


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**

D2.7 – Third federation architecture

Work package	WP2
Task	T2.1
Due date	31/12/2013
Submission date	23/03/2015
Deliverable lead	Brecht Vermeulen (iMinds)
Version	Final
Authors	All WP2 partners
Reviewers	Javier García Lloreda (ATOS) Tim Wauters (iMinds)

Abstract	This document is the third deliverable of the architectural task of the Fed4FIRE project. It defines the architecture that will be implemented during the project's third development cycle. It refines and extends the architecture defined in the first and second development cycle, taking as input requirements from WP3 (Infrastructure community), WP4 (Service community), WP7 (Trustworthiness), WP8 (First Level Support) and task 2.3 (sustainability). Practical experience gained through the development of the architecture of the first and second development cycle has also been taken into account.
Keywords	Architecture, SFA, experiment lifecycle, federation

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)	

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Disclaimer

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

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Executive Summary

This third deliverable of task 2.1 defines the architecture for cycle 3 of the Fed4FIRE project.
It takes as input requirements of WP3 (Infrastructure community), WP4 (service community), WP8 (First Level Support), SLA management and task 2.3 (sustainability) and defines an architecture that copes with as many requirements as possible and will be implemented for cycle 3 of Fed4FIRE. It is an evolution based on the cycle 1 and cycle 2 architecture.

For cycle 3, just as in cycle 1 and 2, discovery, specification and provisioning will be done based on the GENI AM API v2 and v3 standard, while for advanced reservation and extended policy based authorization extensions are currently being made. Regarding the RSpecs, in cycle 3 each testbed can still provide its own RSpec to be used and tools have to be adapted to these RSpecs. However, the RSpecs of the existing and new testbeds are tried to be more equalized if the resource types are similar.

The architecture defines also multiple identity providers with a chain of trust, and a central portal accompanied with an identity provider and directories for tools, testbeds and certificates.

For First Level Support, just as in cycle 1 and 2 the architecture defines facility monitoring which is identical for all testbeds and makes it possible for first level support to have a high level overview of testbed and resource availability.

For monitoring, three types of monitoring have been identified and described. Facility monitoring is used to monitor a testbed as a whole and is e.g. interesting for First Level Support and experimenters to know if a testbed is functioning correctly or not. Infrastructure monitoring is monitoring information provided by the testbed itself to an experimenter (e.g. switch port statistics, spectrum measurements or virtualization infrastructure monitoring) which is normally not available to an experimenter. This infrastructure monitoring is now further specialized in infrastructure monitoring taking into account a specific experiment and thus only interesting for the experimenter (e.g. measure the switch ports only that experiment uses) and infrastructure monitoring of a whole facility (e.g. total traffic on the switch backbone of a testbed or total load of virtual machines). Experimenter measuring is related to monitoring activities that the experimenter can do on the nodes themselves.

For SLA management, the architecture has been more refined and is based on monitoring information coming from the testbeds. This makes it possible to gain experience with simple SLA evaluation based on resource availability during an experiment.

There are now defined standard ways and levels of federation: associated, light and advanced federation to make it possible to have e.g. ad-hoc federation with the light federation method.

As a general conclusion, this 3rd cycle architecture is an evolvement of cycle 1 and 2, with improvements, clarifications and additions on multiple points, but the basic ideas remain the same, which means that implementation and deployment do not have to change radically but will be further go to unified interfaces and workflows.

In order to present this deliverable as a reference architecture document, the complete architecture is described and not only the changes compared to previous architecture versions. For clarification, all updates are highlighted in the relevant sections. Besides this, standard APIs are also published as wide as possible on the internet, to enable interaction wider then Fed4FIRE and as such strengthen the sustainability roadmap. E.g. the AM API is published on github (https://github.com/open-multinet/federation-am-api) where everyone can propose changes with a nightly nice documented compile at https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html .

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Acronyms and Abbreviations

AM	Aggregate Manager
API	Application Programming Interface
CPU	Central Processing Unit
FLS	First Level Support
FRCP	Federated Resource Control Protocol
IMS	IP Multimedia System
NEPI	Network Experiment Programming Interface
NIC	Network Interface Controller
OMF	cOntrol and Management Framework
OML	Open Measurement Library
PI	Principal Investigator
RAM	Random-access Memory
REST	Representational State Transfer
RSpec	Resource Specification
SFA	Slice-based Federation Architecture
SFI	Command line SFA client (meaning of acronym is not documented)
SLA	Service Level Agreement
SOAP	Simple Object Access Protocol
SSH	Secure Shell
VPN	Virtual Private Network
XMPP	Extensible Messaging and Presence Protocol
URN	Unified Resource Name
PI	Principal Investigator
URL	Uniform Resource Locator
SOAP	Simple Object Access Protocol
REST	Representational State Transfer

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Table of Contents

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

List of figures

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

1 Introduction

The Fed4FIRE project is structured around three development cycles (Figure 1). This deliverable is the third in the architectural task of the project, and describes the architecture for the third development cycle. It is an evolution of the architecture that was defined in cycle 1 and cycle 2. Based on practical experience with that architecture, and on the updated requirements listed by WP3, WP4, WP7 and WP8, this third federation architecture includes both revisions of architectural components already supported by the first and second architecture, and also introduces some entirely new concepts. This is the final architecture deliverable for the Fed4FIRE project.

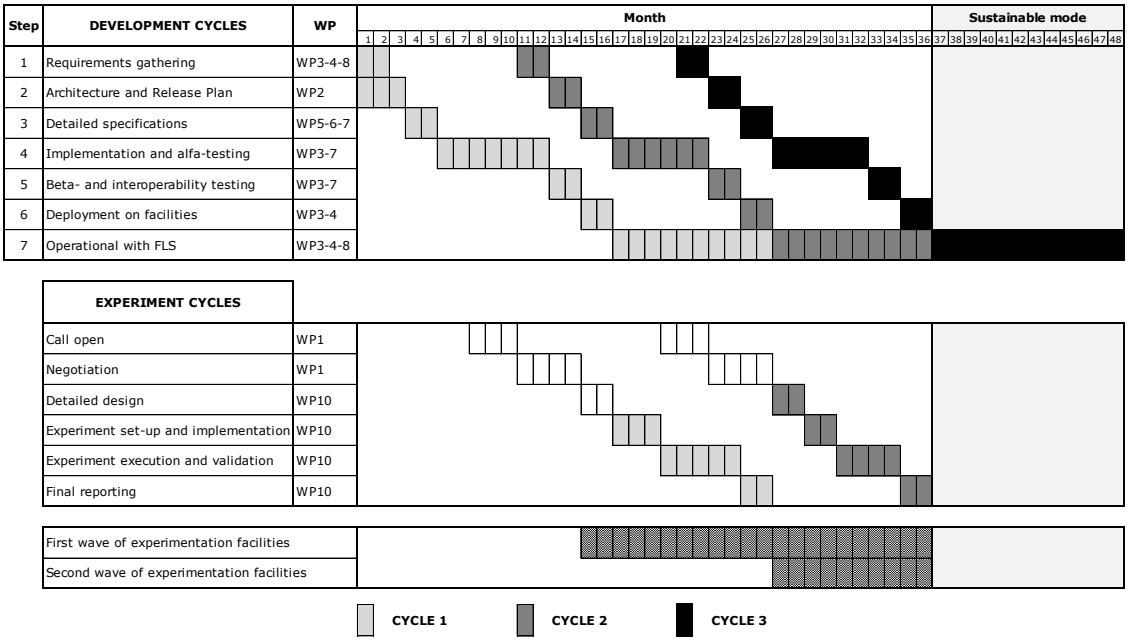


Figure 1: Overview of the three development cycles within Fed4FIRE

Whenever an experimenter performs an experiment, he or she performs a set of sequential actions that will help him or her to execute an experiment idea on a testbed and obtain the desired results. These different actions are considered to be part of what is called “the experimental lifecycle”, which can be fragmented in different functions. Because the federation architecture presented in this deliverable is closely related to these different steps of that experimental lifecycle, the Fed4FIRE definition of the experiment lifecycle is reiterated below.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Function		Description
Resource discovery		Finding available resources across all testbeds, and acquiring the necessary information to match required specifications.
Resource specification		Specification of the resources required during the experiment, including compute, network, storage and software libraries. E.g., 5 compute nodes, 100Mbps links, specific network topology, 1 TB storage node, 1 IMS server, 5 measurement agents.
Resource reservation		Allocation of a time slot in which exclusive access and control of particular resources is granted.
Resource provisioning	Direct (API)	Instantiation of specific resources directly through the testbed API, responsibility of the experimenter to select individual resources.
	Orchestrated	Instantiation of resources through a functional component, which automatically chooses resources that best fit the experimenter's requirements. E.g., the experimenter requests 10 dual-core machines with video screens and 5 temperature sensors.
Experiment control		Control of the testbed resources and experimenter scripts during experiment execution. This could be predefined interactions and commands to be executed on resources (events at startup or during experiment workflow). Examples are: startup or shutdown of compute nodes, change in wireless transmission frequency, instantiation of software components during the experiment and breaking a link at a certain time in the experiment. Real-time interactions that depend on unpredictable events during the execution of the experiment are also considered.
Monitoring	Facility monitoring	Instrumentation of resources to supervise the behavior and performance of testbeds, allowing system administrators or first level support operators to verify that testbeds are performing correctly.
	Infrastructure monitoring	Instrumentation by the testbed itself of resources to collect data on the behavior and performance of services, technologies, and protocols. This allows the experimenter to obtain monitoring information about the used resources that the experimenter could not collect himself. An example of such infrastructure monitoring is the provisioning by the testbed of information regarding the CPU load and NIC congestion on the physical host of a virtual machine resource. The experimenter can only collect monitoring data on the level of the VM, but the testbed provides infrastructure monitoring capabilities that make this data available to the experimenter. This infrastructure monitoring is now further specialized in infrastructure monitoring taking into account a specific experiment and thus only interesting for the experimenter (e.g. measure the switch ports only that

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Function		Description
		experimenter uses) and infrastructure monitoring of a whole facility (e.g. total traffic on the switch backbone of a testbed or total load of virtual machines).
Measuring	Experiment measuring	Collection of experimental data generated by frameworks or services that the experimenter can deploy on its own.
Permanent storage		Storage of experiment related information beyond the experiment lifetime, such as experiment description, disk images and measurements.
Resource release		Release of experiment resources after deletion or expiration the experiment.

This remainder of this deliverable is structured as follows:

- Section 2 describes the inputs and requirements that lead to this architecture definition of cycle 3. The inputs and requirements come from:
  - WP2 (D2.1 and D2.4): the first and second federation architecture [1], [15]
  - WP3 (D3.2 and D3.3-D4.4): Infrastructure community [2], [16]
  - WP4 (D4.2 and D3.3-D4.4): Services community [3], [16]
  - WP8 (D8.4 and D8.7): First Level Support [4], [17]
  - WP7: SLA management
  - WP2 (D2.3 and D2.6): Sustainability [5], [18]
- Section 3 describes the architecture itself, together with basic concepts regarding the Fed4FIRE context. It is described in 4 parts:
  - Resource discovery, resource reservation and resource provisioning
  - Monitoring and measurement
  - Experiment control
  - SLA management
- Section 4 describes the differences between the cycle 1, cycle 2 and cycle 3 architecture.
- Section 5 touches back on the requirements to evaluate how many requirements are tackled by the architecture.
- Section 6 concludes the deliverable
- The following appendices are attached:
  - Appendix A: Requirements from the infrastructures, services and applications communities (D3.4/D4.4)
  - Appendix B Requirements from SLA management
  - Appendix C: Interaction between Policy Decision Point and Aggregate Manager
  - Appendix D: What information does a user credential and slice credential contain
  - Appendix E: Subauthorities in a member and slice authority
  - Appendix F: Evaluation of possible architectural approaches for SLA management
  - Appendix G: Workflow for registering an account and getting a certificate
  - Appendix H: Workflow for creating an SSH key on different operating systems
  - Appendix I: Run a First Experiment
  - Appendix J: Enable apache to authorize SSL certificates

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

2 Overview of inputs

This section will describe all inputs leading to the architectural choice for cycle 3. As stated before, these are the architectural design of the first and second cycle, the additional requirements defined after cycle 2 in WP3, WP4, WP7 and WP8 and the project's task on sustainability.

2.1 First and second federation architecture

When defining the first federation architecture in D2.1 [1], the project evaluated 4 different types of architectures. Based on that evaluation, the architecture for cycle 1 was defined for the different steps in the experiment lifecycle. For cycle 1 it was decided that resource discovery, specification and provisioning would be supported based on the SFA GENI AM API v3 standard, while for advanced reservation and extended policy based authorization extensions would be foreseen. Regarding the resource specification, the aim was to get them as unified as possible in cycle 1 (with GENI RSpec v3 as a guideline), while for cycles 2 and 3 a unified ontology based RSpec definition was identified as the research target.

That architecture also defined multiple identity providers with a chain of trust, and a central portal accompanied with an identity provider and directories (machine and human readable) for tools, testbeds and certificates. These services function as 'broker services'.

For First Level Support, the architecture defined facility monitoring which should be identical for all testbeds and should make it possible for first level support to have a high level overview of testbed and resource availability. Infrastructure monitoring, experiment measurement and experiment control were not yet standardized within WP2's federation architecture for cycle 1. This was however perceived not to be a blocking issue, as the experimenters can deploy their own tools for this and they are not dependent on the testbed providers. In the consequent development step, being the establishment of more detailed specifications by the vertical work packages 5, 6 and 7 (responsible for the development of the federation framework), it was then decided that at least the Federated Resource Control Protocol (FRCP) API was to be adopted for experiment control, and OML for experiment measurement.

The first cycle federation architecture was designed to cover as many requirements as possible. However, since it was not feasible to cover all of them, there were several high priority requirements that were reserved for later cycles. From these, most important non-tackled requirements were considered to be the ones regarding inter-connectivity between testbeds and storage of all kinds of information. It was concluded that both of them would be tackled in cycles 2 and 3 of Fed4FIRE. Some other requirements were identified as being only partly resolved, but were expected to be fully resolved when the ontology based RSpecs/resource descriptions will be deployed in cycles 2 and 3. These specific conclusions should be taken into account when designing the architecture for cycle 3.

The general conclusion of D2.1 and D2.4 was however that the first cycle architecture and way forward already coped with a large number of the requirements that had been set by the other work packages. The practical experience gained with the development of this first and second federation architecture confirmed the statement above: today it is still considered to be a solid basis for the Fed4FIRE federation, and it is not needed to profoundly restructure it. Only refinements, clarifications and additions of some specific new functionality are needed.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

2.2 Requirements from the infrastructures, services and applications community (D3.4/D4.4)

The purpose of the document "D3.4/D4.4 Third input from community to architecture" was to report the results of an exercise that has gathered the third set of requirements from both the Infrastructure and the Services and Applications communities' perspective in order to finalise the construction of federation of FIRE facilities.

Requirements have been collected through surveys, reports and proposal documents, from the following sources:

- ongoing experiments coming from the 1st Fed4FIRE Open Call – 8 experiments involving WP3 and WP4 testbeds
- proposals presented to the 2nd Open Call – 11 proposals from Industry and 8 from Academia
- proposals presented to the 1st SME Open Call – 4 proposals in total involving WP3 and WP4 testbeds .

The survey scorings highlight some of the most important requirements for our community such as the need for uniformity of resource information for resources belonging to different testbeds as far as their presentation is concerned, the availability of orchestration tools to ease the interaction with the federation or the possibility to have detailed monitoring and eventual trouble information (events) during experimentation. Important insights have been found concerning how experimenters experience the interaction with the federation. For example, they seem to greatly appreciate support and guidance when setting up the experiments as well as available documentation and they would like the jFed tool to be extended with further functionality.

Most of the requirements are originating from the first two cycles and their importance has been confirmed in the recent surveys. Appendix A provides an overview of all requirements, collected until now.
They can be summarized as follows:

- Fed4FIRE must provide a clear view on what **node capabilities** are available, and this should be defined in the same way across the federation. It should be possible for the experimenter to **select the resources** he/she would like to include in the experiment.
- Resource discovery must be integrated into uniform tools through **federation-wide APIs**.
- For nodes that have wired and/or wireless network connections to other nodes within the same testbed, it should be possible to identify the physical **topology.** It should also be known how different infrastructures are/can be **interconnected** (e.g. via layer 3 or layer 2).
- Fed4FIRE must provide **hard reservations** of the available resources in a fair, secure and fully automated manner.
- APIs are required to enable **direct instantiation** of both physical and virtualized resources for experiments.
- Fed4Fire must provide the ability to **install a specific custom Linux kernel or distribution on the nodes.** Fed4FIRE must provide the possibility to access a node as root user. Software installation through a packet manager (e.g., apt-get) must be possible. It should be possible to create a binary image of the entire hard disk drive once a node has been fully installed.
- It should be possible to already define what the specific resources should do at startup (install additional software, copy specific files, start specific daemons/tools).
- **Nodes must be accessible via SSH.**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- It must be possible to describe **advanced experiment scenarios by the use of a script that will be executed by a control engine**. This engine should be general enough to support the control of all possible kinds of Future Internet technology.
- Fed4FIRE must provide an **easy way for experimenters to store measures during the experiment runtime for later analysis.**
- **Common characteristics should be stored automatically** during an experiment (CPU load, free RAM, Tx/Rx errors, wireless interference, etc.)
- **Monitoring info regarding the state of the resources should be opened up to all experimenters**, and not only those that were using the resources. This way they can choose the best resources for their experiments.
- **As less overhead as possible should be expected from the monitoring and measurement support frameworks.**
- The user must be able to request **on-demand measurements**. In order to do so, they will need to express that they want agents with such on-demand polling capacities.
- **Access to the data should be properly secured.**
- Experiment configurations should be stored in order to **replay experiments** and compare results of different runs.
- Fed4FIRE must provide the mean of accessing all testbeds within the federation using **one single account** (username/password). Fed4FIRE should also provide authentication by the use of public SSH keys. Access to the Fed4FIRE APIs (discovery, reservation, provisioning, etc.) should also be protected by an authentication mechanism.
- It should be **easy for new experimenters without any affiliation to the federation to create their Fed4FIRE identity.**
- The resources within a Fed4FIRE infrastructure should be able to reach the resources deployed in the other Fed4FIRE infrastructures through a layer 3 Internet connection. **Interconnectivity solutions** should not introduce unneeded complexity in the experiment such as VPN or other tunnels. The ability to conduct **IPv6** measurements and to interact with the nodes of other testbeds over IPv6 should be enabled.
- Per experiment, Fed4FIRE should provide the possibility to r**eserve bandwidth on the links that interconnect specific infrastructures**.

The novel requirements that cannot be mapped entirely on the existing requirements are the following:

- **Monitoring information about network connections** should be available at run-time**.**
- **Remote access to shared data** should be available**.**
- **Orchestration of experiments** between testbeds should be provided in order to trigger specific services or events.
- Fed4FIRE should provide a **standardised way of registering and unregistering external devices** (any device that is not part of the testbed itself, but that is to be included in the experiment, such as dedicated smartphone, tablet, novel product, etc.)
- During an experiment, the experimenter might need to decide if and how to change the amount of computing or network resources allocated to a service.
- **Resources must be described in a homogeneous manner** so that the experimenter can compare the experiments in different testbeds.
- The experimenter should have multiple **resource reservation** options.
- Some **features** could be **executed automatically**. For example, a reserved experiment could start automatically.
- Multiple testbeds of different kinds should be federated together in one experiment through a **set of common tools.**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- **Fed4FIRE tools should be user friendly to the experimenter**.
- It should be possible to **pack up resources in services** offered to the experimenter so that he/she is isolated from the real infrastructure.
- Fed4FIRE must **provide the means for experimenters and third parties to develop and/or deploy applications on top of Fed4FIRE infrastructure.**
- **Experimenters should be able to manage resources they have created** (share/unshare/destroy).
- Fed4FIRE must provide tools to create, view, update, and terminate **monitoring configurations** related to shared resource types or experiments in real time.
- Fed4FIRE should enable the experimenter accessing all resources (hardware and software) with own privileges by performing login only once at the start of the experiment (**single sign on**).
- Fed4FIRE will make the distinction between requests of local users, PhD students from other institutes (research), students (practical exercises), in order to know what kind of experimenter is logging in and from where and apply **policies** accordingly.
- **Interconnections between testbeds** should be in place (WAN links/public Internet access) so that during an experiment, experimenters can invoke services inside and outside a testbed, moving data between testbeds without it going through his personal machine.

2.3 Requirements from First Level Support (D8.7)

First Level Support (FLS) is one of the central functions of the Fed4FIRE federation. It provides a common facility for the logging and resolution of faults. It provides a direct line of communications with experimenters and with the support functions in the individual testbeds. Deliverable "D8.7 Third input to WP2 concerning first level support" [4] considered how the architectural requirements of FLS have been met and described the processes that FLS uses in the operational phase of Fed4FIRE. It considered how, in a live operational environment within the federation, operational support is likely to function. Feedback has been gathered based on the operations of the federation in the previous cycle. This feedback has led to six specific requirements on the project's federation architecture. They are reiterated below.

Process Automation

Currently, FLS resolution processes are initiated by an FLS operator observing a change in dashboard status and creating an appropriate trouble ticket. This approach has the merit that each ticket creation is analysed by an experienced operator. However, the existing dashboard design can potentially lead to missed problems. It is also the case that most of the ticket creation process could be automated. This would still permit operator inspection but, by automating the process, the manual aspects of dashboard observation and ticket creation, which add no significant value, could be eliminated. Two developments would be necessary to automate this process. Firstly a standardized approach to the definition of significant events is needed. An operator will exercise judgment as to what category an event belongs to. This needs to be reflected in a set of rules which would categorise events automatically. Secondly, in order to process events consistently, a common vocabulary to describe such events and also the context in which they occur (time, location, operational state etc.) is required. There is a general open standard for network management which could be applied here, Simple

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Network Management Protocol (SNMP).[1] Since SNMP is widely used in management of distributed ICT systems it is well understood and integrates well with existing tools

Requirement 1: Provide an SNMP trap capability to automate the creation of FLS trouble tickets
Requirement 2: Develop and implement a standardised operational vocabulary to allow the automation of elements of FLS.

Network Testing

The current approach to test-bed reachability testing, associated with fed4FIRE, is based on using a ping test to determine the network reachability of a test-bed. This is reported as a ping latency figure in the dashboard. The actual latency is not of huge significance. Changes to it might be considered relevant but this is not tracked. Assuming that the federation only uses public Internet connectivity for interconnecting test-beds in the federation, this approach represents a reasonable proxy for connectivity testing. Recently, a more sophisticated approach to the network interconnection of some of the federated test-beds has been introduced. This is based on the use of VLANs, set up over dedicated capacity configured between the test-beds, as well as the use of the GÉANT Autobahn service. Autobahn is a Bandwidth on Demand (BoD) service offered by GÉANT, which enables users to request dedicated capacity only when it is required.

Requirement 3: Develop a mechanism for monitoring semi-dedicated connectivity, which is part of the federation, and integrate this information into the FLS dashboard.

Enhanced Reporting

The dashboard provides a summary, quasi real-time, view of the status of the test-beds which are part of the Fed4FIRE federation. The FLS activity, derived from the dashboard state and captured in the Trouble Ticket System (TTS), provides some basis for statistical analysis of the performance of Fed4FIRE. This activity is reported in a monthly report, which also includes trend analysis. This approach to reporting can be described as 'supply led.' The main issue with it is that it is not at all related to experimenters' usage of the federation. Operational support and service performance analysis should ideally be targeted at the service as experienced by users (demand led). There is, therefore, an unfulfilled need to relate the federation performance as monitored by FLS with the performance as experienced by experimenters.

Requirement 4: Create a database of live experiments using the federation to enable FLS to correlate issues detected with experimenters' activity and improve the operational reporting of the federation.
Requirement 5: Develop reporting to include a demand led approach to federation performance.

Pervasive Comment Capability on Dashboard.

In order to provide extended hours coverage and to ensure continuity of service, FLS is provided by a team of operators. Thus, a single incident will be handled, typically, by several people. Although the TTS provides a formal repository for information relating to incidents, the dashboard is the principal tool used by FLS operators. As noted above the dashboard refreshes regularly, and previous information is overwritten. There is a need to have a comment capability associated with the dashboard to allow operators to exchange informal comments associated with the status of a test-

[1] SNMP is defined in RFC 1157

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

bed. These include points such as hand-over notes, informal observations etc. They are essentially of a semi-transitory nature. Whilst is would be possible to use the TTS for this function, it is designed to provide a formal, process-driven record of operational activity. The dashboard is the principle shared tool for FLS operators. It is proposed that the dashboard be enhanced to provide a pervasive comment capable with write access for FLS operators and read access for others. It needs to be pervasive to prevent it being refreshed every time the dashboard is updated.

Requirement 6: Provide an additional dashboard field per test-bed to allow FLS operators to comment on activity related to a test-bed.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

2.4 Requirements from SLA management

Service Level Agreements (SLA) represent a contractual relationship between a service consumer and a service provider. In the case of Fed4FIRE, this relationship needs to be established between the experimenter and the testbeds. The need for SLAs has been touched in the requirement deliverables D3.4 and D4.4 – Third input from community to architecture [16] of WP3 and WP4, but the corresponding implications on the architecture have not been studied there in a profound level of detail. Therefore WP7 decided to further investigate these SLA related architectural requirements more profoundly. Since there is no specific deliverable planned in WP7 regarding architectural requirements, it was decided to present the results of that WP7 study in this section of the architectural deliverable.

SLAs are used when capability is outsourced to a 3rd party provider as a mechanism to increase trust in providers by encoding dependability commitments and ensuring the level of Quality of Service is maintained to an acceptable level. SLA management also provides information for later accounting, depending on the terms and conditions gathered in the SLA and on whether this SLA has been met by all parties or not. For example, this can determine that a certain penalty or reward is applied to one of the parties. SLAs describe the service that is delivered, its properties and the obligations of each party involved. Moreover, SLAs establish that in case the guarantee is fulfilled or violated, rewards or penalties –monetary or not– can be applied, respectively.

The requirements regarding SLA for Cycle 3 have been obtained from different sources:

- **Cycle 2 requirements** have been reviewed and updated according to the trajectory and philosophy followed by Fed4FIRE (e.g. aggregation of SLAs is not required, fees are not supported in the federation)
- The second **review results** in which recommendations about the openness of the federation, agility for testbed federation process and the added value for experimenters it provides were made.
- **Experience with Fed4FIRE portal interaction** through the developed SLA plugin in the SLA lifecycle.
- **Survey** sent to federation testbeds to know their plans regarding SLA adoption (for those which do not have SLAs) and extension (for those which already offer SLA) for Cycle 3. Further analysis of the SLA survey is described below.

In order to know the plans regarding SLAs for each testbed within Fed4FIRE, a survey was sent including questions such as the intention to adopt SLA and what type of SLAs (e.g. new metrics beyond availability) would be offered, the possible penalties or rewards to be applied on the SLA result and the monitoring capabilities needed for SLA evaluation.
The answers provided by the testbeds have been gathered and analyzed, leading to two conclusions.

- **Testbeds without SLA**: Those testbeds that do not offer SLAs in cycle 2 do not have plans to adopt them in cycle 3. Resource reservation is provided with best effort with no guarantees and SLAs are not considered as a priority for the next cycle. The exception to this group is the Ofelia testbed which answered that the adoption of SLAs in cycle 3 was under study.

- **Testbeds with SLA**: Testbeds that already offer SLAs will improve the resource monitoring (that can lead to the offering of more complex SLAs) and plan to give SLA results an impact either on the testbed (i.e. by means of reputation) or on the experimenter (i.e. by providing

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

compensation quota). However, resource reservation will be offered in a best effort manner, SLAs guaranteeing their availability during the experiment.

This is explained in more detail in Appendix B Requirements from SLA management. They can be summarized as follows:

- It should be possibly to classify experimenters into **different profiles**, and offer them specific SLAs based on that profile (possibly for a fee).
- The SLA management system should help testbeds retrieve intelligent information concerning the **testbed usage**. It should provide the means for the testbed to expose a certain SLA according to the available capacity at a given moment.
- The **SLA management should be aware of reservations** for those testbeds who operate on this basis.
- When an experiment goes wrong (time out, etc.), it is essential to try to distinguish whether the problem is originated in the infrastructure (**SLA not met**) or if this is caused by the experimenter's own software or data-set used during the experiment. This is not always clear, but some cases are clear (e.g. if the physical host crashes running a number of virtual machines, then we know that all VMs running on that host are impacted)
- Fed4FIRE must provide the means for infrastructure providers to **publish offerings**, experimenter requirements in terms of QoS and browse/compare offerings. There should be a mechanism for a **negotiation** to agree SLA conditions between the experimenter and each provider.
- Once the user begins the experiment, the SLA management should be able to compare each individual SLA with the monitoring of every testbed involved. In case the SLA is not met, the **SLA management system must be able to clearly identify and make visible which testbed(s) did and did not commit.**
- In case experimenters do not meet their obligations, the SLA management must facilitate the claim for some compensation by all affected testbeds who require so. The other way around, users should also be **informed of SLA violations**.
- Fed4FIRE must provide the means for every experimenter and testbed provider to be aware and **sign the terms and conditions** of the federation
- **Monitoring information** should be provided by testbeds following the **same representation pattern and terminology** so that SLA management can make use of it to evaluate the SLAs.
- The **SLA management should provide information to the central reputation component** in Fed4FIRE.
- In Fed4FIRE, there are low level resource services (e.g. testbeds offer raw resources) but also high-level application services (that run over the resources). Fed4FIRE must be able to deal with SLAs for both kinds of services.
- The SLA management in Fed4FIRE should be **open and based on market standards**

2.5 Requirements from a sustainability point of view (D2.6)

In this phase of the project, Task 2.3 regarding sustainability has delivered its second sustainability plan (D2.6)[18]. This document focused on the different “options” that the federation can choose from in order to make a business plan: value offering towards its stakeholders (experimenters and facilities) and the potential service components, including a first estimation of cost and benefit, that could be supported. A first plausible federator scenario is analyzed and evaluated.
Following key contributions have been made:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- *Identification of the value proposition*. We describe how we identified the value proposition of Fed4FIRE and how this impacts on the continuing and changing needs of experimental users and testbeds. And, in turn, how long term sustainability must plan for such changes.
- *What is the service offering*? Services provided by the federation come with a cost; such costs add to the challenge of sustainability. We propose the key services that are required in the long term and examine the appropriate federation and business models that underpin their delivery.
- *Evaluation of a potential federation scenario.* Based upon the requirements from testbeds and experimenters we make a selection of the most suitable service components and calculate the costs and benefits for facilitator and testbed.

Input has been provided to the architecture in order to take into account technical constraints that are derived from sustainability requirements. Examples are requirement to have an open architecture that is capable of supporting different experimenter communities, being able to reuse existing components, allowing easy adaptation and enhancing existing infrastructures, providing building blocks for new research infrastructures, allowing easy access to the facilities for the experimenters, providing support for facility management by the facility owners, etc.
Task T2.3 could confirm that the sustainability considerations defined in the project's first and second architecture versions are still valid today.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

3 Architecture for Fed4FIRE development cycle 3

3.1 Introduction

The previous section provided an overview of the requirements and constraints imposed on the design of the Fed4FIRE federation architecture for cycle 3. One of the identified basic principles is that the Fed4FIRE architecture should be able to cope with heterogeneous testbed software frameworks, with a focus on adopting the same (standardized) federating interfaces on top of the existing frameworks. This way tools can work with multiple testbeds, orchestration engines should only cope with a single type of interface, and user accounts can be shared over the testbeds. These basic principles were already answered by the architecture of cycle 1, were strengthened even more in cycle 2 and are now confirmed in cycle 3.

To tackle these requirements, and the many other inputs specified in section 2, the federation architecture that is defined in this deliverable encompasses a rather large amount of architectural components. Therefore the detailed architecture discussion has been split up in multiple parts:

- Legend of the architectural diagrams
- Resource discovery, resource reservation and resource provisioning
- Monitoring and measurement
- Experiment control
- SLA management

Before diving into the detail of the second Fed4FIRE federation architecture, a last remark that should be made is that during the design process, careful consideration was given to possible alignment with the work in GENI (http://www.geni.net) in the US. As a result, the third Fed4FIRE federation architecture does not only meet the requirements and other inputs defined by the project in section 2, but it is also interoperable with GENI and many others that have adopted the same framework (Korea, China, Brazil, Japan).

3.2 Legend of the architectural diagrams

Although the architecture is presented in multiple parts, we intend to make these different parts as uniform as possible. As will become clear later in this chapter, this was achieved by visualizing the different architectural parts in the same manner. This section gives some more details about the different elements that are always depicted on the diagrams.

3.2.1 Concept of federations and actors

Before introducing the diagrams of the project's federation architecture, it is important to clarify what we exactly understand by the concept of federation:

- In a federation, testbeds, services, experimenters and authorities have a common trust. E.g. testbeds trust the authorities in the federation (they trust that the certificates which are signed by these authorities contain correct information and thus identity).
- However, this says nothing about policies (who can do what in the federation), only that we trust the entities in the federation.
- A testbed, service, authority or experimenter can be part of multiple federations.
- A federation can be as small as two single testbeds who have a mutual agreement.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- Some federations can be rather short lived, so it should be easy to set them up and tear them down.

Some examples of federations that are in line with the above concept are of course the Fed4FIRE federation, but also the Planetlab federation (Planetlab Europe and Planetlab Central), the Emulab federation (federation based on all testbeds with the same control framework Emulab) and the GENI federation (US testbeds, experimenters and authorities in the GENI project). The existence of these (and other) testbed federations clearly illustrates the need for allowing testbeds, services, authorities and experimenters to be part of multiple federations. E.g. the Planetlab Europe testbed is among others part of the Fed4FIRE and Planetlab federation. The Virtual Wall testbed is among others part of the Fed4FIRE and Emulab federation.

Taking the above interpretation of a federated environment into account, a specific format to depict the project's second and third federation architecture could be established. In every architectural diagram, the same columns are shown vertically (see Figure 2). Each of them corresponds with a specific actor in the federation:

- **Actors inside the federation**:
  - **Testbeds** with **resources** (=nodes, e.g. a server, a WiFi node, etc)
  - **Experimenters** who belong to the federation if they have credentials which are acknowledged by the federation partners
  - **Federator**: central components which ease the federation, e.g. with directory services. It has to be stressed that in the Fed4FIRE approach these federation components are NOT strictly necessary for operating (where operation means experimenters which do experiments on testbeds). These central components are just there for the convenience of both the testbeds and the experimenters.
- **Actors outside the federation**: the architectural design takes into consideration that there are testbeds, experimenters and authorities also outside of the federation. Testbeds inside the federation can choose to trust some of them if they want. This means that Fed4FIRE testbeds have the freedom to grant access to their testbeds to such actors outside of the federation, or they can join additional federations (giving their local userbase the opportunity to use these additional resources outside of the federation).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

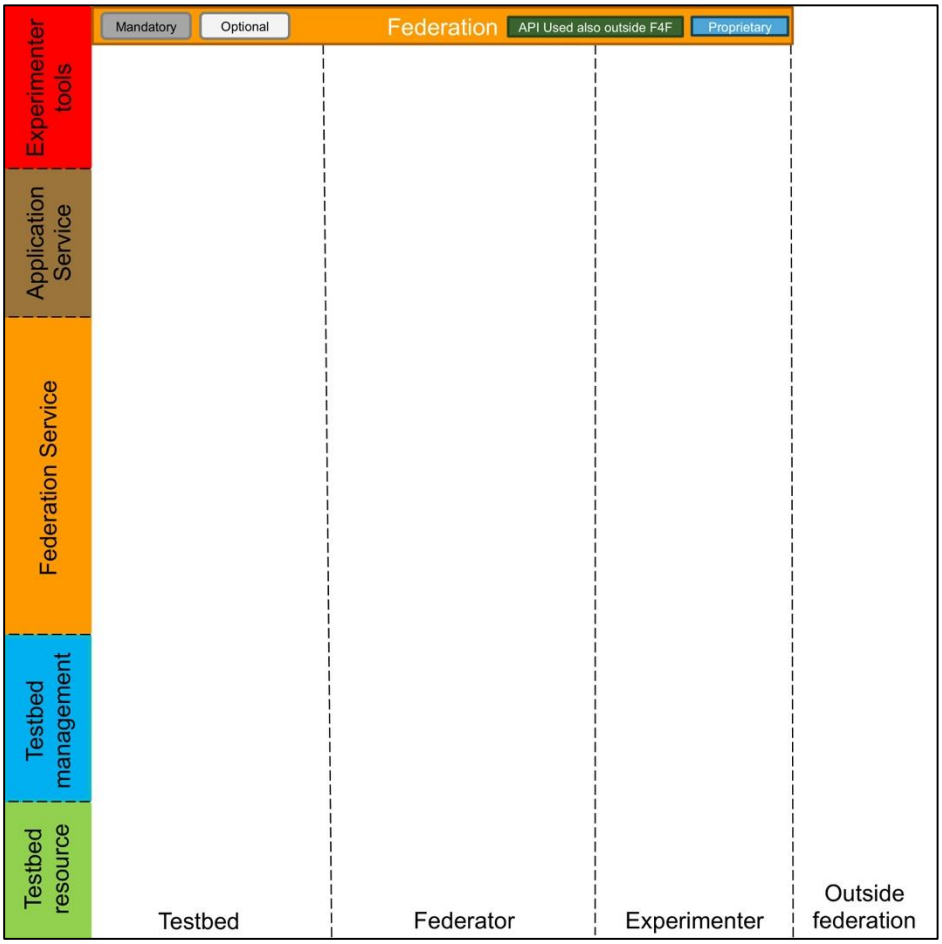


Figure 2: common format of the architectural diagrams

3.2.2 Layers in the architecture

In every architectural diagram, the same horizontal layers are defined (see Figure 2):

- **Testbed resources**: these are the physical and virtual nodes/resources of the testbeds
- **Testbed management**: this is the software framework which manages the resources of the testbed
- **Federation Services**: these are services that enable the federation to operate (e.g. authority services) or that make a federation easier to use (e.g. directory or broker services, documentation, portal, etc.). This is distinct from Application Services. Depending on what it actually is, a Federation Service may provide benefits to testbeds, experimenters, or both.
- **Application Services**: these are services that can be used by experimenters during their experiment (e.g. a service which returns sensor measurements in case of the SmartSantander testbed, or a service that can create a Hadoop cluster on nodes of the Virtual Wall automatically, instead of having the experimenter reserve Virtual Wall resources and then install such a cluster himself). Application services tend to abstract the underlying technical details of the provided services, and only provide direct benefit to an experimenter.
- **Experimenter tools**: this layer contains all tools the experimenters use

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

3.2.3 Colors in the architecture

Figure 3: Enlarged color legend of the architectural diagrams

As can be seen above, the architecture figures contain multiple colors:

- Grey (Mandatory): a component which is mandatory
- Dotted (Optional): a component which is optional
- Green: an API which on the server side is also used outside of Fed4FIRE or which is standardized. This means that an API can only be green if a service exists outside of the Fed4FIRE context that implemented exactly the same API.
- Blue: Proprietary API, or e.g. F4F API which means that there was an agreement in Fed4FIRE about this API interface.

