML, INDL, NDL-OWL and NOVI.

Based on the preliminary work of the Network Description Language (NDL), NML is an information model designed to describe and define computer networks. The model underwent a thorough review and definition process to finally become an Open Grid Forum (OGF) standard. The developers of NML kept it as general as possible, with the possibility of extension in order to customize it for emerging network architectures and novel use cases.

INDL describes computing infrastructures in a technology independent manner. INDL also imports NML, which enables it to seamlessly include the networking part of a computing infrastructure. This ontology adds concepts and relations that are specific to the computing, processing and storage part of an infrastructure, e.g. Processing Component, and Memory Component. INDL further addresses the modeling of resource and service virtualization; it supports description, discovery, modeling, composition, and monitoring of resources.

NDL-OWL was one of the first attempts to design resource life cycle ontology for GENI. Created to support the control framework Open Resource Control Architecture (ORCA), it grew out of the original NDL, however was extended in a number of important ways. The notion of resources and their life cycle was added into the ontology by creating the request, advertisement and manifest models. The models express information about a slice request, about the state of the substrate of the provider or the slice as built, respectively. Importantly, NDL-OWL supports the concept of multiple abstract delegation models, which can be constructed from a single detailed resource description of the substrate generated by the provider. The reason for the models is the need to

preserve the privacy of the provider, allowing it to disclose only certain details of its internal topology. Several levels of abstraction were defined for NDL-OWL advertisements, with the switch being the most commonly used today. Within a switch, a single domain is abstracted into a switch fabric with multiple interfaces facing its peers. This abstraction succeeds in supporting inter-domain path-finding. For more sophisticated topology embedding tasks, more detailed abstract models can be used.

The NOVI information model defines the semantics needed to describe resources and services, policy-based management systems and monitoring capabilities. It further describes communications in the NOVI architecture that focus on the federation of virtualized e-infrastructures. The NOVI information model enables semantic interoperability among the various software components of its architecture. The development of NOVI was driven by requirements including, in particular, the need to support virtualization concepts, context-aware resource selection, and harmonization of monitoring information and measurement units. As a result, the NOVI model comprises three main ontologies: resource, policy and monitoring. NOVI further imports NML for supporting network description. Although it is modular, vendor-independent, and uses the OWL language, the information model is limited to the scope of NOVI architecture. For instance, the resource ontology describes resources in NOVI by differentiating between physical and virtual nodes and network connectivity elements. Thus, extending this ontology to include resources from other domains (e.g. Wi-Fi, IoT) may not be straightforward.

The above overview of the current state of the art in describing resources within federated infrastructures has identified some of the main areas of focus and some drawbacks. A major outstanding problem is that the integration of each approach into current GENI and FIRE platforms is missing and a broader scope of application has not been considered.

3.1.3 Ontology Specification

One of the main objectives of Task 5.2 is the development of an information model to describe resources and their requirements in order to overcome the aforementioned issues. As a result, the international Open-Multinet (OMN) Forum has been established to define a set of upper ontologies by the active contribution of Fed4FIRE and other partners.

As a result and as shown in Figure 5, an upper OMN ontology has been defined that is split into a hierarchy of a number of different ontologies. The omn ontology on the highest level defines basic concepts and properties, which are then re-used and specialized in the subjacent ontologies. Included at every level are (i) axioms, such as the disjointness of each class; (ii) links to concepts in existing ontologies, such as NML, INDL and NOVI (cf. Figure 6); and (iii) properties that have been shown to be needed in related ontologies.

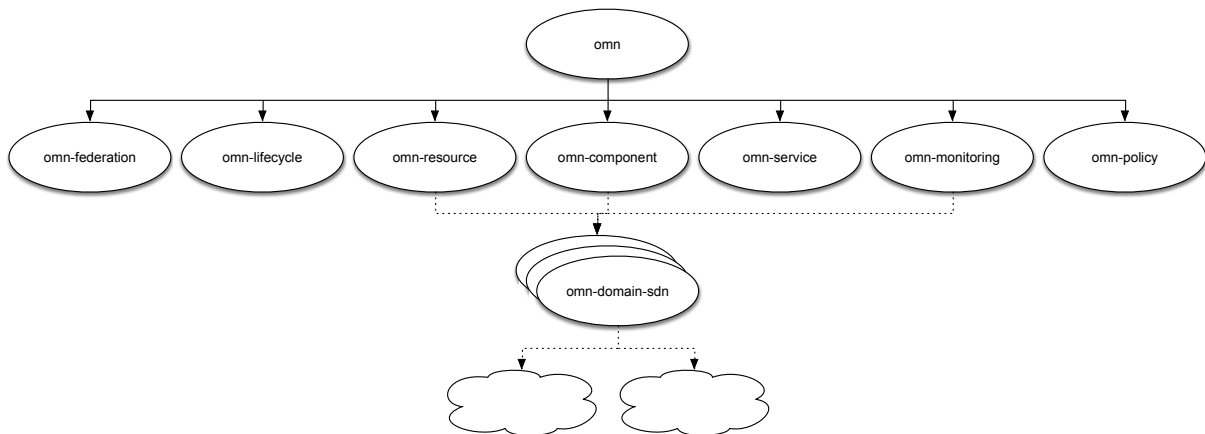


Figure 5: Open-Multinet Upper Ontologies

As next steps, the lower upper ontologies for federations, the life-cycle phases, resources, services, components, policies and monitoring information will further be defined and specialized by domain specific ontologies for Fed4FIRE related technologies such as for SDNs or wireless domains.

As shown in Figure 6, **Error! Reference source not found.**the approach followed in the Open-Multinet Forum does not only import existing work, but also include concepts to convert between XML-based GENI RSpecs and the formal information model. Based on this, in Fed4FIRE initial implementations have been conducted and further extensions for cycle 3 are foreseen that will be described in the next section.

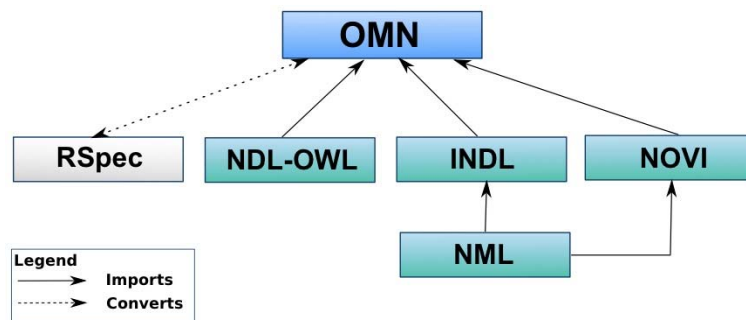


Figure 6 Open-Multinet Ontology Import and Conversion

3.1.4 Software Specification

In order to facilitate the adoption of the conducted work on the formal specification of information within federated infrastructures a number of software integration concepts are possible that have

partially been implemented. As shown on the right side of Figure 7, different layers of the upper ontologies can be integrated in different areas of the Fed4FIRE federation concept. Namely, the formal description of the federation can be embedded into the Fed4FIRE website as RDFa-encoded HTML attributed. Another option is to provide a SPARQL endpoint at this level, extending the existing authorities.xml file or to offer an extended version of the GENI clearing house (CH) API.

Each infrastructure can then provide semantic meta-data by an extended GENI AM GetVersion call or by offering its own SPARQL endpoint that publishes resource information compatible with the Semantic Web community.

Further, in order to discover, provision, control and monitor resources, semantic information models can be exchanged within the GENI AM method calls (RDF/XML serialized), in FRCP calls (JSON-LD serialized) or in OML data streams (using a native RDF/OMSP serialization).

Existing ontologies are being published via <http://open-multinet.info/ontology> and maintained within <https://github.com/open-multinet/playground-rspecs-ontology>.

Further, concrete artifact have already been implemented and are subject of enhancements within the next cycle.

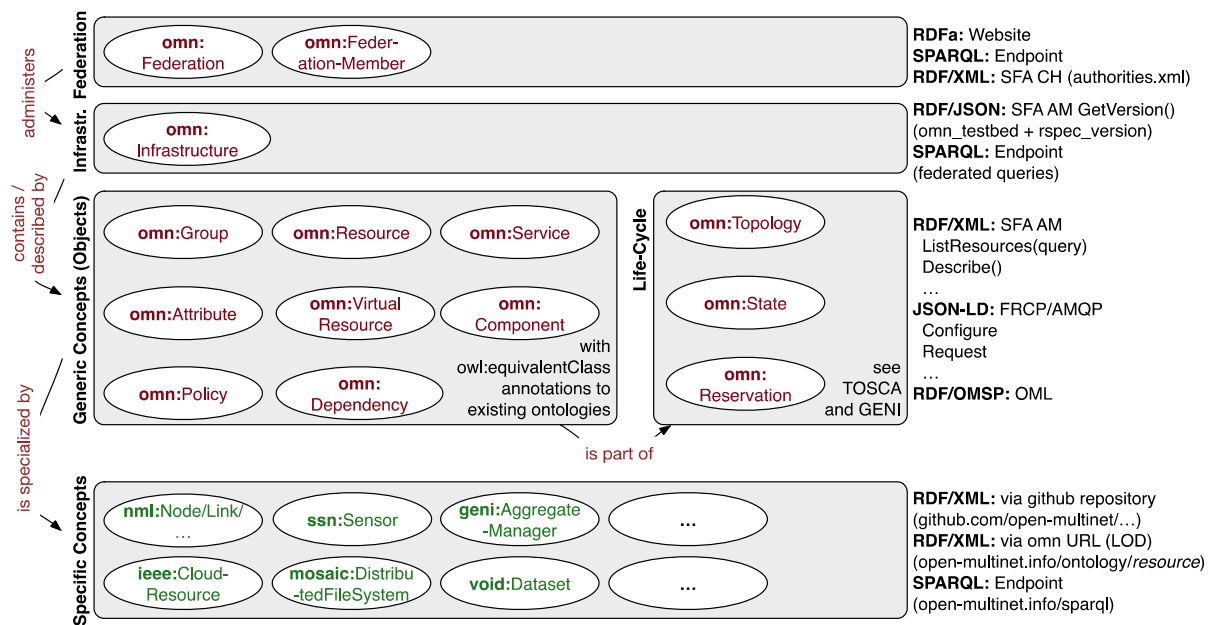


Figure 7 Open-Multinet Software Integration Concepts

3.1.4.1 Translator Service

Appropriate tools are needed to facilitate the transition of non-semantic management systems towards using graph based information models and the integration of semantic management systems into the GENI context. Therefore, another objective in Task 5.2 is the development of such a translation mechanism. These tools should support translating locally used structured, semi-structured and unstructured data models into RDF-based data and the translation from RDF data into GENI RSpecs. This approach has several advantages. The tools (i) automate and speed up the process of converting non-RDF data; (ii) encourage users and developers to migrate their systems to using Semantic Web technologies; and (iii) ensure that the quality of generated RDF data corresponds to its counterpart data in the original system. As a result, TUB has started to develop a translation tool to convert stateless GENI RSpec XML documents into RDF and back using the OMN ontology. The tool

parses the XML tree and converts the tags and attributes to their corresponding classes or properties; it also supports converting the complete GENI resource life cycle messages.

The implementation of the translation tool follows a Test Driven Development (TDD) approach, is included in a Continuous Integration (CI) environment with test coverage analytics, and is offered as a Java based open source library Implementation ("omnlib") in a public maven repository (see github link above). It uses the Java Architecture for XML Binding (JAXB) and Apache Jena to map between XML and RDF and Java objects, and supports a number of APIs: (i) a native API to be included in other Java projects; (ii) a CLI to be used within other applications; and (iii) a REST based API to run as a web service. A preliminary GUI for the later is shown in Figure 8.

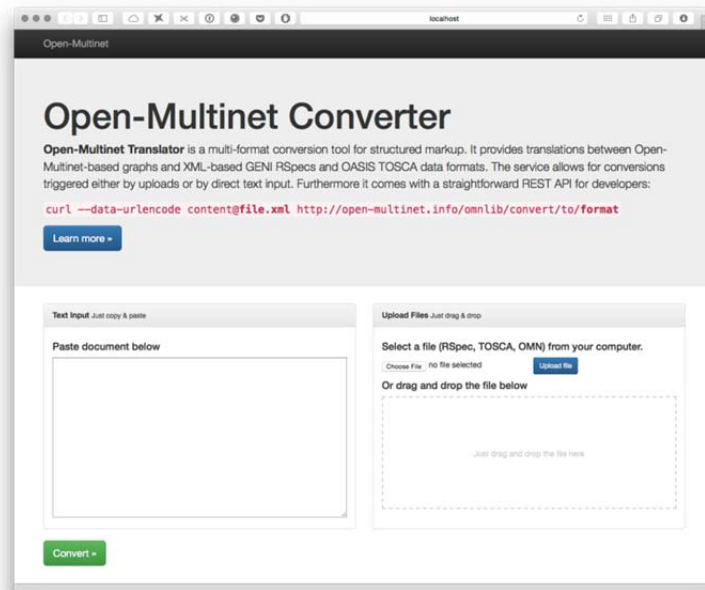


Figure 8 Translator Service

3.1.4.2 Ontology-based Aggregate Manager

In order to evaluate the semantic extensions developed within Task 5.2, an ontology-based GENI AM is being developed as a reference implementation (called FITeagle) that is using the facilitating translation service described above.

As shown in Figure 9 the jFed probe GUI have been used to invoke a semantic query of resources using a newly introduced parameter "geni_query" and the resulting resource description is encoded using RDF/XML. It is planned to further extend this implementation to support the whole SFA life-cycle including information about monitoring capabilities. This will be demonstrated using the jFed user tools.

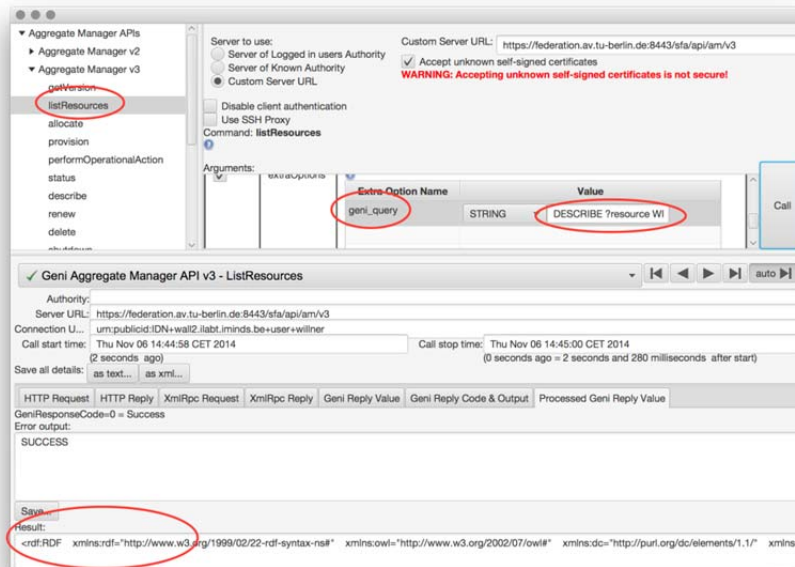


Figure 9 Ontology-based AM calls

3.1.5 Evaluation

The applicability of the proposed ontology has to be validated within the FIRE context. More specifically, requirements from the Fed4FIRE project have been incorporated and, as a result, a number of mappings between GENI RSpecs and the semantic model will be presented in [20].

Besides the functional principle of the converter, its performance is of further interest. The input of the following performance evaluation is based on the RSpec Advertisement published by the Virtual Wall testbed, whose XML serialization is about 2.4 MB in size. In total 212 nodes, including their 619 sliver and 1297 hardware types, were translated.

The evaluation is divided into two parts. The first part includes the conversion of the XML document into a JAXB OM. The measurements were repeated 100 times with 1 second breaks in between and 10 repetitions were executed before filtering out possible start up, initialization and compilation outliers. As shown in Figure 10, this conversion takes, with a 95% confidence interval, 5263 ms +/- 15 ms. As a result, this should further be examined and, if possible, optimized.

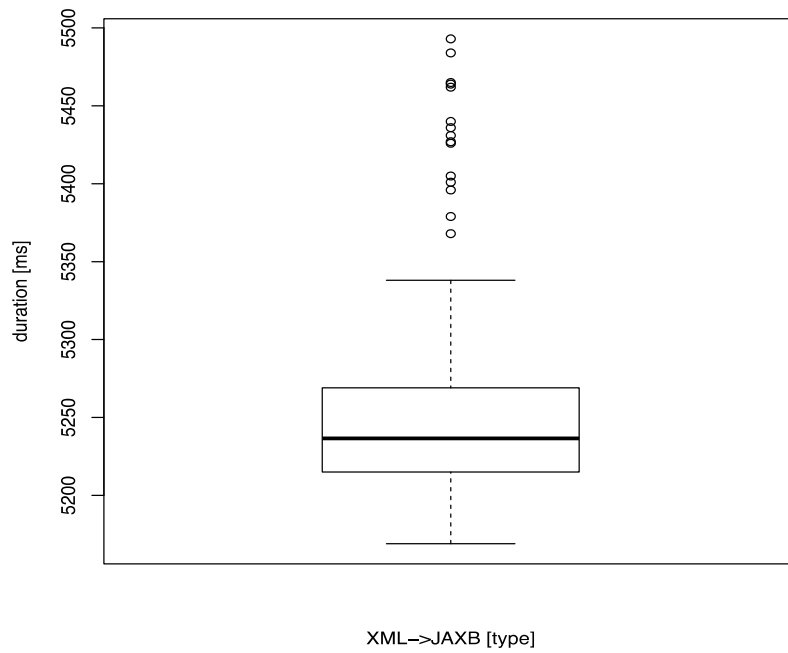


Figure 10 Cost of the XML deserialization

The second part includes the conversion process between the JAXB OM, the RDF graph and the serialization back to XML. The measurements were repeated 1000 times with 100 ms breaks in between and 10 warm-up repetitions. As shown in Figure 11, the most expensive operation is the XML serialization, and the mapping between the OM and the RDF tree takes about 6 ms.

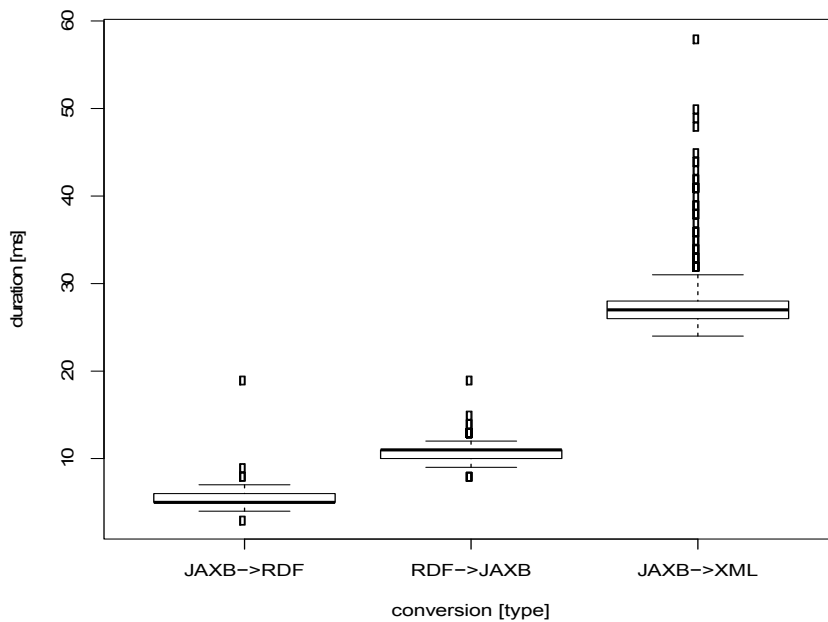


Figure 11 Translation between the JAXB OM, the RDF graph and the XML serialization

3.1.6 Summary and Outlook for Information Modeling in Cycle 3

We have given an overview of the important issue of describing resources within federated infrastructures. Motivated by the concrete field of application of experimental Future Internet research, we have further shortly presented related work on this topic.

The crucial results are twofold. First, we identified that mechanisms developed within the Semantic Web present promising means to address this issue. Second, based on, and integrated with, existing work in the field, we have developed and demonstrated the Open-Multinet Upper Ontology.

The presented work provides potential advantages for infrastructure owners, federation operators, developers, and users. While owners and operators can offer highly heterogeneous resources by specializing existing concepts, developers and users have the possibility to conduct complex queries to discover them. Within a federation, it is possible to enhance this process even further by relating offered descriptions with each other by expressing e.g. equality of resources. Tool developers can re-use existing work available within the Semantic Web to easily explain errors to users, allow handovers between protocols (between SFA and FRCP) or to implement complex resource matching or path finding algorithms without involving functional code.

Our short-term goal is to include support for our ontology in SFA AMs like FITeagle and SFA user tools such as jFed. In the medium term, we want to broaden our approach to include further tools for resource scheduling, experiment control and monitoring and further enhance the ontology to describe more resources within the Fed4FIRE federation. The long-term goals include utilizing our approach in further fields of application, such as federated cloud environments. Towards this goal, we have already extended the translation tool to support the Organization for the Advancement of Structured Information Standards (OASIS), specified by the Topology and Orchestration Specification for Cloud Applications (TOSCA).

This research goes beyond the current approach using RSpecs for resource description.

3.2 Central Reservation Broker Specification (Task 5.3)

3.2.1 Resource Reservation Overview

The Reservation Broker is the overarching service that experimenters will utilize to reserve heterogeneous resources spanning multiple testbeds in the federation based on a multiplicity of selection criteria (time, resource type etc.). Experimenters can significantly benefit from the brokerage service as they will be able to simplify the overall process of identifying and reserving suitable resources for their experiments. The latter is even more important in cases that they need to run (large scale) experiments where resources from multiple testbeds are required to fulfill their needs. An extended description of the brokerage service and its benefits is provided in detail in the predecessors of this document [1] and [2].

Deliverable 5.1 provides taxonomy of the reservation types and their correspondence to testbeds within the Fed4FIRE federation, as well as the high level required functionality of the Reservation Brokering service. Based on the comparison and evaluation of existing tools available to the Fed4FIRE community (NITOS/NICTA Broker, GRID5000 Scheduler, NETMODE Scheduler), the adoption of the NITOS/NICTA Broker was decided, as the most appropriate system in terms of the required Fed4FIRE functionalities. Finally, a detailed description of the NITOS/NICTA Broker's architecture and implementation was provided.

In Deliverable 5.2 a short discussion on the functionality of the Reservation Broker to be supported by the end of Cycle 2 was provided. Adapting the initial NITOS/NICTA Broker implementation, we provide the functional specification of the Reservation Broker, including the broker's architecture, its interactions with other components in the F4F environment, complementary modules to the Reservation Broker like the Reservation plugin in MySlice and considerations and requirements regarding the description of resources with respect to the particular service and the status of implementation.

In this deliverable (i.e. Deliverable 5.4) we provide an update on the functional specifications of the Reservation Broker for Cycle 3, along with complementary modules such as the Reservation plugin. Moreover, based on a questionnaire that was distributed among Fed4FIRE testbeds, we identify and classify reservable resources available in the federation along with the tools or information that is required for supporting the Reservation Broker functionality.

3.2.2 Reservation Broker in Cycle 2

During the second cycle of specification and development, the Reservation Broker was introduced and its role as a *Central Reservation Broker* at the level of federation services was defined with more detail in the predecessor of this document [1]. To this end, in Deliverable 5.2 an initial deployment and functionality of the Reservation Broker was defined and its interactions with other federation services like the *Fed4FIRE Portal*, *Manifold SFA Gateway* and the *Manifold Data Broker* were specified.

In order to attract non-technical users, or users not familiar with the specifics of Fed4FIRE testbeds, Fed4FIRE federation services enable expressing requests for federated slices in a more abstract form (e.g., not specifying the actual substrate resources to be allocated). Hence requests may contain a complete (bound request), partial, or empty (unbound request) mapping between the resources an experimenter might desire and the physical resources available on the Fed4FIRE federation (e.g., I want two communicating 802.11b/g nodes, on June 15th 2015 15:00-18:00 pm).

It is the task of the *Central Reservation Broker* to map such requests to actual physical resources on testbeds that belong to the federation, based on their availability. As described in Deliverable 5.2 the Broker (mapping sub-module) is "*entrusted with the responsibility of (i) splitting efficiently the request among underlying infrastructures and (ii) mapping the corresponding partial unbound requests to the appropriate substrate resources from selected testbeds in the Fed4FIRE federation*"¹. Integration with the *Manifold SFA Gateway* and the *Data Broker* allows the *Reservation Broker* to fetch the latest catalogue of resources for all testbeds in the federation, along with their respective monitoring data, thus enabling its capability to map unbound requests. The interactions between the basic components that facilitate reservation of resources in Cycle 2 can be seen in Figure 12 while they have been described in detail in Deliverable 5.2.

¹ Mapping could also be supported by each testbed.

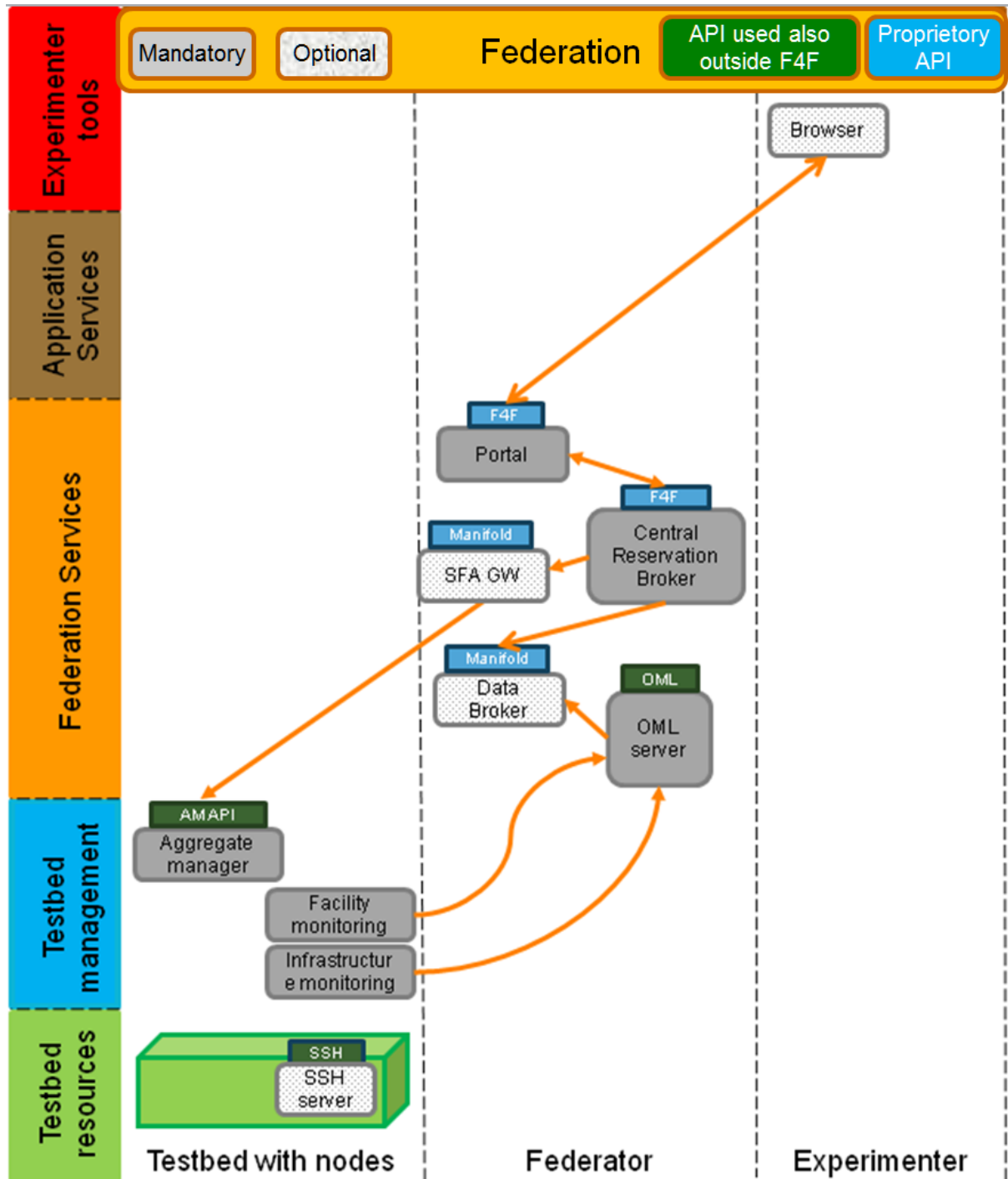


Figure 12 Reservation Broker in the Fed4FIRE Environment (Cycle 2)

3.2.3 Reservation Broker in Cycle 3

Moving on to the 3rd cycle of detailed specifications, several enhancements will be made to the Reservation Broker. The enhancements will be towards:

1. *Enabling a wider set of abstraction in the unbound requests.* So far, the basic level of abstraction was relied on number of required resources, type of resources and

duration/timing these resources should be allocated. The enhancements at the level of abstraction will require the corresponding extensions to the mapping submodule's API, which should support queries containing new characteristics and also to investigate possible implications on the request partitioning/mapping algorithms implemented.

2. *Providing the Reservation Broker as a third party service.* The Reservation Broker can be provided as a service to other tools of the federation, including the various SFA clients. For that purpose appropriate documentation will be provided.
3. *Enabling reservation of resources from a larger set of testbeds.* A crucial factor for rendering the Reservation Broker into a useful federation service to the experimenters, is its capability of supporting many and different types of testbeds. In Cycle 2, most of the efforts were focused on enabling the Reservation Broker for the wireless testbeds (NITOS and NETMODE) and a set of two wired testbeds (PlanetLab and VirtualWall). During Cycle 3 the set of supported testbeds will be expanded to additional heterogeneous testbeds. The inclusion of new testbeds will necessitate the modification of the Reservation Broker's information model and the mapping submodule's algorithms. These should be adapted to the various peculiarities of the new testbeds, so that the Reservation Broker could support the new testbeds in serving unbound requests.
4. *Integration with the Data Broker.* The integration of the Reservation Broker with the Data Broker will be improved in terms of leveraging extra monitoring information, provided by the new testbeds. The deployment of the Data Broker was held in parallel with that of the Reservation Broker in cycle 2, so a basic set of monitoring information has been used so far, including mainly resource availability data. In cycle 3, further monitoring information will be used related to resource utilization, as more and more testbeds expose that kind of information to the federation level services.
5. *Investigate the allocation of inter-domain links.* Currently Layer 2 connectivity is supported among a subset of Fed4FRE testbeds. Within the course of cycle 3 we will investigate the allocation of L2 or L3 links to experimenters, spanning different testbeds, by mean of existing tunneling techniques (e.g., GRE etc.).
6. *Investigate the use of Semantic Technologies for Resource Mapping.* Exploiting semantic annotations for testbed infrastructure allows precise control of the mapping process between requested and available resources. Throughout Cycle 2, we proceeded, in parallel with Task 5.2, to the investigation of semantic aware resource mapping for wireless testbeds. The results of the effort were presented in the 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD) [10]. In Cycle 3 we will further look into the matter, extending the proposed approach to the federated environment.

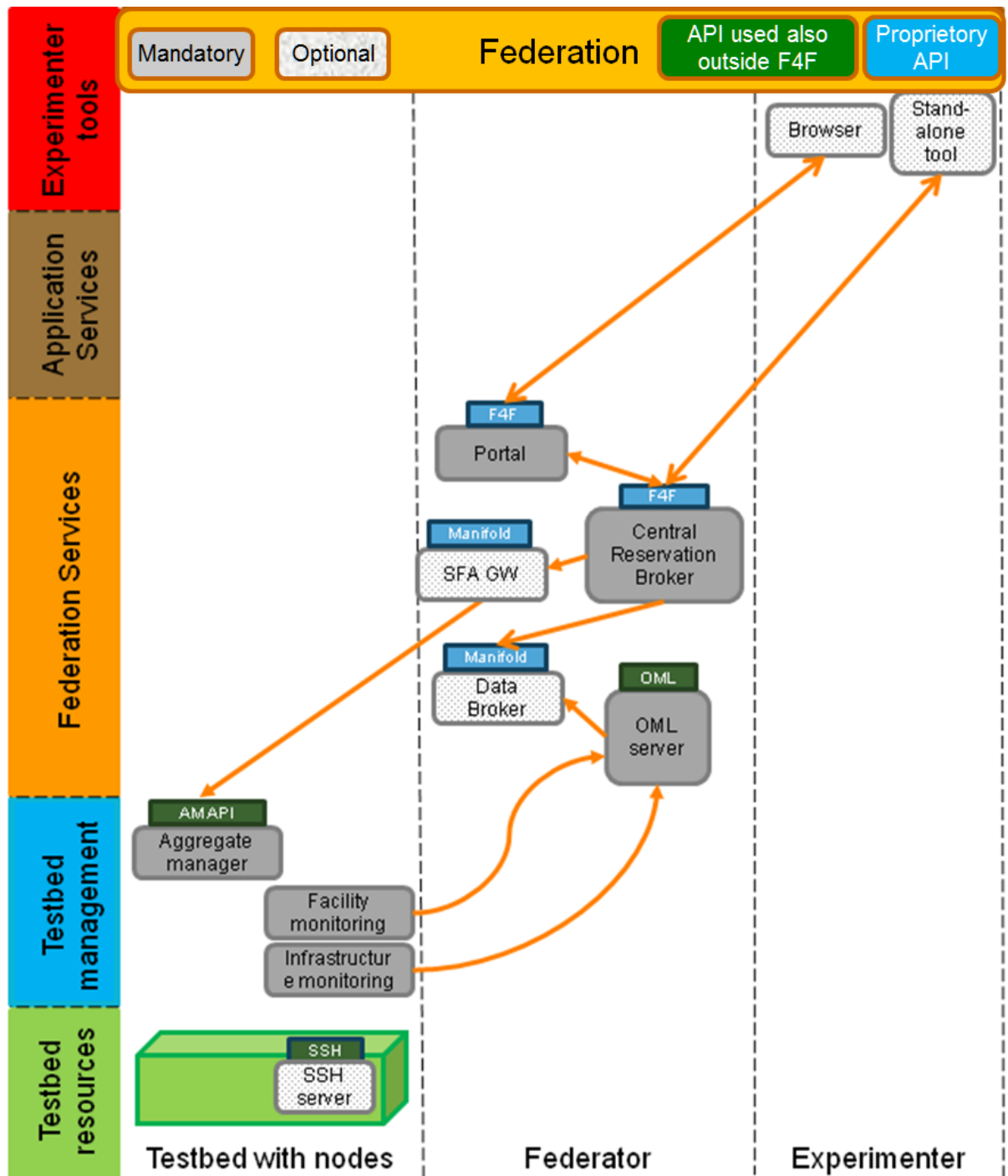


Figure 13 Reservation Broker in the Fed4FIRE Environment (Cycle 3)

3.2.4 Reservation Plugin

In order to provide experimenters with the capability of submitting unbound requests for resources, the "Scheduler" **Error! Reference source not found.** or "Reservation Plugin", integrated with the Fed4FIRE portal, has been extended so that an experimenter is able to express and submit unbound requests for abstract topologies on desired date/time (for exclusively reservable resources) such as "I want two VMs" (instant reservation) or "I want three 802.11b/g nodes from any testbed, on June 15th 2015 from 15:00 to 19:00 pm" (advance reservation). A screenshot of the Reservation plugin, regarding a requested wireless full mesh topology (hence no links provided) from any testbed, from

March 2nd 2015 20:26 pm to March 3rd 2015 20:26 pm, is depicted in the following Figure 14. The request has been successfully submitted to the *Reservation Broker* and the response (mapping solution) is visible to the experimenter on mouse over (e.g., Node-1 has been mapped to omf.nitos.node005). The corresponding bound request is sent immediately to the *Manifold SFA gateway* for actual provisioning.

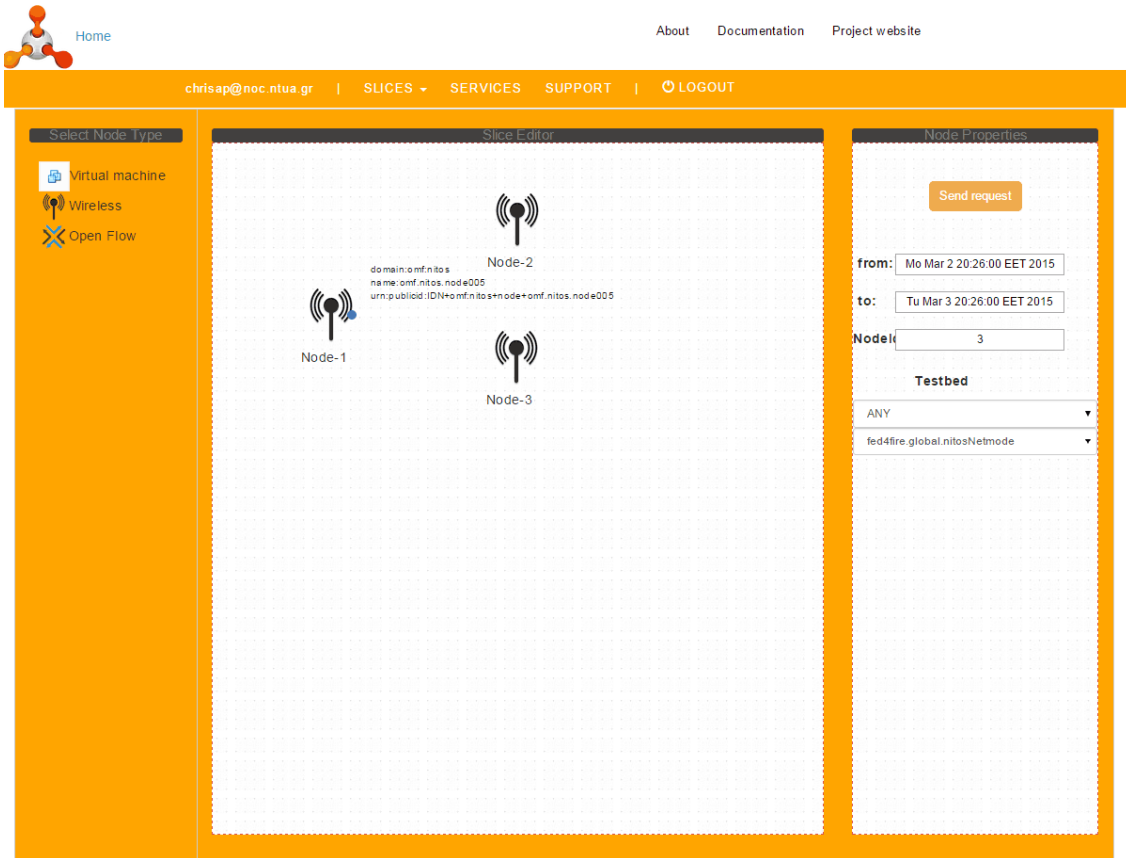


Figure 14 Reservation Plugin: Unbound Request for Wireless Resources

In the same manner, a screenshot of the Reservation plugin, regarding a requested wired topology from any testbed, from March 2nd 2015 20:26 pm to March 3rd 2015 20:26 pm, is depicted in Figure 15.

With regards to Cycle 3, the Reservation Plugin will be updated to support the following features:

1. *Two step procedure for Resource Mapping and Reservation.* The goal is to allow the experimenter (i) inspect the mapping solution suggested by the Reservation Broker and (ii) configure the nodes, before he/she proceeds to actual reservation.
2. *Fine tune the request design.* The Reservation Plugin will allow the user to configure the requested resources (e.g., IP addresses, disk image, boot disk etc.). Resource configuration follows the mapping process, since configuration options are depending on the testbed.
3. *Support large scale slices.* The Reservation Plugin will allow the user to set up fast large scale requests easily, without the need to drag and drop individual resources onto the panel (e.g., I want to reserve 100 virtual machines, irrespectively of their location or testbed, for two weeks).

4. *Support pre-defined topologies.* In the process of creating large scale slices, the user should be able to select connectivity level (graph density) or specific types of topologies (e.g., tree, star, etc.).
5. *Support RSpec import/export.* The user should be able to export the RSpec for a specific slice request and/or load it for repopulating the slice.

Figure 15 Reservation Plugin: Unbound Request for a Virtual Topology

3.2.5 Reservable Resources

To facilitate mapping an experimenter's request to resources available in the federated testbeds environment, additional information is needed from each testbed provider regarding the reservable resources. To classify the type of the resources available a questionnaire was distributed to gather input from Fed4FIRE testbeds. From the collected information, we extracted the type of resources along with the functional and non-functional characteristics that they can provide. In the following, the survey that was given for completion is presented.

The purpose of this survey is to gather information from every testbed for the mapping sub-module of the Reservation Broker. The mapping sub-module is responsible for resource mapping within the

context of F4F; that is:

- (i). Splitting efficiently unbound² user requests among underlying testbeds based on e.g. availability of resources, fairness in the use of testbeds etc.
- (ii). Mapping the corresponding partial unbound requests to the appropriate substrate resources from selected testbeds in the Fed4FIRE federation.

In order to perform mapping/scheduling of the unbound requests a clear description of both virtual and physical resources should be provided. With regards to the physical ones, the description should be provided by the testbed in terms of:

- functional attributes that define characteristics and properties of the testbed resources including static parameters like node type (e.g. router, switch, server), node processor type and capacity, link/path type (e.g. VLAN, L3/L2 VPN), network interface type and number, geographic location etc.
- non-functional attributes that specify criteria and constraints related to the testbed resources including dynamic (real-time) parameters such as resource availability, utilization etc.