3.3 Resource discovery, specification, reservation and provisioning

The first part of the architecture that is presented here aggregates all architectural components needed to support the following first steps of the experimental lifecycle: resource discovery, resource specification, resource reservation and resource provisioning. The architecture itself is depicted in Figure 4. The remainder of this section will first introduce all the corresponding components, and will then go into more details regarding the aspects for which additional information is required (SFA interfaces, mechanisms for authentication and authorization, the new layer for application services).

3.3.1 Introduction of the architectural components

3.3.1.1 Federator

From the perspective of the **Federator**, several components will be provided in one or more central locations for resource discovery, resource requirement, resource reservation and resource provisioning. You can really compare this to the Internet model:

- The basic thing you need is a server with a known IP address, and a client PC with a browser which connects to that server and shows the website.
- A DNS server can come in handy, as it translates e.g. www.cnn.com to the right IP address.
- On top of that, a search engine (e.g. google) is handy to find the right server if you are looking for information on Olympic Games e.g.

So the DNS server and search engine are not necessarily needed to use this internet, but they come in very conveniently.

The same is true for the Federator components. The only things which are needed to use a testbed are an experimenter tool, an authority (can be co-located with a testbed) and the testbed itself. However, the components below offered by the Federator make it more convenient, but they are not explicitly needed to let experimenters use the testbeds. This is also very important for sustainability aspects.

- **Portal**: a central starting place and registration place for new experimenters.
- **Member and slice authority**: experimenters who register at the portal are registered at this authority (as can be seen in Figure 4, there are also other authorities at testbeds). The APIs are not standardized.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- an **aggregate manager (AM) directory** which is readable by computers to have an overview of all testbeds available in the federation (more specifically, of their aggregate managers' contact information).
- a **documentation center** which gives an overview of available tools, testbeds and tips for the experimenter (http://doc.fed4fire.eu)
- **Authority directory**: for authentication/authorization between experimenters and testbeds, Fed4FIRE adopts a trust model where testbeds and authorities establish trust relationships among each other. To support authentication decisions at the testbeds, specific experimenter properties are included in the experimenter's certificate, which is signed by an authority. This approach allows testbeds to implement basic authentication functions or even rules-based authorization if they want to. To ease the creation of such a web of trust across the federation, there should be a trusted location that bundles all root certificates of the federation's authorities. This way a testbed can query/trust the central authority directory to see which root certificates it should trust.
- **Service directory**: directory service for federation and application services.
- **Reservation broker**: a broker for resolving abstract reservation requests within the federation. Note that within WP5 the further details of this component will be specified. Before it was called 'Future Reservation Broker', but the specification clearly indicates that the reservation broker is intended to be used as much for instant reservations as for future reservations, so it might be more suitable to change the name of this element to just "Reservation broker".

From these components only the authority has state of experimenters and experiments, but per definition multiple authorities are supported and installed in the federation. The others are just directory services and documentation to help experimenters and tools find the right testbeds and the portal and future reservation broker, which make the life of an experimenter easy, but which are just some of the usable experimenter tools.

3.3.1.2 Testbed side

At the **testbed** side, we have the following components:

- A testbed provides **resources (=nodes).** These can be virtual or physical, and can be very diverse in terms of technology. In many (but not all) cases, these resources are reachable through SSH.
- A testbed management component (called the **aggregate manager**) that is responsible for the discovery, reservation and provisioning of the testbed's resources. The testbed has the freedom to adopt any desired software framework to implement this functionality, as long as it can expose these functions through the Aggregate Manager interface.
- A testbed may have an **authority** (member and slice) in the federation if it wants to. This makes the testbed independent of the availability of the Federator to allow its own experimenters to participate in the federation. If the testbed however considers this too much of a burden, and is confident that the Federator authority or another authority can provide a reliable and lasting service, then the testbed can choose to rely on the one of the other authorities of the federation to serve its own experimenters.
- The testbed may operate **application services** if it wants to. As will be explained in detail in section 3.2.2, these are services that can be used by experimenters during their experiment (e.g. a service which returns sensor measurements in case of the SmartSantander testbed). Application services tend to abstract the underlying technical details of the provided services, and only provide direct benefit to an experimenter. At the moment it is unclear which mechanism should be adopted for authentication and authorization by application services inside the federation. From the WP2 Architecture point of view it seems preferable that all

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Fed4FIRE federation services would rely on X.509 certificates for this, since this would allow the experimenters to use the same user certificate for the application services as they already do today for the testbed resources. However, before making any final recommendations, it is required that WP7 performs a more thorough feasibility check and technical specification of this aspect.

3.3.1.3 Experimenter side

From the **experimenter** point of view, we can distinguish the following components:

- Some of the tools made available to the experimenter are hosted tools (e.g. the portal, future reservation broker, documentation center, possibly some application services, etc.). The experimenter will make use of these tools through a **browser**.
- Several experimenter tools for resource discovery, reservation and provisioning already exist and run locally on the experimenter's computer. So these are **stand-alone tools** instead of hosted tools. Examples are Omni, SFI, NEPI and jFed. The experimenter has the freedom to choose any tool of his or her preference, it will be supported by Fed4FIRE as long as it is compatible with the adopted AM APIs.

3.3.1.4 Outside of the federation

The last actor depicted in Figure 4 relates to the entities **outside of the federation.** The following components are relevant:

- Just as the testbeds in our federation, the testbeds outside of the federation provide **resources**.
- A **testbed manager** of some sort manages these resources. It can expose its functionality through any desired API. If it supports the AM API, then it is compatible with all Fed4FIRE tools. If it wants, this testbed can decide to authorize (specific) Fed4FIRE members, even without formally belonging to the federation.
- Any party outside of the federation can provide **application services** to both Fed4FIRE and non-Fed4FIRE members. These application services can even rely on Fed4FIRE resources if the external application provider has approval of the specific involved Fed4FIRE testbeds for doing so.
- These application services can expose themselves using any interface that they want. Therefore they can have their own credentials or login/password for authentication and authorization of their users. The **service authority** is then responsible for the management of this service-specific identity information.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

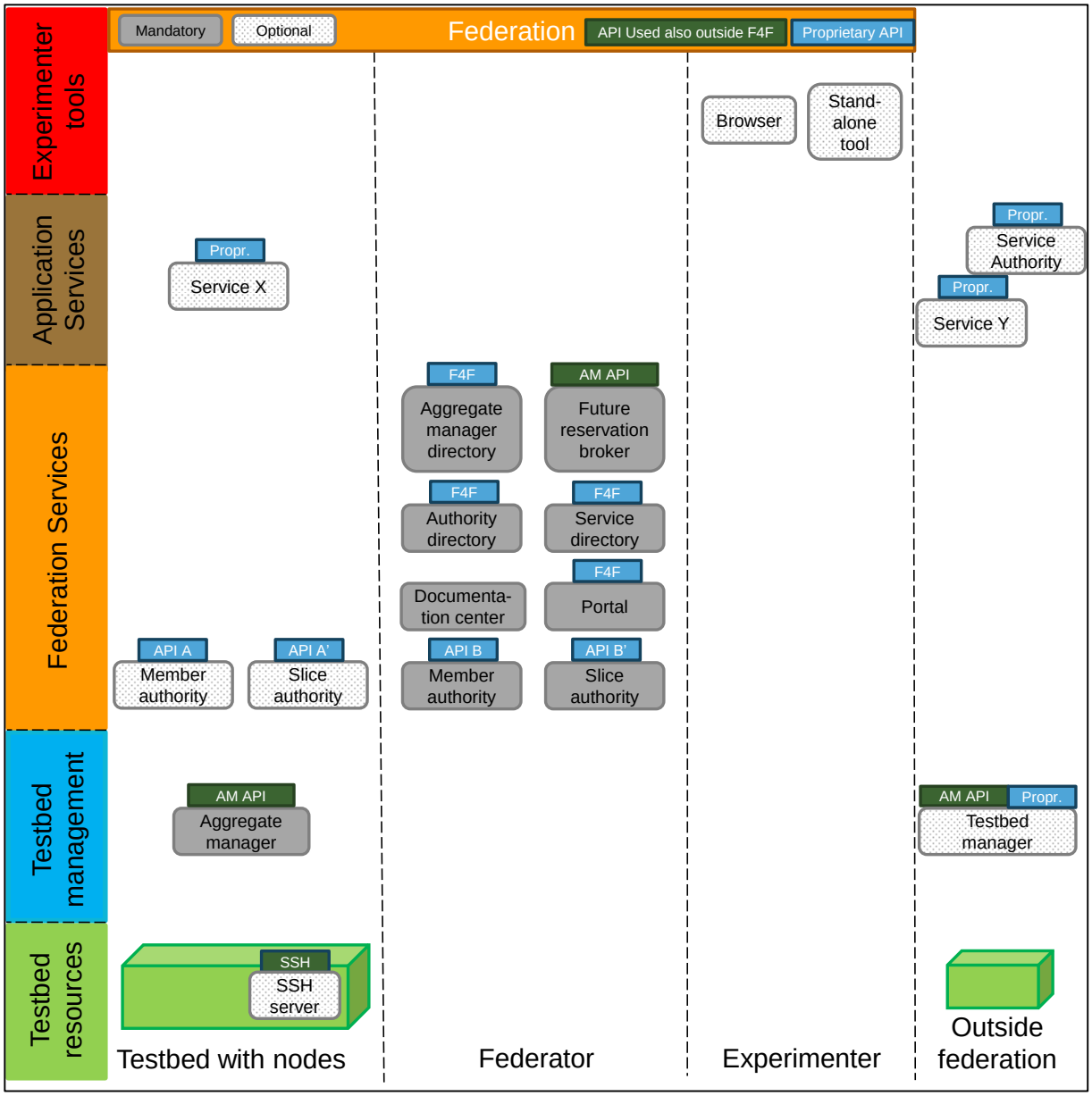


Figure 4: Fed4FIRE cycle 2 architecture for discovery, reservation and provisioning

3.3.2 Basic concepts

Several aspects of the architecture presented in Figure 4 originate from the Slice-based Federation Architecture (SFA) [9]: the Aggregate Manager API, the member authorities and the slice authorities. Given the importance of these components in the Fed4FIRE architecture, some more detail is given in this section.

3.3.2.1 Concepts slice, sliver and RSpec

Three key concepts are slices, slivers and RSpecs. As depicted in Figure 5, a **slice** is the concept that is used to bundle resources belonging together in an experiment or a series of similar experiments, over multiple testbeds.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

A **sliver** is the part of that slice which contains resources of a single testbed. One uses an **RSpec** (Resource Specification) on a single testbed to define the sliver on the testbed. The RSpec and thus the sliver can contain multiple resources. Note that there is an important difference between the concept sliver as defined in the Aggregate Manager API v2 and v3.

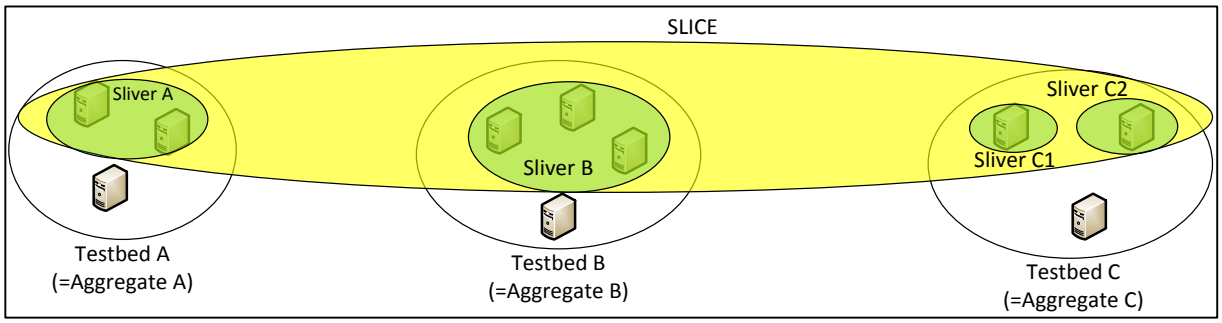


Figure 5: Concepts of slice and sliver

Version 2 of the AM API supported only one sliver per slice per testbed [6]. Hence in this version of the AM API a sliver denotes all the slice's resources on a testbed. (e.g. the deletesliver call deletes all resources of a slice on a single testbed). In the figure above Sliver C1 and C2 cannot exist in AM API v2 (of course, a testbed can decide to work internally as such, but the slivers are not individually addressable through the API).

In version 3 of the AM API, one of the several important problems that was solved is that of adding resources of a testbed to a slice. In AM APIv2, as only a single sliver is supported per testbed and slice, it was not possible to add extra resources to this same slice for that testbed (and remove them afterwards). For this, API v3 now supports addressing of individual slivers, and the SFA calls such as allocate, provision, etc. can now work on multiple slivers on a single testbed. For instance the call "allocate" returns a struct of slivers.

In practice it is a little more complicated since testbeds have the total freedom to adopt one of multiple methods when allocating resources to slivers [8]. If you request an RSpec with multiple resources, a testbed can decide to create multiple slivers, only a single one, or any allocation strategy in between. This has then as a consequence that it depends on the testbed if an experimenter can remove single resources/slivers or only everything at once. E.g. on the virtual wall testbed, if you allocate 2 nodes with a link, you get back 3 slivers (2 for the nodes and 1 for the link).

3.3.2.2 GENI AM API and GENI RSpecs

The current concepts and APIs of the GENI framework are documented on the GENI wiki [10]. Together with an international working group and for this cycle 3, we have created an updated version of the documentation, which can be found at https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html [19]. Those documents contain really the API details, and we decided that it is better to publish them to a wide public instead of putting them in this deliverable. To go to even a broader adoption and a more sustainable way, everyone can now ask for git pull requests (changes or comments) through the common and neutral github platform: https://github.com/open-multinet/federation-am-api

The key components and interfaces are the following:

- GENI Aggregate Manager (AM) API version 3 [7]

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

  - This contains a description of the API for discovery, resource requirements and provisioning.
  - It defines the GENI certificates for authentication. The GENI AM API uses XML-RPC over SSL with client authentication using X.509v3 certificates. This is now documented in more detail at https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html#_basic_am_call_concepts
  - It defines GENI credentials for authorization
  - It defines GENI URN identifiers for identifying and naming users, slices, slivers, nodes, aggregates, and others
- GENI RSpec (resource specification) version 3 [11]. The RSpec comes in three flavors:
  - Advertisement RSpec: for resource discovery (getting a list of all resources)
  - Request RSpec: requesting specific resources
  - Manifest RSpec: describing the resources in an experiment
  - More details can be found here: https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/rspec.html

The services that we identified as 'federation services' are in GENI partly identified as a Clearinghouse [12]:

- An Identity Provider (IdP) provides certificates and PKI key materials to human users, registering them with the GENI federation as GENI users.
- A Project Authority asserts the existence of projects and the roles of members (e.g. PI, Experimenter).
- A Slice Authority provides experimenters with slice credentials by which to invoke AM (Aggregate Manager) API calls on federation aggregates.
- A Service Registry provides experimenters with a 'yellow pages' of URL's of all trusted services of different kinds. In particular, the list of all available aggregate managers trusted by GENI (possibly satisfying particular criteria) is provided.
- A Single-Sign-on Portal, which provides web-based authentication and access to the authorized Clearinghouse services and other GENI user tools.

Fed4FIRE uses as such the GENI AMv3 API (while GENI AMv2 is still supported), and is working together with GENI on a Common AM API as a next version which can be found at https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html .

3.3.3 Details of the adopted mechanisms for authentication and authorization

One of the advantages of the GENI AM API is that authentication and authorization is based on X.509v3 certificates, credentials and a chain of trust. This means that by nature it is distributed and very scalable. Since Fed4FIRE adopts the GENI AM API, it inherently adopts the same mechanisms for authentication and authorization.

More specific, the following authorities will be trusted by the testbeds in cycle 3 of Fed4FIRE:

- iMinds member and slice authority (through https://authority.ilabt.iminds.be)
- Fed4FIRE central member and slice authority (through http://portal.fed4fire.eu)
- PlanetLab Europe member and slice authority (through http://www.planet-lab.eu/)

Currently the APIs on top of these authorities are proprietary.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

3.3.4 Details of the layer "Application Services"

Through the practical experience with the development of the first and second cycle of the project, and through the requirements collected in the scope of WP4 (reported in D4.2 - Second input from community to architecture [3] and D3.3-D4.4 – Third input from community to architecture [16]), the need was identified to introduce a new layer in the architecture that is intended to make it easier for experimenters to use the resources of the federated infrastructure in their experiments. This is the layer called "Application Services" in Figure 4. Some examples might clarify what is exactly meant here.

For instance, in the case of SmartSantander you could have the experimenter reserve the real sensing devices, and collect the data during the course of the experiment. However, SmartSantander also deploys an application service that already gathers all this data or processed versions of them, e.g. historical, maximum, minimum, etc.), and exposes it through an API that can be called by the experimenters.

Another example is that of a Hadoop cluster (big-data framework) on resources of the Virtual Wall testbed. If an experimenter wants to setup such a cluster, he can reserve some nodes on the Virtual Wall, and install a Hadoop cluster on them manually. However, if this is something that many people seem to be doing, and if such an installation is quite labor-intensive, it makes sense for the Virtual Wall administrators or experimenters or application providers to create an application service that can create such Hadoop clusters automatically. In fact, the experimenters shouldn't even be aware of the fact that this cluster is being deployed on the Virtual Wall, from their perspective it should be sufficient to know that they asked for a Hadoop cluster with certain characteristics to be deployed, and that this was actually setup for them.

The introduction of such a new layer in cycle 2 brought some important questions that needed to be answered in the design of the federation architecture. One issue that was solved, is the fact that it is necessary that service providers (federated parties or 3rd parties also as external service providers) can publish their services towards the experimenter community. Another question was the definition of a suitable interface to expose the actual application service to the experimenters. A third unknown factor was how to support the different levels of abstraction of the underlying technology in various conditions. These questions are tackled in the remainder of this section.

3.3.4.1 How to publish application services?

As mentioned above, the services have to be published in order to allow the experimenters to discover them and to decide if they fulfill their needs. A service directory that gathers all the available services and their descriptions is added to the architecture for this. The exact API for this and implementation details can be found on deliverable D5.2 – Detailed specifications regarding experiment workflow tools and lifecycle management for the second cycle. It has to contain a link to the documentation of the service where one can find the used protocol, the method to authenticate and the exact API details.

3.3.4.2 How to expose application services?

Services provide an added value, so the most important aspect of services is the logic they implement (Experiment Service Logic), and the interfaces should only be a means to expose these services. Whereas interfaces might evolve over time, the service logic remains. This allows operating independently of a specific technology and allows more possibilities for service invocation (for example, an orchestrated workflow could be implemented in the future).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

In terms of protocols for the application services, we go along with what is suitable for a specific service. We consider different protocols to provide services: HTTP-REST, SOAP, RPC-XML and others. The documentation of the protocol and API (see 3.3.4.1) is key for using it correctly.

3.3.4.3 High level examples of application services

The service provider exposes the added-value services, deals directly with the experimenter and interacts with the different resources, tools and utilities within each testbed in order to deliver what the experimenter requests. The experimenter might be mainly interested in the functionality the service provides and not so much in the underlying details required to implement it. This allows the experimenter with limited technical skills -or simply not interested in technical details- not to interact directly with the testbeds but rather focus on the service delivery. In this case, the experimenter is not aware or does not care about the different testbeds involved in his experiment (and their internal resources, tools and utilities); it is the service provider who manages all the interactions with the different testbeds.

Examples of this are for example a service provider which offers a service to deploy Hadoop clusters (where the Hadoop user does not know which underlying physical testbeds and resources are used) or a service to read out sensor values of a Smart City (e.g. SmartSantander, where the experimenter is not involved with the real sensors).

Hence, the service provider deals with the whole experiment lifecycle, as a service to the experimenter -discovering testbeds, negotiating reservations and access to the testbeds, uploading input data, running and monitoring the experiment, downloading any output data and presenting the results to the experimenter. The service provider is the single contact point for the experimenter. In this scenario, the experimenter does not need to be identified to the federated environment in order to access the different testbeds, since it is the responsibility of the service provider to authorize and authenticate the user who wants to use the exposed service. It is the service provider who has SLAs and agreements with the testbeds offering physical resources.

These application services are again ‘API only’ services, so the same levels of federation (associated, light, advanced) can applied as described in chapter 4.

3.3.5 Sequence diagrams for resource discovery, specification and provisioning

To further clarify the role of all the architectural components needed for discovery reservation and provisioning, and to illustrate how they interact with one another, the corresponding sequence diagrams are given below. Figure 6 describes the very first steps that a new experimenter would take in the Fed4FIRE federation. It all starts when the experimenter (we will assume that this is a man in the remainder of this discussion to keep the text readable) has a specific idea, and wonders if he could experiment with it on Fed4FIRE (1). To assess if this is actually the case, he visits the project’s documentation center (which is completely open to everyone) and browses the web-based catalogue of Fed4FIRE testbeds (2). Once the experimenter has identified the testbeds that he wants to experiment on, he surfs to the First Level Support dashboard (which is also completely open to everyone) to check if the testbeds are up and running and if they have free resources (3). As a last step of his exploration of Fed4FIRE, he browses to the service directory to look for application services that could minimize the needed development for experimenting with his idea (4). Note that the

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

documentation center, the First Level Support dashboard and the service directory are three different architectural components, but they can (and should) be implemented with cross links.

Since the outcome of all previous steps was positive, the experimenter decides to actually experiment with his idea on Fed4FIRE testbeds (5). The next step therefore will be to get an actual Fed4FIRE user account. For this he retrieves the documentation regarding the registration process from the project's documentation center (6). Based on this guidance documents, he is able to register for an account on one of the Fed4FIRE member authorities (7). Once the administrator of that authority has approved his request, he goes back to that authority and retrieves his X.509-compatbile Fed4FIRE member certificate (8). Now that he has a valid certificate, he only has to decide which tools to use with that certificate to start discovering, reserving and provisioning resources. For this, he surfs again through the documentation center to browse through the catalogue of different experimenter tools that he could use (9). After deciding which specific tool(s) he wants to use for his experiment, the experimenter is entirely ready to start experimenting on Fed4FIRE (10).

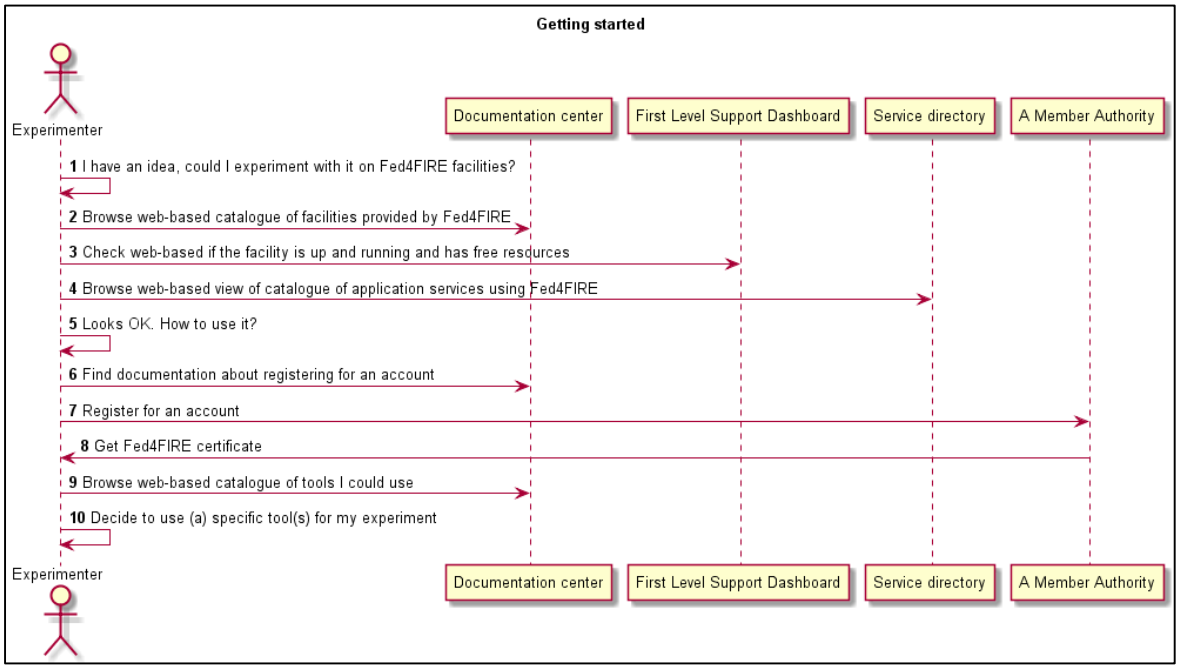


Figure 6: Sequence diagram for initial experiment lifecycle

The next steps to actually setup the experiment are shown in Figure 7. This sequence diagram shows the provisioning right now, without advanced reservation. The first thing that the experimenter has to do is to provide his Fed4FIRE member certificate (which he downloaded before) to the experimenter tool(s) that he wants to use (1). In case of a hosted tool this can be a derivative certificate (e.g. a delegated credential, or a speaks-for credential). Once he has done this, the tool is ready to be used by him. The first thing the tool will want to do is to create a slice for the experiment. The first step for doing so is to learn the contact information of the corresponding member and slice authorities. The tool will retrieve this information from the authority directory (2, 3). The next step will then be to retrieve his user credential from the member authority (4, 5). This credential provides information regarding the specific rights that were given to him by the member authority. One of them is the right to create slices. Using this credential, the experimenter can then register a new slice at the slice authority (6, 7). When creating a slice, an expiration date (end date) has to be given. This can be later made longer (but not shorter).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

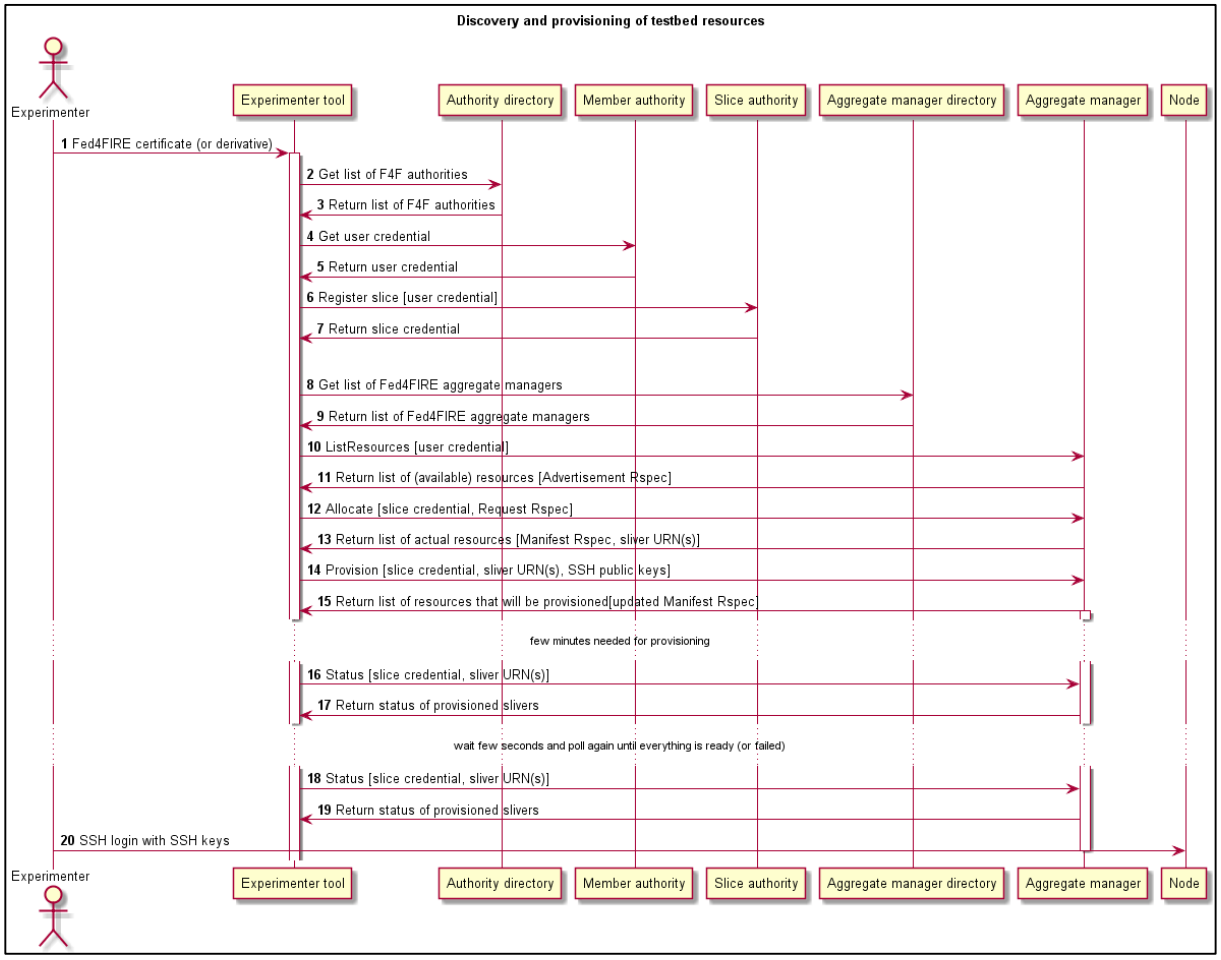


Figure 7: Sequence diagram for discovery and provisioning

Now that the experimenter is the owner of his slice, he can start adding resources to it. For doing this, his tool will first need to retrieve the contact information for the different testbeds (more specifically the aggregate managers) from the aggregate manager directory (8, 9). This information allows the tool to ask for a list of (available) resources at every testbed of the federation (10). This information is returned by the testbed in the form of an Advertisement RSpec (11). Once the tool has presented an overview of all this information to the experimenter, and he has finally selected the specific resources he wants to use, the next step will be to actually request these resources at the corresponding testbeds. For this the tool will create the appropriate Request RSpec, and will request the testbed's aggregate manager to allocate the corresponding resources to the slice of the experimenter (12). To prove that the experimenter has the right to add resources to this slice, he will attach the slice credential that he received from the slice authority in step (7). The aggregate manager will then decide if he can or cannot assign these resources to that slice, and will communicate the corresponding result to the experimenter tool in the form of a Manifest RSpec (13). He also returns the URN(s) of the different slivers if the assignment was successful. However, even in case of a positive response, being allocated does not yet mean that a resource is usable by the experimenter. It first needs to be provisioned. For this the experimenter tool will request the testbed's aggregate manager to provision the resources (14). It will attach the slice credential, sliver URN(s) and the public SSH key of the experimenter to this request. The aggregate manager will respond with an updated Manifest RSpec, describing the list of resources that will be provisioned (15). After waiting a few minutes while the

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

aggregate manger performs the actual provisioning of the resources, the experimenter is eager to start using the resources. For this he will start checking the status of the requested resources (16, 17). Since the resources were not yet ready, he waits a bit longer, and asks for the status again (18, 19). This time the response indicates that the resources are ready to be used. The experimenter then logs in with SSH on the resources (20) using a key pair of which the public key was attached as an argument to the provision request in step (14).

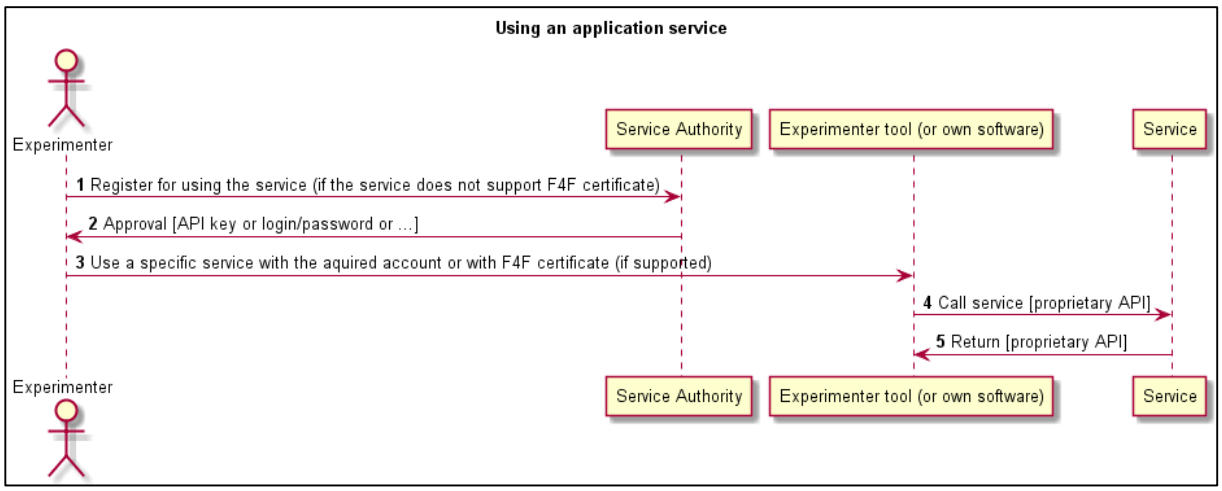


Figure 8: Sequence diagram for using an application service

The last aspect of this part of the federation architecture that needs some more explanation through a sequence diagram is the usage of an application service. As described in section 3.3.1, it is allowed that the service may adopt its own authentication and authorization methods. Inside the federation it is possible that later on it will be obliged to support the Fed4FIRE X.509 certificates.

As depicted above in Figure 8, the first step when using an application service is to register for that service at the corresponding Service Authority (1, 2). Once the needed credentials have been retrieved for the application service (being the experimenter's X.509 Fed4FIRE member certificate or any other credential specific to the application at hand), the experimenter can hand these over to its tool that will make use of the service, together with a request to use a specific service (3). The tool will then call the service (4), and retrieve the results (5).

3.3.6 Resource reservation (in the future or instantly)

With resource reservation we mean the reservation of resources in the future (e.g. next week Tuesday and Wednesday) or instantly (starting right now). Especially testbeds offering physical resources do need this if their occupation is high or if you need very specific resources (e.g. for wireless experiments). This is planned to be implemented for cycle 3 on at least four testbeds (Nitos, NETMODE, PLE, Virtual Wall and w-iLab.t) and probably also on Bonfire.

Currently, the prototypes are being implemented but it seems too early to put definitive API call sequences in this deliverable as it seems that implementation may still vary too much.

It should be noted that the instant reservation as described in the previous sections is just a special case of this with the starting time 'now' instead of in the future. Slice and sliver creation have expiration/end times, so it is a special reservation in the future having as a starting point 'now' till the 'expiration date'.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

3.4 Monitoring and measurement

The second part of the architecture that is presented here aggregates all architectural components needed to support monitoring and measurement. The architecture itself is depicted in Figure 9. The remainder of this section will first introduce all the corresponding components, and will then go into more details regarding the specific architecture of the monitoring for the First Level Support function.