Physical Resources Description: Please indicate the type of testbed infrastructure e.g. (i) physical nodes: server, router, switch, access points, base station, sensor nodes, mimo, etc., (ii) physical links (e.g. 802.11x, Gigabit Ethernet circuits etc.)

Reservable Resources; Please identify the resources available to the experimenter, including e.g. virtualization technology that is used in the testbed (e.g. OpenVZ, Xen, VMware etc.), disk images, OSs are provided, link types etc.

Reservation Duration: Please indicate the reservation model supported for your testbed resources (instant reservations with specific duration, instant reservations with infinite duration, advance reservations, elastic reservations). Please state if it requires the intervention of the testbed administrator.

IP address: Please specify the version of IP that is supported in your testbed (ipv4/ipv6)?

Monitoring data: Please indicate monitoring data regarding the testbed infrastructure (e.g., availability, server load, CPU/ memory usage, link utilization, disk space usage, interface throughput, topology information - connectivity, SNR, PDR etc.) that is currently measured and can be provided as an OML stream.

Reservation information: Please indicate reservation information maintained at the testbed infrastructure (e.g. resource leases).

The wireless testbeds NETMODE, NITOS, w-iLab.t and FUSECO, the wired testbed Virtual Wall and PlanetLab, the OpenFlow testbed OFELIA (i2cat and univbrs), the sensor testbed SmartSantander³, the optical access research testbed UltraAccess⁴ and the cloud testbeds BONFIRE and FUSECO have responded to the given questionnaire and based on their replies the following table was created, which illustrates the tools or information that can be used by the broker to facilitate the request partitioning/mapping process, as well as the type of resources supported by each testbed (shared

² The request that provides no mapping information for requested resources.

³ Due to the nature of SmartSantander testbed that provides only Service layer resources; that is read only access to observations via user subscription, the questionnaire results are not included in following table.

⁴ The questionnaire results of UltraAccess are not included in the table, as the testbed at the time did not match the basic categories identified in the paper.

resources - marked as green, exclusive resources -marked as orange) and the type of reservation (advance or instant).

		Resources	F4F portability	SFA/RSpec	Monitoring OML stream (Infrastructure Monitoring)
Wireless Testbeds	NETMODE	802.11x nodes (advance-finite duration)	x	GENI AM v2 / RSpec v3 (incl. leases) ⁵	node availability (up/down)
	NITOS	802.11x nodes(advance-finite duration)	x	GENI AM v2 / RSpec v3 (incl. leases))	node availability (up/down)
	w-iLab.t	802.11x nodes (advance-finite duration)	x	GENI AM v2 and v3 / RSpec v3	- ⁶
	FUSECO ⁷	<ul style="list-style-type: none"> 802.11x nodes Femtocell BTS/ LTE AP/ nano3G Radio Signal Shield Box/Att. System FUSECO Clients (e.g., Samsung Galaxy S4, Lenovo M93 Tiny) (instant-finite duration) 	x	GENI API v3/ RSpec v3	node availability (up/down)
Wired Testbeds	PLE	Linux-VServer and LXC based virtual machines (instant-4 weeks duration)	x	GENI API v3 / RSpec v3	node availability (up/down)
		Raw PCs (advance-finite duration)			
	Virtual Wall	openVZ containers or XEN virtualization (Emulab) (instant - finite duration)	x	GENI AM v2 and v3/ RSpec v3	current load of backbone experiment switches ⁸
		Raw PCs (instant - finite duration)			
OpenFlow Testbeds	l2cat	OpenFlow enabled topologies (OF-enabled switches/links) Virtual Machines (XEN) (instant - infinite duration)	x (ong.)	GENI AM v3 / RSpec v3 (incl. omf extensions) ⁹	-
	Univbris	OpenFlow enabled topologies (OF-enabled switches/links) Virtual Machines (XEN) (instant - infinite duration)	x (ong.)	GENI AM v3 / RSpec v3 (incl. omf extensions)	-
Cloud Testbeds	BonFIRE	Pre-defined or custom size VMs (OpenNebula) (instant/advance - finite duration)	-	-	Number of running VMs in each server; Total CPU ; Available CPU (not allocated by VMs); Total memory; Free memory; Host availability (up/down)
		Raw PCs (advance - finite duration)			

⁵ <http://nitlab.inf.uth.gr/schema/sfa/rspec/1/ad-reservation.xsd>

⁶ Resource Availability included in RSpec

⁷ For wireless experiments a FUSECO client is needed. FUSECO Playground provides access to physical nodes (Samsung Galaxy S4 or Lenovo M93 Tiny) or instances that running as VM inside the Lenovo M93 tiny. The radio interfaces are available through USB pass through.

⁸ Resource Availability included in RSpec

⁹ <http://www.geni.net/resources/rspec/ext/openflow/3/of-ad.xs>

	FUSECO	Kernel-based Virtual Machine (OpenStack /KVM)	x(ong .)	GENI API v3	node availability (up/down)
--	--------	--	-------------	-------------	-----------------------------

3.3 Resource Provisioning (Task 5.4)

3.3.1 Resource provisioning overview

As commented in D5.1 [1], the solution for resource provisioning is not presented as a tool that provides that service, SFA AM API is a solution that each testbed had to adopt in order to achieve the Fed4FIRE requirements. There were several options presented in that document (SFAWrap, AMsoil, etc.) and much more available to be chosen by each testbed to implement an SFA AM API.

The SFA AM API, described in the appendix of D5.1 [1], is the main tool to provide to experimenters with a common format (interfaces and data types) that allows them to inter operate with the entire federated multiple administrative domains (testbeds) to request for resources. This SFA AM API also allows the testbeds to expose and describe their available resources, which is used in task T5.2.

Above the SFA AM API and the testbeds, there are tools that provide the provisioning service, among others, to the experimenters. These tools cover the experiment life cycle workflow (resource discovery, description, reservation, etc.) and use the SFA AM API to interact with the testbeds.

At the moment of writing this document, almost in the end of cycle 2, all the federated testbeds have implemented or are finishing the implementation of one of these solutions to be able to provision the requested resources. This fulfils the main objective of task T5.4 as expressed in the DoW.

As explained in D5.1 [1], D5.2 [19] and D5.3 [2], there are two tools to perform the different steps of the experiment management lifecycle: the MySlice Portal and jFed. Both tools use the SFA AM API (as of different implementations) to interact with the testbeds and allow the provisioning of the resources exposed. Apart from that, the MySlice Portal, also covers the presentation and acceptance of the SLAs related to the requested resources. The acceptance of the SLA is mandatory to provision the resources.

3.3.2 Provisioning in Cycle 3

Now that the main objective of Task 5.4 has been covered by providing the testbeds the capacity of provide the specific resources requested by the experimenters, the effort in cycle 3 will be focused in providing an alternative way of resource provisioning, by freeing the experimenter of the duty of defining which specific resources from which testbed he wants to request. The aim is that the experimenter simply defines which type of resources he needs and a brokering component decides (orchestrates) to which testbed each particular resource request will be sent to. In this sense, some implemented tools are providing a basic version of this service, while others are studying how to do it. In the following sections these tools are presented.

The main advantage of the orchestrated resource provisioning is that eases the process for the experimenter to select the resources he needs. This can be useful for new experimenters who are not so familiar with the resources details or for low requirement experiments where any of the resources of the server will fit.

On the other hand, the main disadvantage is the loose of decision power of the experimenter. That is, if experimenter can only choose the type of resources he needs, he cannot define details on those resources, as each testbed will provide different configuration options for their resources, even if the resources are of the same type than other testbed. For example, a testbed may offer for its virtual machines the possibility of defining the OS of the image to be loaded, the HD capacity and the amount of RAM in the VM while other just allows defining the HD and RAM or a completely different parameter. For this, a default configuration must exist in each testbed that will be used when no more details are provided.

A possible way to avoid this disadvantage is a trade-off solution for resource orchestration in which the experimenter defines the specific requirements of the resources needed but the location of them and then the orchestrating component searches for the resources that fulfil these requirements among all the testbeds available. The main inconvenient of this solution is that removes the advantage of quick resource requests.

3.3.2.1 Provisioning through Reservation Broker and MySlice

Provisioning of resources in the MySlice portal is done through the Reservation Broker explained in the previous section. Regarding the orchestrated provisioning, some enhancements are being made and expected to continue during cycle 3.

An abstraction request for resources can be made and send to the reservation broker which will be able to map the request to actual specific resources and return the detailed request so the experimenter can make the specific reservation and provisioning. This is done through a REST API that includes a method for accepting abstract requests and returning the mapped set of resources. This mapping submodule can also work as an independent tool to allow 3rd party developers to implement their own mapping algorithms to link abstract requests with specific resources.

Apart from the orchestrated provisioning being developed, MySlice portal also offers the presentation and acceptance of SLAs for the requested resources before they are provisioned. This was presented in D5.2 [6]. For cycle 3 it is envisaged to provide an independent tool for testbed managers to define the SLAs of their resources to be presented and accepted before they are provisioned. The following section provides detailed information about the SLA management module.

3.3.2.1.1 SLA management

During cycle 2, SLAs were only accessible through the Fed4FIRE portal. However, considering the variety of tools that can be used within the federation, this limited experimenters and testbeds to use this tool to accept and offer guarantees on resources respectively. Therefore, changes in the SLA creation process and the parameters sent between SLA components will be implemented so that experimenters can use any federation tool to access SLAs. More details on the changes related to SLA management will be found in the upcoming WP7 deliverable D7.4 [11]– Detailed specifications regarding trustworthiness for the third cycle.

The testbeds offering SLAs are being stored in a directory located in the SLA Collector module and can be easily obtained from a single call to the SLA Collector API. The modification of the data returned with by SFA getVersion call to include a field indicating if the testbed supports SLA is still being considered due to the reduced number of testbeds offering this resource guarantee.

The SLA plugin for the portal will also be improved and extended to integrate the portal functionality upgrades as well as the new SLA management approach to make them accessible from any client tool in the federation. It will provide the same functionalities of the previous version (i.e. presentation of testbed's SLAs, SLA acceptance dialog for experimenters and visualization of SLA evaluations) but with a more clear and appealing GUI.

3.3.2.1.1.1 Showing SLA information of testbeds supporting SLA through the portal

Experimenters will be able to identify which testbeds support SLA using a field in the table presented in the resource reservation view of the portal. This will allow filtering testbeds by the kind of resource guarantee they offer, considering either “best effort” or “SLA”.

Any other client tool of the federation that experimenters can use to reserve resources on testbeds will be able to show that information by retrieving the list of testbeds that support SLA from the SLA Collector module. The presentation of that information will be dependent on the kind of tool selected.

3.3.2.1.1.2 SLA acceptance and evaluation result visualization through the portal

The main changes in the SLA plugin will be focused in its core functionality and therefore will be transparent to the experimenter using the portal, who will be able to accept and visualize both the accepted SLAs and the result of their evaluation in the same way as before.

Once the experimenter selects the resources to be reserved in a testbed, an SLA acceptance dialog will be prompted describing the SLA details. The experimenter will then decide whether accepting or declining the offered SLA. In case the SLA is declined, resources will either be reserved under a best effort service or not allowed to be reserved depending on the policy defined by the testbed. If the SLA is accepted, then it is created using the sliver identifiers of the reserved sources together with the experimenter identifier and stored in the SLA Management module database located in the testbed.

The experimenter will be able to see the accepted SLAs and the result of their evaluation once the corresponding experiment has finished, in the same way it was defined for cycle 2.

3.3.2.2 SLA-Dashboard

SLA-Dashboard is a tool for testbeds in the federation to manage their own SLAs.

Each testbed provider (SLA-Administrator in the tool context) will be able to create its SLA templates which contain the description of the offered guarantees. Then, when reserving resources from a testbed that supports SLAs, the experimenter will be able to select among the offered templates the one that best matches his needs. The testbed provider will be able to see the agreements that have been created with his templates and, after they have been executed, if they're fulfilled or not.

The SLA-Dashboard is a tool that will help the testbed provider to create these templates, to check the agreements that have been created using his templates and to monitor the result of the agreement execution.

3.3.2.2.1 Template-Agreement lifecycle

The provider must generate the template according to his resources in the testbed (or in each testbed, in case there are several). He must decide the offering: metrics that will be provided, which guarantee terms and conditions to fulfil and expiration date. Once decided, he will give a name to this template and create the whole template definition with the SLA-Dashboard tool. The SLA-Administrator can make use of the name to create the template versioning.

The tool doesn't make any template-version management. If the SLA-Administrator wants to create another version from a previous template he will need to introduce it into the tool from scratch.

Once a template has been created, an Experimenter will be prompted through the client tool with the SLA offering, and will have to agree with the terms specified in the agreement, which will create the SLA.

Once the experiment has been finished, both the experimenter (through the client tool) and the testbed provider (from the SLA-Dashboard) will be able to check the result of the SLA evaluations they have agreed.

When violations occur, the SLA Management Module will invoke the Reservation System in order to notify the penalty of the violation or quota change.

3.3.2.2.2 SLA-Dashboard tool description

The SLA-Dashboard will be a web-based application. A simple login page with username and password will be used to authenticate the SLA-Administrator against a local database where the relationship of testbeds and SLA-Administrator usernames will be stored.

3.3.2.2.2.1 Template creation and listing

Each testbed provider should be able to retrieve the list of templates he has created and will be able to create new templates. The following Figure 16: Mock-up templates list and template creation shows a mock-up of the screen that will cover these functionalities.

The mock-up shows a web browser window titled "A Web Page" with a URL bar containing "http://". The page content is titled "Fed4Fire SLA Dashboard". Below the title, there is a navigation bar with "SLA-Administrator" and a "Testbed" dropdown menu. To the right of the navigation bar are three links: "Manage my templates", "See my agreements", and "Logout".

The main content area is titled "Templates" and contains a "Create new" button. Below this is a table listing existing templates:

Template Name	Template Id	Expiration Time	Sliver
Slice1Template	f8b1e038-ff39-47dae023a433f2e	01/01/2015	urn:publicid:IDN+ch.geni.net:CHtest+slice+st5
Slice2Template	a23da34-aa52-f65345645656f4	10/12/2015	urn:publicid:IDN+ch.geni.net:CHtest2+slice+st7
Slice3Template	73bc342-de23-23564e34d26a9	20/08/2020	urn:publicid:IDN+ch.geni.net:CHtest3+slice.st9

Below the table is a "New template" form. It includes fields for "Template Name", "Sliver", and "Expiration date" (with a calendar icon). There is a button "Add property + guarantee term". Below this is a section for "Service Property + guarantee term" with a text input field containing "<gt name>". To the right of this input is a "Constrain" dropdown menu with options: "GT", "LT", "EQ", "LE", "GE", "NE", and "BETW". Further right is a "Value" input field containing "<value>" and a "Create" button.

Figure 16: Mock-up templates list and template creation

A list of the defined SLA templates will be presented to the SLA-Administrator, who can click on each template identifier to see its details.

To create a new template, the SLA-Administrator will need to introduce the template name and the expiration date. He will be able to add as many service properties and guarantee terms as required.

For each guarantee term the SLA-Administrator has to indicate the name, the type of constraint (LT – less than, LE – less equal, GE – greater equal, GT – greater than, EQ – equal, NE – not equal, BETWEEN) and the value (or values in case of BETWEEN) that the guarantee term has to fulfil the described equation.

These will be the conditions an Experimenter will have to accept to generate an agreement.

3.3.2.2.2 *Template detail*

Once a template has been added to the SLA Management Module, the SLA-Administrator will be able to review the details and the assigned Id anytime. This screen won't allow editing the template.

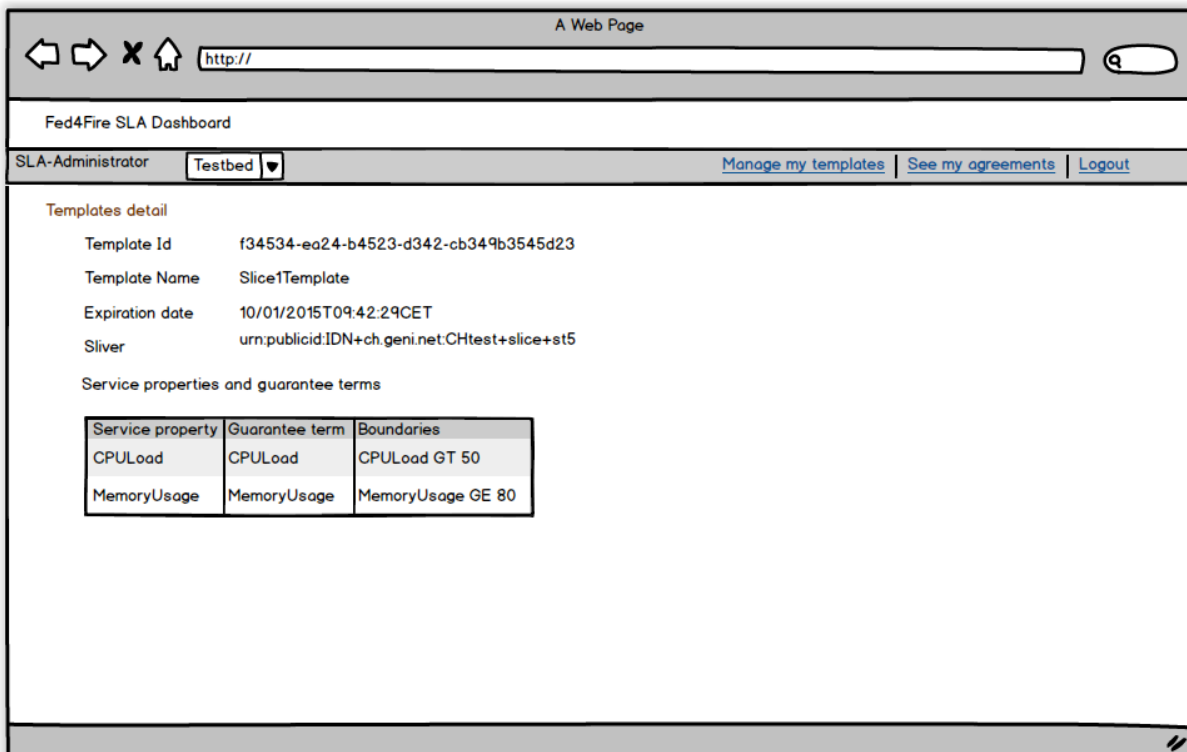


Figure 17: Mockup template detail

3.3.2.2.3 *Agreement listing and details*

The SLA-Dashboard should also be a tool to check the status of the different agreements that have been created and their status (Non-determined – has not been started yet, fulfilled – succeeded evaluation, violated – failed evaluation). The SLA-Administrator will be also able to see which Experimenter has created the agreement.

This SLA-Dashboard is only to be used by the testbed providers. Experimenters will not have access to this tool. They will have access to SLA information through the portal or the different client tools available in the federation (e.g. jFed could be one of these).

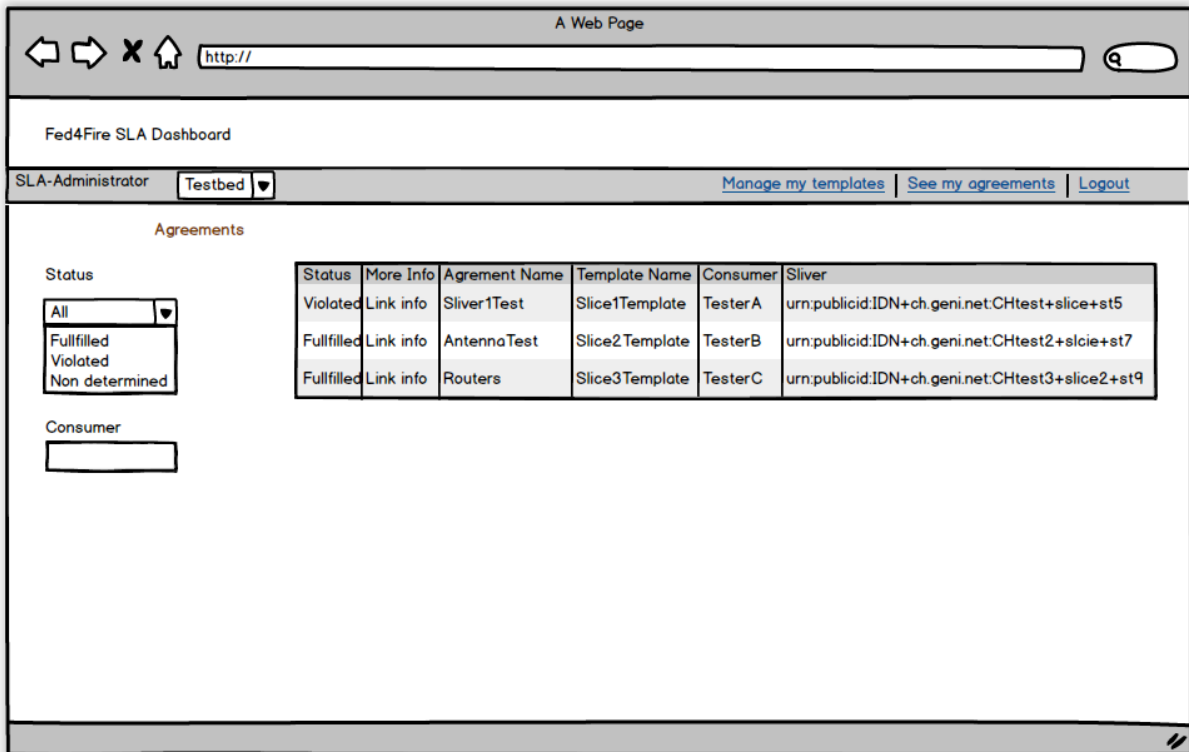


Figure 18: Mockup agreement list per provider

The SLA-Administrator will have a link to retrieve more information about the agreement. As we commented previously, an Experimenter won't be able to modify the SLA, it has to be accepted the way the testbed has defined it.