3.4.1 Introduction of the architectural components

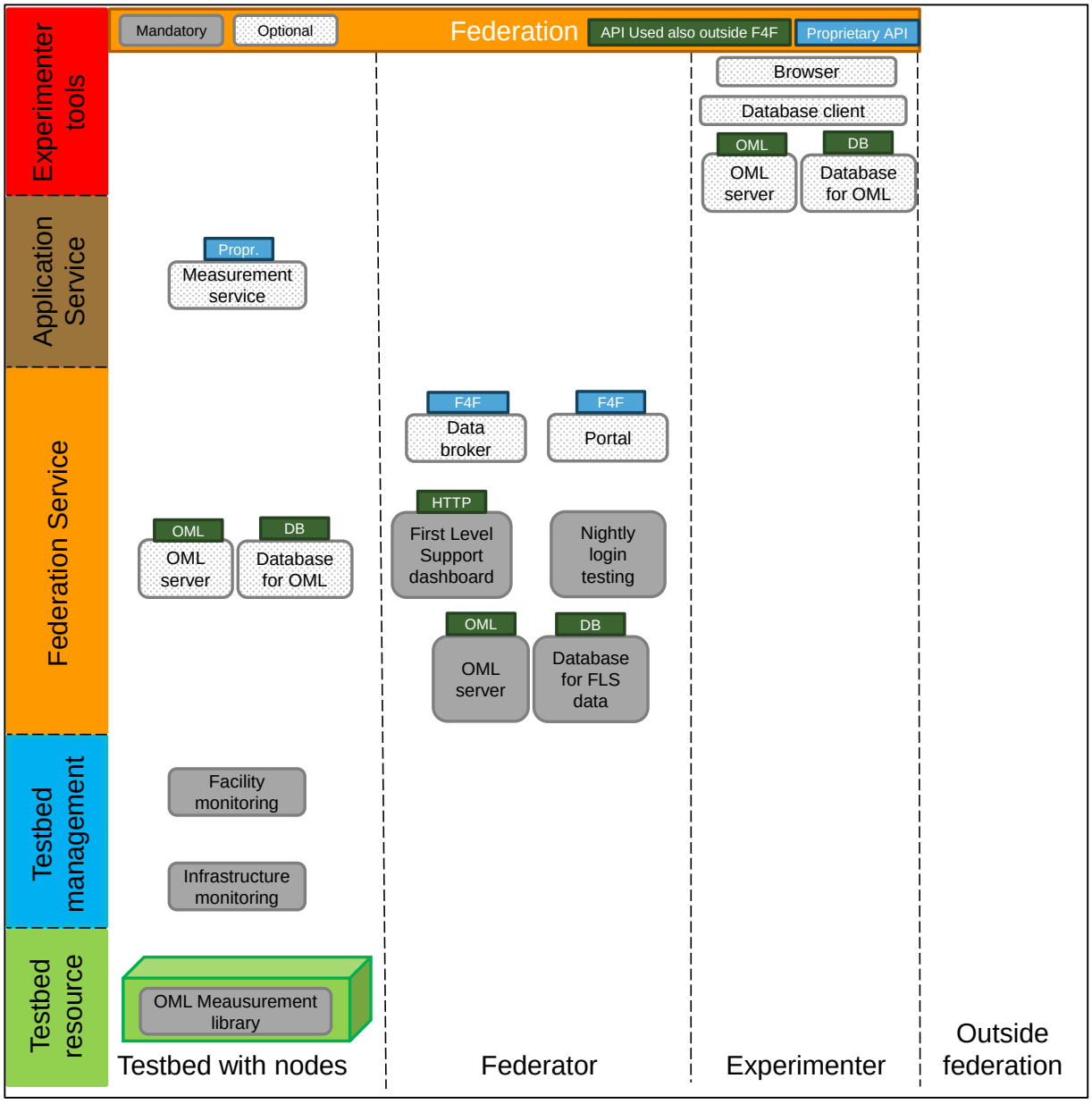


Figure 9: Monitoring and measurement architecture for cycle 3

At the **testbed** side, the following architectural components can be distinguished:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- **Facility monitoring**: this monitoring is used in the first level support to see if the testbeds are up and running. The testbed has the freedom to adopt any solution to gather this type of monitoring data as it sees fit (e.g. an existing monitoring framework such as Zabbix, Nagios or similar), as long as it is able to export that data as an OML stream to the Federator's central OML server, which will store it in a database for First Level Support. In the first cycle of Fed4FIRE, the facility monitoring was rolled out on all testbeds. More details are given in section 3.4.2.
- **Infrastructure monitoring:** Instrumentation of resources by the testbed provider itself to collect data on the behavior and performance of services, technologies, and protocols. This allows the experimenter to obtain monitoring information about the used resources that he could not collect himself. Examples of such infrastructure monitoring data are information regarding the CPU load and NIC congestion on the physical host of a virtual machine resource, the monitoring data of switch traffic, or the gathering of data regarding the wireless spectrum during the course of the experiment. This infrastructure monitoring is now further specialized in infrastructure monitoring taking into account a specific experiment and thus only interesting for the experimenter (e.g. measure the switch ports only that experimenter uses) and infrastructure monitoring of a whole facility (e.g. total traffic on the switch backbone of a testbed or total load of virtual machines).
- **OML measurement library** for experiment measuring**:** this component is intended for measurements which are done by a framework that the experimenter uses and which can be deployed by the experimenter itself on his testbed resources in his experiment. The experimenter can retrieve or calculate this data as he prefers, and can then use the OML measurement library (which should be available on every resource) to easily export it to an OML server with database for storage.
- **OML server**: this component can be configured to be the endpoint of a monitoring or measurement OML stream, and can store this data in several types of **databases** (PostgresQL, SQLite3). The deployment of an OML server by a testbed provider is optional. The best way to retrieve the data should be further investigated in WP6. A common practice today is to get the data directly from the database using the raw database API. This is most likely beneficial in terms of performance, but on the other hand this means that the API for data retrieval differs per underlying database technology. An alternative would be to create an OML-focused API that might be a little less optimal in terms of performance but that can be consistently implemented over a range of database products, and which is in line with the decision to adopt OML as the common interface for measurement and monitoring in the project. A more detailed analysis of the pros and cons of both approaches is needed, together with a feasibility assessment of both approaches.
- The testbed provider that provided an OML server can decide to make this data easy to retrieve for the experimenter by exposing it as a **measurement service** using a proprietary interface. The deployment of such a measurement service is optional.

When focusing on the **Federator**, the following elements of Figure 9 require some further introduction:

- The **FLS dashboard** gives a real-time, comprehensive but also very compact overview of the health status of the different testbeds included in the Fed4FIRE federation. To determine this health status it combines facility monitoring information provided by the testbeds with specific measurements performed by the dashboard component itself. More details about this are given in section 3.4.2. The existence of the FLS dashboard at the Federator level is in line with the project's approach that Federator components should be put in place for convenience, but shouldn't be critical for the federation's operation. The dashboard is indeed a very useful tool to have, but if its operation were disrupted or even discontinued, experimenters would still be just as able to get resources and work on them as when the dashboard was still supported.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Also, since it is a stateless component (it combines data from the different testbeds in a single health overview), it can easily be duplicated or moved. This would only result in a loss of historical information about the federation's health, but no functionality loss would occur.

- The federator provides an **OML server and corresponding database for FLS data** to process and store facility monitoring data of the testbeds to be used by the First Level Support (FLS). These two Federator components (which are depicted in Figure 9) are a necessity to implement the FLS dashboard, but since the dashboard itself is not critical to the federation, neither are the central OML server and corresponding database. This motivates that the Federation level is a suitable place for these components.
- The component for **nightly login testing** provides a second view on the operational status of the federation. Its information is not real-time (typically tests would automatically be performed once or twice a day), but the result of these tests is more thorough then those of the FLS dashboard. This is because in this case the testing module performs an actual experiment (including all steps of the experimental lifecycle related to resource discovery, reservation and provisioning, ending with an actual automatic SSH login on the provisioned resources) and tracks success or issues for any intermediary step of the experiment lifecycle. According to exactly the same principles as for the FLS dashboard, the component for nightly login testing is in line with the approach that Federator components should not be critical for the operation of the federation (disruption in the nightly login testing does not hinder the execution of experiments, this testing could be easily duplicated or moved).
- The **data broker** is an optional component that can be **accessed through the portal**, and which makes it easier for novice experimenters to retrieve their experiment data from the different sources where they might reside (OML servers of the different testbeds that provided infrastructure monitoring, OML servers of the experimenter itself on which the experiment measurements were stored, etc.). According to exactly the same principles as for the FLS dashboard and the component for nightly login testing, the data broker is in line with the approach that Federator components should not be critical for the operation of the federation (disruption in the data broker service does not hinder the direct retrieval of experiment data, the broker service could be easily duplicated or moved).

The **experimenter** himself can also utilize different experimenter tools:

- His **browser** is used to access the data broker through the portal.
- A **database client** allows the experimenter to retrieve his monitoring and measuring data from any OML server where it was stored
- For storing infrastructure and measurement data, the experimenter has the option to deploy his own **OML server and attached database** in order to archive the data himself.

3.4.2 Sequence diagrams regarding measuring and monitoring

The different steps involved in **facility monitoring** are depicted in Figure 10. In short, a monitoring agent on the selected key component(s) of the testbed provides its data to the testbed's facility monitoring component (1). The testbed has the freedom to adopt any facility monitoring framework as it sees fit (Zabbix, Nagios, proprietary software, etc.). This component will calculate the overall testbed health status (2), and will push this health status info to the central OML database accordingly (3). This makes it straightforward for testbeds to support facility monitoring, since the logic is kept locally, and the mechanism to export the result to the central component is OML, which is the same as that for the infrastructure monitoring and experiment measurement functionalities. Testbeds could also easily support multiple central OML servers by just exporting different copies of this OML stream to the different central servers.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

In the next step, an experimenter is then able to browse to the FLS dashboard to look at the status overview of the Fed4FIRE testbeds (4, 5). For depicting this overview, the dashboard first has to collect the corresponding data from the central OML database (6, 7), after which it can display the status of the testbeds to the experimenter (8, 9).

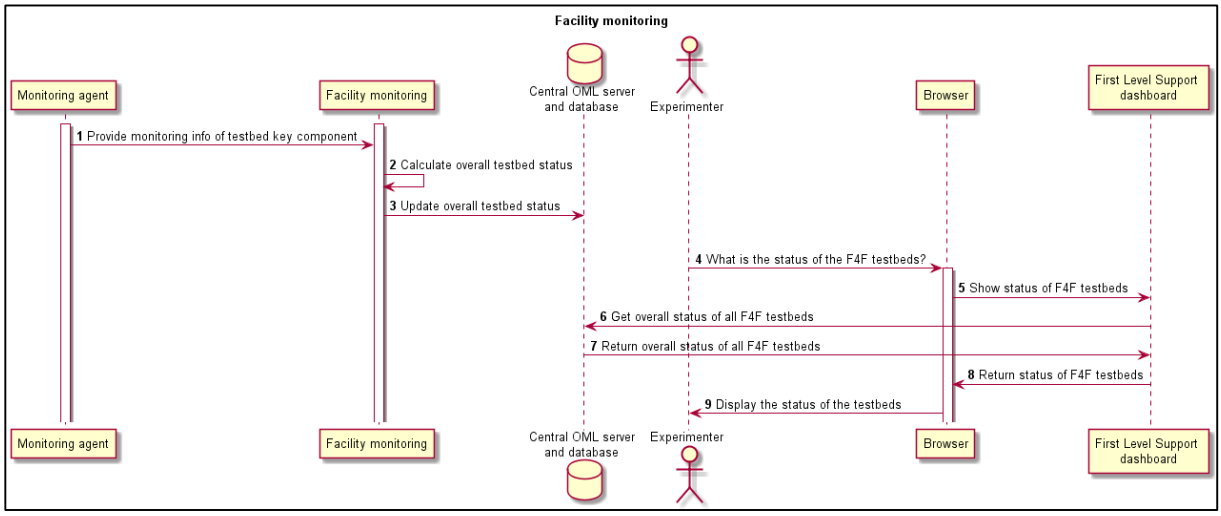


Figure 10: Sequence diagram for facility monitoring

The steps related to **infrastructure monitoring** are summarized in Figure 11. The first step is to collect the data using monitoring agents deployed on the resources. These agents export their data to the infrastructure monitoring framework (1, 2), which will then have to expose it through an OML interface. When requesting resources, the experimenter has to provide information of where the monitoring data should be exported to (exact location of central or experimenter OML server). This then allows the experimenter to use its database client to retrieve his infrastructure monitoring data from the corresponding OML server and database (3-6). In the case the infrastructure monitoring data is spread across different OML servers, the experimenter has the option to easily retrieve all this data through the Fed4FIRE data broker (7-13).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

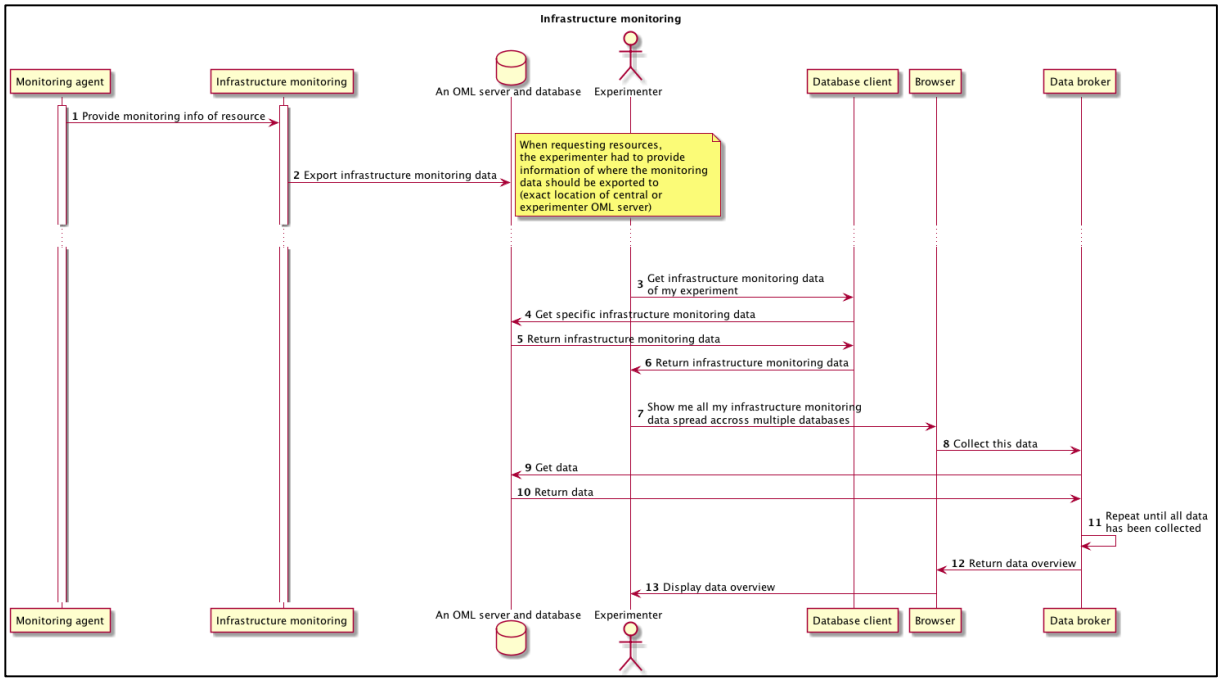


Figure 11: Sequence diagram for infrastructure monitoring

The sequence diagram regarding **experiment measuring** is depicted in Figure 12. In short, it is practically identical to that of infrastructure monitoring that was given in Figure 11. The only important difference is that in this case the data source is any piece of experimenter software that makes use of the measurement library. This library stores the experiment measurements directly in the OML server which was appointed by the experimenter (1). All other steps for retrieving this data as an experimenter are the same is for infrastructure monitoring (2-12).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

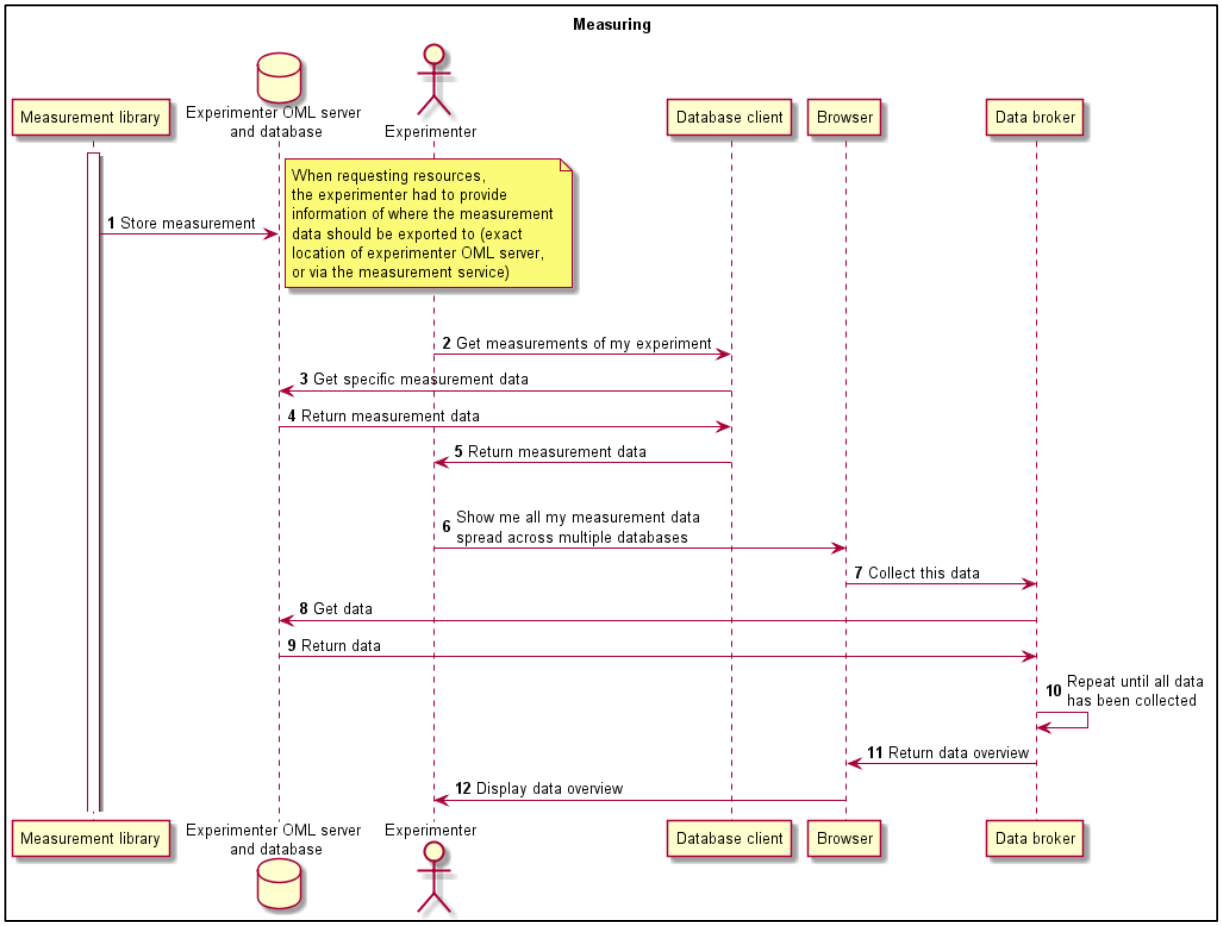


Figure 12: Sequence diagram for experiment measuring

3.4.3 Monitoring and measuring for First Level Support

One of the aspects of this part of the Fed4FIRE architecture that deserves some more details is the specific adopted approach for monitoring and measuring for First Level Support. The software has been completely rewritten for cycle 3 while keeping the same visual layout. Currently the following is deployed as depicted in Figure 13. We monitor 6 things per testbed:

1. ICMP ping to the AM server or some other testbed server: this checks connectivity over the internet to the testbed (if this fails, testbed can likely not be used)
2. AM API GetVersion call: tests the AM component (no credential is needed)
3. AM API Listresources to know the number of free resources (if this is 0, then new experiments cannot be created)
4. Red/Green/Amber internal status of what the testbed provider monitors himself (this is custom per testbed, and based on the testbed's facility monitoring data)
5. Aggregated status: this aggregates all tests, averages (to avoid false alarms !) and sends emails.
6. Login status: result of the last login test per testbed

There is a live visualization at https://flsmonitor.fed4fire.eu and there are also email alarms and long term statistics to dive into the monitoring information of the past.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

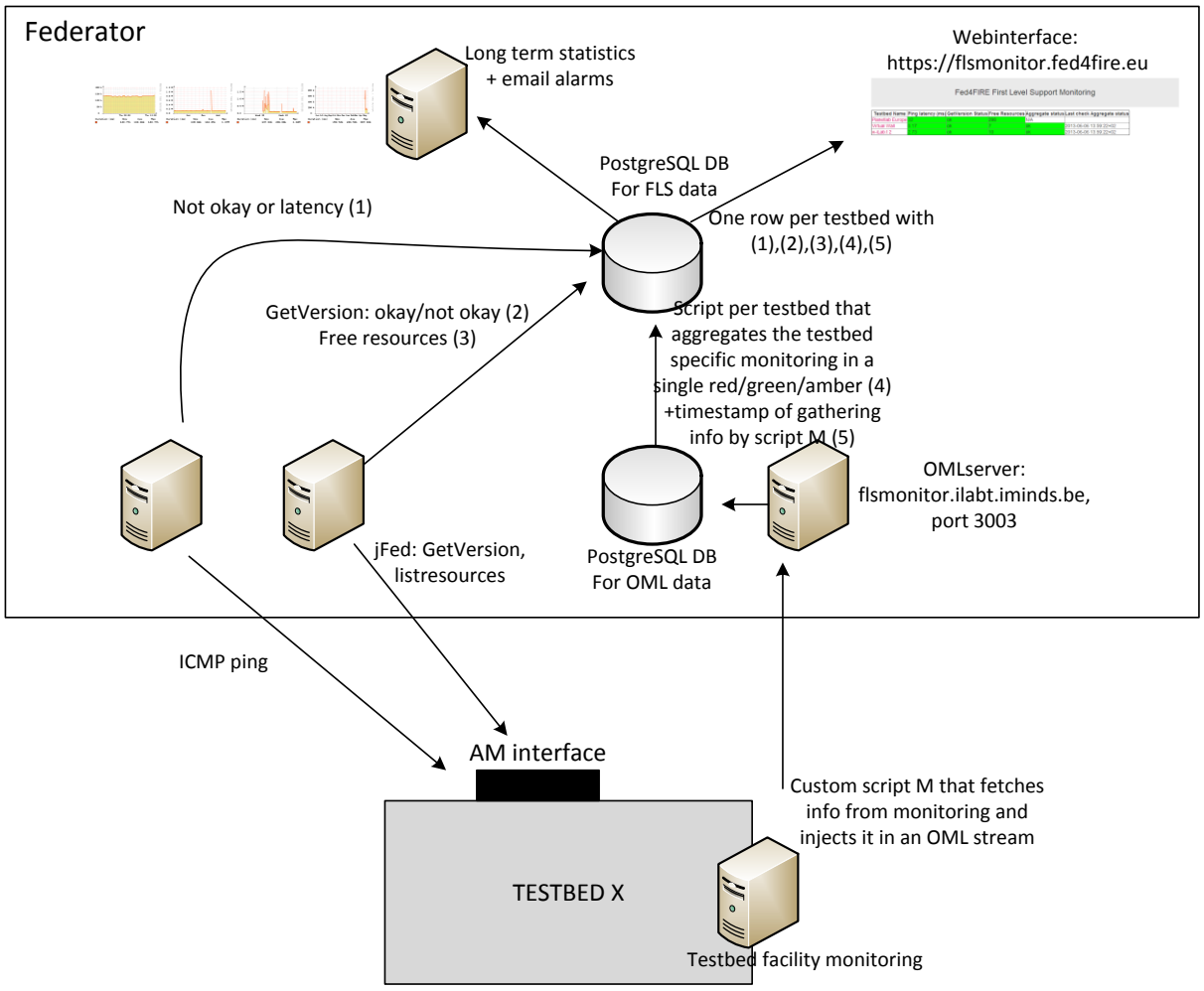


Figure 13: FLS dashboard architecture

Since a picture says more than a thousand words, a screenshot from the FLS dashboard is given below.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Fed4Fire First Level Support Monitor

Testbed Name	Ping Latency (ms)	GetVersion Status	Free Resources	Internal Status	Aggregated Status	Login Status
BonFIRE	24.44	no data	no data	SUCCESS	SUCCESS	no data
C-Lab	52.89	SUCCESS	124	SUCCESS	SUCCESS	WARNING
ExoGENI NICTA	312.39	SUCCESS	20	SUCCESS	SUCCESS	SUCCESS
FUSECO	16.31	FAILED		SUCCESS	FAILURE	FAILURE
Koren	282.69	SUCCESS	3	SUCCESS	SUCCESS	FAILURE
Kreonet Emulab	309.66	SUCCESS	21	no data	SUCCESS	SUCCESS
NETMODE	55.48	SUCCESS	20	SUCCESS	SUCCESS	SUCCESS
NITOS Broker	61.03	SUCCESS	78	SUCCESS	SUCCESS	WARNING
Ofelia (Bristol openflow)	17.52	SUCCESS	-1	SUCCESS	SUCCESS	SUCCESS
Ofelia (Bristol vtam)	17.47	SUCCESS	-1	SUCCESS	SUCCESS	FAILURE
Ofelia (i2CAT openflow)	18.02	SUCCESS	5	SUCCESS	SUCCESS	FAILURE
Ofelia (i2CAT vtam)	20.12	SUCCESS	3	SUCCESS	SUCCESS	WARNING
Planetlab Europe	31.30	SUCCESS	275	SUCCESS	SUCCESS	SUCCESS
SmartSantander	58.88	SUCCESS	0	SUCCESS	SUCCESS	no data
Virtual Wall 1	0.36	SUCCESS	89	SUCCESS	SUCCESS	SUCCESS
Virtual Wall 2	0.20	SUCCESS	34	SUCCESS	SUCCESS	SUCCESS
Virtual Wall 2 (openflow)	2.18	SUCCESS	2	SUCCESS	SUCCESS	SUCCESS
w-iLab.t 2	5.82	SUCCESS	60	SUCCESS	SUCCESS	SUCCESS

This page automatically updates every 10 seconds. (updated 3 times)

Calendar with testbed maintenances and reservations (contact helpdesk AT fed4fire.eu to add maintenance for a testbed)

Figure 14: FLS dashboard screenshot

3.5 Experiment control

The third part of the architecture that is presented here aggregates all architectural components needed to support experiment control. The architecture itself is depicted in Figure 15. The remainder of this section will first introduce all the corresponding components, and will then further clarify them through the appropriate sequence diagrams.

3.5.1 Introduction of the architectural components

At the **testbed** side, the following architectural components can be distinguished in Figure 15:

- The simplest mechanism for experiment control is allowing the experimenter to login to the nodes using SSH, and to control the experiment manually. In this case it is required that an **SSH server** is deployed on the resource.
- The **resource controller** is an agent that runs on the testbed's resource, and which can invoke actions on the request of the experimenter on that resource. These actions have to be communicated to the resource controller in the FRCP protocol. Optionally a resource controller can also invoke actions on another resource which is connected through a communication link. An example is a resource controller running on a node and controlling a sensor through USB.
- These FRCP messages are communicated with the support of an AMQP framework (advanced message queuing protocol, http://www.amqp.org). For this it is advised (but not mandatory) that every testbed deploys its own **AMQP server**. In cycle 1 and cycle 2 we used XMPP, but due to performance and stability reasons this was switched to AMQP.
- When receiving an FRCP message, the resource controller cannot just perform any action that is being requested. It has to verify if the entity that has sent him the FRCP message is actually

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

authorized to request this action. For this the resource controller will contact the **Policy Decision Point** (PDP). This PDP will be able to make this authentication and authorization decision based on the information that was handed to it from the **Aggregate Manager** (which resource belongs to which slice, etc.). PDP is a component introduced in the architecture of cycle 2. It was introduced in order to protect testbeds from unauthorized control of its resources. Without it anyone that knows the messaging topics to which the resource controller is listening, and has knowledge of the FRCP protocol that is being used, would be able to control the resource controller of that resource. In cycle 2 we upgraded from such a security-by-obscurity mechanism to a fully secure authentication and authorization scheme.

- The **experiment control server** is the entity that takes an experiment control scenario as an input, and processes it to define the specific moments when a certain action should be taken to the resources. It will be in charging for sending the corresponding FRCP messages to the appropriate resources at the correct time.

At the **experimenter** side, some experimenter tools can be applied for experiment control

- An **SSH client** allows the experimenter to login to the resources of the testbed himself, and to control the experiment manually.
- An **experiment controller** is identical to the experiment control server described above, but this time it is a local experimenter tool instead of a hosted tool. So the experiment controller takes an experiment control scenario as an input, and processes it to define the specific moments when a certain action should be taken to the resources. It is in charging for sending the corresponding FRCP messages to the appropriate resources at the correct time.
- The **scenario editor** is a tool that enables the construction of the experiment control scenarios that can be fed into the experiment controller or the experiment control server.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

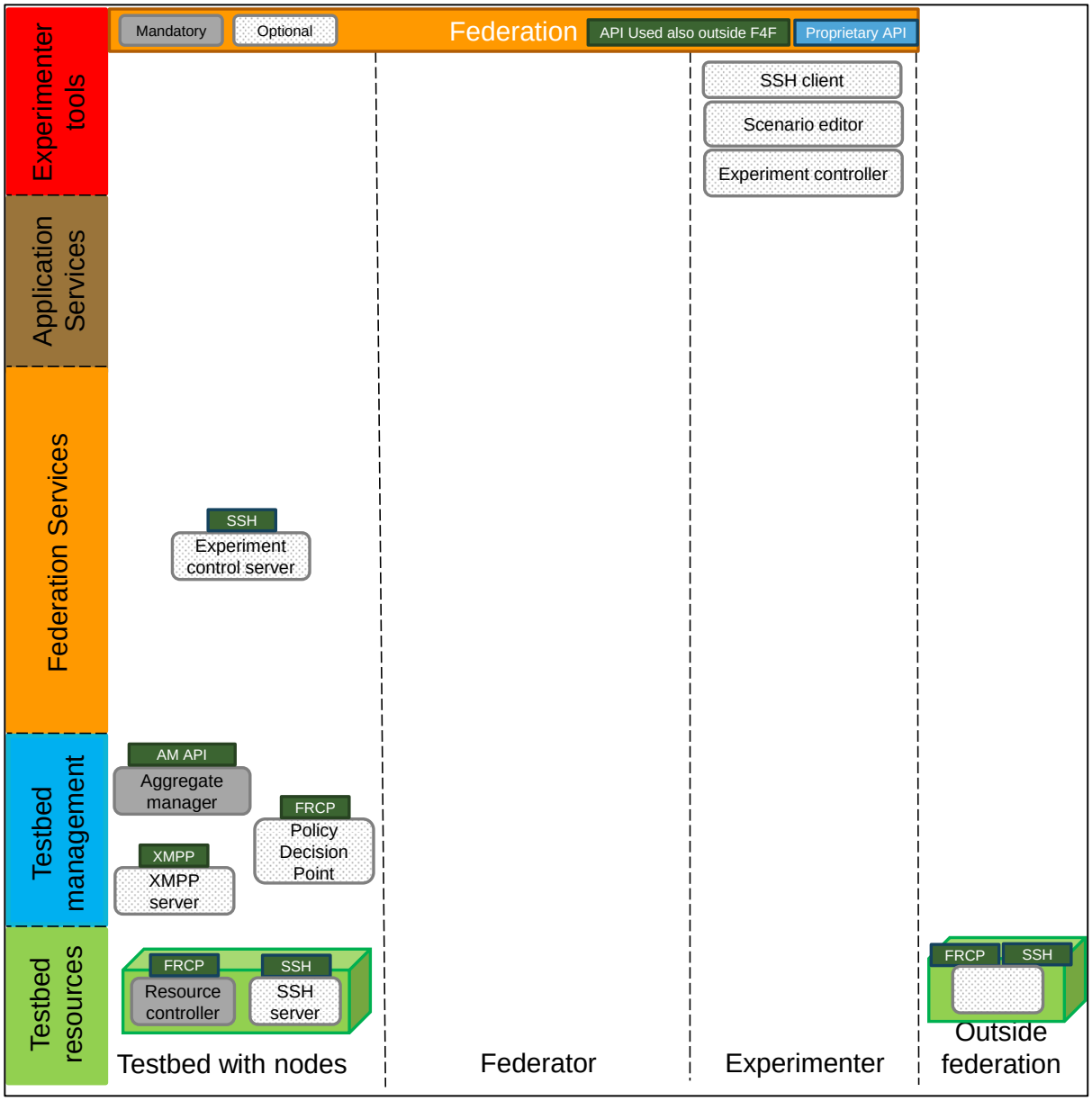


Figure 15: Cycle 2 architecture for Experiment Control

3.5.2 Sequence diagrams regarding experiment control

The simplest way to control an experiment is to log into the nodes with SSH, and to perform all corresponding actions manually. The corresponding sequence diagram is depicted in Figure 16. Note that it only introduces the corresponding interactions on a high level, more details should be given by WP5 and WP7 in their upcoming specification deliverables D5.2 and D7.2. The first step of login into the nodes with SSH is that during node provisioning, that the public SSH key of the experimenter is automatically copied from the AM to the resource (1). In the next step, the experimenter will login to the resource by connecting to its SSH daemon using his own SSH client (2, 3). Once connected, the experimenter can perform any action as he sees fit (4). Once finished, the experimenters logs out of this SSH session by disconnecting from the resource's SSH daemon (5, 6).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

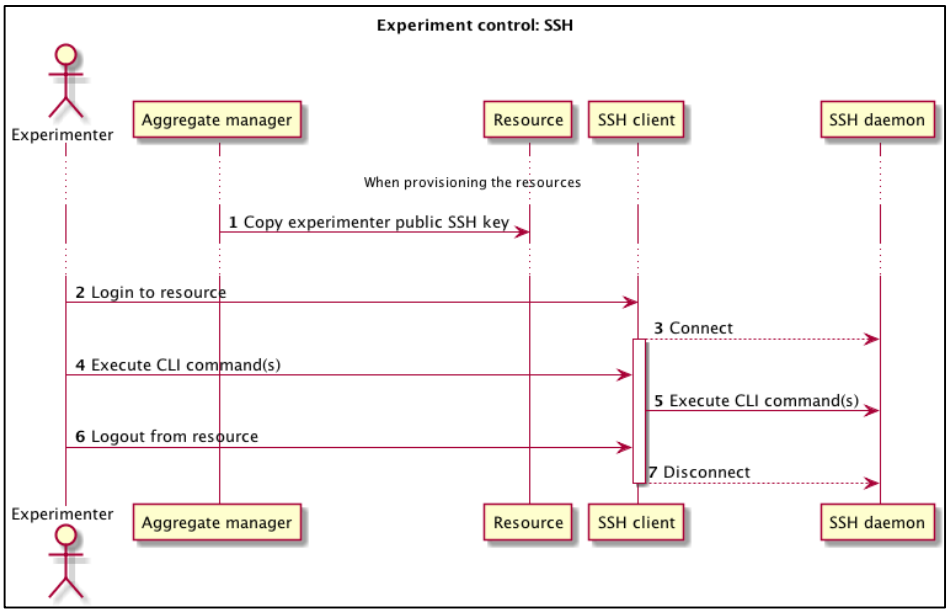


Figure 16: Sequence diagram for experiment control using SSH

The different steps involved in experiment control using an experiment controller are depicted in Figure 17. Note that it only introduces the corresponding interactions on a high level, more details are given in deliverables D5.2 and D7.2. When controlling the experiment with an experiment controller, first of all the Aggregate Manager programs the PDP with the needed information to allow it to make correct authorization decisions. More specifically, the AM informs the PDP about which resource belongs to which slice (1). The next step is then for the experimenter to define its scenario for experiment control (2), and to run it using its experiment controller (3). At due time, this will send the appropriate messages to the resource controller (RC) using the AMQP framework that is in place (4, 5). After receiving such a message, the RC will ask the PDP if it is OK for him to perform the requested action (6, 7). If so, the RC performs it (8). This is then repeated until the experiment control scenario has been entirely performed (9).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

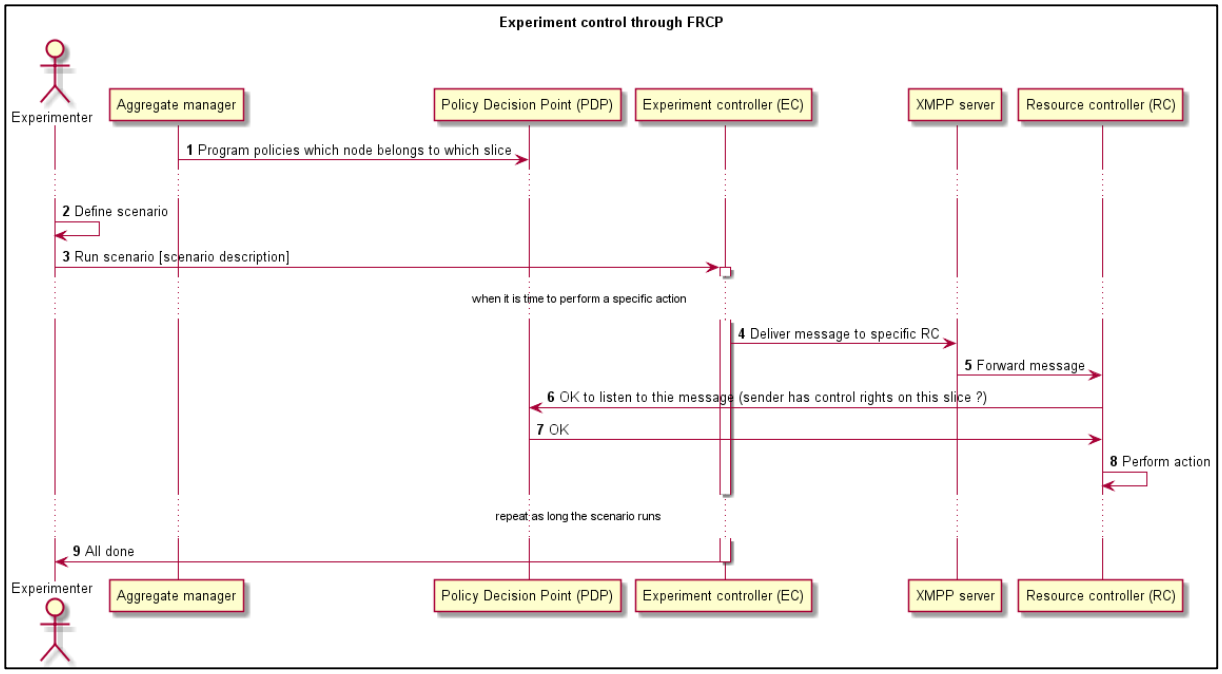


Figure 17: Sequence diagram for experiment control using an experiment controller

3.6 SLA management and reputation services

The part of the architecture that is presented here aggregates all architectural components needed to support SLAs and reputation services. The architecture itself is depicted in Figure 20. The remainder of this section will first introduce the SLA management lifecycle, discuss the adopted SLA management architecture and finally provide an explanation of the reputation service components.