The expiration date should never be later than the expiration date of the template.

The SLA-Administrator, for each testbed, should have a table per guarantee term and should be able to check their evaluation results. This is depicted in Figure 19 with the table below the title of Guarantee terms.

In case a guarantee term is not being fulfilled, the SLA-Administrator should be able to review the monitoring data that has been retrieved. He should be able to check why and when the violation happened. Figure 20 shows how this can be made.



Figure 19: Mock-up detail of an agreement

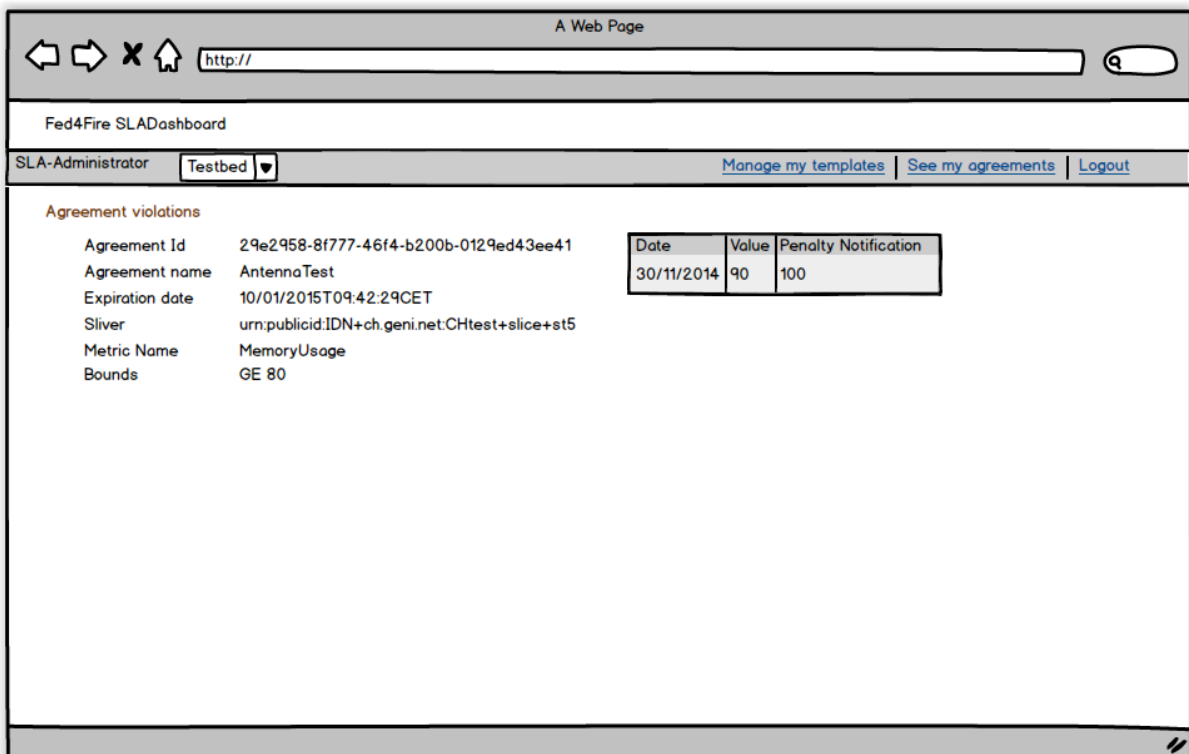


Figure 20: Mock-up detail of agreement violations

3.3.2.3 Provisioning through jFed

Provisioning of resources is already provided by the jFed standalone tool. At the moment of writing this document, jFed is able to provision resources of most of the Fed4FIRE federated testbeds plus non-Fed4FIRE testbeds in a direct way, that is, the experimenter defines the resources for the individual testbeds. However, through the categorization per resource type, the experimenter can also use these icons without choosing a particular testbed (the mapping to testbeds is then done by jFed).

The following table summarizes the resources availability by type:

Table 1: Resources provisionable by jFed

Resource type	Fed4FIRE testbeds	External testbeds
Generic node	BonFIRE PlanetLab iMinds VirtualWall 1 & 2 iMinds WiLab 2	
Physical node	iMinds VirtualWall 1 & 2	Utah Emulab
Virtual Machine	BonFIRE PlanetLab Europe FuSeCo Fokus I2cat VTAM Bristol VTAM	ExoGENI InstaGENI
XEN Virtual Machine	iMinds VirtualWall 1 & 2	InstaGENI
OpenVZ Virtual Machine	iMinds VirtualWall 1 & 2	Utah Emulab InstaGENI
Wireless node	iMinds WiLab 2 Clab Nitos Netmode	
Dedicated Network connection (network edges)	i2CAT OFELIA VLAN Bristol Ofelia VLAN NetMode NITOS Bonfire (EPCC) PSNC	(done through automatic stitching – transparent for the user)

In a screenshot it looks like this:

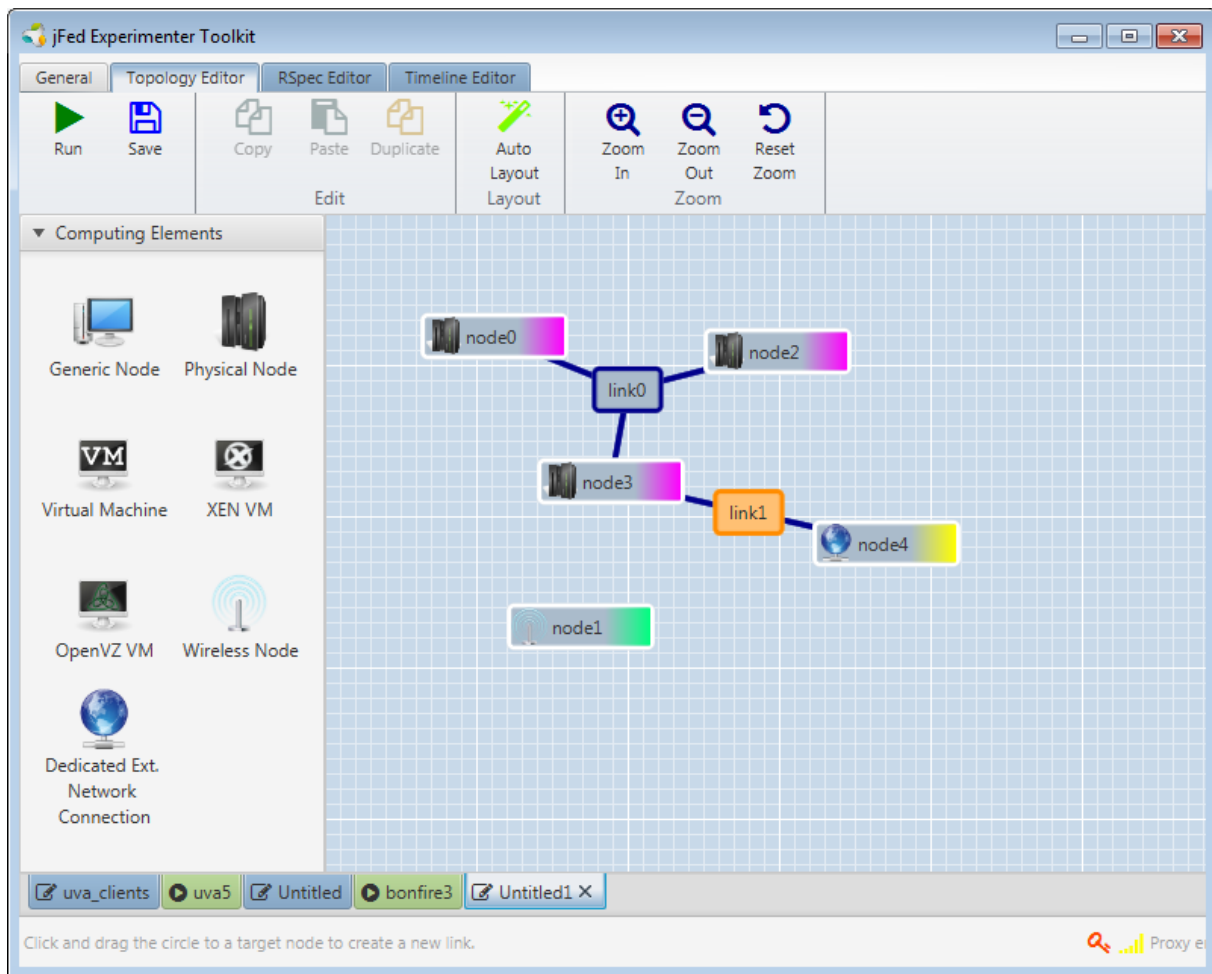


Figure 21: jFed resource provisioning

Dedicated network connections, or network edges, are fictional resources that allow defining connectivity parameters between some of the resources, in case they do not support the fully automatic layer 2 stitching setup

During cycle 3, more resources of the testbeds will be made available for jFed when e.g. the new testbeds do pass the compliancy tests.

Documentation and release notes about jFed can be found in the following places:

- <http://jfed.iminds.be>
- A quick video introduction: http://jfed.iminds.be/jfed_video.html
- In Annex 3 an overview of all features through an experiment walk-through is included

In order to provide provisioning orchestration, it is being studied the possibility of adapting the YourEPM tool to provide it with a REST API to interact with jFed.

YourEPM is a tool aimed to combine the federation application services in an experiment and to run automatic operations over them. However, it is being considered to expand its functionalities to allow making the reservation and provisioning of testbeds resources through the REST API.

Following is a more complete description of YourEPM, detailing its usage, architecture and workflow.

3.3.2.4 YourEPM

YourEPM (Your Experiment Process Manager) is a tool that aims to simplify the creation of experiment processes across different testbeds in the federation. For new experimenters joining Fed4FIRE, develop experiments that combine services and resources offered by two or more testbeds may not result a straightforward process due to the lack of a dedicated tool to use application services in the federation and the complexity of connecting data between testbeds with existing tools

In this sense, YourEPM provides simplified orchestration capabilities of federation services and resources, so that, experimenters can automate their experiments in Fed4FIRE just by defining its workflow without requiring deeper knowledge of each testbed. Despite being focused in application services orchestration, it will also be possible to manage the reservation of resources through already existing tools in the federation such as jFed.

Coming from the BPM (Business Process Modelling) realm, this tool is going to be adapted to the Fed4FIRE context, changing the business definition approach to the experimenters and testbeds scope within the federation.

YourEPM will be available as a central element in the Fed4FIRE architecture accessible from the web portal. The corresponding changes in the user management engine of the tool will be done to support multiple experimenters from different institutions.

3.3.2.4.1 Target usage

YourEPM addresses situations where specific experiments require using a combination application services and reserve resources offered by different testbeds following a predefined workflow, with the possibility of executing local scripting tasks to manipulate and analyze the data.

These experiments of orchestrated services can be carried out by any experimenter that belongs to the federation. Restrictions on the experiment execution will be applied according to established access policies on each testbed. Further restrictions can be set according to the already defined roles in the Fed4FIRE portal:

- **Users:** these are common experimenters with a valid Fed4FIRE certificate. Users are able to define and execute their own orchestrated experiments.
- **PI (Principal Investigator):** PIs are the managers of a specific institution. They can manage defined experiments in their institution, create groups of users, assign tasks to other users or PIs and are also responsible of validating new users belonging to their institution.

A common scenario on the experiment process definition can be as follows: The PI of an institution defines an experiment that may combine the data generated by several application services from remote testbeds, local scripts to process the obtained data and a resource reservation, in a different testbed, to store the results. The PI assigns two users that can develop the scripts, run the complete experiment, verify the execution performance and, if necessary, modify the experiment process definition. At the end of the experiment, the PI is requested to revise and verify the correctness of the results. Once it is approved, the experiment definition is stored for possible future iterations.

3.3.2.4.2 YourEPM architecture

The architecture of YourEPM tool is divided in two layers: Design and Execution.

The Design layer provides functional support for the modelling of experiment compositions based on the BPMN 2.0 [12] standard. This layer is composed by the following elements:

- **BPMN Composition Editor:** enables experimenters to create experiment compositions using the BPMN 2.0 graphical notation and execution semantics. It provides a typical GUI to create, edit and manage experiment compositions, requiring the experimenters to have a basic background in BPM modelling. Composition edition also includes support for creating/updating/deleting features for composition elements, such as service tasks, gateways (exclusive, parallel), flows and events (start, end).
- **Light-weighted Semantic Mediator:** complements the BPMN Composition Editor by providing additional semantic-enabled modelling aids that simplify the modelling process. These assisting features allow experimenters to describe the composition tasks, bind matching services, generate the data flow mapping, and so on.
- **Semantic Knowledge Base:** complements the Light-weighted Semantic Mediator with a semantic repository of application services descriptions. This component provides content access features, including querying and reasoning.
- **BPMN Manager:** provides BPMN model management, including features to validate and complete BPMN models with required executable information (e.g. service bindings, data flow mappings, etc.)
- **BPMN Translator:** provides translation capabilities to other executable composition formats, such as BPEL 1.2/2.0.

The Execution layer contains the Composition Execution component, which deploys, enables and executes the experiment compositions.

Acting as a link between the Design and Execution layers there is another component:

- **Composition Deployer:** belongs to the execution engine, acts as a proxy between the Light-weighted Semantic-enabled Composition Design layer and the Composition Execution layer. It manages the service composition deployment process into the selected Composition Execution layer.

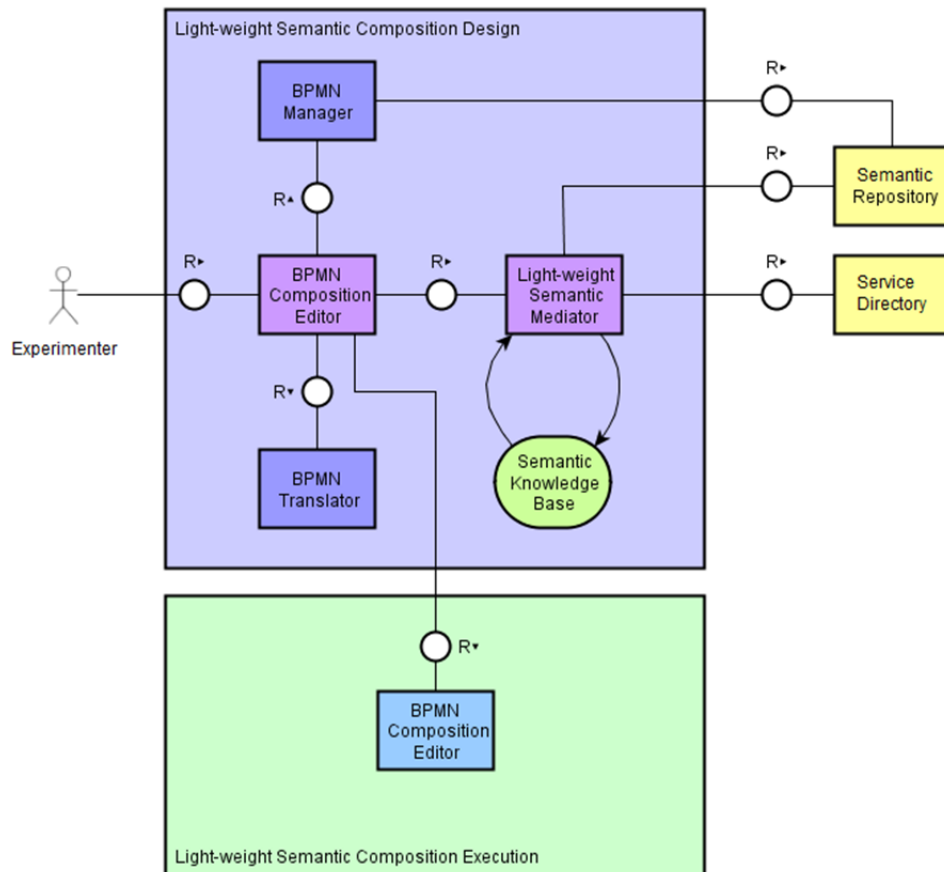


Figure 22 YourEPM architecture components

3.3.2.4.3 Experiment composition workflow

3.3.2.4.3.1 Create a composed experiment

The experimenter, once logged in the tool, can create, open, edit and/or manage (save, delete) experiment models through the BPMN Composition Editor. The design of the composed experiment can be easily done by means of drag and drop elements from the palette whose properties can be adjusted in a dedicated panel. In Figure 23 we can see the canvas of Composition Editor with dragged elements. The output of the composed experiment design will be a file based on the standard notation BPMN2.0. This notation enables the interconnection of any kind of service, either an application service or a resource reservation in a testbed through jFed tool. It is important to note that the aim of this tool is to simplify the way and experiment flow is created.

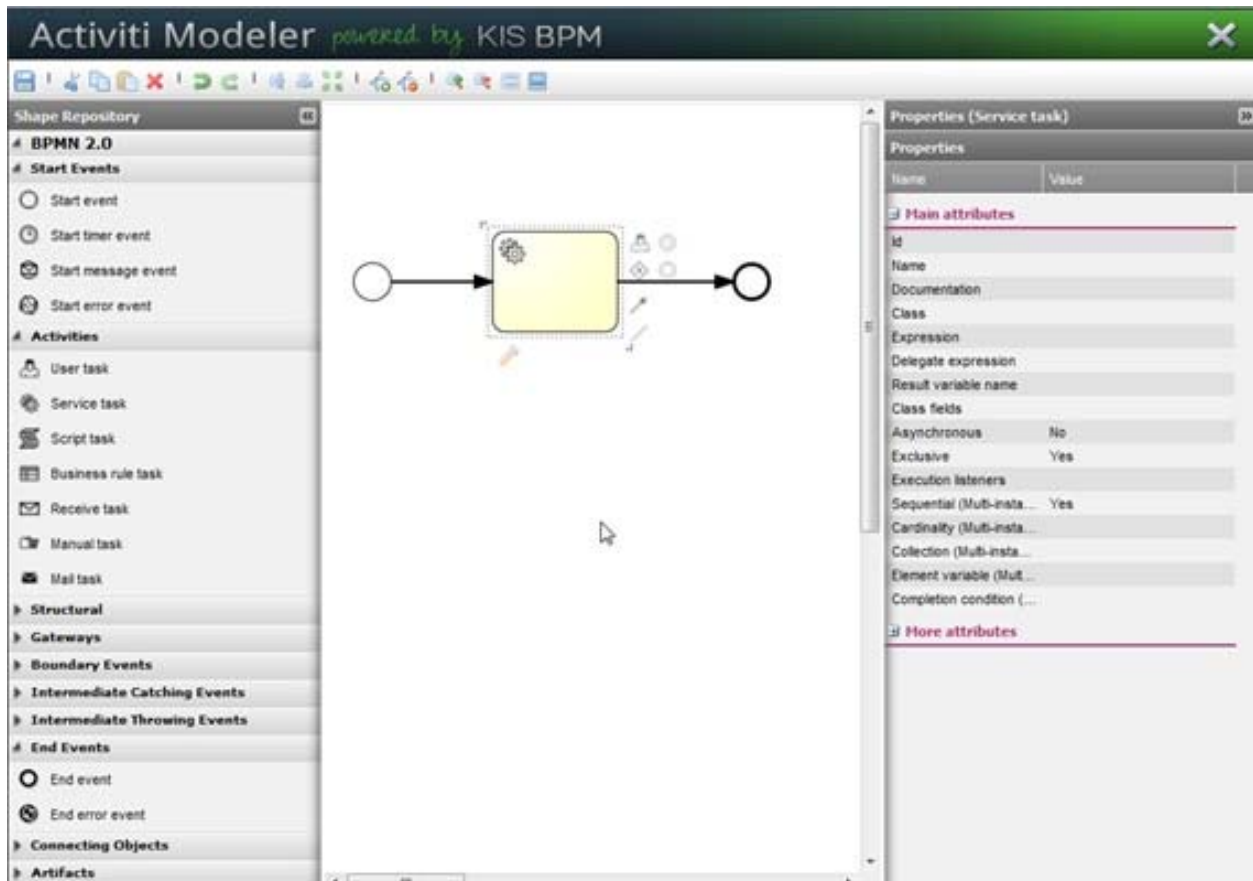


Figure 23: Simple experiment

YourEPM is conceived as a high level orchestration tool and it is not intended to use SFA to access testbed resources. However, with the integration with jFed as a service through a REST API, it will be possible for experimenters to perform resource reservations as part of their composed experiment, leaving the low level control of resources to be done using jFed.