One aspect that should be mentioned before diving deeper in the details of this part of the Fed4FIRE architecture is the fact that during the design it was intended to (if possible) base the Fed4FIRE SLA management solution on existing state-of-the-art SLA solutions such as e.g. Cloud4SOA [13]. Of course such a tool needs to be adapted and integrated with other Fed4FIRE architecture components, but it was perceived that this would definitely be more efficient then developing an SLA management framework from scratch.

To be a bit more specific regarding the Cloud4SOA project, this framework supports cloud-based application developers with multiplatform matchmaking, management, unified application, cloud monitoring and migration. It interconnects heterogeneous cloud offerings across different providers that share the same technology through the concept of adapter that provides a REST-based API for any cloud access. Because of its characteristics it was selected as the starting point for the Fed4FIRE SLA management framework.

3.6.1 SLA Management lifecycle

As depicted in Figure 18, a typical SLA lifecycle can be split in the following phases (which can be grouped and simplified depending on the implementation):

1. *SLA Template Specification*: For a service provider, a clear step-by-step procedure describing how to write an SLA template to provide a correct service description.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

2. *Publication and Discovery*: Publish the provider offer, the customer QoS needs, and possibility for the customer (experimenter) to browse/ compare offers.
3. *Negotiation*: Agreement on SLA conditions between the experimenter and the infrastructure providers.
4. *Resource Selection*: Depending on the chosen SLA, testbed providers select the resources that need to be assigned to the experimenter in order to meet this SLA. The resources can be explicitly selected by the experiment as well, depending on the testbed's policy.
5. *Monitoring and Evaluation of the SLA*: Comparing all the terms of the signed SLA with the metrics provided by the monitoring system, in order to internally prevent upcoming violations and to externally discover potential violations.
6. *Accounting*: This can cover different actions, such as invoking a charging/billing system according to the result, reporting, invoking a reputation system or an internal policy engine tool.

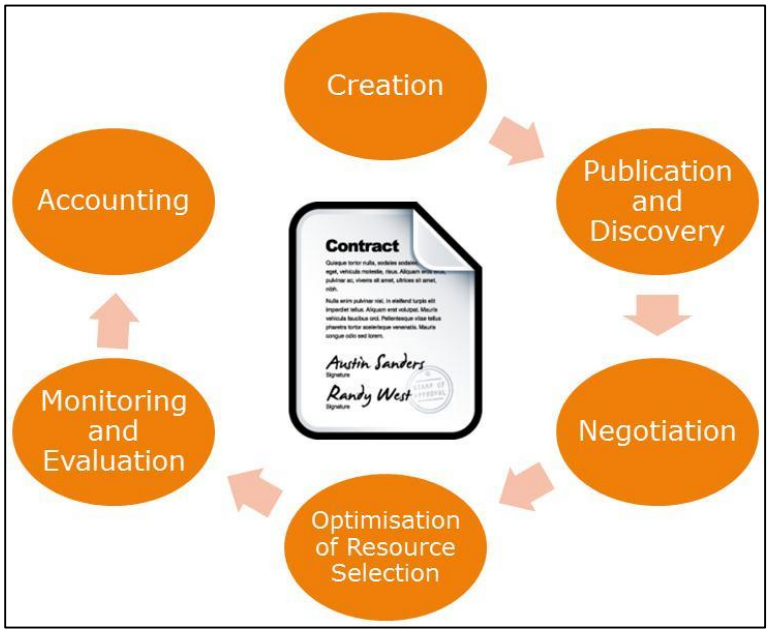


Figure 18: The SLA management lifecycle

3.6.2 SLA Management in Fed4FIRE

The first step of the architectural design of Fed4FIRE's SLA management framework was to evaluate the different possible architectural approaches. This analysis can be found in Appendix F. Based on the corresponding outcomes, the architecture could be defined. It is described in the remainder of this section.

3.6.2.1 Basic principles

In order to integrate SLA management in such a complex federation, Fed4FIRE has decided to implement a simple SLA mechanism that is useful and understandable both by the community of experimenters and by the testbeds. This approach has been inspired by common SLA approaches in the state of the art (e.g. the Amazon EC2 SLAs [14], which guarantees the service is available during a fraction of the service time).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

The main SLA mechanism that is being offered in the federation is to guarantee the availability of resources in a testbed. The defined SLA type is: SLAs guarantee X% availability for Y% of resources during the experiment.

On the other hand, SLA management in Fed4FIRE supports the definition of other types of SLAs including other types of metrics (e.g. CPU, memory) and/or more elaborated constraints. For testbeds to be able to offer more complex SLAs, they are required to be able to monitor and provide the needed metrics so that the SLA management system can evaluate them using the defined constraints.

The business case associated to SLAs in Fed4FIRE, i.e. what happens in case of violations, is very simple. A violated SLA is reported to both the testbed provider and the experimenter and will affect negatively to the testbed's reputation. The possibility of providing compensation quota on resources for the experiment will also be available for testbed providers that desire to implement it.

There will be one SLA per resource reservation between the experimenter and each one of the different testbeds involved in an experiment. The SLA will cover the set of assigned slivers as the RSpec Manifest (see 3.3.5) of a testbed, evaluating the defined constraints by retrieving monitoring information from either a central monitoring database of the federation. If the previous element is not available, the monitoring information could be retrieved instead from a local monitoring database of a testbed. At the end of the experiment, the global information of the SLAs evaluation of all the slivers of the same testbed for the same experiment will be available.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

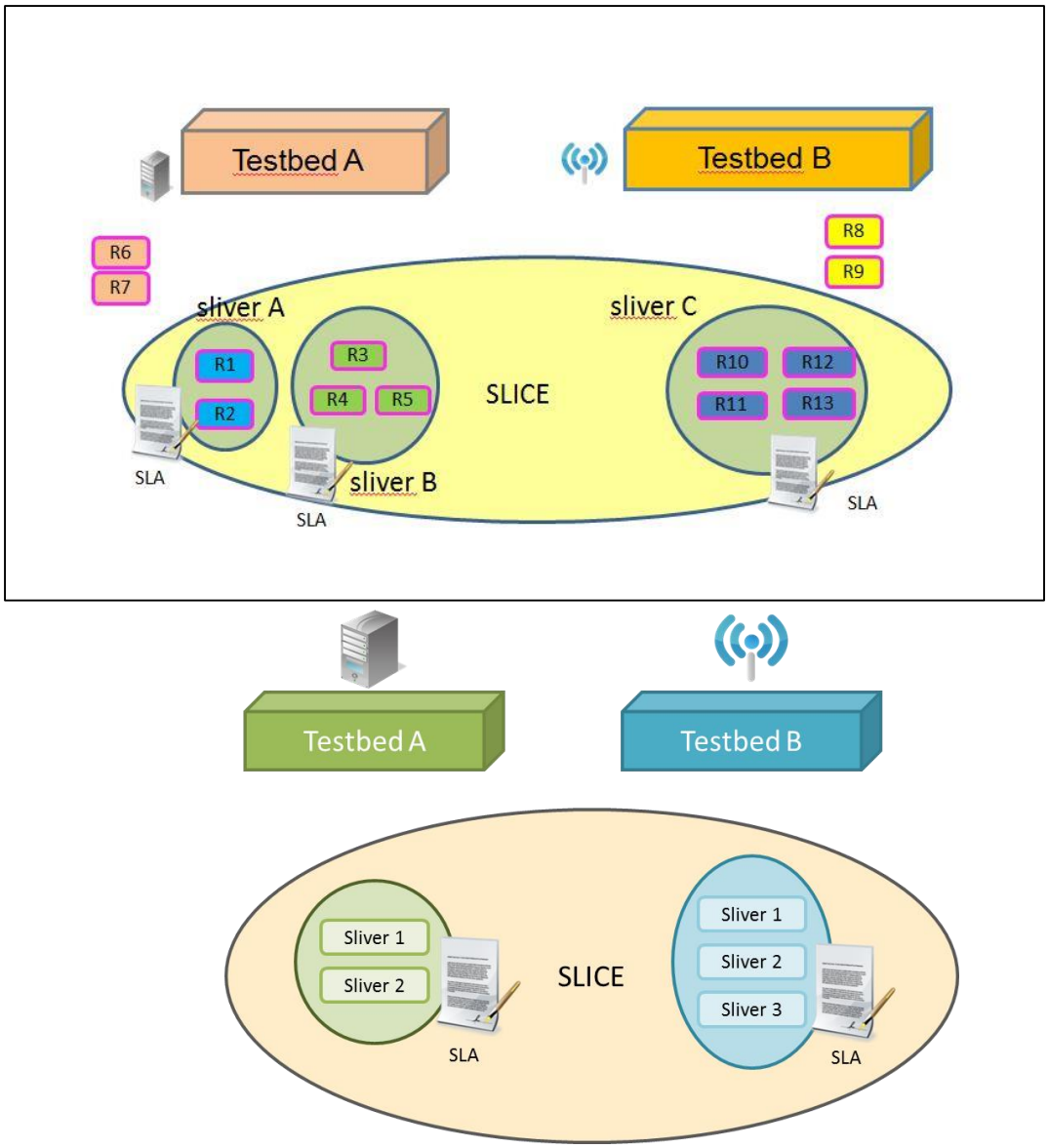


Figure 19: SLA agreements for different testbeds in one experiment

Some requirements needed in order to do the SLA Evaluation are:

- SLA Management should know when the slivers have been provisioned or released and which resources are contained in them. Once the resources of a sliver have been provisioned, they will not change during the entire sliver lifecycle. This is the Rspec Manifest.
- As soon as the experiment begins, the SLA enforcement starts (evaluation) and the SLA Management begins to retrieve monitoring data.
- The SLA Management stops the SLA evaluation at the end of the experiment. It is only at this moment when the evaluation can be requested by the experimenter.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

The evaluation of the SLA is calculated of the following way:

- The status of availability (UP, DOWN) of each resource will be stored by the SLA Management module at every SLA monitoring interval during the time it has been active.
- The uptime rate of each sliver is calculated of the following way:

  - The sum of all the availability values (UP = 1, DOWN = 0) obtained of each resource divided by the SLA monitoring interval during the time this resource has been active.
  - The SLA is met when the X% uptime rate defined in the SLA is fulfilled in at least Y% of the resources.

- Finally, the SLA Evaluation of the different testbeds is shown to the experimenter, who might also request the SLA performance of each sliver for a specific testbed as soon as the slivers are released.

In Cycle 2, the only way an experimenter could accept the SLA offered by a testbed and view the evaluation results once the experiment has finished was to use the Fed4FIRE portal. With the SLA architecture for Cycle 3, the RESTful API of the SLA Collector will be extended in order to allow any federated client to create a new SLA and obtain the evaluation results after the experiment has finished.

The responsibility of knowing the SLA id and start and stop the SLA evaluation through the SLA Collector module is delegated to the client tool, as opposed to the AM of each testbed. Furthermore, one SLA will cover the reserved slivers from a single testbed, instead of resources associated to one sliver. This is because of resources are mapped to slivers in a different way for each testbed (which complicates the later resource-sliver association in SLAs).

Note that SLAs apply to both instant and future reservations. At the end of the experiment, one can evaluate if the uptime/availability of all promised resources was met.

More details about the SLA management for cycle 3 can be found in deliverable D7.4 – Detailed specifications regarding trustworthiness for third cycle.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

3.6.2.2 SLA architectural components

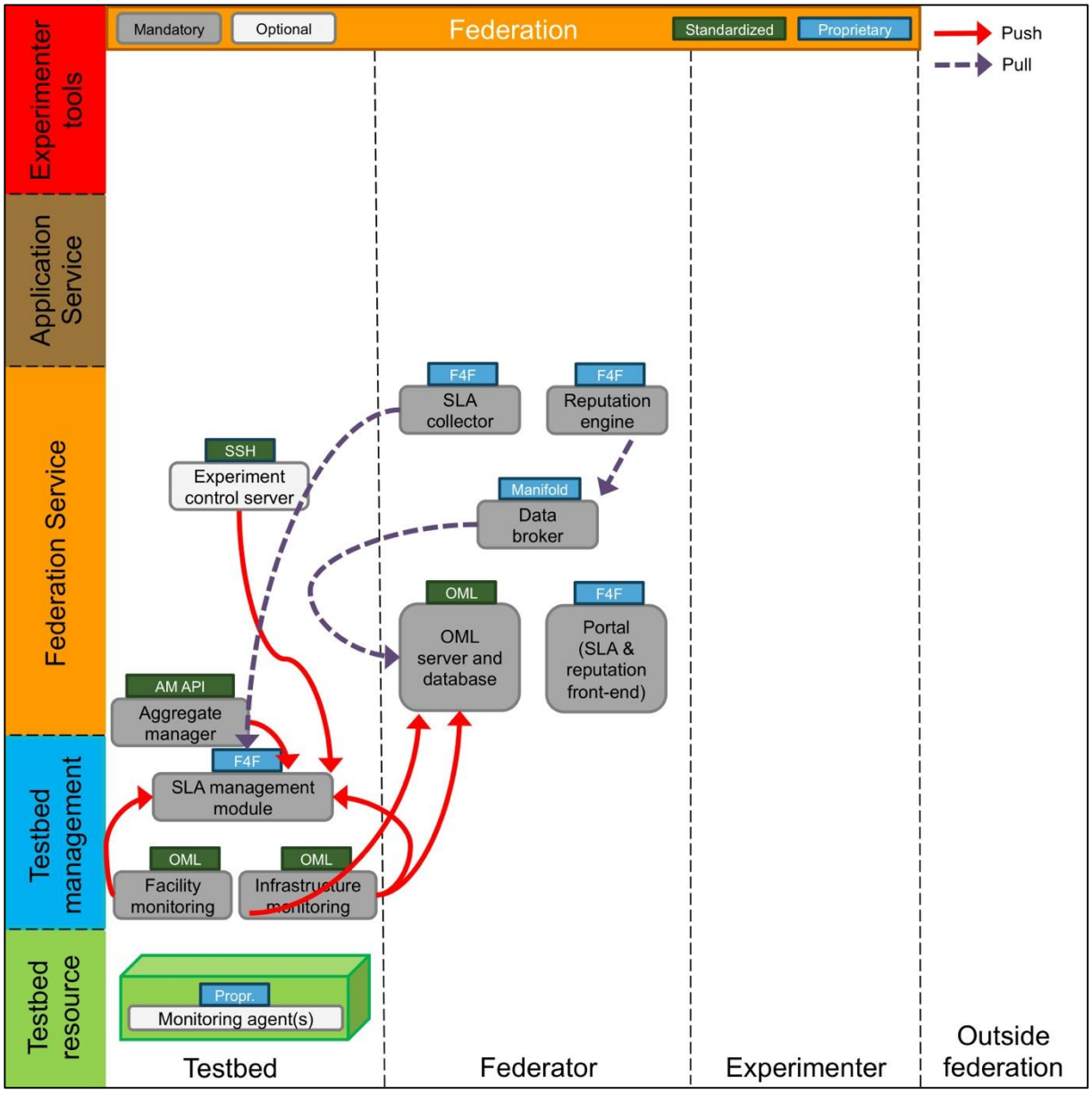


Figure 20: SLA and reputation architecture

As shown in Figure 20, the distributed SLA architecture is composed of four main parts:

- **SLA front-end tool**: the goal of this component is to show the functionalities of the SLA Management to the experimenters through a Graphical User Interface. The SLA front-end tool is also responsible to do the correct request to the SLA Collector and then gather and show the results. To implement the SLA Front-end tool, some plugins will be developed and integrated in the Portal and in the different front-end tools. As shown in Figure 19, the SLA Front-end tool is not envisaged to be a standalone tool, but it is considered to be provided as an integrated part of the portal.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- **SLA Management Module**: this component is responsible for supervising if all the agreements reached are respected. It receives and processes all measurements related to the SLA from the monitoring system. It validates whether the measurements are within the thresholds established in SLA agreement metrics. In case the testbed does not fulfill these conditions during the execution of the experiment, it triggers the appropriate actions in case of violations.
- **SLA Collector**: this component acts as a broker between the different client tools and the different SLA Management modules of each testbed. The goals of this component are:
  - Gather the different warnings and the evaluations when the experimenter wanted.
  - The SLA collector can also provide SLA information to the reputation service if required.
- **SLA Dashboard**: this component allows testbed providers to define their own SLA templates (upon which the actual SLA will be created once the experimenter accepts it) and visualize the status of the active SLAs on their facilities.

The SLA Dashboard is a tool foreseen for service providers to design and implement their own SLAs. This will ensure some independence for testbeds willing to adopt SLAs, who, until now, were assisted by Atos in order to build their commercial offer.

Each testbed provider (SLA-Administrator) will be able to create its SLA templates. The testbed user, also named Experimenter, will be able to select (with another tool) the template 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.

Figure 21**Error! Reference source not found.** shows how the SLA Dashboard will interact with the testbed. The SLA Collector will have in its database the information the IP of each existing testbed. If a testbed provider has several testbeds in the federation (for example, iMinds has two testbeds), the tool will centralise the SLA template setup for all the involved testbeds.

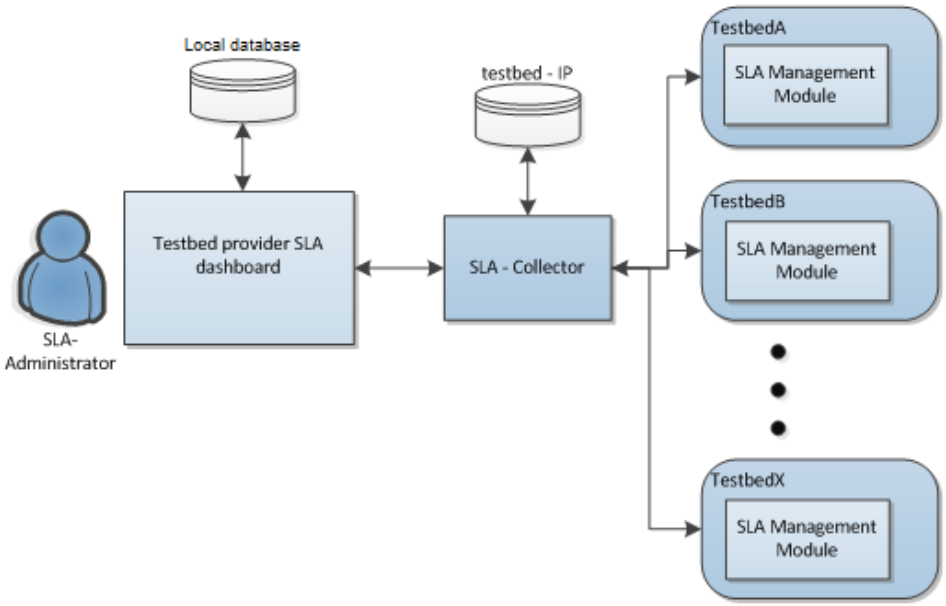


Figure 21: SLA-Dashboard in the Fed4FIRE architecture

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

The SLA Dashboard should store the information to authenticate the SLA Management Module (e.g. iMinds) and associate it to the testbeds (e.g. Virtual Wall, WiLab2).
The SLA-Dashboard will be a web-based application. Any provider should be able to access from any computer that has a connection to the SLA Collector. It will expressly not be integrated with the Fed4FIRE portal. SLA Dashboard is considered as an independent tool where it is possible to have a higher control from where it is accessed. It is foreseen that the Fed4FIRE federation would like to limit the external IP's that have access to this tool or only want allow access within the Fed4FIRE federation; such kind of policies will be easier to implement with and independent tool.

More details on the SLA Dashboard can be found in deliverables D5.4 – Detailed specifications for third cycle ready and D7.4 – Detailed specifications regarding trustworthiness for third cycle.

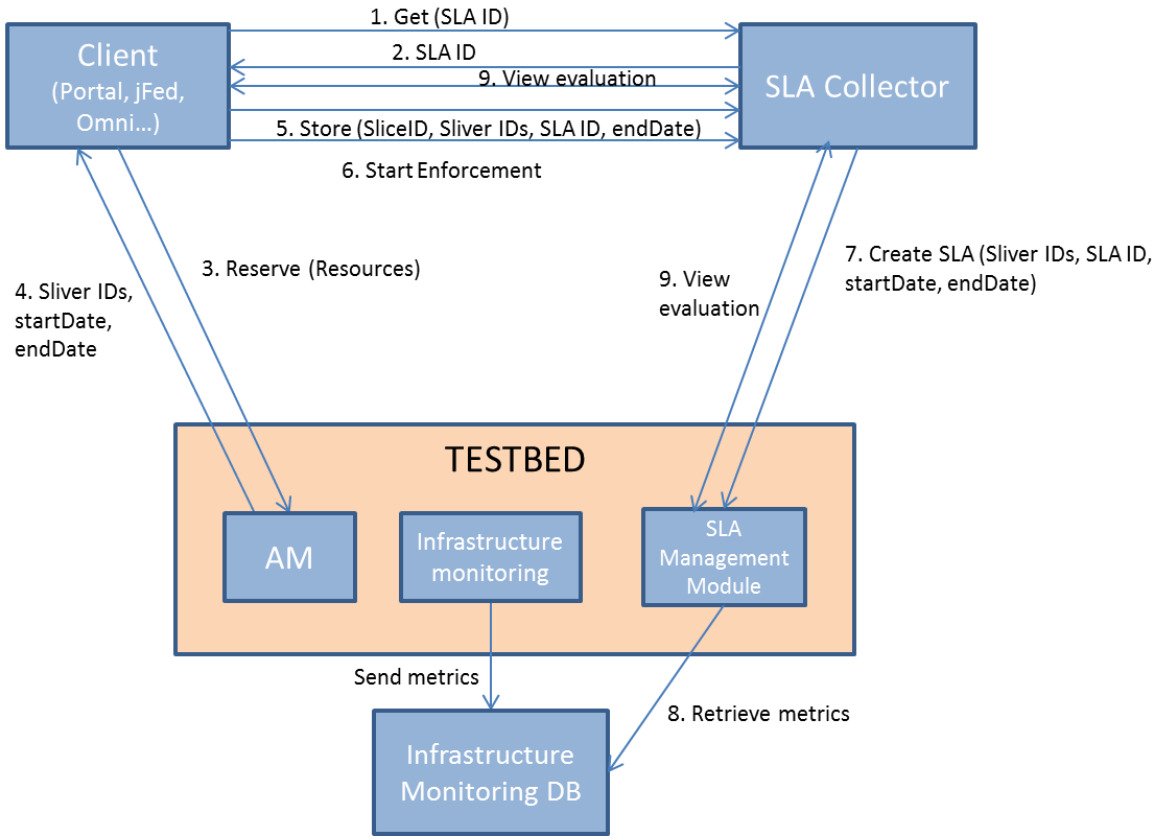


Figure 22: SLA creation and visualization process

3.6.2.3 Illustration of the Fed4FIRE SLA workflow

The SLA workflow is depicted in the following picture. The main concept is that the experimenter will accept the SLA after selecting the resources (browsing the testbeds offering SLAs is also possible) and, once the resources are provisioned, the SLA evaluation is based on monitoring information for those particular resources. Once the sliver is released, he experimenter can request the SLA evaluation and, for those slivers having failed, the specific violations can be shown.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

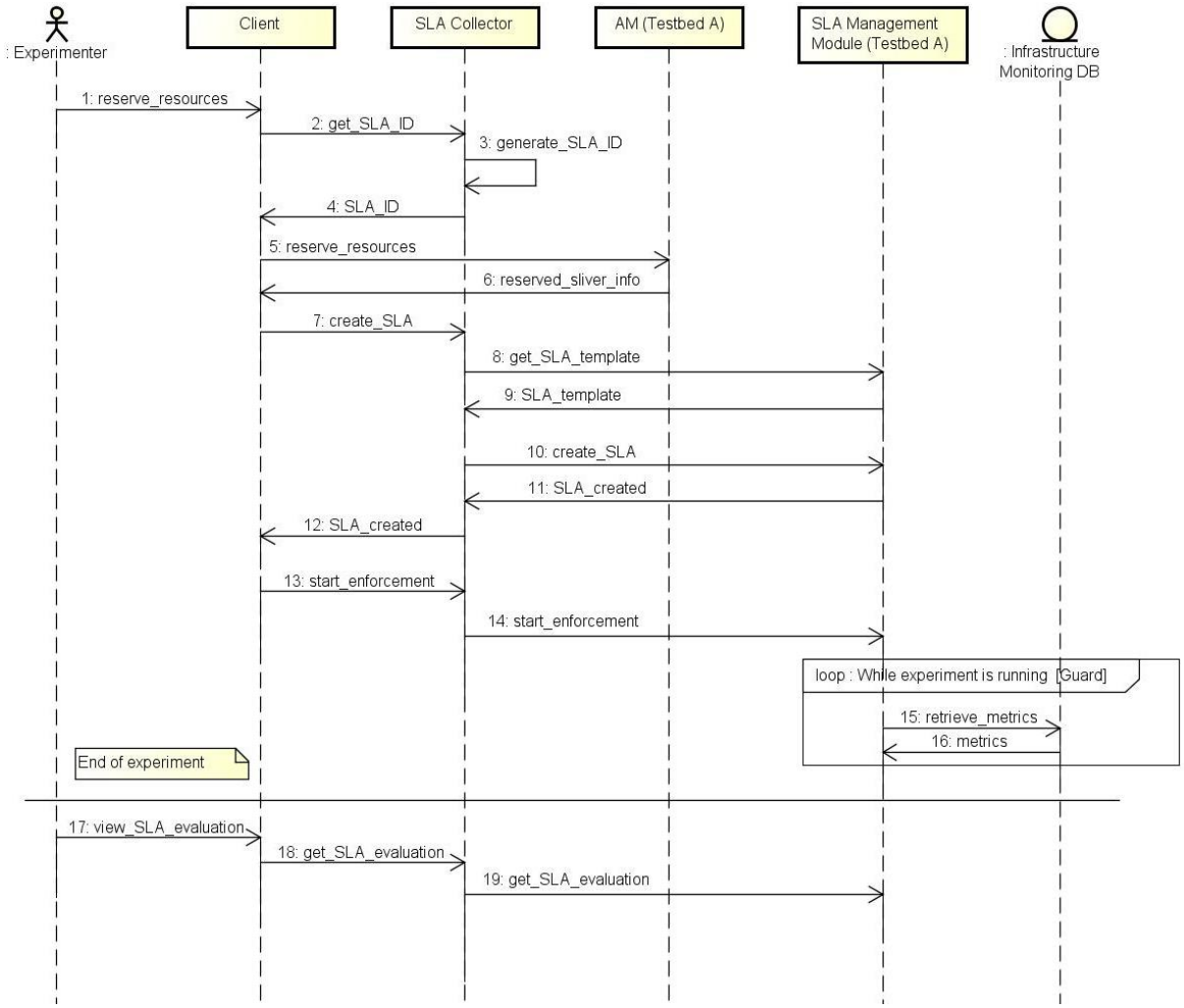


Figure 23: The SLA management lifecycle

3.6.3 Reputation Service in Fed4FIRE

The reputation service in Fed4FIRE aims to provide mechanisms and tools towards building trustworthy services based on the combination of Quality of Experience (QoE) and monitoring data. The developed mechanisms and tools will reflect the end users (experimenters) perspective with the objective of empowering the users/experimenters to select testbeds based on dynamic performance metrics. These metrics will offer a "smart" user support service that provides a unified and quantitative view of the trustworthiness of a facility.
In order to achieve that, the service will mainly focus on building reputation-based trust utilizing:

1. Raw monitoring data (e.g. information to experimenters on up-time, usage etc. that results into site popularity) and SLA information
2. Users' feedback regarding their Quality of Experience (QoE) and service received.

The main interactions of this service (depicted as the **reputation engine** on Figure 20) with the other Fed4FIRE components include communication with: (1) the monitoring data broker from which measurement data for the reserved resources will be obtained with respect to user's experiment (2) the SLA service from which will be obtained SLA information regarding violations, etc. and (3) the Portal

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

which will serve both as a place for displaying testbeds' reputation scores (statistics) and as a feedback page for users' Quality of Experience (depicted on Figure 20 as the **reputation front-end**, which is an integrated part of the portal).
More information regarding the reputation service is provided in deliverable D7.2 - Detailed specifications regarding trustworthiness for second cycle and D7.4 - Detailed specifications regarding trustworthiness for third cycle..

3.7 Layer 2 connectivity between testbeds

In Fed4FIRE different methods of layer 2 connectivity are supported:

- Stitching layer 2 vlans
  - Fully automatic
  - Halfway stitching to a stitching point
- Automatic (E)GRE tunnels
- Manual GRE tunnels or VPNS

3.7.1 Layer 2 stitching vlans

This method is based on VLANs which are provisioned and then stitched together at points where they meet. The figure below shows the workflow:

- The experimenter draws in an experimenter tool a link between two nodes on different testbeds (which is translated in an RSpec)
- When the tool starts provisioning, it first calls the Stitching Computation Service (SCS) which calculates a route between the two testbeds based on the layer 2 paths it knows. The SCS augments the RSpec with this information
- The tool then knows also intermediate hops in the path (e.g. Geant, internet2) and can call them to set up the path.
- In the end, all parts of the links and nodes become ready, and the experiment is ready.

For this fully automatic stitching, the vlan numbers are dynamically chosen based on free vlan overviews, tries and retries.
The SCS software can be found at https://wiki.maxgigapop.net/twiki/bin/view/GENI/NetworkStitchingSoftware and in Fed4FIRE an SCS is available at http://scs.atlantis.ugent.be:8081/geni/xmlrpc. To see the detailed calls, and augmented RSpecs, one can have a look at the jFed Probe and jFed experimenter GUI to see this in action.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

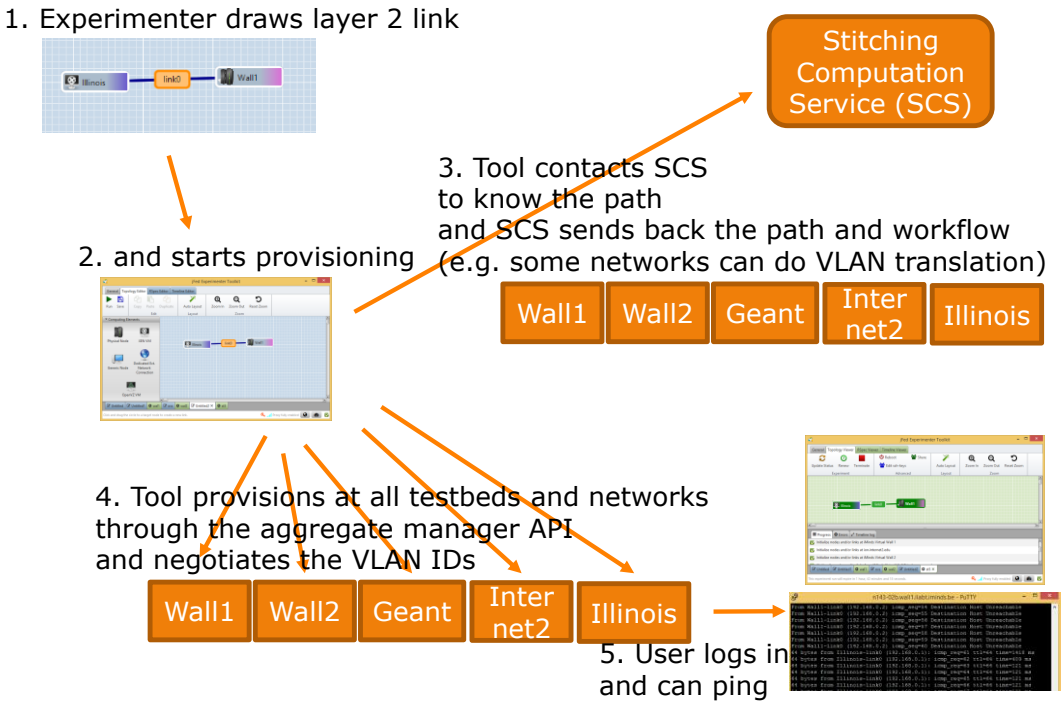


Figure 24: Automatic layer 2 stitching

For testbeds which do not support this fully automatic stitching end-to-end (including vlan choice), we have implemented also halfway stitching through the 'dedicated external network connection' icon in jFed, see figure below. In this case it is possible to select manually the testbed and vlan of the destination, eg.. i2cat vlan 188. Based on documentation, the experimenter has to know where vlan 188 at i2cat is connected to (e.g. port 23 on openflow switch C).

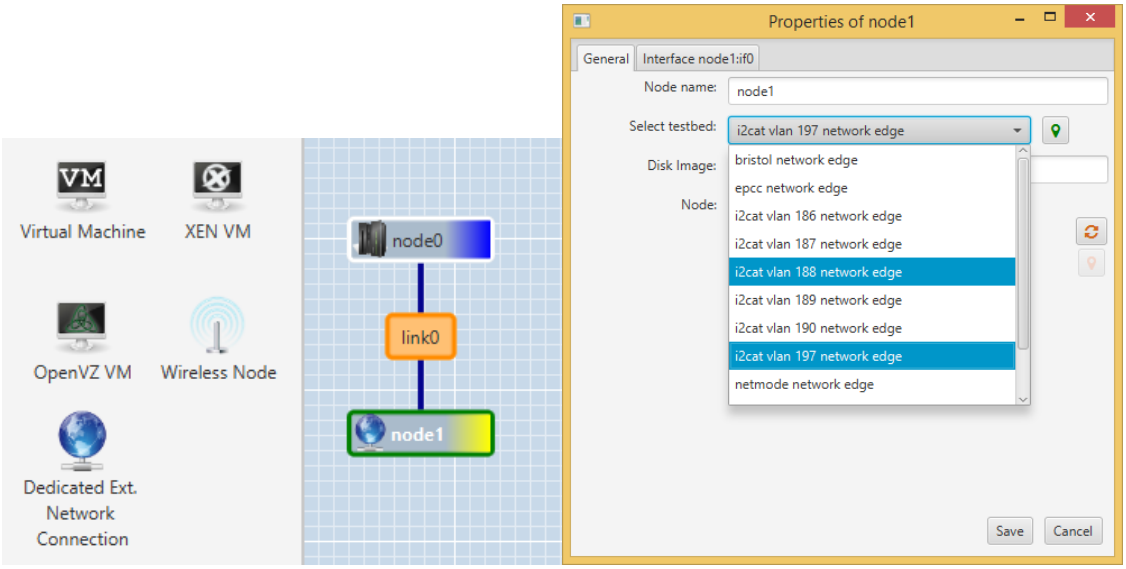


Figure 25: Halfway layer 2 stitching

The advantage of this layer 2 stitching, is that we have an end-to-end native layer 2 path with reserved bandwidth.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

3.7.2 Automatic (e)GRE tunnels

The emulab based testbeds like Virtual wall and w-iLab.t also make it possible to use automatic GRE tunnel setup. For this, it is possible to set in jFed the link-type to gre-tunnel, as shown below. The advantage is that the testbeds set up the tunnel for you and configure the IP addresses.

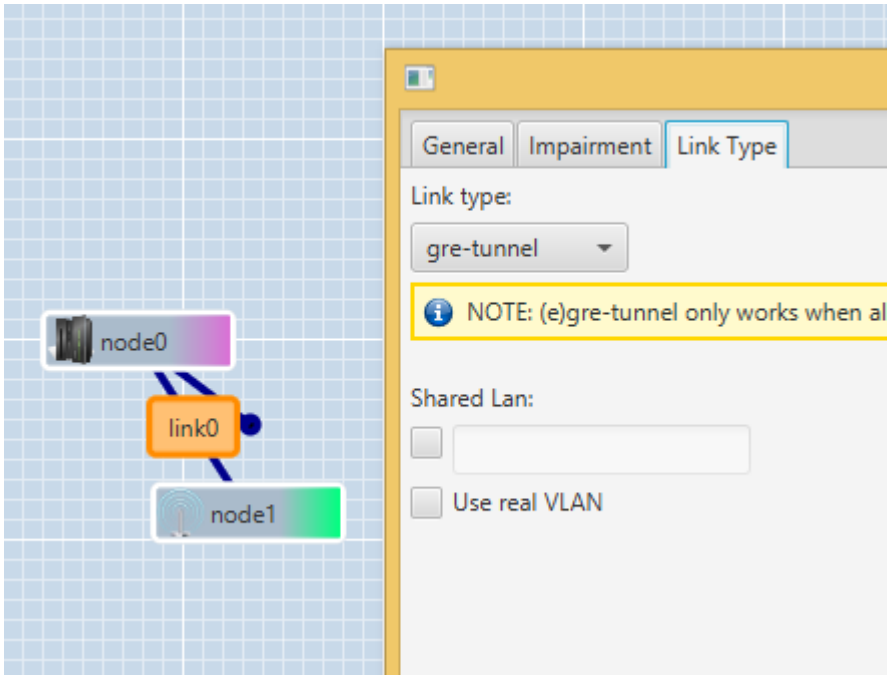


Figure 26: Automatic GRE tunnel in jFed

And the link in the RSpec looks then as follows:

```
<link client_id="link0">
  <component_manager name="urn:publicid:IDN+wall1.ilabt.iminds.be+authority+cm"/>
  <component_manager name="urn:publicid:IDN+wilab2.ilabt.iminds.be+authority+cm"/>
  <interface_ref client_id="node0:if0"/>
  <interface_ref client_id="node1:if0"/>
  <link_type name="gre-tunnel"/>
  <property source_id="node0:if0" dest_id="node1:if0" capacity="10000"/>
  <property source_id="node1:if0" dest_id="node0:if0" capacity="10000"/>
</link>
```

Figure 27: Automatic GRE tunnel in RSpec

The cave-at for this, is that the IP addresses of the nodes should be fully reachable for each other, typically meaning public IP addresses at both sides. (for virtual machines, the hypervisor should have a public IP as this is the one setting up the tunnel)

3.7.3 Manual GRE tunnel setup

An experimenter can also set up manually a GRE tunnel between two nodes, on the condition that the IPs are reachable for each other (firewall open, direct routing)

This is as simple as:

```
Node A:
ip tunnel add netb mode gre remote 172.19.20.21 local 172.16.17.18 ttl 255
ip link set netb up
ip addr add 10.0.1.1 dev netb
ip route add 10.0.2.0/24 dev netb
```

Node B:

```
ip tunnel add neta mode gre remote 172.16.17.18 local 172.19.20.21 ttl 255
ip link set neta up
ip addr add 10.0.2.1 dev neta
ip route add 10.0.1.0/24 dev neta
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Other setups, e.g. using openvpn or tinc are of course also possible.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

4 Compliance and federation of testbeds with Fed4FIRE

For cycle 3, we have now also defined in detail the categories for technical compliance of testbeds with Fed4FIRE (this does not define the policies).

4.1 Definition of testbed types

A testbed is a combination of hardware and testbed management software.
We make a difference in two types of testbeds which could join the federation or be compatible with Fed4FIRE:

- Type A: Testbeds with 'SSH/FRCP/openflow controlled resources':
  - Ability to share resources between different users
  - Shared over time or in parallel (multiplexing, slicing)
  - Concept of credentials and dedicated access (e.g. ssh)
- Type B: 'API only' testbeds:
  - A service with an API (proprietary or standard)
  - Concept of credentials

NOTE: we avoid discussions and paradigms as service or resource testbeds as 'Everything is a resource' and 'everything is a service' can confuse people too much.

The distinction between the two testbed types is relatively simple to understand. If you have resources where you can log in with SSH, that can be controlled with FRCP or with openflow, then you have a type A testbed. If you have a testbed with an API with resources not belonging to type A, then it is a type B testbed.
E.g. the Virtual Wall which provides physical or virtual machines with SSH access is type A, while SmartSantander providing a proprietary REST API to fetch the measurement results is a type B testbed.

4.2 Types of federation

We define for both testbed types three types of federation:

- Associated testbeds
- Light federation
- Advanced federation

4.2.1 Associated testbeds

This is a very basic step, and basically means we mention the testbed on the Fed4FIRE website.

- No real federation (e.g. no credential exchange, no testing, …)
- Only mentioning the testbed on the Fed4FIRE website and linking to the testbed specific documentation
- Testbed has to organise its own support

4.2.2 Light federation

Light federation is the same for type A and type B testbeds. The testbeds need to fulfill the following:

- Support for Fed4FIRE credentials in a client based SSL API (clients are authenticated based on the certificate they use in the SSL connection)
  - X.509 certificates, e.g. derived PKCS12 version which can be loaded in a webbrowser or other HTTPS tool
  - API is not the AM API

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- Testbed has to provide documentation for experimenters (on a webpage maintained by the testbed)
  - Testbed description
  - Documentation on the specific API
  - URLs of the API
  - A basic experiment showing the testbed, in a tutorial format
- Policies: everyone with a valid F4F certificate can execute the basic experiment that is document without extra approval
- Facility monitoring
  - API will be tested from central location, if testbed has internal monitoring, send a summary through OML to the central OML server
- Connectivity: a public IPv4 address is needed for the API server

The Fed4FIRE federation offers the following to the testbed:

- Test credentials for testing federation
- Information on enabling PKCS12 authentication (see Appendix G)
- Central monitor dashboard
- Min. 1 client tool exporting PKCS12 credentials from the X.509 certificate
- At least 1 authority to provide credentials
- Central documentation linking to all testbeds
- Central support (google group, NOC) for first help and single point of contact

This light federation makes it possible to have a simple way to federate with Fed4FIRE and as such testbeds can easily join ad-hoc and dynamically for a short time.

4.2.3 Advanced federation

Here we will make a difference between advanced federation for type A ('SSH/FRCP/openflow controllable resources') and type B ('API only') testbeds.

4.2.3.1 Advanced federation for type A ('SSH/FRCP/openflow' controllable) testbeds

- Support for AMv2 or AMv3 (or later versions)
  - Authentication, authorization: X.509 certificates, slice and user credentials, accepting root certificates of the main F4F authorities
  - Resource description and discovery: RSpec definition
  - Provisioning (instant): through the AM API
  - Control: through SSH with ssh public/private keys put in the API calls, FRCP control or openflow: point a controller for a switch
- Documentation (on a webpage maintained by the testbed)
  - Testbed description
  - RSpec description
  - URLs of the AM API
  - A basic experiment showing the testbed (and with a F4F tool), described as a tutorial
- Policies: everyone with a valid F4F certificate can execute the basic experiment without extra approval
- Facility monitoring
  - AM API tested from central location, if testbed has internal monitoring, send a summary through OML to the central OML server

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- Connectivity: public IPv4 for AM, public IPv4 or IPv6 for ssh login (exceptions for VPN can be granted, but then the ssh gateway of the F4F federation will be a permanent client of the VPN)
- Testbed has to provide basic support on the testbed functionalities towards experimenters

Optional features that a testbed can support in this Advanced federation:

- Infrastructure monitoring (details in WP6)
- Advanced reservation (details in WP5)
- SLA (details in WP7)
- Reputation (details in WP7)
- Permanent storage (not yet defined)
- Experiment control
  - FRCP enabled images
  - AMQP server
  - Policy decision point (PDP)
- Layer 2 connectivity between testbeds
  - VLAN stitching (federation runs stitching computation engine)
  - Tunnels (egre or gre option in RSpec link)

The Fed4FIRE federation offers the following to the testbed:

- Testing tools for the AM API, test credentials, ...
- Nightly testing when federated
- Central monitor dashboard
- Min. 1 client tool having support for all federated infrastructure testbeds
- At least 1 authority to provide credentials
- Ssh gateway (to bridge e.g. to IPv6, VPNs, ...)
- Central documentation linking to all testbeds
- Central support (google group, NOC) for first help and single point of contact

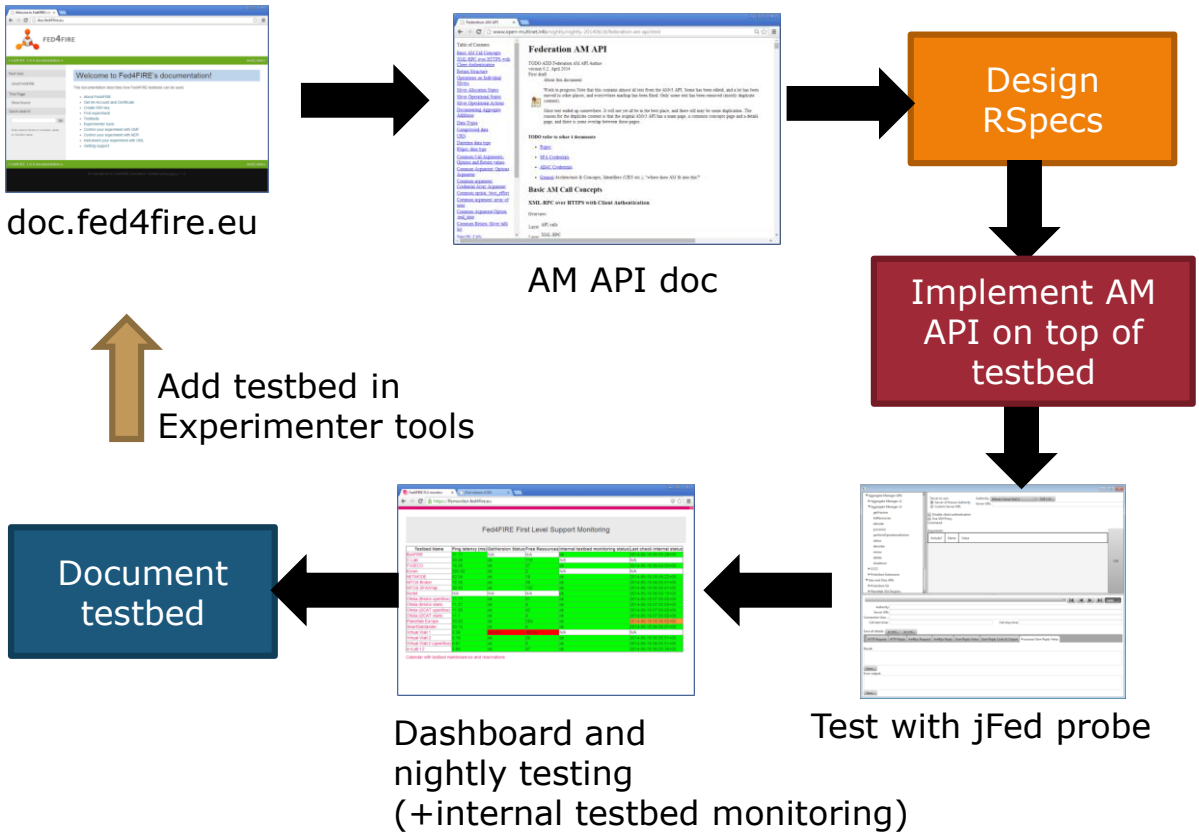
4.2.3.2 Advanced federation for type B ('API only') testbeds

In cycle 3, service orchestration will be implemented based on the 'YourEPM' (Your Experiment Process Model) tool which is designed to provide high level service orchestration for experimenters, based on open standards such as BPMN (Business Process Model and Notation) and BPEL (Business Process Execution Language).
YourEPM is presented as an independent tool available as a central component in the federation that allows experimenters to orchestrate services offered by one or several testbeds by means of a web GUI, automatically obtaining the required information to make use of those services. YouEPM tool will look for available services in the Service Directory, retrieving the service description from the specific URL provided by each testbed. The communication with the services from the tool will be done using general wrappers to specific technologies (i.e. REST, SFA). This tool may also be integrated with jFed tool to extend the orchestration to include testbed resources.
In order for YourEPM to use application services available in the federation, type B testbeds which want to have an advanced federation with Fed4FIRE have to provide a description of the service API, so the tool can invoke it automatically. Experimenters will then have the possibility of using YourEPM tool, which will be linked from the Fed4FIRE portal, to use and orchestrate those services.

4.3 Workflow for federation

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

The scheme below highlights the typical workflow for a new testbed, starting with the documentation of existing testbeds from an experimenter perspective to learn how things should work.



© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

5 Main differences with cycle 1 architecture and D2.1

Throughout this deliverable, the description of the third federation architecture of the project has been intentionally described as something on its own, without referring to how the architecture looked like before. The reason for this is that we wanted to make it easy for people novel to the project to understand exactly how our current architecture looks like and operates. We feared that focusing on the deltas would possibly confuse the reader.

However, we are aware that for people that have been involved in the project since its very beginning, it is interesting to get a clear view on how the architecture for cycle 3 compares to that for cycle 1. The most important differences are therefore listed below:

- Definition of member and slice authority instead of identity provider
- Definition of federation model and introduction of the concepts outside of the federation
- Definition of slice and sliver
- Removal of the brokers layer and introduction of the federation services layer and application services layer
- Federator instead of central location(s)
- Definition of currently three authorities in the Fed4FIRE federation
- Rename certificate directory to authority directory
- Introduction of documentation service in the federator
- Rename testbed directory to aggregate manager directory
- The cycle 1 architecture did not capture some finer details defined by WP5 regarding experiment control: it does not indicate that FRCP is the adopted API for this functionality, and that an experiment controller is to be foreseen on every resource. Also, this adopted approach needed to be made more secure when transferring it from the single-testbed to the testbed federation domain.
- Similarly, the first architecture did not yet capture the decision to implement facility monitoring by exporting the corresponding data as OML streams, and collecting them in a central OML server that is queried by the FLS dashboard.
- As requested by the WP4 community, services were contemplated. This resulted in the addition of the application services layer next to the federation services.
- Even in cycle 1 some architectural components were optional, others were mandatory. However, the cycle 1 architecture didn't depict them in a different way.
- In experiment control, the PDP component was introduced.
- Sequence diagrams were introduced to show the detailed workflow of the particular parts of the architecture (on some parts D2.1 was lacking these details).
- SLA management and reputation services were introduced.
- A detailed setup was introduced for the First Level Support dashboard.
- Detailed technical examples of the concepts of delegation, handover between AM and PDP and subauthorities as a base for distinction between the experimenters have been introduced in cycle 2 (appendices C, D and E).
- Description in detail of the workflows for setting up an account, creating SSH keys and doing a first experiment are included (appendices G, H and I).

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

6 Main differences with cycle 2 architecture and D2.4

Throughout this deliverable, the description of the third federation architecture of the project has been intentionally described as something on its own, without referring to how the architecture looked like before. The reason for this is that we wanted to make it easy for people novel to the project to understand exactly how our current architecture looks like and operates. We feared that focusing on the deltas would possibly confuse the reader.

However, we are aware that for people that have been involved in the project since its very beginning, it is interesting to get a clear view on how the architecture for cycle 3 compares to that for cycle 2. The most important differences are therefore listed below:

- The AM API and RSpecs have been detailed at https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html and are open for external improvements and discussions at https://github.com/open-multinet/federation-am-api to have the best chance for sustainable adoption
- Infrastructure monitoring is now further finegrained as global infrastructure monitoring and experiment specific infrastructure monitoring
- The SLA architecture has been further refined
- There is now a clear definition of 2 types of testbeds ('SSH/FRCP/openflow controllable' testbeds and 'API only' testbeds) and 3 levels of federation/compliancy with Fed4FIRE, together with the necessary documentation.
- Based on input of the First Level Support task, the dashboard and testing for FLS has been updated
- For layer 2 connectivity, stitching of VLANs and GRE tunnels are now supported and described
- The requirement inputs and evaluation has been updated to reflect the architecture of cycle 3
- Appendices have been updated with current screenshots, tutorial, etc.
- An appendix has been added with information on how to provide facility monitoring to the First Level Support dashboard

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

7 Requirements which are fulfilled with the architecture in cycle 3

This goal of this chapter is to analyze to which degree the federation architecture for the second cycle of the projects meets all requirements and constraints imposed on it in chapter 2. This allows us to assess where we are with the project, and to identify the issues that must definitely be addressed in cycle 3. In order to keep the length of this deliverable within reasonable limits, it was chosen to include this analysis formation into the different related appendices:

- Appendix A: Requirements from the infrastructures, services and applications communities (D3.4/D4.4)
- Appendix B: Requirements from SLA management

To annotate the degree in which a requirement is fulfilled in this cycle 3 architecture, the requirements in the appendix have been color coded to show which are fulfilled by the architecture and which are not:

- In green, the requirements which will be fully fulfilled by the proposed architecture
- In orange the requirements which are partially fulfilled
- In red the requirements which are not yet fulfilled.

In the table below, the counts of these color-coded annotations are summarized, in order to give a quick overview of the big picture in terms of covered requirements in cycle 3. Note that for some sources the requirements were presented in a more aggregated fashion, resulting in lower absolute numbers in terms of total requirements (e.g. WP3/WP4 did this for its requirements on experiment workflow and lifecycle management, measurement and monitoring, trustworthiness and interconnectivity). Other inputs have chosen to present their requirements in a much finer-grained fashion (e.g. the SLA requirements), resulting in much higher totals. So comparing the absolute numbers between requirements categories does not make much sense. But the spread per category gives a good estimate of how well certain aspects of the overall desired Fed4FIRE federation framework are tackled today.

Requirements (see Appendices) have been scored now based on input of open call experimenters.

Table 1: Overview of how well the different requirements are fulfilled by the architecture of cycle 3

Requirements category	Total # requirements	# covered	# partially covered	# not covered
Experiment workflow and lifecycle management	42	30	8	4
Measurement and monitoring	17	9	6	2
Interconnectivity	6	3	2	1
SLA requirements	29	17	0	12

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

8 Conclusion

This third deliverable of task 2.1 defines the architecture for cycle 3 of the Fed4FIRE project.
It takes as input requirements of WP3 (Infrastructure community), WP4 (service community), WP8 (First Level Support), SLA management and task 2.3 (sustainability) and defines an architecture that copes with as much requirements as possible and will be implemented for cycle 3 of Fed4FIRE. It is an evolution based on the cycle 1 and cycle 2 architectures.

In chapter 2 all these requirements are listed, while in chapter 3, we describe the architecture for the different steps in the experiment lifecycle (discovery, specification, reservation, provisioning, monitoring and measurement, experiment control and SLA management). For cycle 3, just as in cycle 1 and 2, discovery, specification and provisioning will be done based on the SFA GENI AM API v3 standard, while for advanced reservation and extended policy based authorization extensions are currently being made. Regarding the RSpecs, in cycle 3 each testbed can still provide its own RSpec to be used and tools have to be adapted to these RSpecs, but there is a trend to equalize the RSpecs when testbeds have the same kind of resources.

The architecture defines also multiple identity providers with a chain of trust, and a central portal accompanied with an identity provider and directories (machine and human readable) for tools, testbeds and certificates.

For First Level Support, the architecture defines facility monitoring which should be identical for all testbeds and should make it possible for first level support to have a high level overview of testbed and resource availability.

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

For SLA management, a first architecture is introduced which is based on monitoring information coming from the testbeds. This should make it possible to gain experience with simple SLA evaluation.

There are now defined standard ways and levels of federation.

Chapter 5 highlights the main differences with the cycle 1 architecture and D2.1, while chapter 6 highlights the differences with the cycle 2 architecture and D2.4.

Chapter 7 lists a short summary about the requirements which are fulfilled and which are not, and refers to the appendix for the detailed list.

As a general conclusion, this 3rd cycle architecture is an evolvement of cycle 1 and 2, with lots of improvements, clarifications and additions on multiple points, but the basic ideas remain the same, which means that implementation and deployment do not have to change radically but will be further go to unified interfaces and workflows.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

References

[1] B. Vermeulen, W. Vandenberghe, T. Leonard, et al. “D2.1 – First federation architecture”, deliverable of the FP7 Fed4FIRE project, December 2012

[2] C. Scognamiglio, M. Sioutis, S. Bouckaert. “D3.2 – Infrastructures community federation requirements, version 2”, deliverable of the FP7 Fed4FIRE project, December 2013

[3] F. Lobillo, M. Saywer, G. Francis, et al. “D4.2 – Second input from the Services and Applications community to the architecture”, deliverable of the FP7 Fed4FIRE project, November 2013

[4] D. Davies. “D8.4 – Second input to WP2 concerning first level support”, deliverable of the FP7 Fed4FIRE project, September 2013

[5] J. Van Ooteghem, B. Naudts, W. Vandenberghe et al. “D2.3 – First Sustainability Plan”, deliverable of the FP7 Fed4FIRE project, July 2013

[6] “GENI Aggregate Manager API Version 2 details”, http://groups.geni.net/geni/wiki/GAPI_AM_API_V2_DETAILS, last accessed February 3rd 2014

[7] “GENI Aggregate Manager API Version 3”, http://groups.geni.net/geni/wiki/GAPI_AM_API_V3, last accessed February 3rd 2014

[8] “Operations on Individual Slivers”, http://groups.geni.net/geni/wiki/GAPI_AM_API_V3/CommonConcepts#OperationsonIndividualSlivers, last accessed February 3rd 2014

[9] L. Peterson, R. Ricci, A. Falk and J. Chase. “Slice-Based Federation Architecture”, available for download on http://groups.geni.net/geni/attachment/wiki/SliceFedArch/SFA2.0.pdf, last accessed February 3rd 2014

[10] “GENI Design Activities”, http://groups.geni.net/geni/wiki/GeniDesign, last accessed February 3rd 2014

[11] “GENI RSpec version 3”, http://www.geni.net/resources/rspec/3/, last accessed February 3rd 2014

[12] “Clearinghouse”, http://groups.geni.net/geni/wiki/GeniClearinghouse, last accessed February 3rd 2014

[13] E. Kamateri, N. Loutas, D. Zeginis, et al. “Cloud4SOA: A semantic interoperability PaaS solution for multi-Cloud platform management and portability”, Service-Oriented and Cloud Computing Lecture Notes in Computer Science, Volume 8135, pp 64-78, Springer, 2013

[14] “Amazon EC2 Service Level Agreement”, http://aws.amazon.com/s3-sla/ , last accessed February 3rd 2014

[15] B. Vermeulen, et al. “D2.4 – First federation architecture”, deliverable of the FP7 Fed4FIRE project, December 2013

[16] C. Scognamiglio, F. Lobillo, et al. “D3.4 D4.4 – Third input from community to architecture”, deliverable of the FP7 Fed4FIRE project, October 2014

[17] D. Davies. “D8.7 – Third input to WP2 concerning first level support”, deliverable of the FP7 Fed4FIRE project, October 2014

[18] J. Van Ooteghem, et al. “D2.6 – Second Sustainability Plan”, deliverable of the FP7 Fed4FIRE project, November 2014

[19] W. Van de Meerssche, et al., “Federation AM APIs”, https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix A: Requirements from the infrastructures, services and applications communities (D3.4/D4.4)

Experiment Workflow and Lifecycle Management

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
When discovering the different resources that Fed4FIRE can offer me for my experiment, I require …										
'1-3'	ST.1.003	That the above view on node capabilities is the same across the different testbeds of the federation. This means that when describing the characteristics of resources, all testbeds should adopt the same units (e.g. represent RAM always in MB, and not sometimes in MB and sometimes in GB) and use the same parameter names for aspects that mean the same (e.g. always talk about "RAM", and not "RAM" on some testbeds, "working memory" on some others and just "memory" on a third group of testbeds). Resources must be described in a homogeneous manner so that the experimenter can compare the resources in different testbeds, giving the experimenter a view into the internal steps the provisioning of his resources go through.	Fed4FIRE must able to deploy an experiment on multiple testbeds selected by the experimenter.	H	2.86	2.00	2.38	1.50	2.50	**3.00**
'1-1'	ST.1.001	That I can browse some kind of resource catalogue to look for appropriate resources on a high level. Such a catalogue is limited to information such as: testbed X is a testbed for WiFi experiments in an office environment, testbed Y is a testbed for testing cloud applications, etc.	Data sharing, connectivity and experiment management across multiple testbeds - Data management & migration of data	H	3.00	2.55	2.38	1.00	2.50	**2.50**

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'1-2'	ST.1.002	That Fed4FIRE provides a detailed view on what node capabilities are available on every testbed of the federation (e.g. mentioning information for every resource of a testbed regarding CPU speed, RAM, supported 802.11 technology, optical networking interfaces, etc), including if possible detailed information about the full set of configurable QoS parameters and QoS interfaces and information such as a detailed view on each hardware or software component on every available node on every testbed of the federation (e.g. modem radio model, modem radio driver and interoperability with linux kernel modules like pktgen)			3.00	2.27	2.25	1.50	2.00	**2.37**
'1-6'	ST.1.006	That for nodes that have static network connections to other nodes in the same testbed, that it should be possible to identify the corresponding physical topology. In the wired domain this means that you can know how the nodes are connected to each other. For wireless resources this means that you know which resources are in transmission range of each other.			1.71	2.00	1.88	1.50	3.00	**1.93**
'1-5'	ST.1.005	That I know the location of the site where resources are located. Per site, this location information can be exactly the same for all resources.			2.14	1.55	1.50	1.00	1.00	**1.60**
'1-4'	ST.1.004	That next to browsing through information about what is available, that I can actively search for the existence of resources with certain characteristics by defining a specific query (e.g. something that is similar to an SQL query, e.g. select resources from all testbeds where RAM >= 8 GB)			2.29	1.64	1.38	0.00	1.00	**1.57**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'1-9'	ST.1.009	That I can assess which testbeds/resources are more reliable then others (both in terms of provided hardware, software, and wireless interference, possibly based on historical health information about the resources and their environment)			1.71	1.55	1.88	0.00	1.50	**1.57**
'1-8'	ST.1.008	For virtual resources, that I know their physical host and the actual location.			1.86	1.09	1.38	1.50	1.50	**1.40**
'1-7'	ST.1.007	That I have location information about the actual resources that I will use. For example , in wireless nodes the accuracy is particularly important (1 m accuracy)			1.14	1.36	1.00	0.00	1.50	**1.13**
When selecting and reserving resources that I want to include in my Fed4FIRE experiment, I require …										
'2-11'	ST.1.020	That I can easily reserve resources across multiple testbeds using the same common tools. These should also be as user-friendly as possible, abstracting the complexity of the underlying infrastructures for me as much as possible. This way I can focus on the experiment design itself instead of learning how to work with numerous testbed-specific tools.	Scheduling simultaneous access to multiple testbeds. This requirement is also related to resource management on testbeds, SLA management, enforcing limits, making experimenters accountable for the resources they request and use.	H	3.00	2.27	2.38	1.50	0.00	**2.27**
'2-1'	ST.1.010	That when browsing through the resource descriptions, that I can manually select every node that should be added to my experiment. Think of an experience similar to online shopping and putting resources in your shopping cart.			2.57	1.91	2.38	1.00	1.00	**2.07**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'2-13'	ST.1.022	That I can use a single Fed4FIRE account to select and reserve resources at all different testbeds of the federation. So even when using one common tool for reservation at the different testbeds, I don't want to remember a different username/password combination for every testbeds, and I also don't want to register again at every testbed that I want to use. Of course, registering for that one Fed4FIRE account should also be straightforward. Experimenters will use a single set of credentials for all operations on the platform (experimenting, monitoring, etc.) regardless of the tool used (handover). The authentication and authorisation processes to grant access to resources should be as light as possible for the experimenter. Finally, any Identity information must be protected within Fed4FIRE.	It should enable the creation of an account on different testbeds and/or services and link them to a unique Fed4FIRE user. ID management and mapping. Privacy and data protection are also important. For example, passwords must never be sent by email.	H	2.57	1.55	2.63	1.50	0.50	**2.00**
'2-5'	ST.1.014	That I can reserve resources. It is OK for me that they are shared with others (soft reservation, e.g. requesting a virtual machine that will be deployed on a physical server that is used by other experiments also), as long as I know that I will also have guaranteed access to them.			2.29	1.91	2.25	0.50	0.00	**1.87**
'2-8'	ST.1.017	That situations are avoided where a have to wait days or weeks before being able to use the testbed because of long reservations of others.			3.00	1.36	1.75	1.50	1.50	**1.87**
'2-12'	ST.1.021	That when reserving resources across multiple testbeds, that there is guidance in finding the first appropriate time when all the resources that I want across the testbeds would all be available.			2.86	1.09	2.38	1.50	0.50	**1.83**
'2-10'	ST.1.019	That a reservation is approved or rejected quickly (within a few minutes).			3.00	1.18	1.88	1.50	1.00	**1.80**
'2-15'	ST.1.024	That the testbeds and/or the federation guarantee a certain Service Level to me regarding the execution of my experiment (availability of resources, reliability of resources			2.71	1.18	1.75	1.50	1.50	**1.73**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
		(uptime/downtime), responsiveness of support services, privacy guarantees, etc).								
'2-16'	ST.1.025	That I can dynamically scale my resources up and down according that what my experiment needs during its execution. For instance if a server deployed on a VM gets overloaded, I should be able to assign more resource (RAM, CPU cores, etc.) to that running VM, and/or should be able to add a second VM to my running experiment on which I deploy a second instance of that server.	This could imply scaling resources - adding and removing them from testbeds or adding / removing a whole testbed	H	2.29	1.73	1.88	0.50	0.00	**1.70**
'2-2'	ST.1.011	That I can select suitable resources for inclusion in my experiment by defining a specific query (e.g. something that is similar to an SQL query, e.g. select all resources from Virtual Wall where nr_ethernet_cards >= 6)			2.43	1.73	1.63	0.00	0.50	**1.67**
'2-7'	ST.1.016	That next to adding resources to my experiment right now (instant reservation), that I can also define a reservation for any moment in the future (future reservation, e.g. tomorrow from 9AM-5PM).			2.00	2.09	1.00	0.50	1.50	**1.63**
'2-6'	ST.1.015	That I can reserve resources. They have to be exclusively assigned to me (hard reservation, e.g. reserving a virtual machine that will be deployed on a physical machine that is dedicated to your experiment only)			2.71	1.27	1.00	1.50	1.50	**1.57**
'2-14'	ST.1.023	That if testbeds decide to assign me a certain reservation quota (e.g. based on my profile such as student, post-doc, professor, paying customer, etc), that I can request a temporary increase of my quota if really need it (e.g. before a paper deadline)			2.29	1.27	1.13	0.50	1.50	**1.43**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'2-3'	ST.1.012	That I can temporarily install my own equipment at a Fed4FIRE testbed for testing, and select it to be included in my experiment. My own equipment might be for example mobile devices in a wireless experiment. The fact that this kind of resource appears and disappears might not alter the course of the experimentation. When these devices appear, they might become part of the infrastructure (e.g. as sensing nodes and thus producing information). This also applies for VMs, for example, since experimenters should be able to dynamically create VM (resources), name them and use that name to interact with them.	It relates to registration of resources in testbeds, specification of resources, advertising	H	2.00	1.55	1.25	0.50	0.00	**1.40**
'2-4'	ST.1.013	That the mechanism for registering my own equipment at a testbed is standardized, allowing me to register that equipment at different testbeds in exactly the same manner.			2.29	1.36	1.00	1.00	0.00	**1.37**
'2-17'	ST.1.026	That if I reserved a number of resources at a testbed, that I can divide them over different independent experiments that I am doing at the same time. Experimenters can have more than one experiment running on a given testbed at the same time. He should be able to easily address/group the resources from one experiment. This means that experimenters should be able to use reserved resources for more than one experiment	Could be solved if experimenters are allowed to create slices. Provisioning must be done more than one in this case, so that each experiment starts from a clean state	M	1.86	0.82	1.50	0.50	1.00	**1.23**
'2-9'	ST.1.018	That I can reserve nodes exclusively for myself for a longer period (days or weeks)			2.57	0.64	0.38	0.00	0.00	**0.93**
When using the resources that I included in my Fed4FIRE experiment, I require …										
'3-1'	ST.1.027	That I can SSH to my nodes. Experimenters should be able to use different tools to prepare and run their experiment. For example, have access to CLI tools to perform standard actions	A portal and an API is not enough for experimenters	M	3.00	2.55	2.63	1.50	1.50	**2.53**
'3-2'	ST.1.028	That I have root access to my nodes. This allows me to perform any action on the nodes that I			3.00	2.55	2.50	1.50	1.50	**2.50**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
		want (install new applications, device drivers, load additional kernel modules, etc).								
'3-3'	ST.1.029	That I can use a single public/private SSH key pair to access my resources on all the different testbeds			3.00	2.18	2.25	1.00	0.00	**2.17**