All experimenter models are stored within a Repository with the appropriate access policies. The Composition Editor also supports creating, updating and deleting features of composition elements such as service tasks, manual tasks to be done by designed experimenters, gateways (exclusive, parallel), flows and events (start, end, error).

3.3.2.4.3.2 Discover application services through the Service Directory

Experimenters can discover available services through the Service Directory integration in YourEPM. A link to a special view of the Service Directory services will be available in the editor, where an overview of the services will be presented. Experimenters can then select the one that fits their needs or perform a keyword search to filter the results.

Currently, application services on the federation are based on REST APIs. In order to automatically identify the parameters of each service, a detailed description have to be provided using a REST API modelling language, for which RAML [13], Swagger [14] and WADL [15] are being considered since they represent three of the most popular options available.

Each application service will have a detailed view where the service description with the available tasks will be shown. It will contain:

- The service endpoint (URI).
- The accessible resources.

- The allowed methods to be executed on them (e.g. GET, POST, PUT, DELETE, etc.).

Although a semantic search of services is supported by YourEPM through the Light-weighted Semantic Mediator and the Semantic Knowledge Base, the definition of application service ontology in the federation would be required. Due to the complexity of this task, the possibility of adding the semantic search for services will be studied during cycle 3, taking into account the number of available services and the benefits presented by this approach.

3.3.2.4.3.3 Task binding

Once a task from an application service is selected, the experimenter will bind it to the task block in the BPMN Composition Editor. Saving the experiment after a task binding operation will keep the block bound to the assigned task.

A REST wrapper will be developed to translate the native Web Service calls in YourEPM to the appropriate REST calls provided in the application service definitions.

3.3.2.4.3.4 Generate BPMN 2.0 file

After the composed experiment has been defined and the tasks bound to the services, the experimenter generates a BPMN 2.0 file compliant with the associated services. The file contains all the standard tags indicating the wrappers that will translate the task calls and parameters.

The generated BPMN 2.0 XML file contains first the endpoint of the service or services that will be used, followed by the connection of BPMN elements with the service requirements. Finally a description of the graphical section for the BPMN Compositor Editor is generated.

3.3.2.4.3.5 Deploy and execute the BPMN 2.0 file

To execute the generated BPMN 2.0 file, the experimenters need to load and deploy it into YourEPM tool. During the experiment, manual tasks such as data verification or parameter input could have been defined and assigned to specific users. Whenever the experiment encounters one of these tasks, it stops the execution waiting for the user input.

After the experiment finalization, any defined output information will be available for involved experimenters to consult.

3.3.3 Policies

One of the main problems in federation systems is how each of the federated testbeds can control and prioritize the utilization of its resources by internal and external users. The counterpart of being able to reach more resources from a testbed is that other testbeds can use your resources leaving the local users without resources for their experiments.

A way to avoid these problems is the definition of policies that limit the amount of resources shared or that can be requested depending on the identity of the requester or any other criterion.

Policies can act at the moment of exposing resources (resource discovery), limiting what the federated testbeds can see from your own facility and thus limiting what they can request. Policies can also act at the moment of provisioning them, analyzing the requests and validating if they are served or not.

In both cases, the policies are acting in the testbed side, placed between the federation system and the aggregate managers. This situation prevents the necessity of a centralized system for policy management to which all the testbeds have to accommodate. On the contrary, it allows each testbed the freedom to implement or deploy any tool it may consider adequate in terms of functionality, compatibility, ease of deployment, etc., even the option of not deploy any policy having no restrictions on the request and provision of resources.

The only need for a centralized or standard system could be the exposing of the policies of all the federated testbeds so experimenters can know prior to request resources which are the restrictions of each testbed.

Since the common adopted tool for resource description, discovery and provision is the SFA API, in this section we present two tools that are compatible or easily adaptable to it.

3.3.3.1 *pyPElib*

PyPElib (python Policy Engine library) is a small library to help programmers to use abstractions provided by the library to build rule-based Policy Engine(s) within a certain scope of action. PyPElib is available as an open source repository under the LGPL license in [16].

pyPElib allows enforcing policies based on rules, which can also trigger actions or logs in a similar way to iptables.

The main function component of pyPElib is the rule table. The rule table encapsulates a set of policies of a certain scope which will be evaluated and produce a True/False result. The policies are defined by the rules that compose the rule table plus a default ACCEPT/DENY policy in case none of the rules match during the evaluation process. The rules may interact with other classes, functions or modules of the application to obtain extra information. Rules can be modified, reordered or deleted on the fly and – if required – persisted using a back-end.

The evaluation process receives an object which encapsulates the basic information to be checked – in the context of resource provisioning, this object would be the RSpec with the resources request.

When a request is received, the RSpec is parsed to extract the conditions of the request and those are verified against the rules in the rule table in the defined order.

There are two types of rules:

- Terminal rules: if the evaluated condition matches, the rule lookup loop is broken and the rule result value, ACCEPT or DENY, is returned. Example rule 3 in Figure 24.
- Non-terminal rules (action rules): they do not break the rule lookup loop in case of the evaluated condition matches; they perform actions (including the modification of the RSpec)

without returning any value (a typical example is for logging and statistical purposes). Example rule 2 in Figure 24.

The other main component is the mapping, which contains the basis association between keywords and objects, functions or static values. This mapping is defined by the user of the library according to the rules' needs.

A final component is the persistence module, which permits the storage of the rules, mappings and rule tables for future uses.

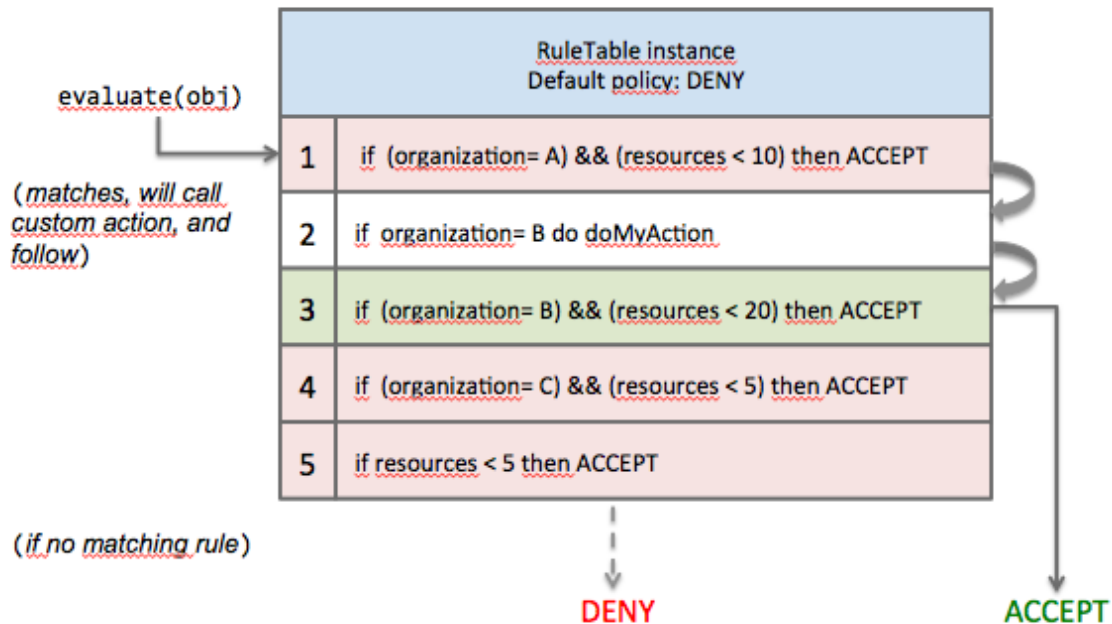


Figure 24: pyPElib rule table

3.3.3.2 SFATables

SFATables [17] is a tool for defining access and admission control policies in an SFA network, in much the same way as iptables is for IP networks. It uses the familiar paradigm of a firewall and provides a flexible interface for resource owners to specify, query and analyze access policies for their resources. SFATables is open source and is available under the PlanetLab repository at [18].

SFATables sit “in front” of the testbeds’ resources, intercepting resource requests and accepting, denying or modifying them like a firewall. It allows the resource providers to express and enforce the policies across their resources and the resource consumers to discover the policies that apply to them so they can take these policies into account when requesting resources.

Policies are defined as filters that operate on RSpecs and can transform them into a new RSpec that satisfies a given set of constraints.

In the same way as pyPElib, running SFATables on an RSpec can produce one of three results:

- Accept decision, if the RSpec satisfies the policies
- A modified RSpec, consistent with the configured policies, along with an accept decision
- A deny decision

The policy engine can be configured with two sets of policy rules:

- **Outgoing rules set:** applied on the RSpec output by ListResources. They are invoked before the RSpec is returned to the caller and can modify it to hide resource information that the caller should not see.
- **Incoming rules set:** applied to a resources request before the RSpec is passed to the Aggregate Manager. They can filter or modify incoming resource requests based on the caller's identity.

A SFAtables configuration consists of lists of rules that are applied to incoming and outgoing RSpecs. Each rule consists of a 'match', to evaluate a given request against a certain set of criteria and a 'target', to perform a corresponding action. Rules are manipulated by using a command line interface. SFAtables comes with a default set of matches and targets, which can be extended using a simple interface.

The basis for deploying rules using SFAtables is the library of matches and targets. A set of such rules constitutes a 'policy', which is a portable piece of information that can be exchanged with users, peers, and policy makers to make resource allocation and peering a more effective process.

A match specification consists of a 'context', a set of arguments, and a 'processor.' The context defines the information associated with a request that this match operates on, i.e. the input parameters to the match. The arguments define values specific to the rule. The processor refers to the program that actually evaluates the match. A configuration is simply a set of match-target pairs.

Targets are specified just like matches.

Matches and targets are associated with specific contexts. A target may use a variety of criteria to process a request, and may need to look them up in the SFA database. The 'context' contains an expression that isolates the items that a match or target may refer to.

Finally, resource provisioning service in extended FitSM format is presented Annex 2 (section 6).

3.4 Experiment control (Task 5.5)

3.4.1 Approach

This document specifies the features that are desirable as part of Cycle 3 of the project, in regard to Experiment Control. In this section we will consider two very different angles, that we argue need both to be taken in consideration.

On the one hand, as far as Fed4FIRE is concerned, Experiment Control is addressed through a handful of **tools**, and that we summarize for reference in the next section, where we emphasize their respective positioning with respect to Experiment Control, given that Experiment Control is only one part of the overall Experiment Life Cycle – namely, some of these tools target more the early stages of resources discovery and provisioning, while some others were initially more targeting Experiment Control per se.

On the other hand, since we are dealing with specifications, it is crucial to keep in mind the various types of **usages** that need to be addressed, so that a given user can be expected to experience a smooth learning curve, starting as beginner and then gaining experience and confidence with the set of tools.

For these reasons, we start with a brief presentation of the tools dimension, focusing of course on Experiment Control, and then identify what we consider are the two major kinds of usages – in a nutshell, beginners and advanced users - and the implications, in terms of specifications, on the gateways that are needed between the various tools, so that the resulting toolset is as integrated as possible with respect to these target usages.

3.4.2 Current position of the various tools involved

For the record, here is the list of tools that are taken into account in these specifications.

3.4.2.1 Portal / MySlice

MySlice is the tool behind the Fed4Fire portal; in its current state MySlice is particularly helpful in that it provides a single point of entry in the system when undertaking a new experiment. Once registered and logged into MySlice, a user can interact with the various facilities of the federation in a uniform way, without having to provide any additional credentials. One can thus gain an overall perspective of the available resources, together with a thorough indication of their respective specifics – as exposed in the various underlying *RSpecs* – and of their availability, in terms of physical resources, as well as over time for reservable resources.

In this respect, it is fair to state that MySlice addresses more the early stages of the experiment, in other words that belongs more in the Control Plane than in the Experimental Plane, in that at this point in time at least, once the resources are selected and obtained, MySlice has limited features for actually running an experiment in its details. It plays however a rather central role for any user who needs more than one testbed to implement her experiment, and to newcomers as well because it provides a unique global view of the available resources.

3.4.2.2 jFed

JFed provides features that are more evenly balanced in the Control Plane – Experimental Plane dimension. Using its interactive tool (a.k.a. GUI), one can draw topologies, probe the various testbeds to discover physical resources, and create virtual resources onto which the topology will be mapped. It is then possible to interact with these virtual resources for triggering commands remotely, and

thus controlling one's experiment. JFed also provides a command-line tool (a.k.a. CLI) that is less relevant with respect to the Experimental Plane since, at this point in time at least, it focuses mostly on slices management features like creation, renewal and deletion.

In the perspective of the present document, jFed clearly has more powerful features than MySlice in terms of Experiment Control. Also it was historically designed primarily as a GUI, and so its CLI capabilities regarding experiment control are more limited than what is offered in the GUI. jFed tries to offer the experimenters a graphical interface to experiment control as can be seen below:

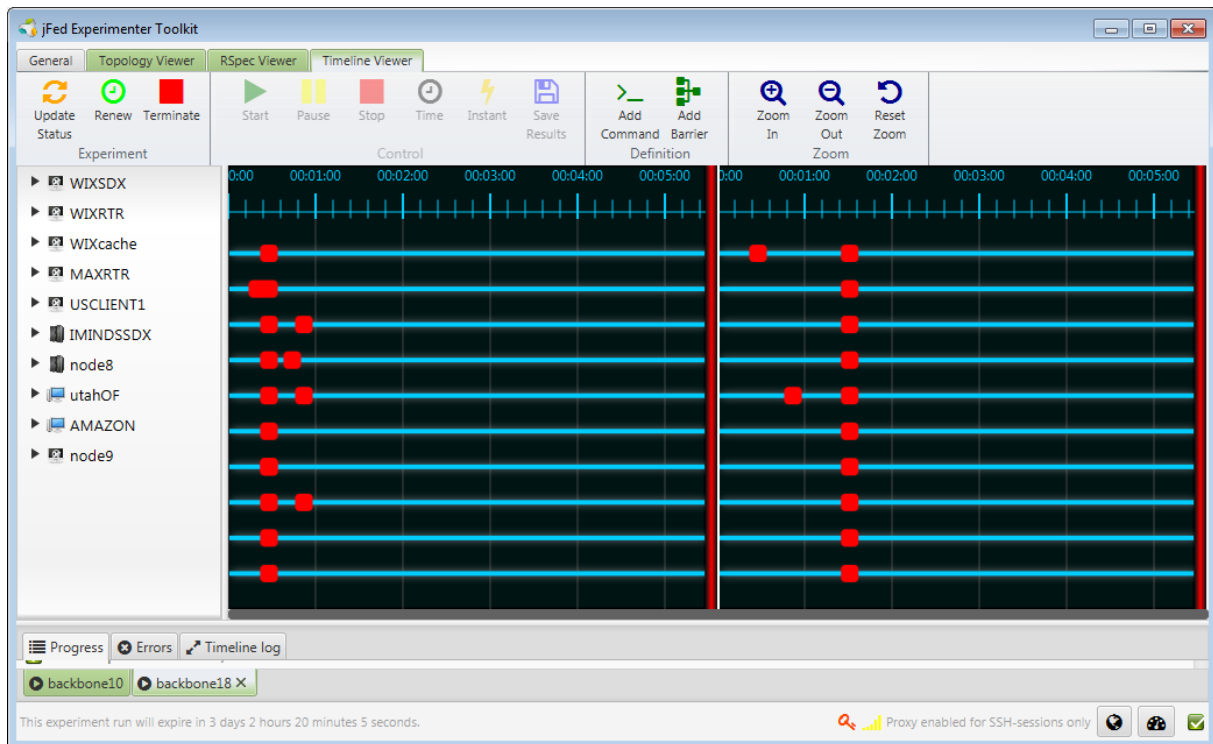


Figure 25: jFed GUI for experiment control

The nice thing is that the commands can be saved in the RSpec, so to easily reproduce experiments in the future:

```

jFed Experimenter Toolkit
General Topology Editor RSpec Editor Timeline Editor
Run Save Format Code Verify RSpec Search Search & Replace
Code
xmlns:uri="http://www.protogeni.net/resources/rspec/ext/uri/1" xmlns:jfed
command="http://jfed.iminds.be/rspec/ext/jfed-command/1"
xmlns:client="http://www.protogeni.net/resources/rspec/ext/client/1" xmlns:jfed-ssh-
keys="http://jfed.iminds.be/rspec/ext/jfed-ssh-keys/1" xmlns:jfed="http://jfed.iminds.be/rspec/ext/jfed/1"
xmlns:sharedvlan="http://www.protogeni.net/resources/rspec/ext/shared-vlan/1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.geni.net/resources/rspec/3 http://www.geni.net/resources/rspec/3/request.xsd ">
3 <node client_id="node0" exclusive="false" component_manager_id="urn:publicid:IDN+wall2.ilabt.iminds.be+authority+cm">
4 <sliver_type name="emulab-xen"/>
5 <location xmlns="http://jfed.iminds.be/rspec/ext/jfed/1" x="121.0" y="175.5"/>
6 <jfed-command:experimentCommand barrierSegmentOrderNumber="0" tag="Command 1">
7 <jfed-command:command>iperf -c</jfed-command:command>
8 <jfed-command:duration>
9 <jfed-command:time>0.0</jfed-command:time>
10 <jfed-command:duration>
11 <jfed-command:startingTime>
12 <jfed-command:time>0.0</jfed-command:time>
13 <jfed-command:startingTime>
14 </jfed-command:experimentCommand>
15 <jfed-command:experimentCommand barrierSegmentOrderNumber="0" tag="Command 2">
16 <jfed-command:command>sudo apt-get install dstat</jfed-command:command>
17 <jfed-command:duration>
18 <jfed-command:time>0.0</jfed-command:time>
19 <jfed-command:duration>
20 <jfed-command:startingTime>
21 <jfed-command:time>58000.0</jfed-command:time>
22 <jfed-command:startingTime>
23 </jfed-command:experimentCommand>
24 <jfed-command:experimentCommand barrierSegmentOrderNumber="1" tag="Command 3">
25 <jfed-command:command>iperf -c node0</jfed-command:command>
26 <jfed-command:duration>
27 <jfed-command:time>0.0</jfed-command:time>
28 <jfed-command:duration>
29 <jfed-command:startingTime>
30 <jfed-command:time>48000.0</jfed-command:time>
31 <jfed-command:startingTime>
32 </jfed-command:experimentCommand>
33 </node>
34 <jfed-command:experimentBarrierSegment orderNumber="0" tag="Barrier segment 0"/>
35 <jfed-command:experimentBarrierSegment orderNumber="1" tag="Barrier segment 1"/>
36 </rspec>
sdx1 Untitled X
Click and drag the circle to a target node to create a new link. Proxy enabled for SSH-sessions only

```

Figure 26: jFed RSpecs for experiment control

3.4.2.3 NEPI

NEPI on the other hand is a tool that was initially designed as a CLI-only tool, and it has no GUI at all. It allows to entirely specifying the details of one experiment, including in terms of fine-grained orchestration dependencies, and in terms of data collection. NEPI aims at experiment repeatability, and was deeply influenced by the notion of the *runnable paper*, meaning that NEPI can be used to design an experiment script that can be run anytime, without human attendance.

For all these reasons its positioning on the Control Plane – Experimental Plane axis is predominantly towards the latter, although it is also possible to issue SFA commands to actually create a slice, but without any fancy way to browse for resources.

3.4.2.4 Reservation Broker

Finally, we will consider the Reservation Broker as a tool that can be leveraged for improving the Experiment Control experimenter experience. Of course, this tool clearly belongs in the Control Plane only, since its purpose is to satisfy global requirements, both about resource characteristics and in terms of simultaneous availability for reservable resources. However, at this point in time, this tool remains to be more tightly integrated into the overall experiment workflow, and we will see below how we envision that the potential benefits of such an integration can reach down to the Experimental Plane arena.

3.4.3 Usages

For the sake of simplicity, as mentioned above we consider two major types of usage, as a user's fluency with the facilities and tools increases, starting with the learning phase and evolving to a

production phase where actual papers are getting written or where experimental code is submitted to real-size conditions.

3.4.3.1 Learning phase – tutorials and beginners

Obviously a system as complex as a federation of experimental testbeds cannot be grasped in an eye blink. Our past experience with similar systems has taught us that attracting new users is key to success, and at that point potential users, even very knowledgeable researchers or engineers need visual tools in order to quickly assess the technical offer, be it about resources, capabilities and tools. Key points here seem to be:

- **Credentials management:** all the technologies involved need to rely on some form of authentication (SFA certificates, SSH keys, login/passwords), and it should be very obvious now that substantial energy has been devoted to minimizing the burden of providing such credentials all along the experiment lifecycle. In simpler words, users, and especially beginner users, can accept to spend in the order of 3 minutes – and preferably less – for getting all set credentials-wise, but will not tolerate to have to repeat the process for each tool, nor *a fortiori* to see their train of thought continually interrupted by prompting for passwords or certificates.
- **Visual tools:** if only for getting an accurate sense of the resources diversity, either geographically – because even in a virtual world, location does matter – or technologically – what exactly are the settings that I am able to change on this virtual node, or on this wireless node – a visual tool can obviously do a very effective job for a newcomer. Particularly so, as opposed to having to learn convoluted APIs before one even knows what this new technology can be useful for.

3.4.3.2 Production phase

Having convinced a potential user of the adequacy of our collection of available testbeds for her experimental needs, it is as much important to provide tools that are also suitable for production, be it for scientific or academic work, or for more industry-oriented purposes for pre-competitive solutions. These needs often exhibit another set of requirements, and namely among the key factors again:

- **Repeatability:** a given experimental scenario must be easily repeatedly played over and over again, either exactly identical for gathering meaningful results in a probabilistic environment, or with small variations when one studies the impact of a given parameter on the overall results.
- **Scalability:** likewise, it is in most cases interesting to be able to run a given scenario in a topology that involves a flexible number of resources, for example when studying the impact of size on a given measurement.
- **Batch-oriented:** another much-desired feature of Experiment Control tools is the ability to run an experiment in unattended mode, for example at night-time when the resources are otherwise less heavily used, or just as a consequence of repeatability. Which as a side-effect implies **credentials management** here again, obviously, since in unattended mode nobody sits in front of the screen to enter a password.

3.4.4 Specifications

3.4.4.1 *Common features for different usages*

In order to address the common desired property of simple credentials management, we expect that it should be possible to perform **user account creation** using a **single tool** (although several tools may provide the feature). In addition, in the case of the portal which by definition runs in the Fed4Fire infrastructure and thus in a separate computing space, the user must have the ability to **download all necessary credential** materials into her own laptop, so that using a separate tool like NEPI down the experiment lifecycle is possible without caring any further about such credentials (except of course for renewing them over time).

3.4.4.2 *Features for the learning phase*

Given the toolset as briefly exposed earlier, several combinations of tools can be considered when putting together learning material, or when running tutorial sessions, and they are:

1. Either jFed in standalone mode, since in its present form it provides some great features that give a very vivid sense of the exciting capabilities offered by Fed4FIRE, or
2. A combination of MySlice and jFed, which would add very fancy perspectives on the diversity and specifics of the set of resources available throughout the federation, including wireless or IoT-related resources.

JFed already provides quite a decent environment for such first steps, and is also able of setting up its own credentials, so solution 1 is deemed readily available. In order to make solution 2 a viable alternative, some additional features need to be taken care of. In the first place, using jFed out of credentials initiated in MySlice still needs small developments. Second, it would be very handy to have the ability to seamlessly invoke jFed locally from a MySlice session, with the set of resources attached to the slice already exposed to jFed (i.e. without the need for a query to the infrastructure), where the user could proceed from there using the rich features of the jFed GUI.

3.4.4.3 *Features for the production phase*

For this kind of usage it is of course an option to keep on using jFed for a production phase. There are however some limitations to that approach, with respect to the needs as defined earlier, that directly result from jFed's orientation as a GUI tool, which makes it not too suitable for example for unattended mode, and to a lesser extent for scalability.

Given again the current toolset as a background, it is rather natural to consider the following tool chains:

1. A combination of MySlice and NEPI, where resource selection is done in MySlice, and the results are forwarded as an input to NEPI;
2. A combination of MySlice + Reservation broker + NEPI, that adds to the previous solution the ability to leave actual resource mapping to the Reservation Broker, from higher-level specifications.

As far as solution 1 is concerned, we have identified several integration levels between MySlice and NEPI. In its simplest implementation, such a solution could be obtained in its simplest form by providing a button in MySlice that lets the user download a python file, describing the list of actual resources involved in the slice, which could be imported by the NEPI script as a regular python module import. In a slightly more advanced variant, the NEPI script could contact the MySlice infrastructure – using the manifold API – to retrieve the same information programmatically, thus

removing the need for any UI interaction. A third option would be to have MySlice generate a template NEPI script, and this option would be useful for people getting acquainted with NEPI. Finally, we are still studying the possibility to run a NEPI script right from MySlice, but at this point in time this is still a very prospective idea that seems at first glance to be rather tricky to implement.

As far as solution 2, we plan on adding the ability in MySlice to interact with the Reservation Broker so that the new feature can be widely available right from the portal. This of course requires some changes to the portal's logic, and new ways to express resources requirements in a form that is of higher level: users would e.g. say that they need 10 wired nodes and 4 wireless nodes in the A range, and the Reservation broker would translate this into actual node names and a timeslot.

As a final note on this matter, let us mention that a direct interaction between NEPI and the Reservation Broker could make perfect sense quite obviously as well, but for practical reasons we will have to postpone and to consider such a feature as part of a future spiral cycle.

3.4.5 Summary and Conclusion

Fed4FIRE has chosen a very concrete and pragmatic approach based on iterative cycles in a spiral model. At this point in time it seems that each of the Fed4FIRE tools, taken individually, has a rather complete set of features, and for the time being, namely for the next cycle we will focus on improving **interoperability** and gateways between the current set of tools, in an attempt to provide more consistent and **more seamless tool chains**, targeting specific usages.

In this respect it is worth noting also that, as far as Experiment Control is concerned, the general topic of policies is not deemed as a hard requirement at this point. It is our feeling that simple policies can easily be implemented at the level of each individual testbed, without the need – for the time being again – for a sophisticated policies scheme that would allow to formalize and reconcile such policies. Or at least we have no clear view on how policies would impact the tools that deal with Experiment Control as we have presented them in this section.

3.5 User Interface / Portal (Task 5.6)

3.5.1 Fed4FIRE Portal

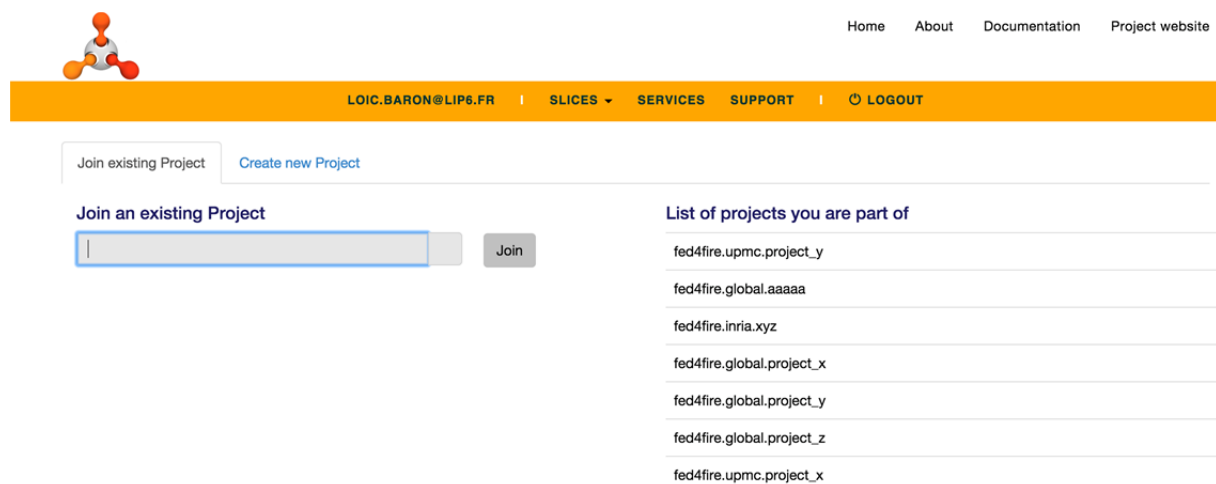
The Fed4FIRE portal acts as a single point of access to services of the federation will interact with all the WP5 tasks.

3.5.1.1 Projects management

The Fed4FIRE Portal will implement a new feature that will allow users to create as many slices as they want without approval within a project. The administrators' approval will only be necessary to create such a project. According to users' feedback, this feature will ease the use of the portal.

The workflow for projects management will be the following:

A basic user will be able to view a list of the projects he/she is a member and the existing projects in order to join one.



The screenshot shows the Fed4FIRE portal interface. At the top, there is a navigation bar with links for Home, About, Documentation, and Project website. Below this is a user profile bar for LOIC.BARON@LIP6.FR with options for SLICES, SERVICES, SUPPORT, and LOGOUT. The main content area has two tabs: 'Join existing Project' (selected) and 'Create new Project'. Under 'Join existing Project', there is a search input field and a 'Join' button. To the right, a list titled 'List of projects you are part of' shows several project IDs: fed4fire.upmc.project_y, fed4fire.global.aaaaa, fed4fire.inria.xyz, fed4fire.global.project_x, fed4fire.global.project_y, fed4fire.global.project_z, and fed4fire.upmc.project_x.

This user willing to join a project will select a project and click on the join button. Then a request will be sent to the administrators of the project.

The project administrators will be able to validate or reject this request.



The screenshot shows the 'Management' page for a project: 'Project: fed4fire.upmc.project_y'. It has tabs for 'About', 'Users', 'Slices', and 'Requests' (selected). Below the tabs is a table with columns: ID, Type, Authority, Info, Date, and Status. The table contains one row with ID '1', Type 'join', Authority 'fed4fire.upmc.project_y', Info 'John Doe <john.doe@fed4fire.eu>', Date 'Mar. 10, 2015, 8:10 a.m.', and Status. Below the table are two buttons: 'Validate' (with a checkmark) and 'Reject' (with an X).

A basic user will also be able to create a new project. This action will generate a request sent to the administrators of the authority under which the project creation has been requested. However, if the user has the administration rights (PI) on the selected authority, the project will be created directly.

[Join existing Project](#)[Create new Project](#)

Create a new Project

The administrator of the authority will be able to validate or reject this project creation request.

Management > Institution: fed4fire.upmc


[About](#) [Users](#) [Projects](#) [Slices](#) [Requests](#)

From your authorities

ID	Type	Authority	Info	Date	Status
1	<input type="checkbox"/> project	fed4fire.upmc.new_project	John Doe <john.doe@fed4fire.eu>	Mar. 20, 2015, 10:20 a.m.	

Once the project is created, the user can create as many slice as he/she wants within this project.

Experiment > Request a new Slice

Project: 

3.5.1.2 Reservation Broker

As mentioned in Task 5.3, the integration between the Portal and the Reservation broker will be further developed in order to ease the reservation of Fed4FIRE resources.

3.5.1.3 Experiment Control

As mentioned in Task 5.5, the Portal and the Experiment Control Tools will be further integrated in order to improve the ease of use and to allow repeatability of experiments.

3.5.2 Authority Portal

The iMinds authority portal (<https://authority.ilabt.iminds.be>) is currently used as one of the main entry points for experimenters to create an account and then start the jFed tool.

Figure 27: Authority Portal for account creation

Figure 28: Authority Portal for quickstarting jFed

4 Conclusion and Future Plans

The experiment workflow and life cycle management is specified via a set of tools, services and platforms in conformance with the emerging industry standard for service management in federated IT infrastructures. This strategy is in line with the plans for the project sustainability phase, 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 (D5.1) and enhanced (D5.2) specifications of tools for experiment lifecycle management.

This development occurred within the project environment concurrently with the

- development of cycle 3 architecture,
- on-going work for Federation Board specification;
- joint work with GENI and with OMN Forum,
- sustainability of Fed4FIRE infrastructure supported by FitSM for service- and process orientation, and lastly, by
- Discussions of interworking with other federations within and outside the FIRE.

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

However, it should be noted that the major difference of cycle 3 evaluation from that of cycle 1 and 2 consists in the change of operational paradigm. The cycle 3 paradigm is that of an operational federation of facilities, open access and sustainability phase, 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 Calls and from the SME Calls.

Since it is hard to underestimate the importance of these “early adopters” the future work must be continued 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), *Detailed specification for second cycle ready (Deliverable 5.3)*, EU, Fed4FIRE Consortium.
- [3] Fed4FIRE (2015), *Initial Federation Board Specification*, (Deliverable 2.13), EU, Fed4FIRE Consortium
- [4] Fed4FIRE. (2014). *Second Federation Architecture (Deliverable 2.4 version 8)*. EU: Fed4FIRE Consortium.
- [5] FOAM: <http://groups.geni.net/geni/wiki/OpenFlow/FOAM>
- [6] Fed4FIRE (2014) *Report on first cycle developments of the services and applications community (Deliverable 4.3)*. EU: Fed4FIRE Consortium.
- [7] *FitSM: Standards for lightweight IT service management*, on-line at <http://www.fedsm.eu/fitm>
- [8] Wikipedia contributors, "Operational definition," *Wikipedia, The Free Encyclopedia*, http://en.wikipedia.org/w/index.php?title=Operational_definition&oldid=644847075 (accessed February 15, 2015)
- [9] [Unifying Management of Future Networks With Trust \(pages 193–212\)](#) Laurent Ciavaglia, Samir Ghamri-Doudane, Mikhail Smirnov, Panagiotis Demestichas, Vera-Alexandra Stavroulaki, Aimilia Bantouna and Berna Sayrac, Article first published online: 27 DEC 2012 | DOI: 10.1002/bltj.21568
- [10] M. Giatili, C. Papagianni, S. Papavassiliou, "Semantic Aware Resource Mapping for Future Internet Experimental Facilities", Accepted for presentation in IEEE CAMAD, Athens 1-3, 2014
- [11] Fed4FIRE. (2014). *Detailed specification for third cycle ready (Deliverable 7.4)*. EU: Fed4FIRE Consortium
- [12] <http://www.omg.org/spec/BPMN/2.0/>
- [13] <http://raml.org/>
- [14] <http://swagger.io/>
- [15] <https://wadl.java.net/>
- [16] pyPElib. <https://github.com/fp7-ofelia/pypelib>
- [17] S. Bhatia, A. Bavier, L. Peterson and S. Sevinc, "SFATables: A firewall-like policy engine for federated systems", 2011
- [18] SFATables repository. <http://git.planet-lab.org/?p=sfa.git;a=tree;f=sfatables;hb=HEAD>
- [19] Fed4FIRE. (2014). *Detailed specifications regarding experiment workflow tools and lifecycle management for the second cycle (Deliverable 5.2 version 12)*. EU: Fed4FIRE Consortium.
- [20] Willner, A., Papagianni, C., Giatili, M., Morsey, M., Grosso, P., Al-Hazmi, Y., & Baldine, I. (2015). The Open-Multinet Upper Ontology - *Towards the Semantic-based Management of Federated Infrastructures*. In IEEE (Ed.), *10th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (to be accepted)*. Vancouver, Canada.

5 Annex 1: Extending FitSM with Service Policies

Indeed, in a service oriented world an OLA can be seen as a placeholder for policies. A federation may use different methods to keep a record of all policies that are agreed upon and that are maintained, that is managed, enforced, modified, etc. in full accordance with the policy life cycle. However keeping policies aligned with an OLA helps federation to migrate from pure access control to a process control, which in turn facilitates another migration path, namely from point correctness to process correctness as required by the sustainability of a federation.

In the below we demonstrate pragmatically how to associate policies to services and how to differentiate them from those policies that constitute OLA. The starting point will be to look at every service description and extend it with relevant policy fields, which then will be collected and harmonised before being associated with OLA. In order to show a concrete example we shall consider a purely hypothetical service briefly described in Figure 29.

User Alice has developed a big data analytics software package that she wants to run concurrently on as many nodes of a federation as possible however within certain cost-utility envelope. The workflow of this big data analytics is as follows: SENSE modules being deployed on nodes collect primary metrics of node operation and feed those to STAT module, which does certain statistical analysis of primary metrics and distributes the results to CTRL modules. Depending on user-configured thresholds the CTRL modules decide on i. configurations of SENSE modules; ii. deployment of new SENSE modules or stopping existing SENSE modules; iii. deployment of new CTRL modules or stopping existing CTRL modules. All workflow modules operate in slotted time. There is always only one STAT and at least one SENSE and one CTRL module. The workflow operation stops after pre-defined number of time slots (normal operation) or on impossibility to continue the operation (fault condition) For the sake of this example it is enough to consider that the cost-utility envelope is being defined within the workflow like this: the STAT module computes certain workflow utility metric, while deployment of each new SENSE or CTRL bears certain cost.

Figure 29 Sample service technology

According to FitSM the service based on the described technology is specified in Table XXX , the coloured rows are added to cater for relevant policies

Table 2 sample service description

Basic Information	
Service name	CCA - Cost-efficient Cloud Analytics
General description	Self-orchestrated set of sense, analyse, decide modules configurable for cloud monitoring tasks
User of the service (role)	Cloud administrator or developer
Service access policies	Administrator: no constraints Developer: on valid credentials
Service management	
Service Owner	Cloud owner if used by administrator, developer otherwise
Contact information (internal)	Contact within a cloud
Contact information (external)	Developer only (on wish)

Service status	Current status
Service Area/Category	Administrator: OLA support Developer: task monitoring
Service agreements	Administrator: all enforced SLA's offered by a cloud Developer: configurable (thresholds and counts)
Service usage policies	Administrator: per offered SLA's Developer: that of resource usage (quota time reservation<...)
Detailed makeup	
Core service building blocks	SENSE, STAT, CTRL modules
Additional building blocks	Additional (replicated) SENSE and CTRL modules
Service packages	Administrator: CCA packaged with SLA protected cloud services Developer: CCA packaged with developer defined workflow
Dependencies	Administrator: as defined in OLA Developer: TBD
Dependency policies	Number of SENSE and CTRL modules varies on load subject to usage policies, STAT module is unique (single).
Technology Risks and competitors	
Cost to provide (optional)	Cost of deployment of SENSE and CTRL, communication overhead
Funding source (optional)	Administrator: cloud owner Developer: TBD
Pricing (optional)	TBD
Pricing policies	Flat Reputation based Load based Utility based ...
Value to customer (optional)	Cost efficient and configurable monitoring system
Risks	DoS
Competitors	Ceilometer (Zabbix), Nagios, Ganglia, etc.

In this work we conjecture that highlighted rows containing relevant policies are generic enough to be applied to a reasonably wide spectrum of federation services. Let us note however that the above service specification is purely speculative and is used only as an illustration of the FitSM concept and template - now being enhanced with policies. A remark should be made on Competitors field, namely those competitors can be easily packaged with CCA as variants of SENSE, the CCA then provides dynamic reconfiguration of the system.

Obviously the above four types of policies are service specific. For SLA monitoring they will need to monitor service specific KPI's, and, as it is well known the decision on target KPI's for service improvements is hard.