'3-7'	ST.1.033	Fed4FIRE must provide the means for experimenters and third parties to develop and/or deploy applications on top of Fed4FIRE infrastructure. This should be done in such a way that these external providers find it attractive to eventually join the federation. An experimenter might be able to install own pieces of software (or software downloaded from the internet) on resources of the facilities involved in his experiment. The experimenter will also have the possibility to manually complete the experiment data and to setup initial data-sets over different facilities. An interface through which the experimenter can enter these data must be provided for this and the data entered must be stored in the selected facility to be used at runtime.	This involves suppliers which might become part of the core federation players	H	2.71	2.36	1.75	1.50	1.50	**2.17**
'3-12'	ST.1.038	That I can easily use my resources across multiple testbeds using the same common tools. These should be as user-friendly as possible, abstracting the complexity of the underlying infrastructures for me as much as possible. This way I can focus on the experiment itself instead of learning how to work with numerous testbed-specific tools. In general, Fed4FIRE tools should be user friendly to the experimenter		H	2.43	1.82	2.38	1.00	2.00	**2.07**
'3-5'	ST.1.031	That I can choose to have a specific Linux distribution on my nodes (e.g. latest Ubuntu LTS release)			2.43	2.00	1.88	1.00	0.50	**1.90**
'3-8'	ST.1.034	That I can take a binary image of the hard drive of my nodes, and that I can store these for later re-use (so flashing the image back later on)			2.57	1.64	1.63	0.50	1.50	**1.77**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'3-11'	ST.1.037	That I can allow other people of my work team that are involved in the experiment to use the resources that I have reserved and deployed. I should be able to specify which resources should be shared, and which not. Resource groups should be shared by default with a group of people the experimenter belongs to. Resources might be shared among several experimenters or used exclusively, depending on the testbed offer and on the experimenters' needs.	Could be solved if users from the same sub-authority can get slice credentials for slices created by others	M	2.00	1.73	1.63	0.00	2.00	**1.67**
'3-10'	ST.1.036	That during the deployment of my resources over different facilities, that my initial data sets can be automatically loaded to all these resources.			2.71	1.18	1.63	0.00	1.50	**1.60**
'3-9'	ST.1.035	That I can define what a node should automatically do at start-up (bootstrap scripts).		H	3.00	1.27	1.38	0.00	0.00	**1.53**
'3-6'	ST.1.032	That I can choose to use a custom Linux kernel on my nodes (e.g. with my own performance upgrade patches to the kernel)			2.29	1.64	1.00	0.50	0.00	**1.43**
'3-4'	ST.1.030	That I can choose to have Windows installed on my nodes			2.14	1.45	0.88	0.00	1.50	**1.37**
When controlling the execution of my experiment in an orchestrated manner, I require …										
'4-3'	ST.1.041	That the description of the above orchestration is described in a human-readable way. This description should also be uniform across the different testbeds.			2.57	1.91	1.75	0.50	2.00	**1.93**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'4-1'	ST.1.039	That I can define the behaviour over time of a distributed experiment in a single script, which can be started automatically at any desired moment, and will be automatically translated to the corresponding triggers at the nodes at the appropriate time. So e.g. describing in a single script that the 5 client nodes in an experiment should gradually increase their load on the server that they are testing in the experiment. This will be done automatically, without the experimenter login in to these 5 nodes and gradually increasing this load manually. It should be possible to write a workflow or other description of an experiment which allows it to be run from start to finish including resource provisioning, resource control, service interaction, monitoring and data collection.	Proposed workflow system could offer this if well integrated with existing / lower level components	L	2.43	2.00	1.63	0.50	2.00	**1.90**
'4-2'	ST.1.040	That I can define the behaviour of a distributed experiment in a single script, based on events (e.g. value above threshold). This can be started automatically at any desired moment, and will be automatically translated to the corresponding triggers at the nodes at the appropriate moment. So e.g. describing in a single script that a server should scale up to a VM with more CPU power and RAM when the load of the clients on the server becomes higher than a certain threshold. Orchestration should involve different testbeds, for example using the results from one testbed as inputs or triggers for the other			2.14	1.82	1.63	0.50	2.00	**1.77**
'4-4'	ST.1.042	That the above description of the orchestrated control of the experiment can also include other aspects that will be performed automatically. This includes selection, reservation and deployment of resources; monitoring of the resources and collection of measurement data during the experiment.		L	2.43	1.27	1.38	0.50	2.50	**1.60**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Measurement & Monitoring

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
When capturing the results of my experiment (monitoring and measuring data), I require …										
'5-11'	ST.2.011	That the overhead of any monitoring and measurement tool is minimal. These tools should have a negligible impact on the results of my experiment.			2.86	2.36	2.25	1.50	2.50	**2.40**
'5-9'	ST.2.009	That other aspects related to the successful execution of my experiment are continuously monitored, and that I am automatically informed in case of any errors. Examples are: when a selected resource could not be instantiated, when there is a problem with the interconnectivity between the used testbeds, when a used testbed goes down during the experiment, when there is a sudden peak of wireless interference, etc. This might be important when analysing anomalies in the experiment results.			3.00	2.27	1.88	1.50	1.00	**2.20**
'5-15'	ST.2.15	That I can store experiment configurations in order to repeat experiments and compare results of different runs. The experiment setup should be made easy in order not to repeat the same procedure every time (same tools, same user, etc.)		M	2.86	1.73	2.38	0.50	2.50	**2.13**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'5-7'	ST.2.007	That the overall health status of the different testbeds (testbed up or down, has free resources left, etc.) is continuously monitored by the federation, and that in case of issues I am informed of this. Fed4FIRE will deal with errors and exceptions occurred during an experiment and raise alerts providing comprehensive information back to the experimenter, including information from several facilities. Capability to collect information from different testbeds in order to furnish an open platform with all the gathered product data.		M	2.14	1.91	2.25	1.50	2.00	**2.03**
'5-3'	ST.2.003	That by default some common characteristics of my resources are stored automatically for later analyses during experiment runtime (CPU load, free RAM, Tx errors, etc).			2.43	1.45	2.50	1.50	1.00	**1.93**
'5-2'	ST.2.002	That Fed4FIRE makes it easy for me to retrieve and store data that I measured during the runtime of the experiment. This means that it should be easy to store my measurement somewhere in a way that the data is clearly related to the experiment ID, but without needing to establish connections to certain databases manually from within my code, and without needing to know the specific experiment ID that belongs to my current experiment.			2.57	1.73	2.13	0.50	1.00	**1.90**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'5-8'	ST.2.008	That the overall health status of the different testbeds (testbed up or down, has free resources left, etc.) is continuously monitored by the federation, and that in case of issues the corresponding testbeds try to solve them asap.			2.00	1.73	2.38	0.50	1.50	**1.87**
'5-12'	ST.2.012	That I can store and access my experiment monitoring data and other measurements on a data service on the federation, which is accessible during the experiment (temporarily data storage by the federation). Fed4FIRE must provide storage facilities for the experimenter to retrieve historical data concerning an experiment data-set and results. Data centres have to keep log of data stored and access to external resources. It is important for audits and historical data tracking	This requirement involves data management and archiving. Questions arise about the long-term storage of experiment data - access control, management, curation, who pays for long term storage.	M	2.29	1.64	2.13	0.50	2.00	**1.87**
'5-10'	ST.2.010	That when an error requiring manual intervention is reported to me as part of the previous step, that I am guided through the process for recovery.	Operational errors, such as failure of he infrastructure, impossibility to store or retrieve data are some examples of these errors	L	2.29	1.91	1.75	0.50	1.50	**1.83**
'5-14'	ST.2.014	That access to my stored data is properly secured. Experiments must be kept confidential if required, the privacy of experiments, data sets and results should be guaranteed.		L	2.71	1.09	1.88	1.50	2.50	**1.80**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'5-17'	ST.2.017	That I am made aware if my storage capacity is running out.			2.86	1.18	2.00	0.50	2.00	**1.80**
'5-4'	ST.2.004	That for the above monitoring, that I can select and configure how this data should be collected (always at a specified interval, only after a certain event or alarm, define some specific filters, etc). Fed4FIRE must provide tools to create, view, update, and terminate monitoring configurations related to shared resource types or experiments in real time. Monitoring data should be reportable for visualisation and analysis purposes with several reporting strategies (only alarms, all data, filters, etc.) in real time in order to provide accurate information and ease the analysis process. The experimenter might create own aggregated/composite monitored elements out of the available ones when designing the experiment, deciding what to monitor, defining some filtering possibilities as well, and providing the destination endpoint to send the information to. Monitoring information must cover information from different facilities and services. Monitoring metrics should be compatible across different facilities. For example, Monitoring computing & network resources' capacity. The monitored data during an experiment runtime will be available to the experimenter for all components involved in the experiment. If not aggregated, at least monitoring information from all involved testbeds must be provided. Testbeds must be able to publish infrastructure status through an API	As most of the monitoring in cycle 1 is done by using OML streams, experimenters should be able to create and/or configure new/existing OML Measurement Points in real time (maybe via FRCP)		2.43	1.36	1.88	0.50	1.50	**1.70**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'5-13'	ST.2.013	That Fed4FIRE makes it easy for me to retrieve and store data that I measured during the runtime of the experiment. This means that it should be easy to store my measurement somewhere in a way that the data is clearly related to the experiment ID, but without needing to establish connections to certain databases manually from within my code, and without needing to know the specific experiment ID that belongs to my current experiment.		M	2.14	1.45	2.13	0.50	1.00	**1.70**
'5-1'	ST.2.001	That the internal clocks of resources across multiple testbeds are synchronized very accurately; Comment from MobileTrain: "Different experiments need different time accuracy. Sometimes NTP is enough, whereas other times it is necessary a GPS module on each node."			1.71	1.45	1.13	1.50	3.00	**1.53**
'5-5'	ST.2.005	That I might create own aggregated/composite monitored elements out of the available ones when designing the experiment, deciding what to monitor, defining some filtering possibilities as well		H	1.86	1.45	1.38	0.00	1.00	**1.40**
'5-16'	ST.2.016	That I can share my stored data with specific others (individuals and/or groups), or even make them publically available			1.43	1.64	1.38	0.00	1.00	**1.37**
'5-6'	ST.2.006	That information about external wireless interference during the execution of my experiment is automatically provided for me.			0.57	0.82	1.13	1.50	2.50	**1.00**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Interconnection

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
When focusing on the connectivity of the resources that will be included in my Fed4FIRE experiment, I require …										
'6-1'	ST.4.001	That resources at different testbeds are interconnected on layer 3 (IP)			2.86	1.64	1.38	1.50	1.50	**1.83**
'6-3'	ST.4.003	That I can know the type of interconnections that are available between the testbeds (layer 2 and/or layer 3, NAT or VPN included, dedicated direct link, connected through Géant with or without bandwidth reservation, connected over the public Internet, …)			2.57	1.36	2.13	0.50	1.50	**1.80**
'6-5'	ST.4.005	That my resources are directly reachable, without any network address translation (NAT) or virtual private network (VPN) in between. So actually I require that all resources have a public IPv4 or IPv6 address.			2.14	2.09	1.38	0.50	1.00	**1.73**
'6-6'	ST.4.006	That if an issue arises with the interconnection between my used testbeds, that I am automatically informed about this.			2.29	1.55	1.88	0.50	1.00	**1.70**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

SURVEY REQ ID	REQ ID	DESCRIPTION	COMMENTS	CYCLE 2 PRIORITY	Average 1st open Call survey (a)	Average 2nd Open Call survey results (Industry) - b	Average 2nd Open Call survey results (Academic) -b	Average 1st SME Open Call winners (c)	Average 1st SME Open Call survey (d)	GLOBAL AVERAGE
'6-4'	ST.4.004	That I can configure a specific bandwidth on the interconnections between the different testbeds used in my experiment. As long as the links behave as configured, I don't really care what the testbed has to do behind the curtains to implement this (reserve guaranteed bandwidth in case of limited capacity on the interconnecting link, or limit the bandwidth in case of a high capacity on that same link).			1.71	1.73	1.50	0.50	1.00	**1.53**
'6-2'	ST.4.002	That resources at different testbeds are interconnected on layer 2, or that such a layer 2 connection can be automatically created for me (in a way that all the underlying technical details are abstracted for me)			1.57	0.82	1.38	0.00	0.00	**1.03**

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix B Requirements from SLA management

Methodology for requirements

The requirements regarding SLA for Cycle 3 have been obtained from different sources:

- **Cycle 2 requirements** have been reviewed and updated according to the trajectory and philosophy followed by Fed4FIRE (e.g. aggregation of SLAs is not required, fees are not supported in the federation)
- The second **review results** in which recommendations about the openness of the federation, agility for testbed federation process and the added value for experimenters it provides were made.
- **Experience with Fed4FIRE portal interaction** through the developed SLA plugin in the SLA lifecycle.
- **Survey** sent to federation testbeds to know their plans regarding SLA adoption (for those which do not have SLAs) and extension (for those which already offer SLA) for Cycle 3. Further analysis of the SLA survey is described below.

Survey review

In order to know the plans regarding SLAs for each testbed within Fed4FIRE, a survey was sent including questions such as the intention to adopt SLA and what type of SLAs (e.g. new metrics beyond availability) would be offered, the possible penalties or rewards to be applied on the SLA result and the monitoring capabilities needed for SLA evaluation.

The answers provided by the testbeds have been gathered and analyzed, leading to two conclusions.

- **Testbeds without SLA**: Those testbeds that do not offer SLAs in cycle 2 do not have plans to adopt them in cycle 3. Resource reservation is provided with best effort with no guarantees and SLAs are not considered as a priority for the next cycle. The exception to this group is the Ofelia testbed which answered that the adoption of SLAs in cycle 3 was under study.

- **Testbeds with SLA**: Testbeds that already offer SLAs will improve the resource monitoring (that can lead to the offering of more complex SLAs) and plan to give SLA results an impact either on the testbed (i.e. by means of reputation) or on the experimenter (i.e. by providing compensation quota). However, resource reservation will be offered in a best effort manner, SLAs guaranteeing their availability during the experiment.

Area / Functional Domain	Field	Requirement id	Requirement statement	Requirement description	It affects	Fulfilment cycle 2	Coverage at the end of Cycle 3	Comments on coverage

Trustworthiness	SLA Management	ST.3.001	Profile-based offering	Different SLAs will be offered depending on the experimenter's profile: e.g. academia will be offered Best Effort and industry will have Premium services offered. More or less resources are provisioned according to the SLA.	Authentication and profiles belong to the security domain. The user profile is present in this process so that the testbed can accept or reject the request	N	N	The same SLA will be shown to all experimenters regardless of their profile. Testbeds do not provide different types of offers.
Trustworthiness	SLA Management	ST.3.002	Prioritization of requests	There will be a prioritization of some users' requests over others (premium over normal). Moreover, resources will be assigned or reserved for a user until the experiment finishes, unless he makes a use against the testbed policies.	Authentication and profiles belong to the security domain. The user profile is present in this process so that the testbed can accept or reject the request	N	N/A	This requirement concerns capacity management at the testbed and is beyond SLA management scope
Trustworthiness	SLA Management	ST.3.003	Deeper insight of facility usage	The SLA management system should help facilities retrieve intelligent information concerning the testbed usage (e.g. track historical SLA info per experimenter)	Monitoring. User information should be provided to the SLA management	N	Y	The SLA dashboard will allow the visualization of SLA information available at each testbed
Trustworthiness	SLA Management	ST.3.004	SLA template specification	Fed4FIRE must provide a clear step-by-step procedure describing how to write and SLA template for infrastructure providers		N	Y	The SLA dashboard will allow the creation of SLA templates for infrastructure providers
Trustworthiness	SLA Management	ST.3.005	SLA establishment	The SLA system should not be bureaucratic, it should not add additional administrative burden for experimenters. Simple, costless and short SLA establishment.		Y	Y	The SLA tools allow an easy SLA creation and visualization for the experimenter

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Trustworthiness	SLA Management	ST.3.006	SLA feedback for reservation	For reservation-based testbeds, a reservation is a time slots on resources. The SLA management should help keep track of the booking diary, by providing the information of whether reserved resources have been actually used or not and raise alarms in case of conflict. Tentative reservations could be offered at a lower price (price in terms of quota). Besides, this should work even if the reservation system is vague "2 weeks sometime in the next month".	This requirement implies that the SLA management will have to indicate to BonFIRE that there is an adjustment to make on the quota. Moreover, additional monitoring in Fed4FIRE needs to be done: the info concerning the reservations, the cancellations, how much time in advance reservations are made, etc. is in the dB but there is no current process to interpret it.	N	N	This requirement concerns resource management at the testbed. The SLA will only be informed of reservations for the calculation of the SLA but no feedback to the reservation system will be implemented
Trustworthiness	SLA Management	ST.3.007	SLA for reservation and Best Effort for availability	The SLA management should be aware of reservations for those testbeds who operate on this basis. This means that there will be a previous phase before the experiment in which the reservation will take place and an according SLA has to be setup.	The SLA management must be integrated with reservation mechanisms.	N	Y	SLAs support reservation of resources and will be stablished at the reservation moment. SLAs cannot be modified in case there are changes in the testbed capacity once the SLA is accepted. Reservation system not fully operational until cycle 3
Trustworthiness	SLA Management	ST.3.008	SLA offerings and pricing	Possibility of offering Best Effort and other Premium alternatives in exchange of a fee	Extended monitoring is required for this. Conciliation is beyond the scope of SLA management.	N	N	Only a general unique SLA will be offered by facilities adopting SLAs. No fees involved in Fed4FIRE

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Trustworthiness	SLA Management	ST.3.009	Accountability	When an experiment goes wrong (time out, etc.), it is essential to distinguish whether the problem is originated in the infrastructure (SLA not met) or if this is caused by the experimenter's own software or data-set used during the experiment.		N	N	Not achievable, testbeds cannot monitor this kind of issues
Trustworthiness	SLA Management	ST.3.010	SLA publication and discovery	Fed4FIRE must provide the means for infrastructure providers to publish offerings, experimenter requirements in terms of QoS and browse/compare offerings. User must be aware of available offerings (including costs associated with resources provided) using specific criteria (e.g. privacy and cost).		N	Y	Possible using SLA dashboard tool. However no testbeds are offering more than one SLA.
Trustworthiness	SLA Management	ST.3.011	SLA monitoring	Once the user begins the experiment, the SLA management will be able to compare each individual SLA with the monitoring of every testbed involved. In case an SLA is not met, then the actions defined in the SLA for the testbed that did not meet the SLA must be invoked. For this, the SLA management will require to be interfaced with external systems that will actually carry out these actions.	The SLA evaluation depends on the information provided by the monitoring system for every facility. The SLA tool needs to be aware of the start and end times of the experiment	Y	Y	This is the basic mechanism of the SLA tool

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Trustworthiness	SLA Management	ST.3.012	Unmet SLA by providers	In case the SLA is not met, the SLA management system must be able to clearly identify and make visible which facility/ies did and did not commit. In case an SLA is not met, then the actions defined in the SLA for the testbed that did not meet the SLA must be invoked.	For this, the SLA management will require to be interfaced with external systems that will actually carry out these actions. This applies also for reservation.	Y	Y	Individual information for each facility is provided. Connection with Reputation Engine and quota for testbeds that support it.
Trustworthiness	SLA Management	ST.3.013	Unmet SLA by experimenters	In case experimenters do not meet their obligations, the SLA management must facilitate the claim for compensation by all affected testbeds who require so.		N	N/A	No testbed provides mechanisms to monitor experimenters.
Trustworthiness	SLA Management	ST.3.014	SLA evaluation for individual testbeds. SLA management, monitoring and capacity planning	The SLA management will allow knowing how much capacity is required for the SLAs offered and what SLAs a testbed can offer with the current available capacity.		N	N/A	This requirement concerns capacity management at the testbed and is beyond SLA management scope
Trustworthiness	SLA Management	ST.3.015	SLA negotiation based on offerings	There will be a mechanism for a negotiation to agree SLA conditions between the experimenter and each provider.		Y	Y	Experimenters have to agree SLA offered by the testbed before being able to reserve resources.
Trustworthiness	SLA Management	ST.3.016	Experimenter's obligations	The SLAs must gather, for each facility, the obligations of the experimenter. For example, the user must be associated to a site (e.g. academic institution) in the particular case of PlanetLab.		N	Y	Currently no testbeds are imposing obligations to experimenters but this is covered
Trustworthiness	SLA Management	ST.3.017	SLA violation notification	Users will be informed of SLA violations		Y	Y	This will be done individually for each testbed using any client tool in cycle 3

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Trustworthiness	SLA Management	ST.3.018	Aggregate SLAs and monitoring	Fed4FIRE must provide the means to manage and monitor an aggregated SLA lifecycle by joining partial SLAs provided by the underlying infrastructures and services. Monitoring system values related to the SLAs is required.		N	N/A	Not required in Fed4FIRE, SLAs are always individual for each facility and experiment
Trustworthiness	SLA Management	ST.3.019	SLA establishment	Fed4FIRE will allow the establishment of an SLA between the experimenter and facility providers. Before the experiment begins, all related SLAs must have been signed (including reservation SLAs).		Y	Y	Fed4FIRE portal already implements this requirement. Support from any client tool will be available for cycle 3
Trustworthiness	SLA Management	ST.3.020	Operational planning	In case an experiment involves several facilities and at least one of them is reservation-based, Fed4FIRE should ensure that none of the involved testbeds plans maintenance operations that could affect the experiment during reservation time		N	N	This should be notified by the testbed in the reservation process and is beyond the scope of SLA management
Trustworthiness	SLA Management	ST.3.021	Offering based on capacity	The SLA mechanism should provide the means for the testbed to expose a certain SLA according to the available capacity at a given moment.	Monitoring. This information should be provided to the SLA management	N	N	Testbeds only support one type of SLA, regardless of available capacity
Trustworthiness	SLA Management	ST.3.022	Integration with existing tools	The SLA mechanism must adapt to the different policy engines (e.g. quota system) existing in different facilities, if any. If a central policy component is included in Fed4FIRE, the SLA will interact with it.		N	Y	A general purpose API is available to be used by any federation engine

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Trustworthiness	SLA Management	ST.3.023	SLA signing entity (on behalf of the experimenter)	The SLA mechanism in Fed4FIRE must contemplate the fact that experimenters can follow hierarchies as far as registered users are concerned (e.g. institutions, projects, research fellows) and that, in some cases, the signature of a high-level user is valid for all underlying users. This policy might be different for every testbed.	This is related to authentication mechanisms. There should probably be an SLA responsible in every user group, understandable by Fed4FIRE	N	Y	This has to be contemplated in client tools. SLAs already support entity signing
Trustworthiness	SLA Management	ST.3.024	SLA Management	SLA management should provide machine readable information in order for the monitoring to know what parameters to measure in order to fulfil the SLA and anticipate upcoming failures. This information must be automatically provided before provision is achieved.		N	Y	Monitoring metrics are automatically retrieved from either a central or testbed local database
Trustworthiness	SLA Management	ST.3.025	SLAs for tools provided by software suppliers	Fed4FIRE will provide the means to setup SLAs also for applications developed by third party suppliers provided that they become part of the federation	The central architecture of Fed4FIRE, which must enable this new kind of players. It also affects the Fed4FIRE sustainability definition as these stakeholders will become core players once they federate.	N	Y	A SLA API is available to be used for any federated client software used by the experimenter
Trustworthiness	SLA Management	ST.3.026	Reputation	The SLA management will provide information to the central reputation component in Fed4FIRE	The central reputation component	N	Y	Integration with Reputation Engine will be done in cycle 3

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Trustworthiness	SLA Management	ST.3.027	Different kinds of services	In Fed4FIRE, there are low level resource services (e.g. testbeds offer raw resources) but also high-level application services (that run over the resources). Fed4FIRE must be able to deal with SLAs for both kinds of services.	The central architecture will enable both types of services	Y	Y	SLA tools support services but no SLA will be offered for application services in cycle 3.
Trustworthiness	SLA Management	ST.3.028	Openness and Reusability	The SLA management in Fed4FIRE should be open and based on market standards	The testbeds and the central architecture	Y	Y	The SLA management tool is based on REST technology and WSAG standard
Trustworthiness	SLA Management	ST.3.029	Security	Only federated experimenters and client tools should be able to access the SLA API therefore proper security mechanisms must be available for authentication (i.e. Fed4FIRE certificates)		N	Y	Authentication mechanisms being developed in cycle 3

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

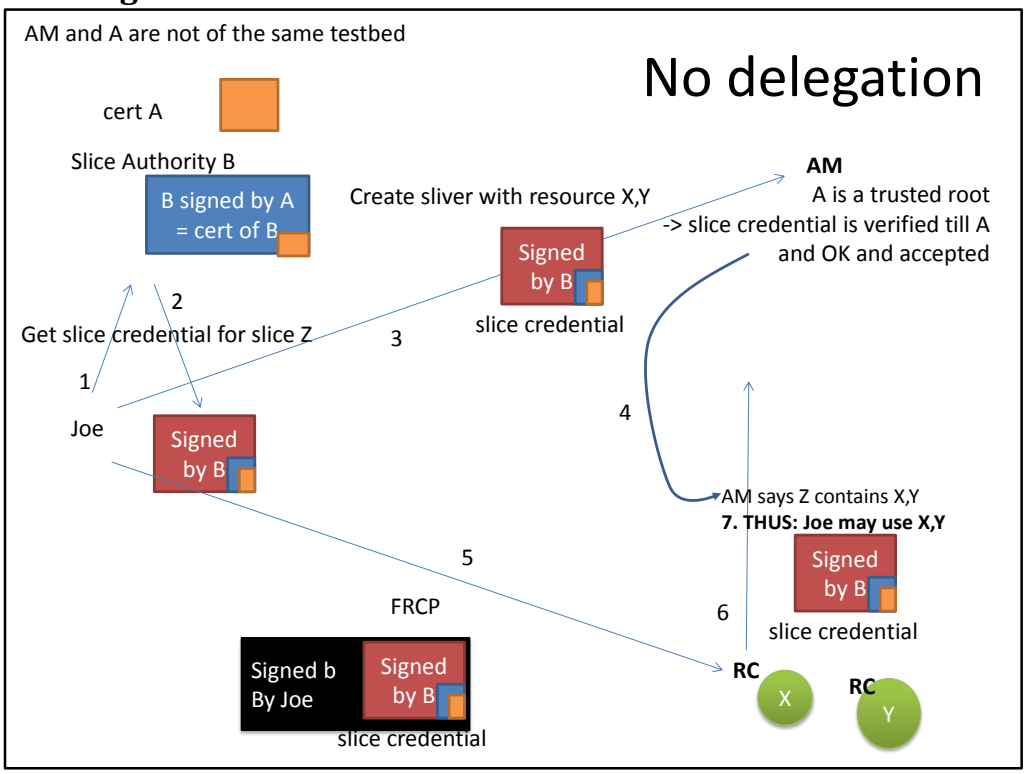
Appendix C: Interaction between Policy Decision Point and Aggregate Manager

Simply put, a slice is a container owned by a slice owner and issued by a Slice Authority - think of it like a shopping bag that can be filled with resources from multiple testbeds. The Slice Owner can go to a testbed's AM and request resources in that slice. If the AM agrees, it will allocate resources for the slice owner and bind them internally somehow to the slice. After this, the slice owner can go to another testbed and request more resources in the same slice. The slice then contains two sets of resources at two testbeds, and the associated slice credential is a way of proving the ownership or rights on the slice, and therefore to the resources at both testbeds allocated to the slice.

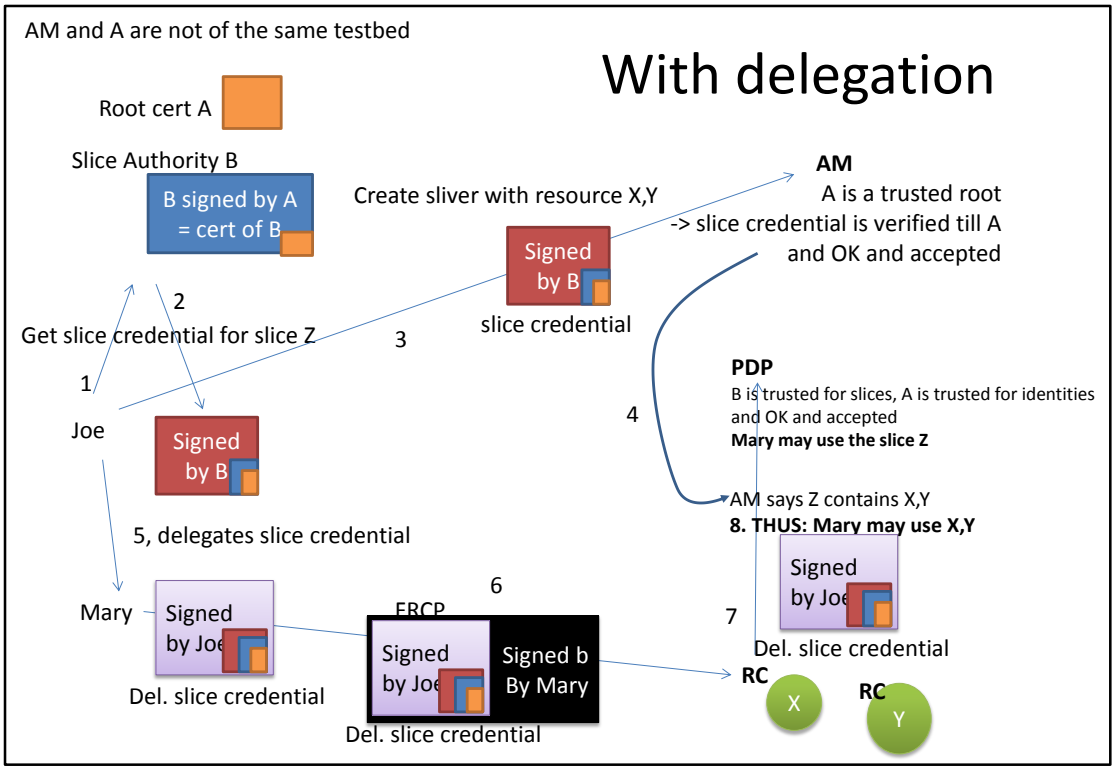
The PDP uses the SFA slice credential as a common access token in multiple testbeds. The PDP installed at all testbeds who want it, and it protects an OMF RC at the testbed - this RC is the point of entry for users of resources. This approach (and the PDP) is not connected with SSH keys at all - the PDP is another way of accessing resources reserved in SFA and supports the access via FRCP and OMF. Testbeds can support SSH access, FRCP access, or both - it is up to them which ones they support, but SSH and FRCP access to resources are completely separate. The PDP is intended to bridge resource request and allocation using SFA, and resource usage in FRCP. The PDP provides a way for an AM to declare the binding of resources at a testbed to a slice, so when the slice owner presents a slice credential, the PDP will be able to decide whether they can access the resources.

There is no need for a common account in this pattern, because the slice is the common mechanism by which resources at multiple testbeds can be requested and used, and the access credential for this is the slice credential. However, when the owner of a slice wants to provide access ot its resources to other experimenters also, some additional measures need to be taken: As depicted below, these can be based on different approaches: without delegation of the slice credential, or with delegation. The corresponding steps are depicted in the figures. More details will be given in the context of WP7.

No delegation



With delegation of slice credential



© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix D: What information does a user credential and slice credential contain

Slice credential

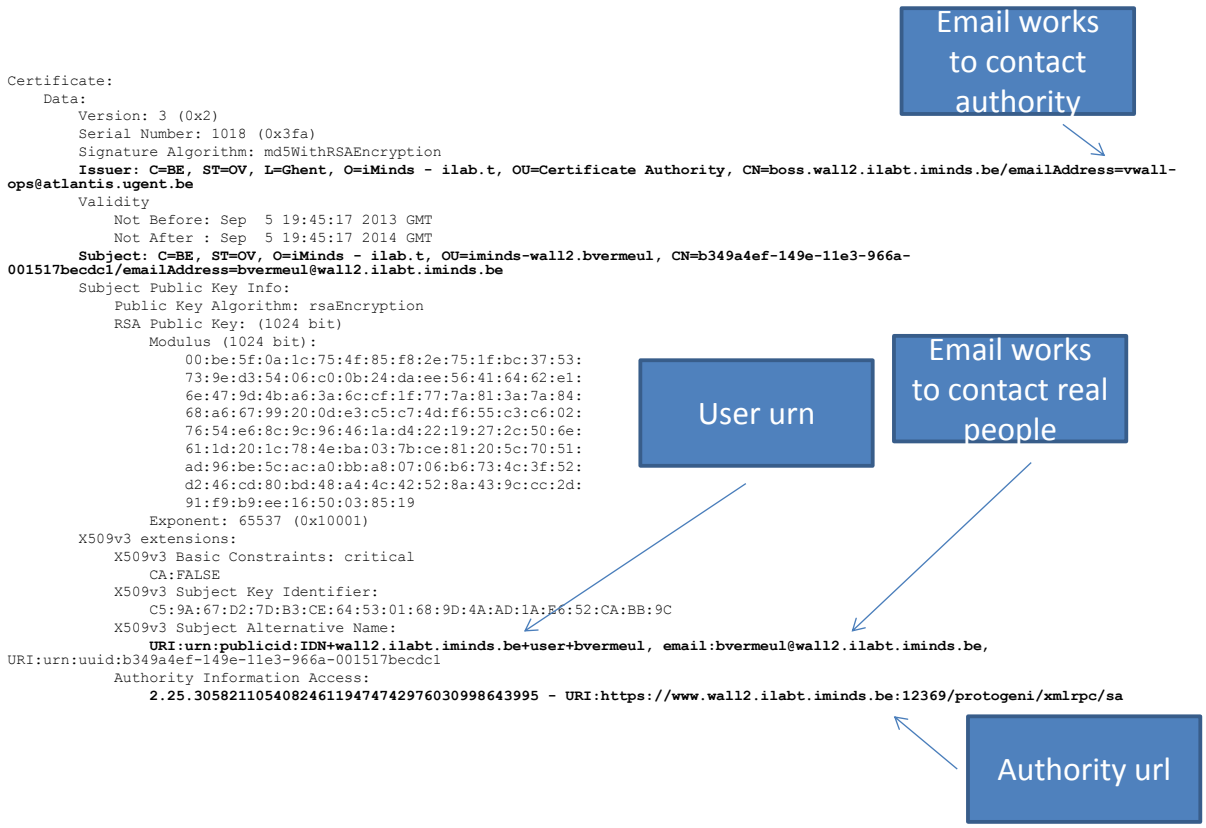
A slice credential is an X.509 certificate, and is therefore intended to contain information about the slice in a way that any recipient of the credential can authenticate it. In other words, it is a document that allows the transfer of information about a slice in a trusted manner. As depicted below, the slice credential contains information about the (sub)authority that issued the slide, the validity period of the slice, the slice name and the slice owner. For further details we refer to WP7.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3621 (0xe25)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=BE, ST=OV, L=Ghent, O=iMinds - ilab.t, OU=Certificate Authority, CN=boss.wall2.ilabt.iminds.be/emailAddress=vwall-
ops@atlantis.ugent.be
        Validity
            Not Before: Nov 11 12:24:44 2013 GMT
            Not After : May  4 13:24:44 2019 GMT
        Subject: C=BE, ST=OV, O=iMinds - ilab.t, OU=iminds-wall2.pdptest2, CN=a12e2ec1-4ad4-11e3-883a-
001517becdc1/emailAddress=brecht.vermeulen@iminds.be
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:dd:28:80:bf:b3:27:6a:59:f9:ca:e9:89:d5:12:
                    51:aa:db:84:15:fa:33:f7:ba:21:68:71:98:71:b5:
                    e5:ad:73:41:f6:f2:99:fa:7a:6a:0b:f2:db:d7:4f:
                    db:83:b8:8e:57:7b:d3:50:ea:d5:18:df:29:5e:f8:
                    8b:46:37:c0:a3:e4:4d:2e:4e:67:6e:fa:6f:57:23:
                    a7:ee:85:08:7e:9b:3a:25:8b:c8:ea:a0:96:4a:68:
                    48:cf:73:c5:a0:ab:f6:b1:fa:38:79:65:5f:ab:1d:
                    f1:70:bb:6b:e4:3f:df:27:dc:9e:45:5f:1d:5e:47:
                    db:20:22:de:68:61:4a:f3:2f
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                42:36:EA:B6:6B:D5:C0:44:88:33:D1:2D:2E:98:47:32:F3:1F:BF:76
            X509v3 Subject Alternative Name:
                URI:urn:publicid:IDN+wall2.ilabt.iminds.be+slice+pdptest2, URI:urn:uuid:a12e2ec1-4ad4-11e3-883a-001517becdc1
            X509v3 Basic Constraints: critical
                CA:FALSE
```

User credential

A user credential is an X.509 certificate, and is therefore intended to contain information about the user in a way that any recipient of the credential can authenticate it. In other words, it is a document that allows the transfer of information about a user in a trusted manner. Below more details are given about the information that is contained in the user credential that is returned by the member authority of the Virtual Wall. As depicted below, that user credential contains information about the (sub)authority that issued the user credential, the validity period of the credential, the user's contact information and the user's public key. Note that this presence of valid contact information is critical when accountability is to be supported. For further details we refer to WP7.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015




```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1018 (0x3fa)
        Signature Algorithm: md5WithRSAEncryption
        Issuer: C=BE, ST=OV, L=Ghent, O=iMinds - ilab.t, OU=Certificate Authority, CN=boss.wall2.ilabt.iminds.be/emailAddress=vwall-
ops@atlantis.ugent.be
        Validity
            Not Before: Sep  5 19:45:17 2013 GMT
            Not After : Sep  5 19:45:17 2014 GMT
        Subject: C=BE, ST=OV, O=iMinds - ilab.t, OU=iminds-wall2.bvermeul, CN=b349a4ef-149e-11e3-966a-
001517becdc1/emailAddress=bvermeul@wall2.ilabt.iminds.be
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (1024 bit)
                Modulus (1024 bit):
                    00:be:5f:0a:1c:75:4f:85:f8:2e:75:1f:bc:37:53:
                    73:9e:d3:54:06:c0:0b:24:da:ee:56:41:64:62:e1:
                    6e:47:9d:4b:a6:3a:6c:cf:1f:77:7a:81:3a:7a:84:
                    68:a6:67:99:20:0d:e3:c5:c7:4d:f6:55:c3:c6:02:
                    76:54:e6:8c:9c:96:46:1a:d4:22:19:27:2c:50:6e:
                    61:1d:20:1c:78:4e:ba:03:7b:ce:81:20:5c:70:51:
                    ad:96:be:5c:ac:a0:bb:a8:07:06:b6:73:4c:3f:52:
                    d2:46:cd:80:bd:48:a4:4c:42:52:8a:43:9c:cc:2d:
                    91:f9:b9:ee:16:50:03:85:19
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Subject Key Identifier:
                C5:9A:67:D2:7D:B3:CE:64:53:01:68:9D:4A:AD:1A:E6:52:CA:BB:9C
            X509v3 Subject Alternative Name:
                URI:urn:publicid:IDN+wall2.ilabt.iminds.be+user+bvermeul, email:bvermeul@wall2.ilabt.iminds.be,
URI:urn:uuid:b349a4ef-149e-11e3-966a-001517becdc1
            Authority Information Access:
                2.25.305821105408246119474742976030998643995 - URI:https://www.wall2.ilabt.iminds.be:12369/protogeni/xmlrpc/sa
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix E: Subauthorities in a member and slice authority

Basic idea

At least the emulab slice authority and the GENI GPO slice authority do support sub-authorities to make authorities and experimenters further scalable.
The idea is based on putting projects/sub-authorities in place, and experimenters can belong to one or more projects. Each project has one or more PIs or project responsible which can approve other members of that project.

Scalability

The scalability comes in two ways:

- Authorization can be based on the sub-authority, this means that a particular testbed (aggregate manager) can let in all members of a specific project/sub-authority, but not other experimenters of that authority.
- Project creation has to be approved by authority administrators, but approval for members in the project is done by the project PI (delegation)

Practical example

A practical example may make it clearer:

- iMinds' Virtual Wall has a lot of users: iMinds people, students, industry members, Fed4fire partners, …
- Without sub-authorities, if a user comes at another AM with a virtual wall certificate, authorization has to be done on the individual member (e.g. with attributes or based on email)
- With sub-authorities, the following can be done: a project 'networks4ever' (e.g. a European project) is created at the virtual wall, and contains a number of members. This project has agreed with Bonfire and Nitos that the members of the project can use the resources of those testbeds. At that moment, Bonfire and Nitos authorize incoming sliver requests based on the sub-authority 'networks4ever' signing the slice credential. So, if new members join the project, or leave the project, the 'networks4ever' PI can just add or drop them in the sub-authority, while Nitos and Bonfire just check the sub-authority (no matter if the project contains students, industry, …). All resource usage can then also be done in the responsibility of 'networks4ever' e.g.
- In the emulab authority it is not yet forbidden, but will be in the future, but in the GENI GPO authority, you can get a credential, but if you do not belong to a project/sub-authority, you cannot create slices. (and projects/sub-authorities can have a limited life-time, e.g. during a real project, or during a practical exercise for students or demonstration). E.g. the tutorials at GEC use projects which are limited to one week after the conference, so you can test at home, but only for one week under the tutorial authorization.

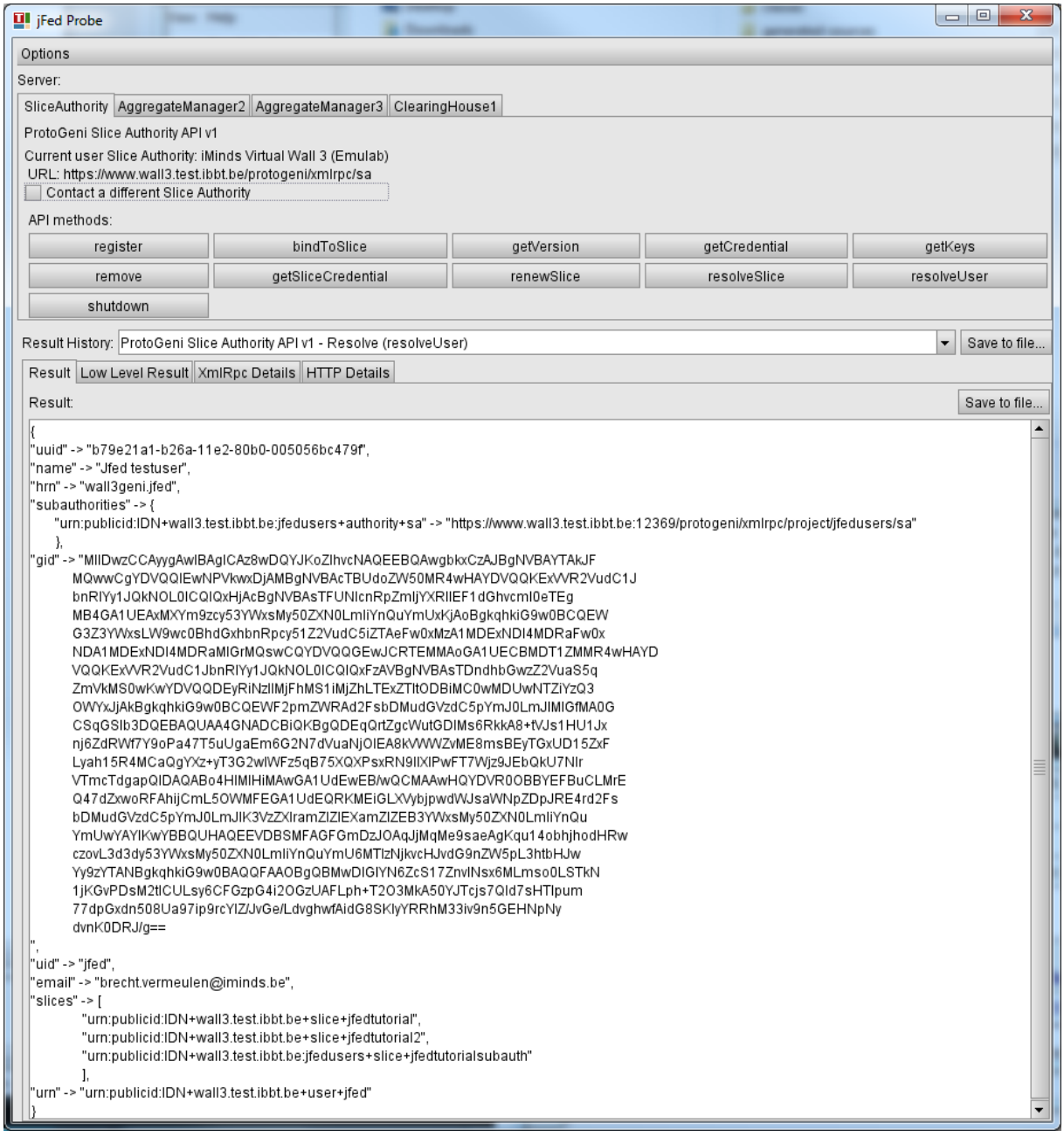
How to do this with the APIs by using jFed ?

At the end of cycle 1 the usage of subauthorities has already been explored and documented in the jFed tutorial (http://jfed.iminds.be). Since it describes how subauthorities can be used in real life in detail, it is a very useful additional illustration of the concepts discussed in this appendix. Therefore

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

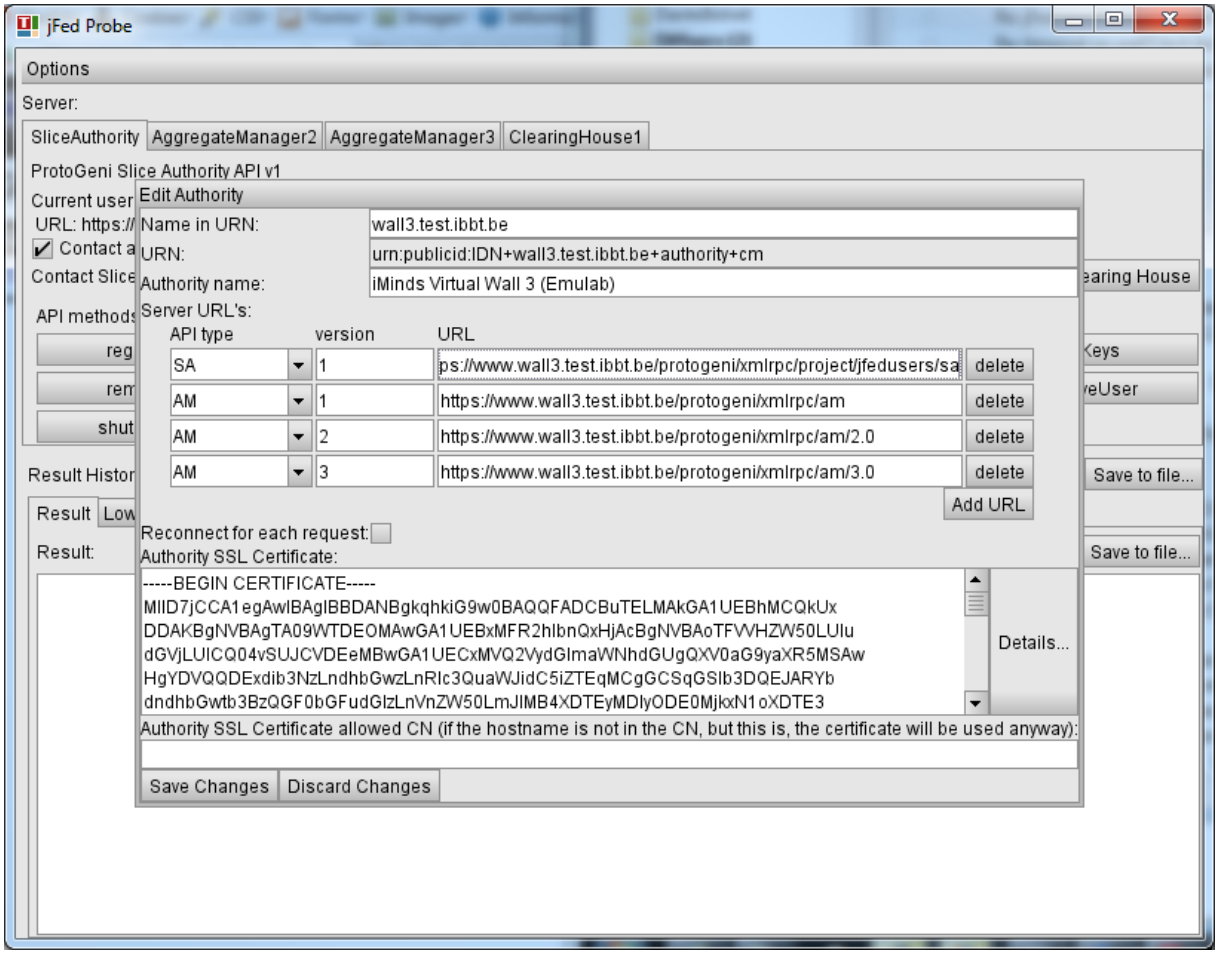
the most important parts of that tutorial have been copied below. Note that this tutorial is intended as an illustration of these architectural concepts, to make them as clear as possible. Such information can be situated somewhere between architectural descriptions and more detailed specifications. The remainder of this appendix is the extract of that jFed tutorial:

To demonstrate the concept of sub-authorities: A user can belong in multiple projects/sub-authorities. E.g. if you call 'resolveUser', you can see the subautorities to which a user belongs:



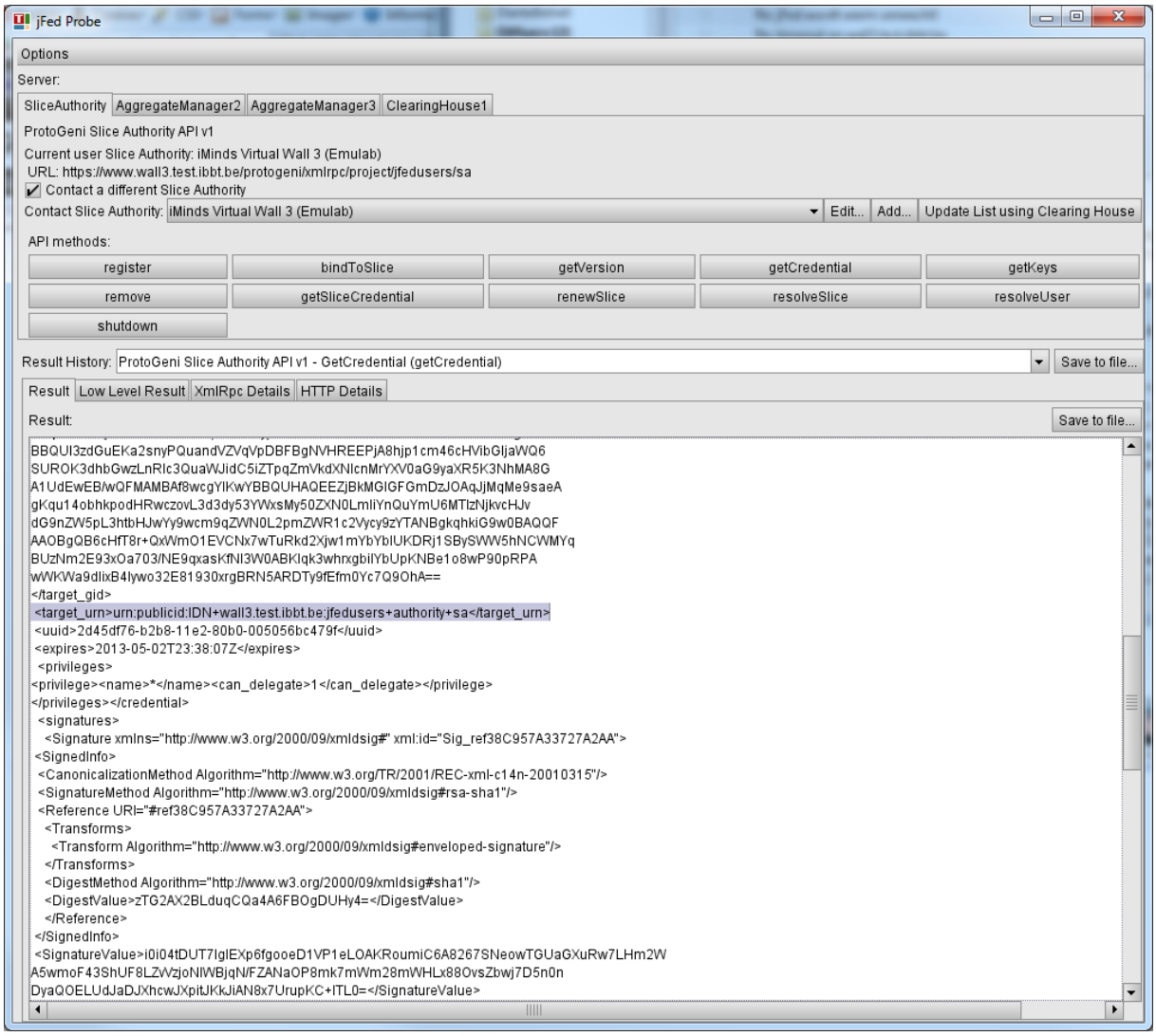
If you edit the slice authority, and put for the SA, a line as seen in the resolveUser (but without port 12369): https://www.wall3.test.ibbt.be/protogeni/xmlrpc/project/jfedusers/sa, you can create slices on this subauthority.
The advantage of this is, that other federating authorities can give access to a subpart of an authority, e.g. only slices belonging to jfed users are authorized.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

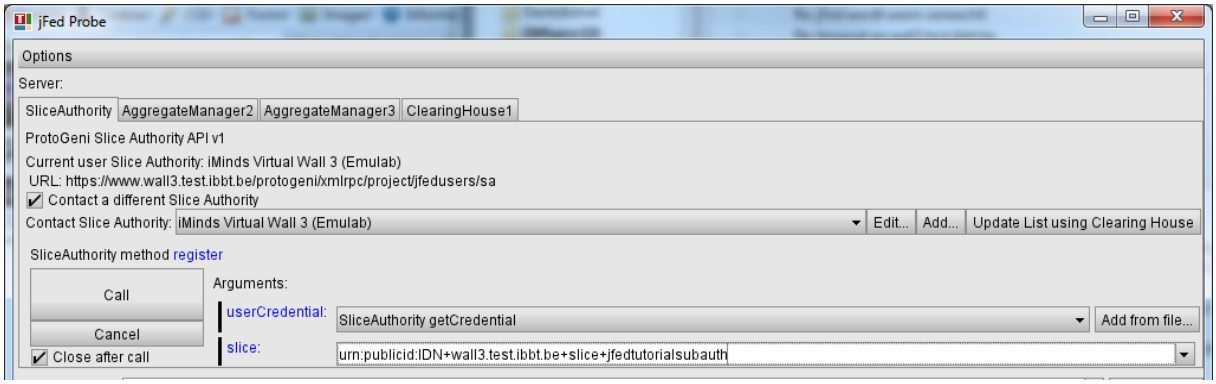


First, do a getCredential on this subauthority:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

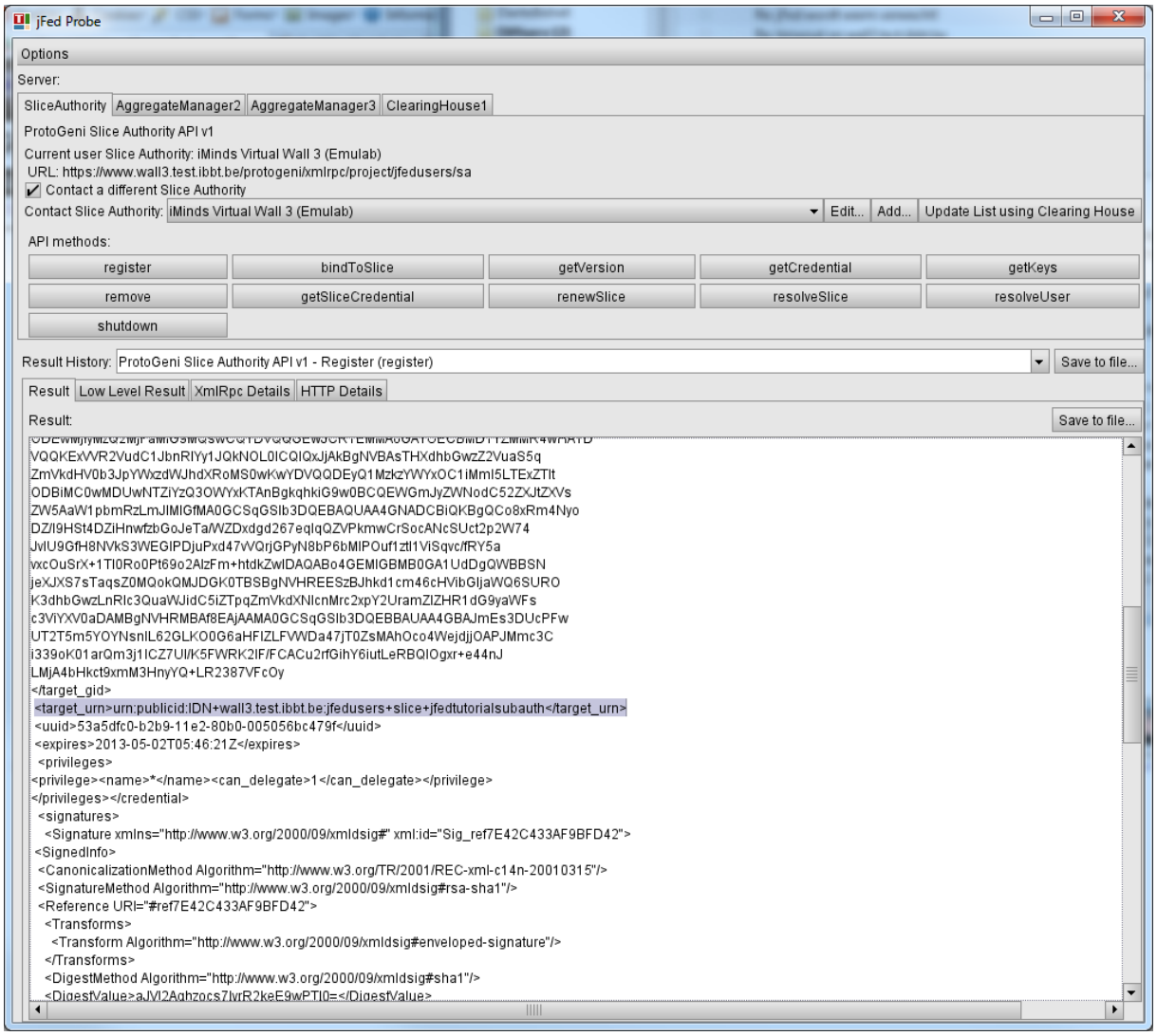


Then register a slice:



And you receive a slice from the subautority, which has a URN: urn:publicid:IDN+wall3.test.ibbt.be**:jfedusers**+slice+jfedtutorial

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



The details of this slice credential look like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<signed-credential                          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.protogeni.net/resources/credential/credential.xsd"
xsi:schemaLocation="http://www.protogeni.net/resources/credential/ext/policy/1
http://www.protogeni.net/resources/credential/ext/policy/1/policy.xsd">
  <credential xml:id="refB39085D4FD641534">
 <type>privilege</type>
 <serial>2807</serial>
 <owner_gid>MIIDwzCCAyygAwIBAgICAz8wDQYJKoZIhvcNAQEEBQAwgbkxCzAJBgNVBAYTAkJF
MQwwCgYDVQQIEwNPVkwxDjAMBgNVBAcTBUdoZW50MR4wHAYDVQQKExVVR2VudC1J
bnRlYy1JQkNOL0lCQlQxHjAcBgNVBAsTFUNlcnRpZmljYXRlIEF1dGhvcml0eTEg
MB4GA1UEAxMXYm9zcy53YWxsMy50ZXN0LmliYnQuYmUxKjAoBgkqhkiG9w0BCQEW
G3Z3YWxsLW9wc0BhdGxhbnRpcy51Z2VudC5iZTAeFw0xMzA1MDExNDI4MDRaFw0x
NDA1MDExNDI4MDRaMIGrMQswCQYDVQQGEwJCRTEMMAoGA1UECBMDT1ZMMR4wHAYD
VQQKExVVR2VudC1JbnRlYy1JQkNOL0lCQlQxFzAVBgNVBAsTDndhbGwzZ2VuaS5q
ZmVkMS0wKwYDVQQDEyRiNzllMjFhMS1iMjZhLTExZTItODBiMC0wMDUwNTZiYzQ3
OWYxJjAkBgkqhkiG9w0BCQEWF2pmZWRAd2FsbDMudGVzdC5pYmJ0LmJlMIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEqQrtZgcWutGDlMs6RkkA8+tVJs1HU1Jx
nj6ZdRWf7Y9oPa47T5uUgaEm6G2N7dVuaNjOlEA8kVWWZvME8msBEyTGxUD15ZxF
Lyah15R4MCaQgYXz+yT3G2wIWFz5qB75XQXPsxRN9IlXIPwFT7Wjz9JEbQkU7Nlr
VTmcTdgapQIDAQABo4HlMIHiMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFBuCLMrE
Q47dZxwoRFAhijCmL5OWMFEGA1UdEQRKMEiGLXVybjpwdWJsaWNpZDpJRE4rd2Fs
bDMudGVzdC5pYmJ0LmJlK3VzZXIramZlZIEXamZlZEB3YWxsMy50ZXN0LmliYnQu
YmUwYAYIKwYBBQUHAQEEVDBSMFAGFGmDzJOAqJjMqMe9saeAgKqu14obhjhodHRw
czovL3d3dy53YWxsMy50ZXN0LmliYnQuYmU6MTIzNjkvcHJvdG9nZW5pL3htbHJw
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

```
Yy9zYTANBgkqhkiG9w0BAQQFAAOBgQBMwDIGIYN6ZcS17ZnvINsx6MLmso0LSTkN
1jKGvPDsM2tlCULsy6CFGzpG4i2OGzUAFLph+T2O3MkA50YJTcjs7QId7sHTIpum
77dpGxdn508Ua97ip9rcYlZ/JvGe/LdvghwfAidG8SKIyYRRhM33iv9n5GEHNpNy
dvnK0DRJ/g==
</owner_gid>
 <owner_urn>urn:publicid:IDN+wall3.test.ibbt.be+user+jfed</owner_urn>
 <target_gid>MIIDbTCCAtagAwIBAgICA1YwDQYJKoZIhvcNAQEEBQAwgbkxCzAJBgNVBAYTAkJF
MQwwCgYDVQQIEwNPVkwxDjAMBgNVBAcTBUdoZW50MR4wHAYDVQQKExVVR2VudC1J
bnRlYy1JQkNOL0lCQlQxHjAcBgNVBAsTFUNlcnRpZmljYXRlIEF1dGhvcml0eTEg
MB4GA1UEAxMXYm9zcy53YWxsMy50ZXN0LmliYnQuYmUxKjAoBgkqhkiG9w0BCQEW
G3Z3YWxsLW9wc0BhdGxhbnRpcy51Z2VudC5iZTAeFw0xMzA1MDIxNzIxMTJaFw0x
ODEwMjMxODIxMTJaMIG6MQswCQYDVQQGEwJCRTEMMAoGA1UECBMDT1ZMMR4wHAYD
VQQKExVVR2VudC1JbnRlYy1JQkNOL0lCQlQxIzAhBgNVBAsTGndhbGwzZ2VuaS5k
ZW1vc3ViYXV0aG9yaXR5MS0wKwYDVQQDEyQxMTdhZGUyNi1iMzU1LTExZTItODBi
MC0wMDUwNTZiYzQ3OWYxKTAnBgkqhkiG9w0BCQEWGmJyZWNodC52ZXJtZXVsZW5A
aW1pbmRzLmJlMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCoMaf/jZRRGFbI
us5cPrHS2I1q8K4hGcCEWFPQESg1KS8pe8II4Zzv+Q0bB8+28ECF/avrVOgUIKVd
H6LH9qHIBS02v7XcOlsOhyIp0C6AMt9YJe6DUOAyHL3ePaEZ7WGhlzyL8kfXieR1
7eQLSz3QAwdfrj5Re0Y2FFAa0htbvwIDAQABo4GAMH4wHQYDVR0OBBYEFIC+vg+M
AUayuXkTZSVk9hKJKF/rME8GA1UdEQRIMEaGRHVybjpwdWJsaWNpZDpJRE4rd2Fs
bDMudGVzdC5pYmJ0LmJlOmpmZWR1c2VycytzbGljZStkZW1vc3ViYXV0aG9yaXR5
MAwGA1UdEwEB/wQCMAAwDQYJKoZIhvcNAQEEBQADgYEAg1IOS13GViRRgMcqIYwz
b1ALKds7OzeXgba3MnjJ+67ZUadxFyvvcv0o+CESGq9FFpbXS5X688nuTYUxPTRc
HS27liaLy0IIFnh908Q87NMorIyXFkO8/232P1P+0tynGtVCexQcQRUFKauJ1S7O
/P9CouSgNfv6XSc/UyJrL/w=
</target_gid>
 <target_urn>urn:publicid:IDN+wall3.test.ibbt.be:jfedusers+slice+demosubauthority</target_urn>
 <uuid>118d7e15-b355-11e2-80b0-005056bc479f</uuid>
 <expires>2013-05-03T00:21:12Z</expires>
```


Interpret slice credential

If you put a
-----BEGIN CERTIFICATE-----
-----END CERTIFICATE-----
around the gid, you can look into them with
openssl x509 -text -in gid1.pem

User credential

```
<owner_urn>urn:publicid:IDN+wall3.test.ibbt.be+user+jfed</owner_urn>
```

Below you see that the slice authority has signed the user certificate, and you can find an email address of the slice authority administrators and the user (jfed@wall3.test.ibbt.be)
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number: 831 (0x33f)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=BE, ST=OVL, L=Ghent, O=UGent-Intec-IBCN/IBBT, OU=Certificate Authority, CN=boss.wall3.test.ibbt.be/emailAddress=vwall-ops@atlantis.ugent.be
    Validity
      Not Before: May 1 14:28:04 2013 GMT
      Not After : May 1 14:28:04 2014 GMT
    Subject: C=BE, ST=OVL, O=UGent-Intec-IBCN/IBBT, OU=wall3geni.jfed, CN=b79e21a1-b26a-11e2-80b0-005056bc479f/emailAddress=jfed@wall3.test.ibbt.be
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
      RSA Public Key: (1024 bit)

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

```
            Modulus (1024 bit):
               00:c4:a9:0a:ed:66:07:16:ba:d1:83:94:cb:3a:46:
               49:00:f3:eb:55:26:cd:47:53:52:71:9e:3e:99:75:
               15:9f:ed:8f:68:3d:ae:3b:4f:9b:94:81:a1:26:e8:
               6d:8d:ed:d5:6e:68:d8:ce:94:40:3c:91:55:96:66:
               f3:04:f2:6b:01:13:24:c6:c5:40:f5:e5:9c:45:2f:
               26:a1:d7:94:78:30:26:90:81:85:f3:fb:24:f7:1b:
               6c:08:58:5c:f9:a8:1e:f9:5d:05:cf:b3:14:4d:f4:
               89:57:20:fc:05:4f:b5:a3:cf:d2:44:6d:09:14:ec:
               d9:6b:55:39:9c:4d:d8:1a:a5
            Exponent: 65537 (0x10001)
         X509v3 extensions:
            X509v3 Basic Constraints: critical
               CA:FALSE
            X509v3 Subject Key Identifier:
               1B:82:2C:CA:C4:43:8E:DD:67:1C:28:44:50:21:8A:30:A6:2F:93:96
            X509v3 Subject Alternative Name:
               URI:urn:publicid:IDN+wall3.test.ibbt.be+user+jfed, email:jfed@wall3.test.ibbt.be
            Authority Information Access:
               2.25.305821105408246119474742976030998643995                -
URI:https://www.wall3.test.ibbt.be:12369/protogeni/xmlrpc/sa

      Signature Algorithm: md5WithRSAEncryption
         4c:c0:32:06:21:83:7a:65:c4:b5:ed:99:ef:20:db:31:e8:c2:
         e6:b2:8d:0b:49:39:0d:d6:32:86:bc:f0:ec:33:6b:65:09:42:
         ec:cb:a0:85:1b:3a:46:e2:2d:8e:1b:35:00:14:ba:61:f9:3d:
         8e:dc:c9:00:e7:46:09:4d:c8:ec:ed:02:1d:ee:c1:d3:22:9b:
         a6:ef:b7:69:1b:17:67:e7:4f:14:6b:de:e2:a7:da:dc:62:56:
         7f:26:f1:9e:fc:b7:6f:82:1c:1f:02:27:46:f1:22:88:c9:84:
         51:84:cd:f7:8a:ff:67:e4:61:07:36:93:72:76:f9:ca:d0:34:
         49:fe
-----BEGIN CERTIFICATE-----
MIIDwzCCAyygAwIBAgICAz8wDQYJKoZIhvcNAQEEBQAwgbkxCzAJBgNVBAYTAkJF
MQwwCgYDVQQIEwNPVkwxDjAMBgNVBAcTBUdoZW50MR4wHAYDVQQKExVVR2VudC1J
bnRlYy1JQkNOL0lCQlQxHjAcBgNVBAsTFUNlcnRpZmljYXRlIEF1dGhvcml0eTEg
MB4GA1UEAxMXYm9zcy53YWxsMy50ZXN0LmliYnQuYmUxKjAoBgkqhkiG9w0BCQEW
G3Z3YWxsLW9wc0BhdGxhbnRpcy51Z2VudC5iZTAeFw0xMzA1MDExNDI4MDRaFw0x
NDA1MDExNDI4MDRaMIGrMQswCQYDVQQGEwJCRTEMMAoGA1UECBMDT1ZMMR4wHAYD
VQQKExVVR2VudC1JbnRlYy1JQkNOL0lCQlQxFzAVBgNVBAsTDndhbGwzZ2VuaS5q
ZmVkMS0wKwYDVQQDEyRiNzllMjFhMS1iMjZhLTExZTItODBiMC0wMDUwNTZiYzQ3
OWYxJjAkBgkqhkiG9w0BCQEWF2pmZWRAd2FsbDMudGVzdC5pYmJ0LmJlMIGfMA0G
CSqGSIb3DQEBAQUAA4GNADCBiQKBgQDEqQrtZgcWutGDlMs6RkkA8+tVJs1HU1Jx
nj6ZdRWf7Y9oPa47T5uUgaEm6G2N7dVuaNjOlEA8kVWWZvME8msBEyTGxUD15ZxF
Lyah15R4MCaQgYXz+yT3G2wIWFz5qB75XQXPsxRN9IlXIPwFT7Wjz9JEbQkU7Nlr
VTmcTdgapQIDAQABo4HlMIHiMAwGA1UdEwEB/wQCMAAwHQYDVR0OBBYEFBuCLMrE
Q47dZxwoRFAhijCmL5OWMFEGA1UdEQRKMEiGLXVybjpwdWJsaWNpZDpJRE4rd2Fs
bDMudGVzdC5pYmJ0LmJlK3VzZXIramZlZIEXamZlZEB3YWxsMy50ZXN0LmliYnQu
YmUwYAYIKwYBBQUHAQEEVDBSMFAGFGmDzJOAqJjMqMe9saeAgKqu14obhjhodHRw
czovL3d3dy53YWxsMy50ZXN0LmliYnQuYmU6MTIzNjkvcHJvdG9nZW5pL3htbHJw
Yy9zYTANBgkqhkiG9w0BAQQFAAOBgQBMwDIGIYN6ZcS17ZnvINsx6MLmso0LSTkN
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

1jKGvPDsM2tlCULsy6CFGzpG4i2OGzUAFLph+T2O3MkA50YJTcjs7QId7sHTIpum
77dpGxdn508Ua97ip9rcYlZ/JvGe/LdvghwfAidG8SKIyYRRhM33iv9n5GEHNpNy
dvnK0DRJ/g==
-----END CERTIFICATE-----

Slice credential

<target_urn>urn:publicid:IDN+wall3.test.ibbt.be:jfedusers+slice+demosubauthority</target_urn>
<expires>2013-05-03T00:21:12Z</expires>

In the URN (:jfedusers), you see that it is a slice 'demosubauthority' within the subauthority jfedusers of wall3.test.ibbt.be.

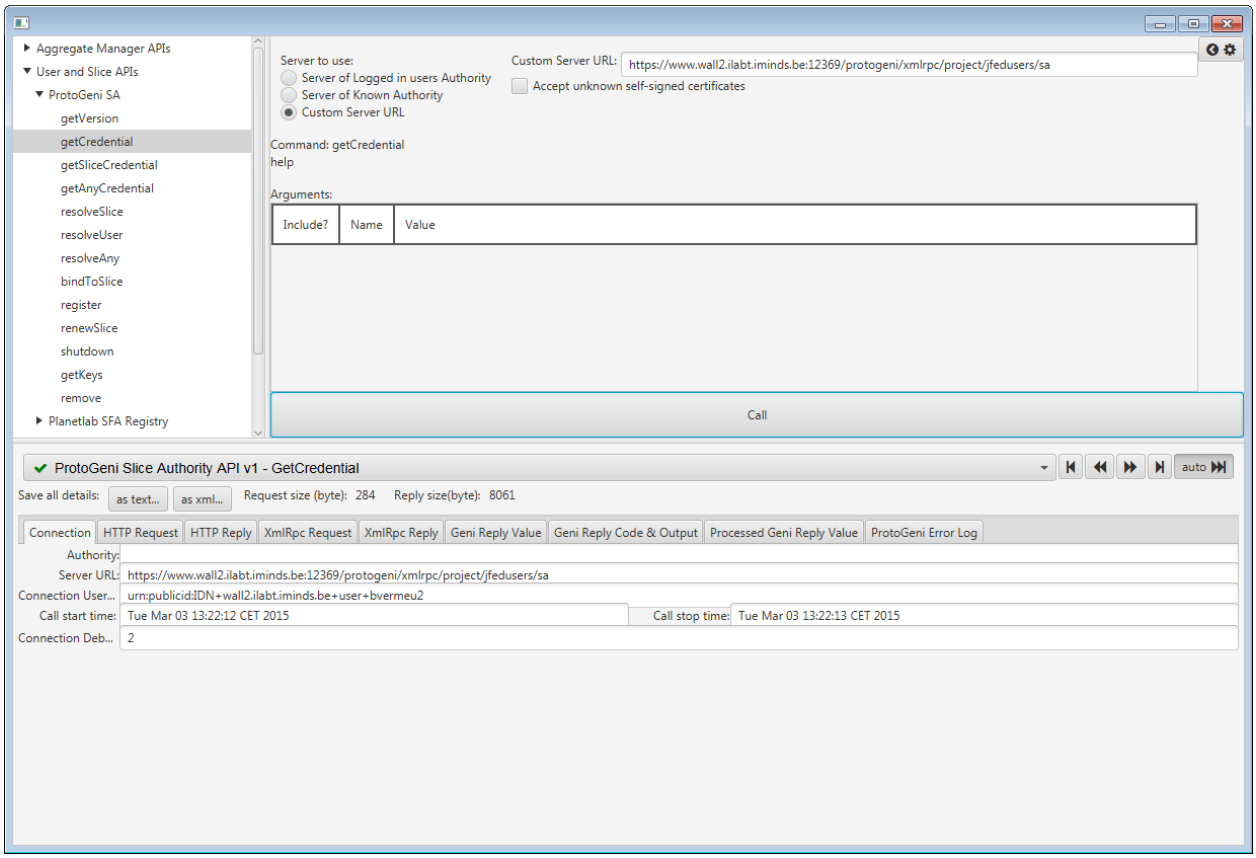
Certificate:
Data:
Version: 3 (0x2)
Serial Number: 854 (0x356)
Signature Algorithm: md5WithRSAEncryption
Issuer: C=BE, ST=OVL, L=Ghent, O=UGent-Intec-IBCN/IBBT, OU=Certificate Authority, CN=boss.wall3.test.ibbt.be/emailAddress=vwall-ops@atlantis.ugent.be
Validity
Not Before: May 2 17:21:12 2013 GMT
Not After : Oct 23 18:21:12 2018 GMT
Subject: C=BE, ST=OVL, O=UGent-Intec-IBCN/IBBT, OU=wall3geni.demosubauthority, CN=117ade26-b355-11e2-80b0-005056bc479f/emailAddress=brecht.vermeulen@iminds.be
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
RSA Public Key: (1024 bit)
Modulus (1024 bit):
00:a8:31:a7:ff:8d:94:51:18:56:c8:ba:ce:5c:3e:
b1:d2:d8:8d:6a:f0:ae:21:19:c0:84:58:53:d0:11:
28:35:29:2f:29:7b:c2:08:e1:9c:ef:f9:0d:1b:07:
cf:b6:f0:40:85:fd:ab:eb:54:e8:14:20:a5:5d:1f:
a2:c7:f6:a1:c8:05:2d:36:bf:b5:dc:3a:5b:0e:87:
22:29:d0:2e:80:32:df:58:25:ee:83:50:e0:32:1c:
bd:de:3d:a1:19:ed:61:a1:97:3c:8b:f2:47:d7:89:
e4:75:ed:e4:0b:4b:3d:d0:03:07:5f:ae:3e:51:7b:
46:36:14:50:1a:d2:1b:5b:bf
Exponent: 65537 (0x10001)
X509v3 extensions:
X509v3 Subject Key Identifier:
80:BE:BE:0F:8C:01:46:B2:B9:79:13:65:25:64:F6:12:89:28:5F:EB
X509v3 Subject Alternative Name:
URI:urn:publicid:IDN+wall3.test.ibbt.be:jfedusers+slice+demosubauthority
X509v3 Basic Constraints: critical
CA:FALSE
Signature Algorithm: md5WithRSAEncryption
83:52:0e:4b:5d:c6:56:24:51:80:c7:2a:21:8c:33:6f:50:0b:
29:db:3b:3b:37:97:81:b6:b7:32:78:c9:fb:ae:d9:51:a7:71:
17:2b:ef:72:fd:28:f8:21:12:1a:af:45:16:96:d7:4b:95:fa:
f3:c9:ee:4d:85:31:3d:34:5c:1d:2d:bb:96:26:8b:cb:42:08:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

16:78:7d:d3:c4:3c:ec:d3:28:ac:8c:97:16:43:bc:ff:6d:f6:
3f:53:fe:d2:dc:a7:1a:d5:42:7b:14:1c:41:15:05:29:ab:89:
d5:2e:ce:fc:ff:42:a2:e4:a0:35:fb:fa:5d:27:3f:53:22:6b:
2f:fc
-----BEGIN CERTIFICATE-----
MIIDbTCCAtagAwIBAgICA1YwDQYJKoZIhvcNAQEEBQAwgbkxCzAJBgNVBAYTAkJF
MQwwCgYDVQQIEwNPVkwxDjAMBgNVBAcTBUdoZW50MR4wHAYDVQQKExVVR2VudC1J
bnRlYy1JQkNOL0lCQlQxHjAcBgNVBAsTFUNlcnRpZmljYXRlIEF1dGhvcml0eTEg
MB4GA1UEAxMXYm9zcy53YWxsMy50ZXN0LmliYnQuYmUxKjAoBgkqhkiG9w0BCQEW
G3Z3YWxsLW9wc0BhdGxhbnRpcy51Z2VudC5iZTAeFw0xMzA1MDIxNzIxMTJaFw0x
ODEwMjMxODIxMTJaMIG6MQswCQYDVQQGEwJCRTEMMAoGA1UECBMDT1ZMMR4wHAYD
VQQKExVVR2VudC1JbnRlYy1JQkNOL0lCQlQxIzAhBgNVBAsTGndhbGwzZ2VuaS5k
ZW1vc3ViYXV0aG9yaXR5MS0wKwYDVQQDEyQxMTdhZGUyNi1iMzU1LTExZTItODBi
MC0wMDUwNTZiYzQ3OWYxKTAnBgkqhkiG9w0BCQEWGmJyZWNodC52ZXJtZXVsZW5A
aW1pbmRzLmJlMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCoMaf/jZRRGFbI
us5cPrHS2I1q8K4hGcCEWFPQESg1KS8pe8II4Zzv+Q0bB8+28ECF/avrVOgUIKVd
H6LH9qHIBS02v7XcOlsOhyIp0C6AMt9YJe6DUOAyHL3ePaEZ7WGhlzyL8kfXieR1
7eQLSz3QAwdfrj5Re0Y2FFAa0htbvwIDAQABo4GAMH4wHQYDVR0OBBYEFIC+vg+M
AUayuXkTZSVk9hKJKF/rME8GA1UdEQRIMEaGRHVybjpwdWJsaWNpZDpJRE4rd2Fs
bDMudGVzdC5pYmJ0LmJlOmpmZWR1c2VycytzbGljZStkZW1vc3ViYXV0aG9yaXR5
MAwGA1UdEwEB/wQCMAAwDQYJKoZIhvcNAQEEBQADgYEAg1IOS13GViRRgMcqIYwz
b1ALKds7OzeXgba3MnjJ+67ZUadxFyvvcv0o+CESGq9FFpbXS5X688nuTYUxPTRc
HS27liaLy0IIFnh908Q87NMorIyXFkO8/232P1P+0tynGtVCexQcQRUFKauJ1S7O
/P9CouSgNfv6XSc/UyJrL/w=
-----END CERTIFICATE-----