6 Annex 2: Resource Provisioning Service

Since resource provisioning in cycle 3 deals with policies as explained in section 3.3.3 explicitly we demonstrate on this example how concrete WP5 service can be represented in a service portfolio using extended FitSM format introduced in Annex 1 (Section 5)

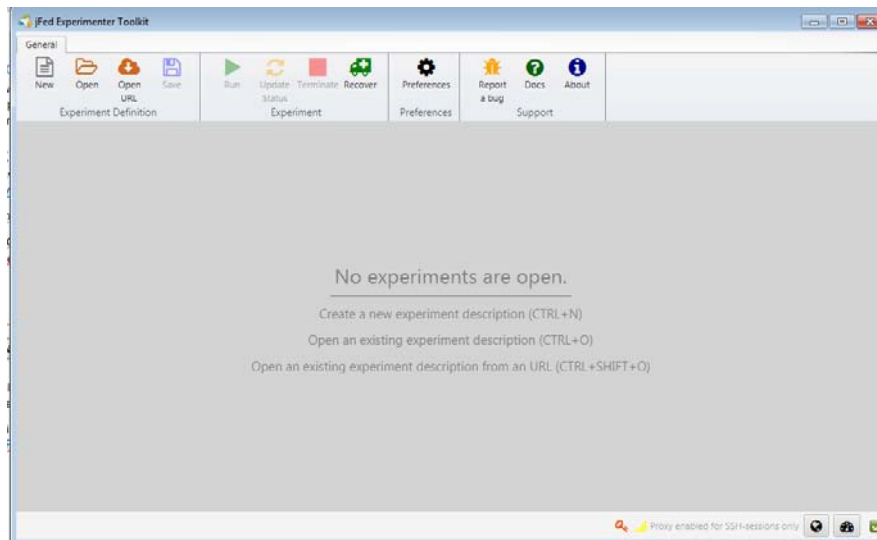
Basic Information	
Service name	Resource provisioning
General description	Resource provisioning allows to the accredited user the instantiation of resources from one or several testbeds to deploy his experiment. It can be direct (user selects specific resources) or orchestrated (user defines requirements and service provides the best fitting resources).
User of the service (role)	<p>The final user of the Resource Provisioning service is the Experimenter when requests resources for his experiment.</p> <p>The Testbed Manager or Administrator indirectly uses the service by configuring the Aggregate Manager and its policies to offer the resources to be provisioned.</p> <p>The direct Fed4FIRE components that invoke the service are the users' GUIs (MySlice Portal, jFed) and CLIs (Omni), the reservation broker, policies engines (PDO, SFAWrap, pyPElib), service directory and reputation engine.</p>
Service access policies	<p>Administrator: no constraints</p> <p>Experimenter: on valid credentials</p>
Service management	
Service Owner	<p>Task Leader: i2CAT</p> <p>Each testbed owns the resource provisioning service for its resources.</p>
Contact information (internal)	carlos.bermudo@i2cat.net
Contact information (external)	https://portal.fed4fire.eu/
Service status	<p>Cycle 1: conception</p> <p>Cycle 2: service development for some testbeds</p> <p>Cycle 3: integration</p>
Service Area/Category	<ul style="list-style-type: none"> • Resource provisioning. • Resource description. • Resource specification • Infrastructure monitoring
Service agreements	Best-match search for the orchestrated provisioning. Orchestrated provisioning requires finding the resources that match experimenters' requests and provision them.
Service usage policies	Experimenter: per offered SLA's or other policies defined in the

	testbed. Administrator: defines the policies
Detailed makeup	
Core service building blocks	For the direct provisioning, testbed directory and resource directory. For orchestrated provisioning, resource discovery and reservation broker.
Additional building blocks	SFA Registry, SFA AMs
Service packages	This service will offer two modes of resource provisioning (direct and orchestrated) each one can be a different package although with some common functions.
Dependencies	Services dependencies: Resource Discovery, Resource specification and Infrastructure monitoring to know the availability of resources to be provisioned. Software dependencies: SFA implementation (SFAWrap or AMsoil), MySlice Portal, plugin for MySlice, jFed, monitoring of available resources.
Dependency policies	Number and type of resources provisioned may vary subject to the policies defined in each testbed.
Technology Risks and competitors	
Cost to provide (optional)	Cost of implementation of SFA API in the testbed.
Funding source (optional)	EU
Pricing (optional)	Free of charge
Pricing policies	Utility based
Value to customer (optional)	Cost efficient and configurable monitoring system
Risks	The resource provisioning relies on the correct authentication of the users in order to provide resources. If this authentication is overpassed, resources could be provisioned to unauthorized user. Availability of resources must be monitored correctly, otherwise users can be granted access to resources already provisioned to other experiments. For the provider, the risk is a loss of revenue since unauthorized users may consume resources, for the user it is a risk that his experiment is potentially disclosed to unauthorized users.
Competitors	There are specific solutions for resource provisioning in other testbed-based projects. Most of those solutions are part of testbeds that will be part of Fed4FIRE federation, so the potential risk of competitors is reduced as they are assimilate by this solution.

7 Annex 3: jFed functionality

This annex walks through a jFed experiment while showing all the functionality. It is based on tutorials we give to experimenters.

When starting jFed, you should see the canvas as shown below after logging in:

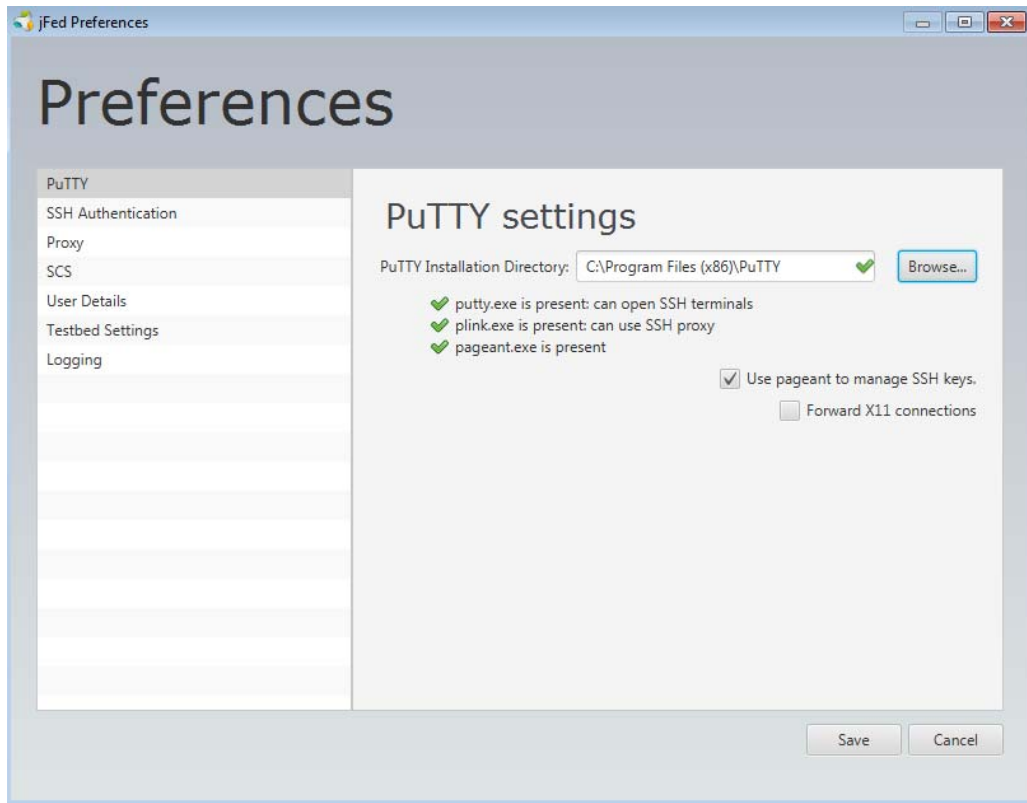


First we will check if our preferences for SSH terminal are okay. Go to 'Preferences' (button on the top). Depending on the platform, the preferences might be opened automatically the first time you run jFed.

For **Windows users**, the SSH terminal that will be used is PuTTY and jFed should have found it as shown below (all checks should be green). If not, then please install PuTTY from the following link: <http://the.earth.li/~sgtatham/putty/latest/x86/putty-0.64-installer.exe>

For **MAC/Linux users**: the terminal should be automatically chosen.

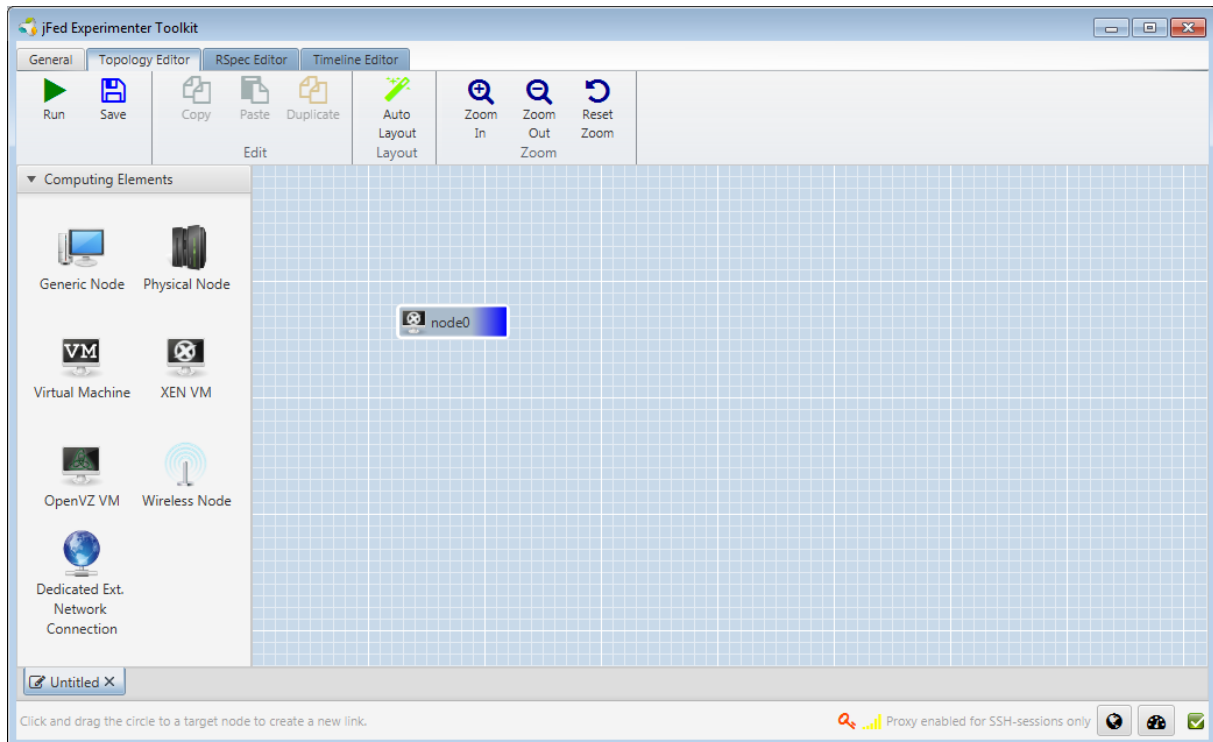
Note: check that 'Use pageant/ssh agent to manage SSH keys' is checked.



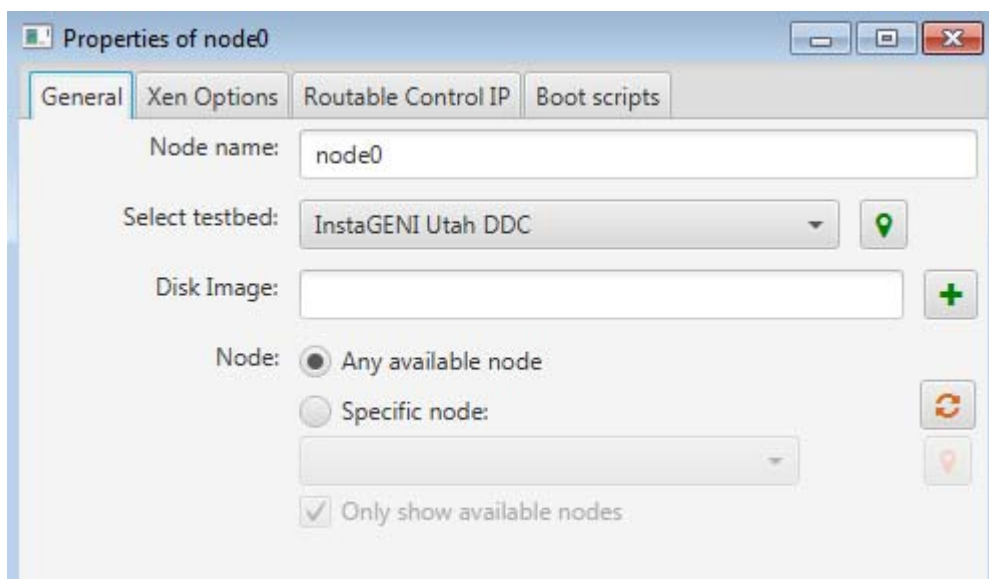
First basic experiment

Define the experiment

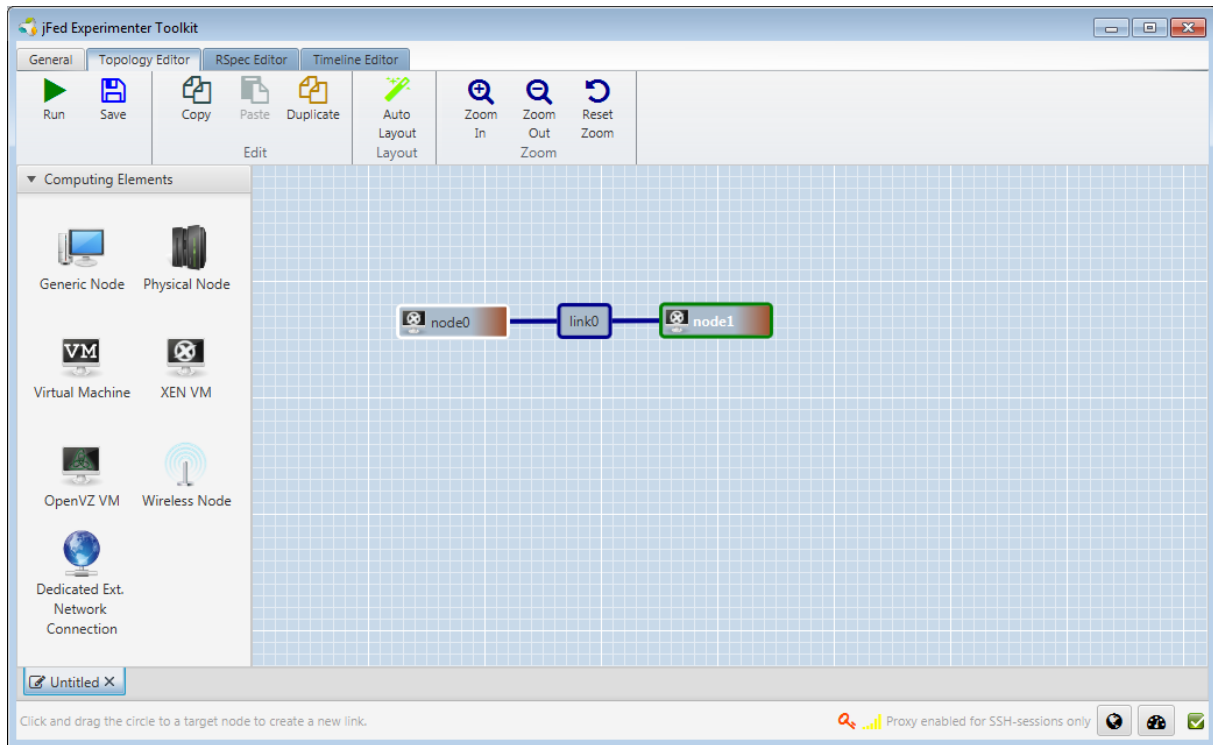
When you click **New**, you get a blank canvas where you can draw your experiment. Let's drag in a XEN VM from the left side to the canvas. For more specific experiments you can right click and configure the node, but for now let it in the default settings.



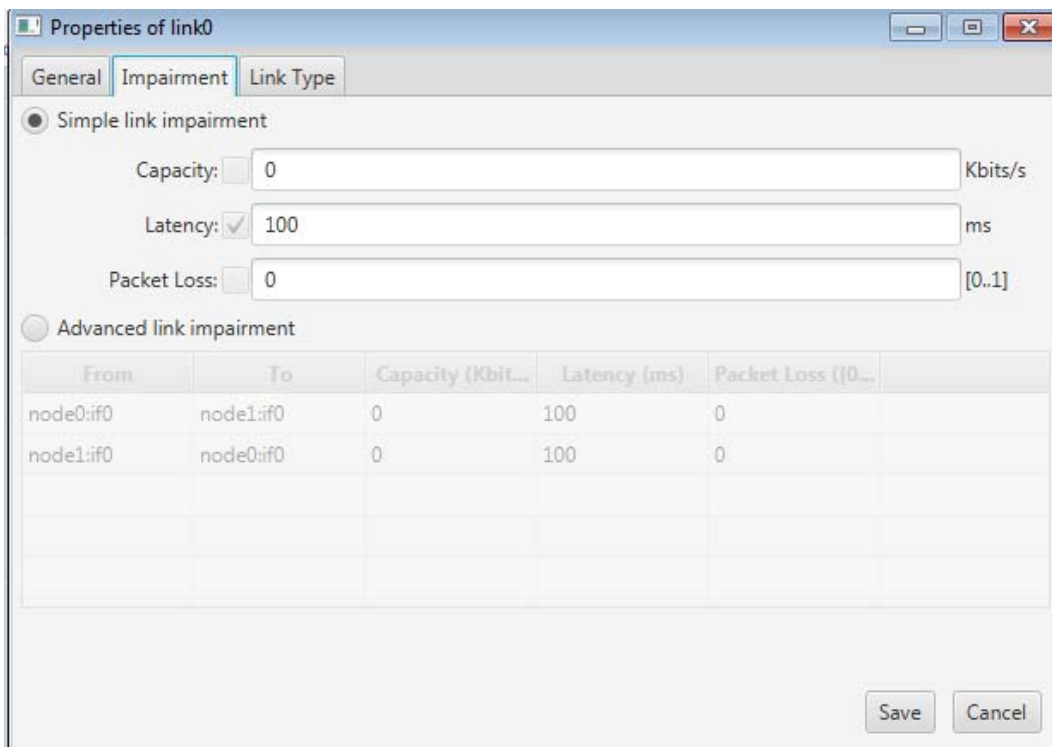
Now right click the XEN VM, and choose the rack that you have been appointed and save. You will see that the color on the right side of the node changes depending on the testbed you chose.



Now duplicate the node, by first selecting the node, and the clicking the duplicate button on the top. Then drag a link between the two nodes.

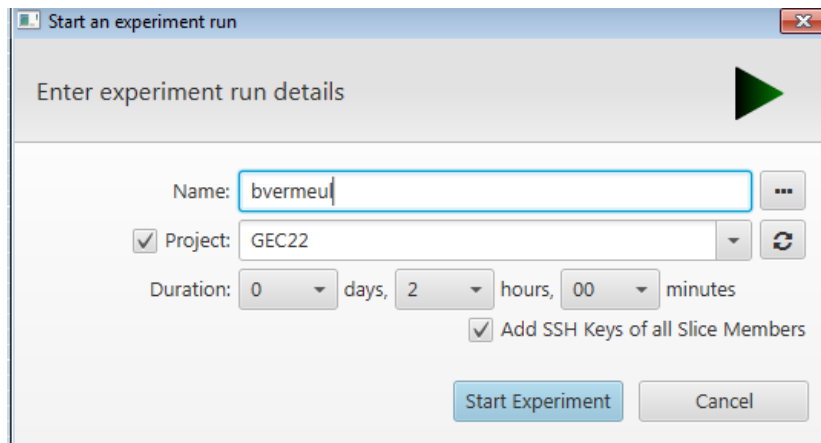


Now double click the link to configure a latency of 100ms on the link in the 'Impairment' tab.

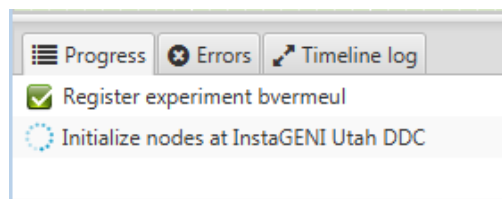


Run the experiment

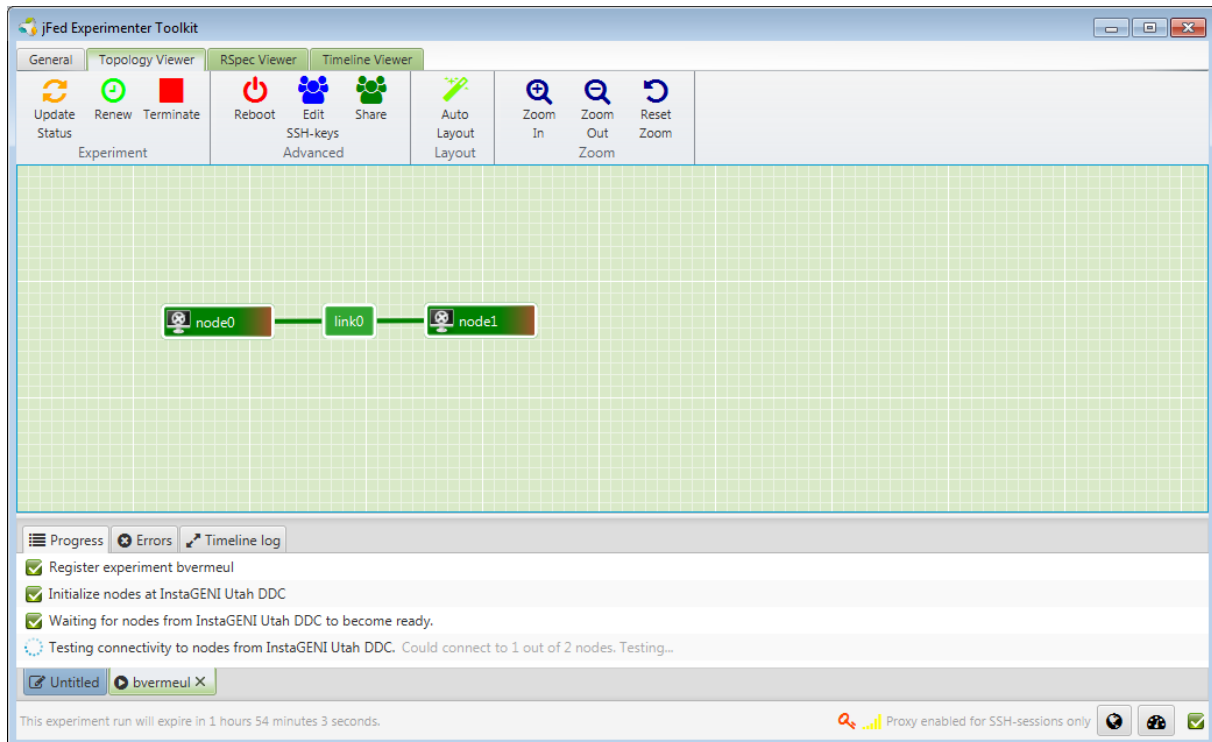
Now we will run this experiment by clicking the 'Run' button in the top bar. You will see that you need to choose a unique name for your experiment and choose the project and duration (default is 2 hours).



Click start and at the bottom you will see the progress of the experiment setup (and errors if there would appear some).



After a couple of minutes, the two VMs should be up and running, indicated by the green color of the nodes.



Login on the nodes

We now can simply login on the VMs by double clicking the nodes. Try to ping to the other node and see if the round trip delay (RTT) is 200ms (100ms one way).

```

brechtv@node0: ~
Using username "brechtv".
Authenticating with public key "imported-openssh-key" from agent
Welcome to Ubuntu 12.04 LTS (GNU/Linux 3.2.0-56-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

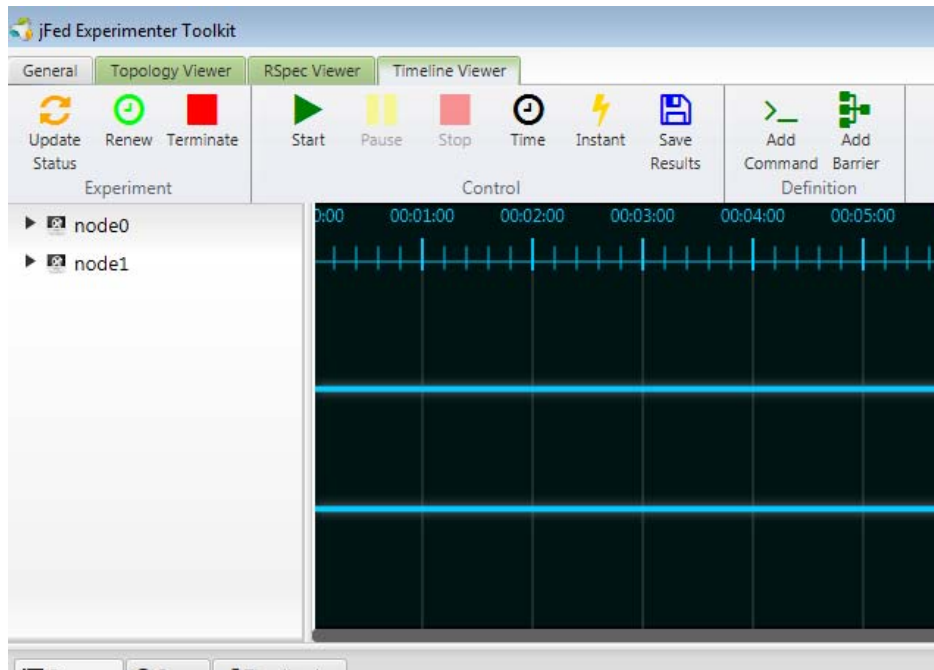
brechtv@node0:~$ ping node1
PING node1-link0 (192.168.0.2) 56(84) bytes of data:
64 bytes from node1-link0 (192.168.0.2): icmp_req=1 ttl=64 time=402 ms
64 bytes from node1-link0 (192.168.0.2): icmp_req=2 ttl=64 time=200 ms
64 bytes from node1-link0 (192.168.0.2): icmp_req=3 ttl=64 time=200 ms
64 bytes from node1-link0 (192.168.0.2): icmp_req=4 ttl=64 time=201 ms
^C
--- node1-link0 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 200.886/251.214/402.009/87.063 ms
brechtv@node0:~$

```

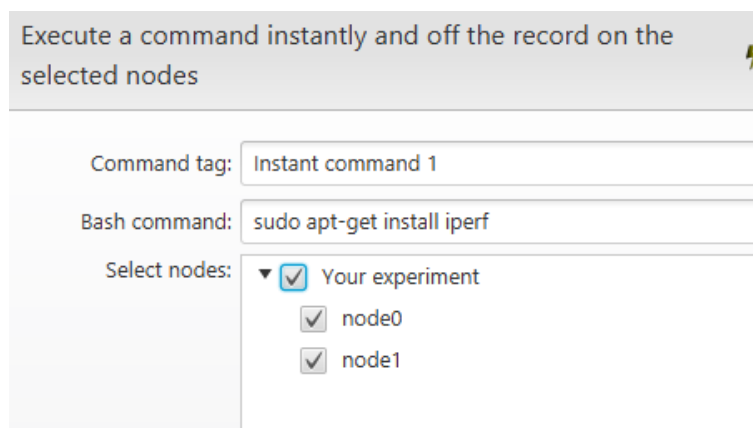
Install and run iPerf

We can install manually iperf on the nodes, but jFed has a nice feature to install easily software on multiple nodes.

Go to the 'Timeline Viewer' by clicking the tab on the top, and click 'Instant'.



Fill in 'sudo apt-get install iperf' and check both nodes and click ok at the bottom.



Now on one node, run iperf server:

```
iperf -s
```

On the other:

```
iperf -c hostname_other_machine (may depend on which node you are logged in).
```

This will start a 10 second TCP traffic flow.

Which bandwidth do you measure and what did you expect?

Learning more on your experiment and more functionalities

- Going more in detail:
 - Go back to the design mode (blue tab), and go to the rspec editor: you can also manually change things, save the RSpec, etc
 - If you right click a node and configure it, you can also select images
 - For XEN VMs you can configure RAM, extra disk, routable control IP. Also Exogeni can be selected under virtual machine and configured. (we will not start such an experiment)
 - Go to the running experiment (bottom green tab), in RSpec view and verify details on nodes, login, RSpec manifest
 - Information on options for a running node:
 - Node reload = reload the image for that node (=reformat the node)
 - Node info = detailed ssh info + interface info
 - Node reboot = simple reboot of the node
 - create image = will take an image of your node that you can use in new experiments (we won't do this now)

How to get info on your experiment on the node itself

The previous section described how to get information about nodes and your experiment from jFed. Now we will try to get this same information on the node itself (e.g. to use in scripts you want to use) We use the 'geni-get' tool for this.

Geni-get supports the following commands:

```
geni-get commands
{
  "client_id": "Return the experimenter-specified client_id for this node",
  "commands": "Show all available commands",
  "control_mac": "Show the MAC address of the control interface on this node",
  "geni_user": "Show user accounts and public keys installed on this node",
  "getversion": "Report the GetVersion output of the aggregate manager that allocated this node",
  "manifest": "Show the manifest rspec for the local aggregate sliver",
  "slice_email": "Retrieve the e-mail address from the certificate of the slice containing this node",
  "slice_urn": "Show the URN of the slice containing this node",
  "sliverstatus": "Give the current status of this sliver (AM API v2)",
  "status": "Give the current status of this sliver (AM API v3)",
  "user_email": "Show the e-mail address of this sliver's creator",
  "user_urn": "Show the URN of this sliver's creator"
}
```

You can try some, e.g.

```
geni-get slice_email
```

Now, we will demonstrate a more advanced script which parses the manifest RSpec and shows you th info on your node:

```
wget http://doc.ilabt.iminds.be/ilabt-documentation/_downloads/geni-get-info.py
chmod u+x geni-get-info.py
./geni-get-info.py
```

And you should see info on interfaces and IP addresses and so.

Experiment management

- Renew in jFed can extend the experiment lifetime (for all slivers)
- In jFed, you can Edit ssh keys on the nodes if you want to add other users ('edit ssh keys button')
- In jFed you can share the experiment with other people in your project, to make others member of the slice and be able to recover it in jFed e.g.

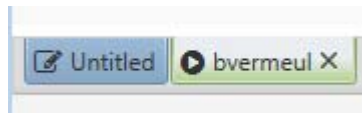
Let's break things

- Verify that eth0 is your control interface (where you are logged in)
- Turn off: 'ifconfig eth0 down', your ssh connection will be lost
- The next steps might not work for everyone, but you can try them:
 - Now, in jFed right click the node and click reboot
 - Right click and 'show console' to show you how it boots
 - After that you can access the node again

Ending the experiment

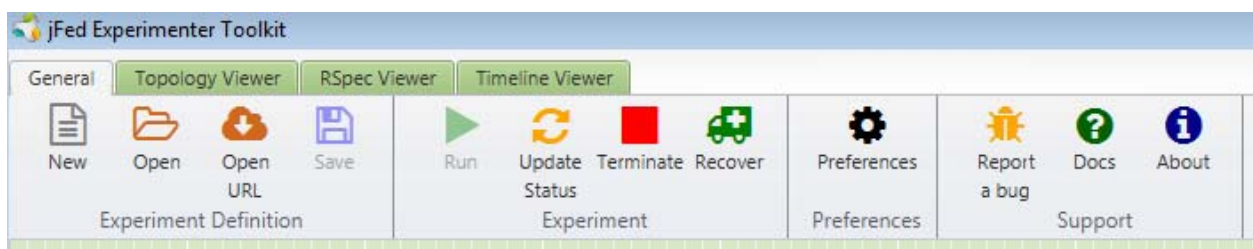
To release your resources before the endtime of your experiment, you can click the `Terminate` button at the top in jFed. After that the nodes will become black and if your ssh connection is still open, you can see that the node will be shutdown.

At the bottom of the jFed window, you see multiple tabs: the blue ones are experiment definitions, the green/red ones are experiment runs. You can click to switch tabs.



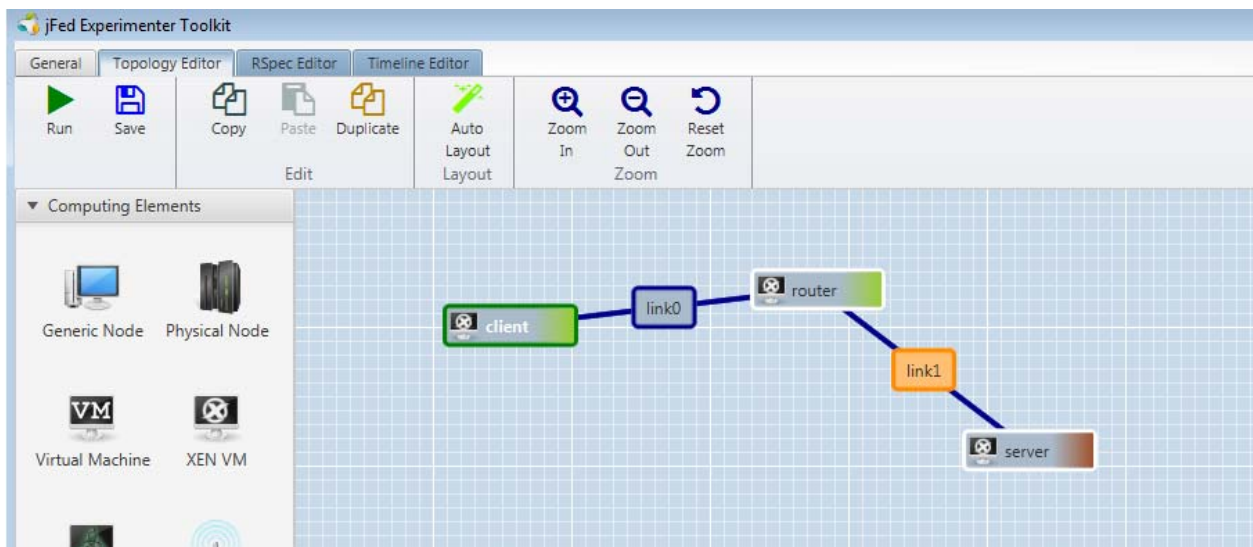
If something goes wrong?

Push the bug report button on the general tab and report what went wrong.



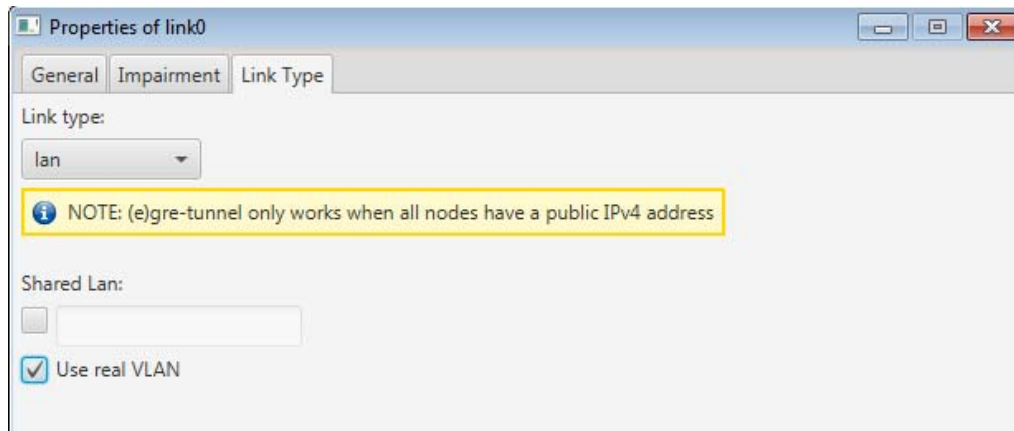
Going to a multi-testbed experiment and up-scaling

Now we will do a more advanced experiment using VMs on multiple testbeds. Design your experiment as follows:



So, we define a server in one testbed, and then a router and client on the other testbed. As you can see, you can define names for the routers (do you find how?). Verify the IP addresses that jFed has chosen by double clicking the links.

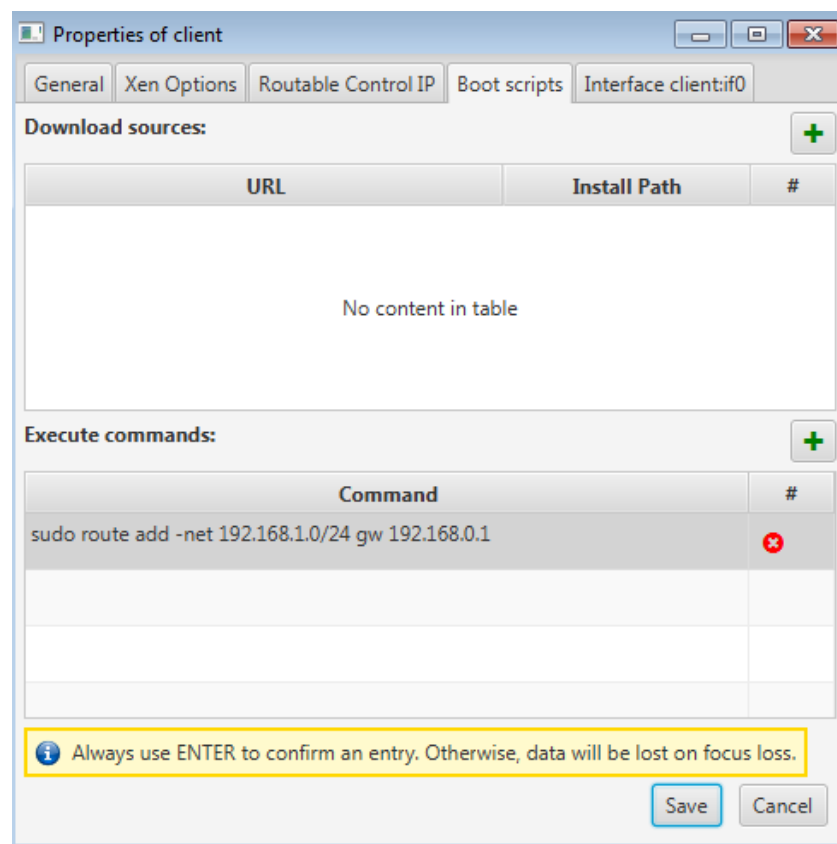
Very important for the next part of the tutorial: check for the link between client and router that in the link properties – link type, the ‘Use real vlan’ check is marked. This will force the two VMs to be installed on different physical hosts.



Now run the experiment, verify that you can ping from the server to the router (watch the latency). Configure the proper routing on client and server. This depends on your exact topology, but the commands are like this:

```
sudo route add -net 192.168.1.0/24 gw 192.168.0.1
```

In the future, you can also configure this before running your experiment, by using the node boot scripts option. Double click the node, go to the boot scripts:



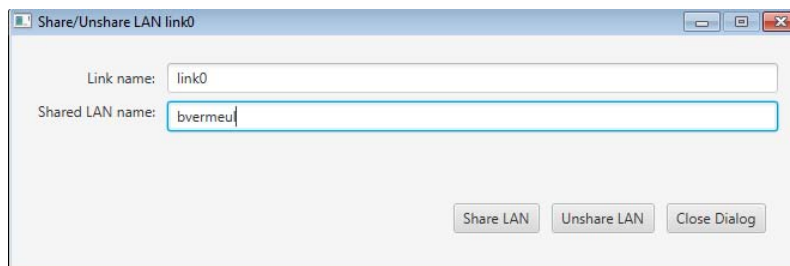
Note: Stitching bandwidth is by default set to 10Mb/s in jFed. You can change this in your experiment definition by clicking the link and changing the link impairment – bandwidth.

Upscale your experiment

Share the LAN

Now we want to add more clients. We will open up the network between client and router ('sharing the vlan') by right clicking the green link and choosing 'share lan':

You have to choose a unique name for the link (and remember it) and type it in the 'Shared LAN name' box.



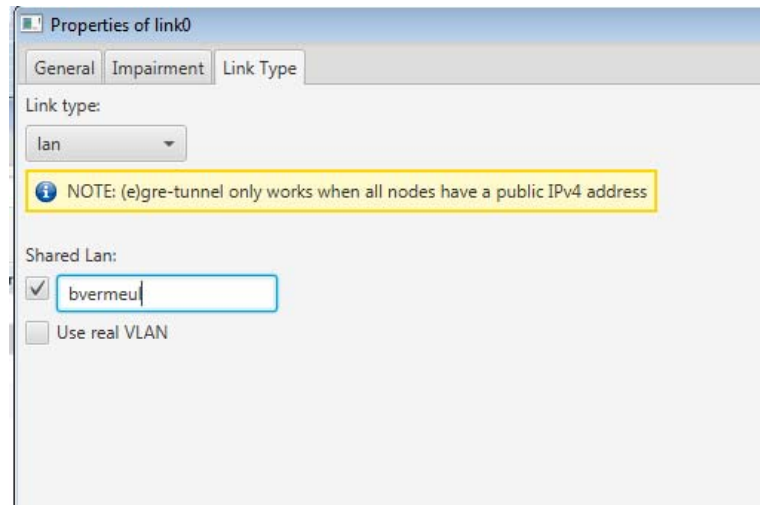
Upscale

We will create now a new experiment which will contain two nodes that will be added to the client vlan that we opened up in the previous section. **Of course, they should be on the same rack as the client and router in the previous part.**

We have to edit the IP addresses so the new nodes do not clash with the existing ones, and they should be in the same subnet. Easiest to do this, is double click the link.

Interface ID	IP Address	Netmask
client2:if0	192.168.0.3	255.255.255.0
client3:if0	192.168.0.4	255.255.255.0

To link this new network with the existing one, go to the 'Link type' tab and select and fill in the LAN unique shared lan name you used in the previous section.



Verify after login in on the nodes that you can ping the existing IP addresses. You might need to add a route to reach the server.

It should be straight forward from this tutorial that you can easily scale up further:

- Use the duplicate button in jFed
- Use bash scripting to create RSpecs
- Use geni-lib to make upscaled RSpecs

More advanced

In this last part, have a look into more advanced preferences in jFed you can configure:

- Ssh authentication: add your own key
- Proxy: in case of firewall problems or to access IPv6 nodes
- Configure SCS
- Testbed settings: exosm setting to chose the central exogeni broker or go to the specific rack controllers
- Ssh agent forwarding to login from node to node automatically: login on a node, and then ssh to another IP address in your topology