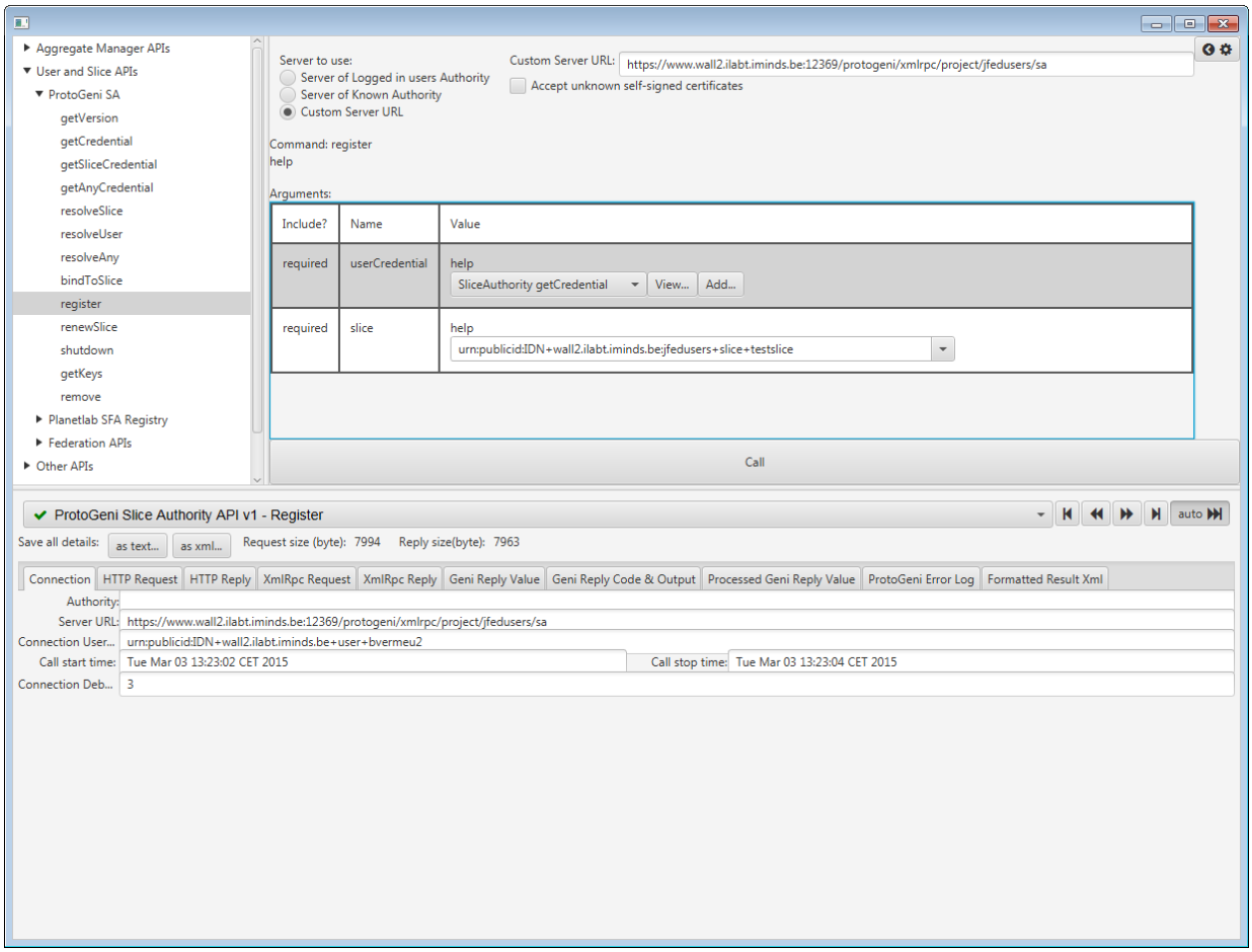
In a newer jFed probe, it looks like this:
Getcredential call
Use custom server URL:
https://www.wall2.ilabt.iminds.be:12369/protogeni/xmlrpc/project/jfedusers/sa
(jfedusers is the name of the project)

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



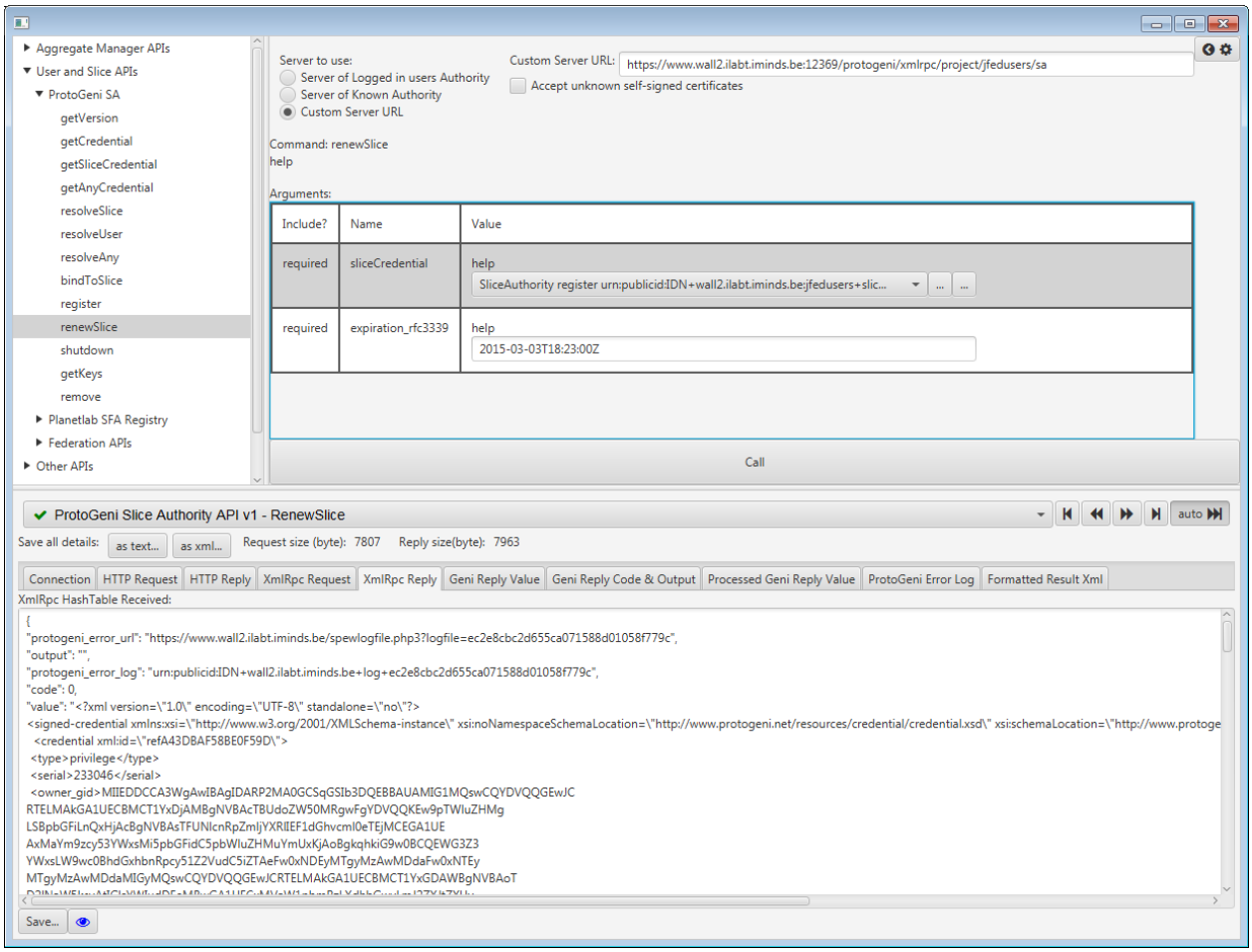
Then get a slice in the same subauthority:
Call register
Same custom server URL
Then Edit the slice name and add **:jfedusers+** and the slice name (testslice in the example)
Click call

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



To make the slice longer, chose ‘renewSlice’ call and type in a new experiation date.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix F: Evaluation of possible architectural approaches for SLA management

Two different approaches for the implementation of SLA management in Fed4FIRE have been considered. Both are similar as far as requirement coverage is concerned but they have different implications for the federation and the testbeds, which will be analysed the remainder of this subsection.

Before we go deeper into the different architectural possibilities, we should also clarify that, regardless of the architecture chosen, the SLA mangement itself can be centralised or distributed, depending on how thick the federation layer is. In the case of Fed4FIRE, with a multi-provider, multi-technology and multi-resource/service situation within only one experiment, both are possible:

- **Fed4FIRE provides federated centralized SLAs**: An SLA is agreed between the experimenter and the federation as an entity. The federation deals with underlying testbeds as a sort of SLA broker.
- **Fed4FIRE provides federated SLA setup**: An SLA is agreed between the experimenter and every single testbed involved in an experiment having adopted SLAs. The federation ensures there is an SLA established with every testbed before the experiment begins.
  In this case, on an experiment basis, the SLAs should be agreed between the experimenter and all the SLA-enabled testbeds involved in the experiment. This means that the SLAs are not agreed between the experimenter and some federation entity. Instead, the federated SLA management makes sure that, before an experiment begins, all required SLAs have been agreed between the involved parties. The main advantages of this approach, as far as we see it, are:
    - The federation layer is kept as thin as possible like this, which is in line with the current WP2 approach.
    - There are no billing/conciliation functionalities in Fed4FIRE federation for the moment (some testbeds intend to operate commercially; those testbeds will deal directly with experimenters for billing and conciliation).
- **Fed4FIRE provides a federated SLA tool**: This approach is related to the previous one (Fed4FIRE provides federated SLA setup), but in this case the federation does not ensure that there is an SLA established with every testbed before the experiment begins. Every testbed having adopted SLAs will expose its SLAs and will manage its SLA commitments individually. With this approach, the federation only provides the set of tools for a testbed to implement SLA management. It's up to the testbed to use it or not.

The following table gathers a comparison of these alternatives.

	Advantages	Disadvantages	Selected as final approach
Federated centralized SLA	• Easier for the experimenter	• More complicated to implement	
Federated SLA setup	• Fits Fed4FIRE philosophy (light federation layer)	• Less easy (more bureaucratic) for the experimenter	
Federated SLA tool	• Fits Fed4FIRE philosophy (light federation layer) • Testbeds have the freedom to implement it or not.	• More bureaucratic for the experimenter and more difficult to	X

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

	• Less modifications of the existing tool are required since currently it only works for one provider	implement for the testbeds	

Table 2: Comparison of approaches for implementing SLA Management

Taking into account a federated SLA tool as the most suitable approach for Fed4FIRE, the different architectural possibilities can be presented.

Centralised approach

In the centralised approach, the SLA management is a centralised component of the architecture. This can be implemented by considering the SLA management module like an upper layer to SFA. In this case, there is only one SLA Management module, which has advantages and disadvantages that are detailed bellow:

- Advantages:
  - There is only one SLA Management module in Fed4FIRE, which is easier for maintenance.
  - This option requires less implementation effort from testbeds.
  - The experimenter retrieves SLAs from only one centralised place.

- Disadvantages:
  - It implies important modifications in the state-of-the-art SLA solution (which does not currently operate in a multi-provider environment) and integration within a testbed whenever a new testbed is incorporated to the federation, depending on the monitoring system used by this testbed and whether it is SFA compliant or not.
  - The needed communication with the Aggregator Manager (AM) of each testbed and the possible different monitoring agents is complex.

Regarding the current technical situation, this approach implies:

- The project's MySlice portal and other experimenter tools (e.g. jFed, Omni, SFI, Flack) need to be able to display SLA options for an experimenter. In the particular case of the portal, new MySlice plugins to be integrated in existing forms are required.
- The SLA Management module needs to develop new SFA calls in order to be able to provide the following functionalities: publication, discovery and negotiation of SLAs requirements. That is to say, through this interface, SLA management communicates with the AM of the different testbeds and gets information about which SLAs and resources are offered by each testbed, and even just before doing the SLA negotiation between both parties (experimenter and testbed), it may allow knowing if the resources required by the experimenter are available or not.
- API Rest: this API is needed for the creation and the management of SLAs.
- OML Interface: to communicate with the different monitoring agents in order to obtain monitoring information.

Distributed approach

The second option considered is to include the SLA management as a distributed component inside each testbed. The distributed approach has advantages and disadvantages that are detailed bellow.

- Advantages:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

  - Thin federation layer.
  - Flexibility: it's up to every testbed to incorporate the SLA module.
  - This option requires fewer modifications in the existing SLA solution since it currently deals with only one provider.
  - It is easier to adapt within the PDP mechanisms deployed at each testbed.
  - It allows a testbed to use the SLA management outside of Fed4FIRE for its own purposes.

- Disadvantages:
  - It requires more implementation effort from the testbeds.
  - This option requires more coordination between testbeds in order for all implementations to operate in a federated and homogeneous manner. Otherwise the interaction for the experimenter would be more difficult.

Regarding the current technical situation, this approach implies:

- The project's MySlice portal and other experimenter tools need to be able to display SLA options for an experimenter. In the particular case of the portal, new MySlice plugins needs to be implemented.
- A new catalogue database containing all the available resources/services offered by the different testbeds is required in every testbed. This catalogue needs to be interfaced with client tools.
- Each testbed adopting SLAs manages them in a distributed manner. There is no aggregated processing of SLAs at a federated level.
- The experimenter agrees only one general SLA per experiment, which is split in as many SLAs as testbeds are involved in the experiment. In the end, there will be one SLA between the experimenter and each testbed involved in the experiment.
- The experimenter gets the SLA evaluation information individually from every testbed, for all testbeds with SLA mechanisms involved in the experiment. Aggregation of results could be tackled in cycle 3.

Comparison

The following table summarizes the main advantages and disadvantages of these different solutions:

Approach	Advantages	Disadvantages	Selected as final approach
Centralized approach	• Easier for maintenance (only one SLA management module) • Experimenter retrieves SLAs from only one centralised location.	• More complicated to implement (communication with all the AM of each testbed and all the possible different monitoring agents). • It does not fit Fed4FIRE 's WP2 philosophy (more complex federation layer)	

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Approach	Advantages	Disadvantages	Selected as final approach
Distributed approach	• Thin federation layer • Flexibility: it's up to every testbed to incorporate the SLA module. • Less modifications in the existing SLA tool (it currently deals with only one provider) • It is easier to adapt within the PDP mechanisms deployed at each testbed.	• More implementation effort from testbeds in order to operate in a federated and homogeneous manner.	X

Table 3: Comparison of architectural approaches for implementing SLA Management

The distributed approach seems to be more suitable and flexible for cycle 2 of Fed4FIRE. Besides, this solution can evolve towards a more centralized management approach (by aggregation) in case it is required in the future.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix G: Workflow for registering an account and getting a certificate

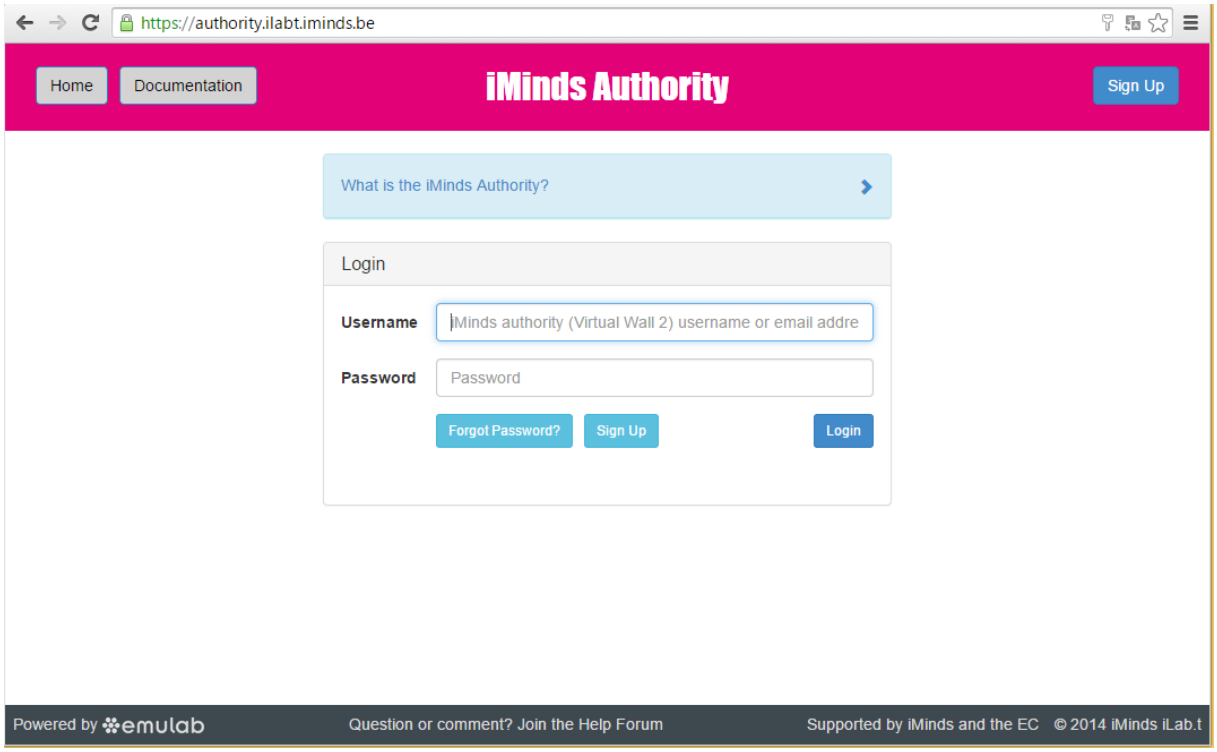
This appendix shows one of the possible workflows to get an account and certificate from a Fed4FIRE authority. Note that this presented workflow is intended as an illustration of the architectural concepts described in this deliverable, to make them as clear as possible. The actual content of this appendix can be situated somewhere between architectural descriptions and more detailed specifications. Therefore it is likely that this information (possibly in more detail) will also become part of WP5's specifications for cycle 2 in deliverable D5.2.

Introduction

Fed4FIRE works with X.509 certificates to authenticate and authorize experimenters on testbeds. In this section we will guide you through creating such a certificate. Testbeds can decide which certificates from which authorities they accept and as such federations of testbeds and authorities are established. Fed4FIRE is such a federation.
In the end, you will have a file on your local PC which contains this certificate information and this can then be used in experimenter tools.
This procedure has to be run only one.

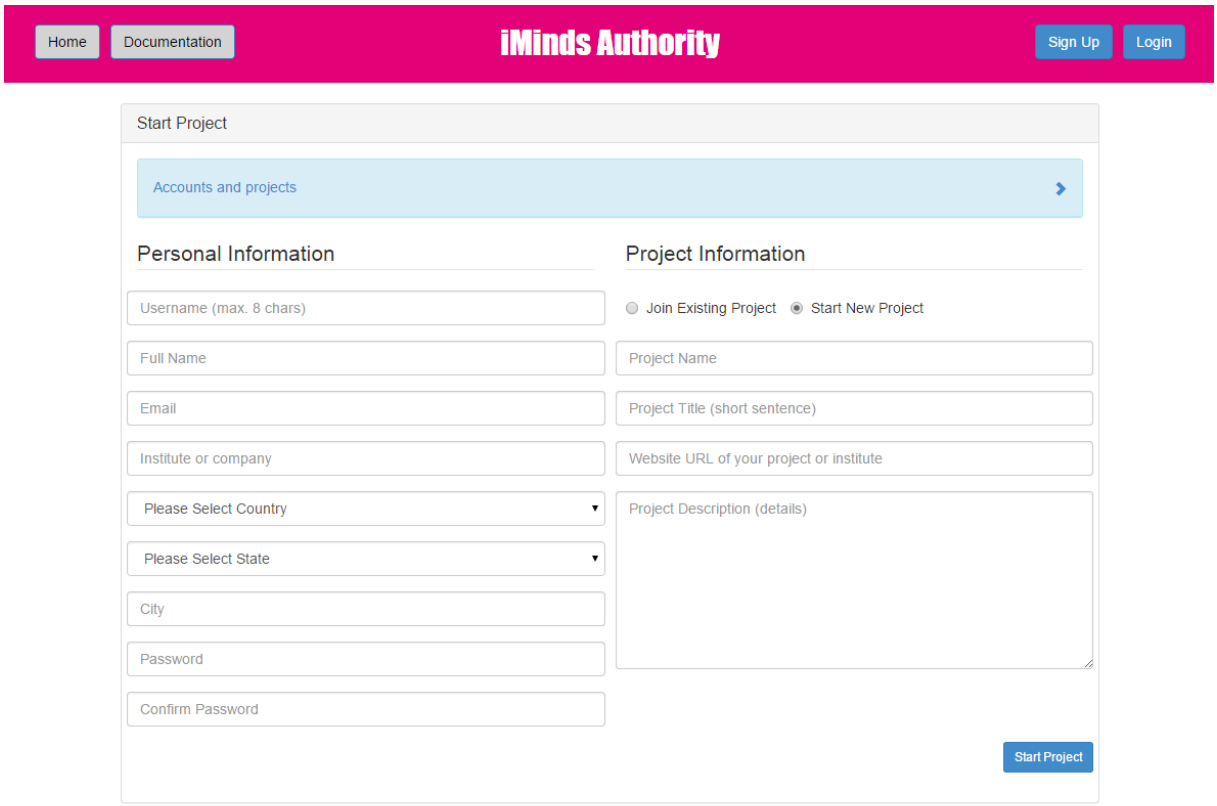
Register for an account

The current authority which can be used in the Fed4FIRE federation is located at https://authority.ilabt.iminds.be.



We click on `Sign up`.

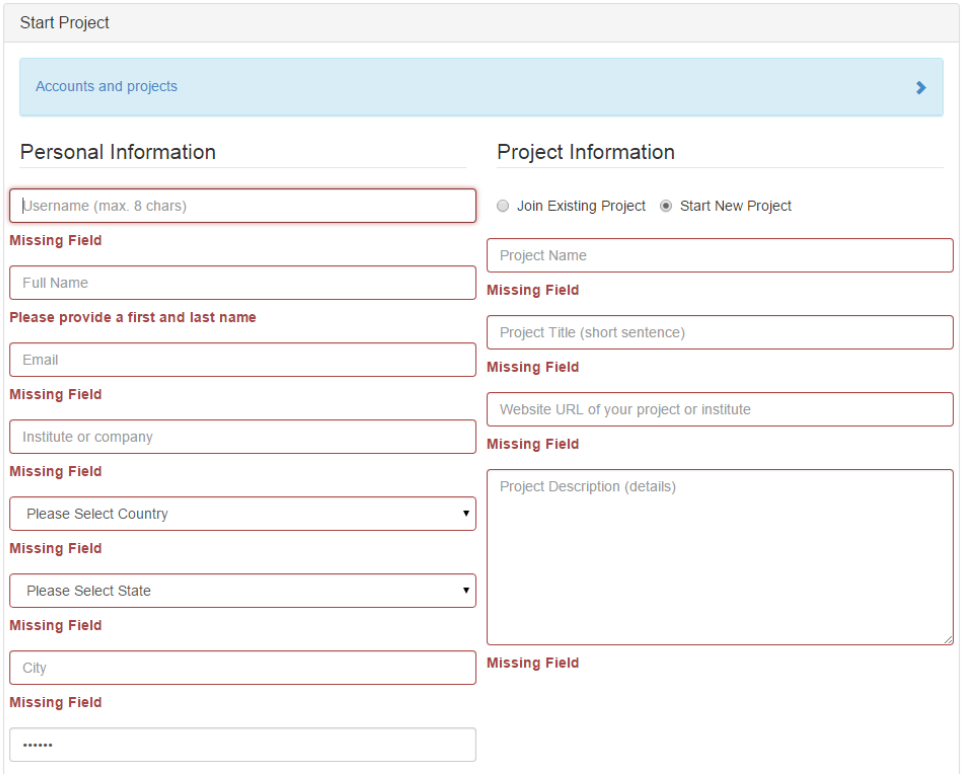
© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



The authority has the concept of Projects which bundle multiple experimenters. In a Project, the PIs (Principal Investigators) can approve new experimenters for that project, without Fed4FIRE administrators needing to approve this.
Here we will start with creating a new project by clicking Start New Project. If you have been invited to a Project, just click Join an Existing Project (or you might have received a specific URL). Now, we have to fill in our account information.

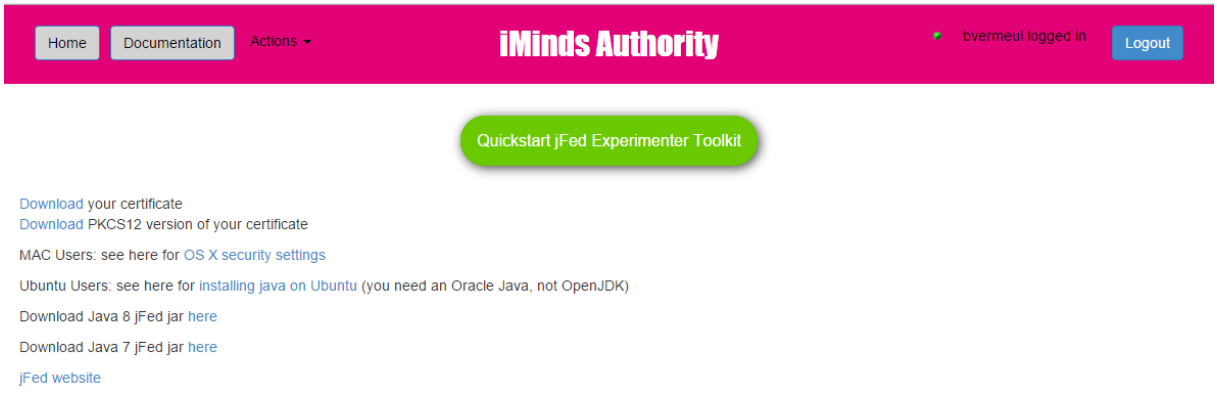
If there are some problems when submitting the form, this will be indicated clearly:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



If all goes well, you will receive an email in your mailbox to verify your account. This will contain a code you need to type in at the portal to verify your account.

After your account has been approved by an administrator (or the PI of the project you joined), then you can easily start up jFed with the right credentials. (or download your signed certificate to use with another tool)



© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix H: Workflow for creating an SSH key on different operating systems

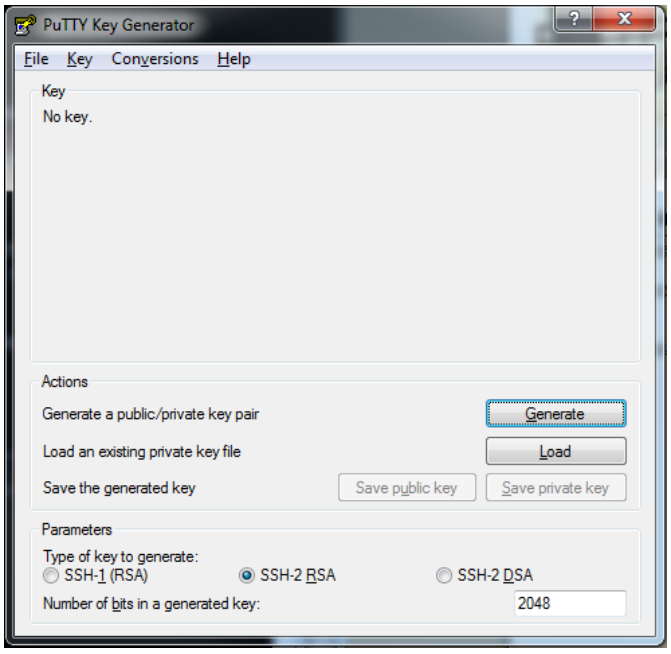
Introduction

In Fed4FIRE, access to testbed resources is done through SSH and more specifically SSH based on public/private key authentication. This section will guide you through the process of generating an SSH key. If you already have an SSH key, you can skip this. Note that this presented workflow is intended as an illustration of the architectural concepts described in this deliverable, to make them as clear as possible. The actual content of this appendix can be situated somewhere between architectural descriptions and more detailed specifications. Therefore it is likely that this information (possibly in more detail) will also become part of WP5's specifications for cycle 2 in deliverable D5.2.

Windows platform: PuTTY

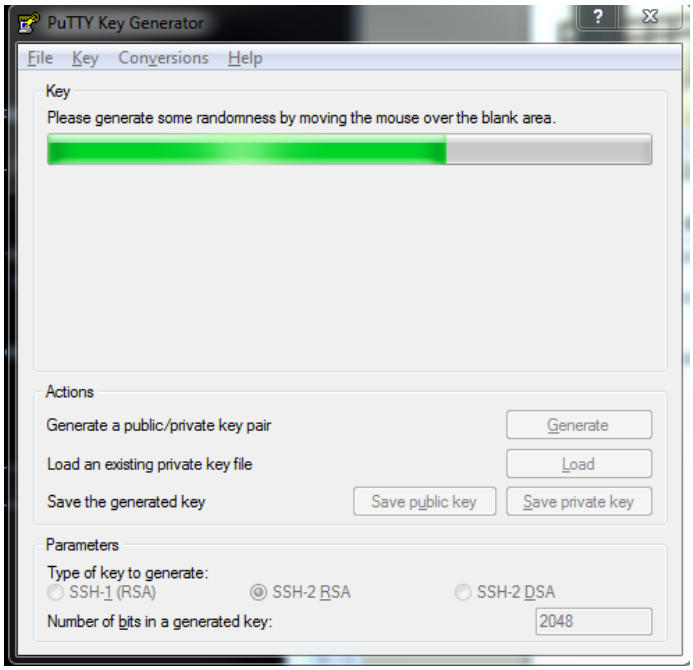
PuTTY is a widely used SSH client for Windows and it includes the tool PuTTYgen to create an SSH key. You can download PuTTY from http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html and you can download an installer of the latest version here.

To generate an SSH key, start the tool PuTTYgen.

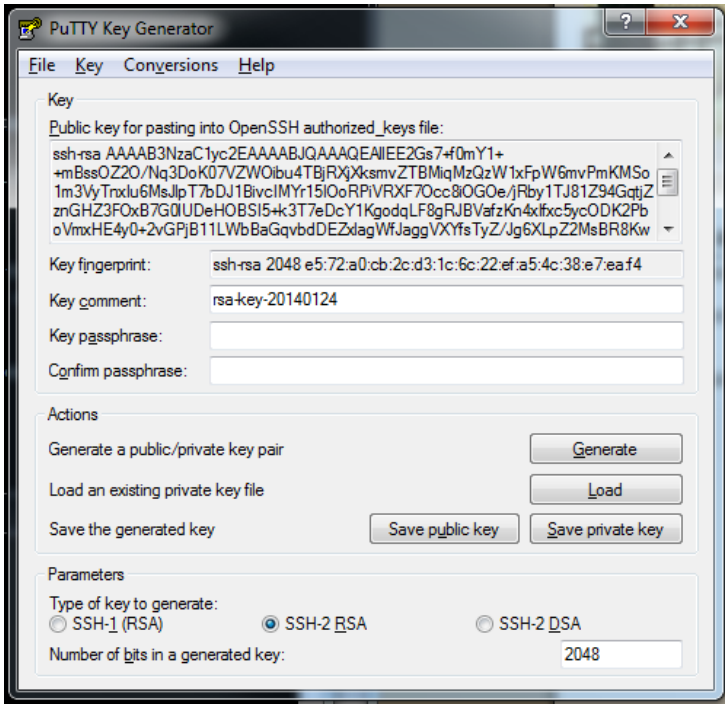


If you click on `Generate`, it will start collecting random information while you move your mouse over the blank session as indicated.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

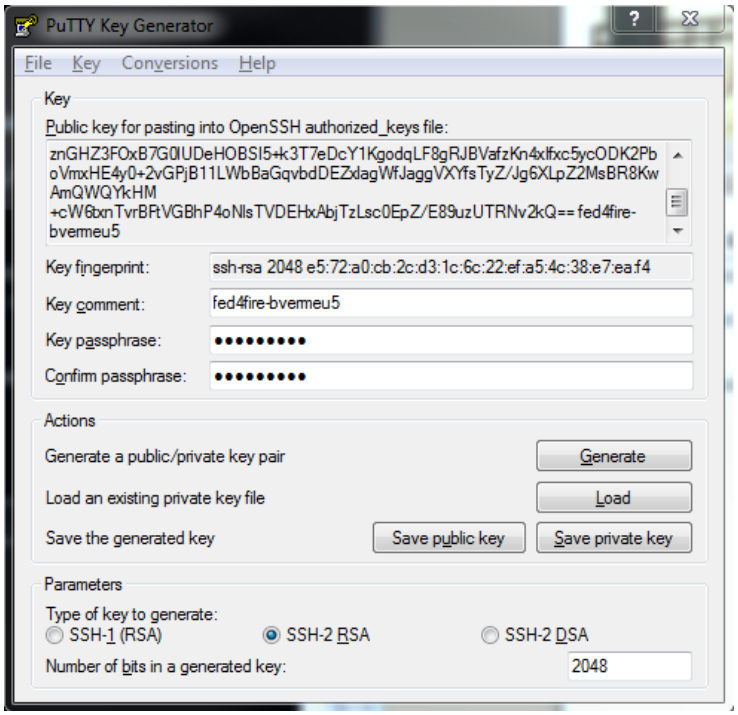


When it has enough random information, it will create your key.



Now you should change the `Key comment`, e.g. something as fed4fire-youraccount, and provide a passphrase on your SSH key. After that, click `Save private key` and save it to a file on your PC, e.g. `fed4fire_youraccount.ppk`. Copy also the public key on top (in the frame below `Public key for pasting into OpenSSH authorized_keys file`) and save it to a local file, e.g. `fed4fire_youraccount.pub`. This is NOT the same format as the button `Save public key` will generate !

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



UNIX platforms (Linux/Apple OS X)

For UNIX platforms, creating an SSH key can be done through a command line tool that you run in a terminal. Run the command `ssh-keygen -t rsa`. The default file locations are normally okay, so you can press enter to accept them. Provide a passphrase on the SSH key.

Afterwards you have a file with the public part of your key (e.g. `id_rsa.pub`) and a file with the private part of your key (e.g. `id_rsa`). The latter may never be distributed.

Output looks like:

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/bvermeul2/.ssh/id rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/bvermeul2/.ssh/id rsa.
Your public key has been saved in /home/bvermeul2/.ssh/id rsa.pub.
The key fingerprint is:
4b:da:d0:55:4c:1b:fd:14:e6:6f:dd:79:c9:14:26:b0
bvermeul2@node0.singlenodedeb64.bvermeul.wall1.ilabt.iminds.be
The key's randomart image is:
+--[ RSA 2048]----+
|           +=o =.|
|           .oo* o|
|          .E.  = |
|        . .    o B|
|       . S      +*|
|        = .     ..|
|       . o        |
|                  |
|                  |
+-----------------+
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix I: Run a First Experiment

Introduction

The remainder of this appendix describes how one can run an experiment in real life on the federation (using jFed). This appendix is intended as an illustration of the architectural concepts described in this deliverable, to make them as clear as possible. The actual content of this appendix can be situated somewhere between architectural descriptions and more detailed specifications. Therefore it is likely that this information (possibly in more detail) will also become part of WP5's specifications for cycle 2 in deliverable D5.2.

When you have a local Fed4FIRE certificate (see ***Get An Account and Certificate***) and created an SSH key (see ***Create SSH key***), we can go on with a first experiment to see if you can access testbed resources. For this we will use the jFed tool.

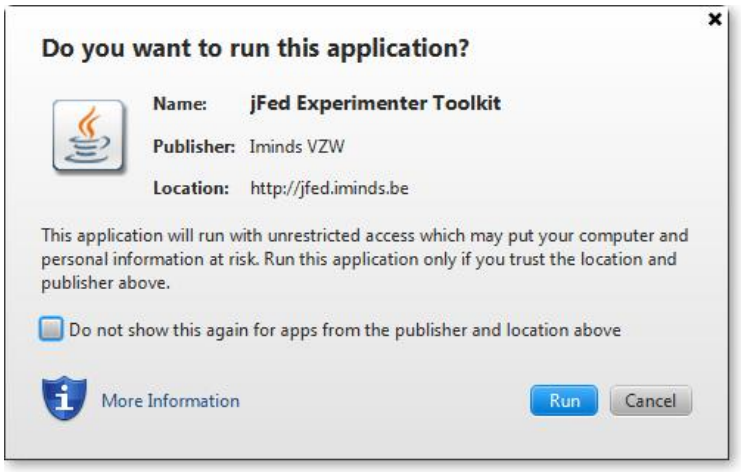
Start up jFed

You need to have a recent java 7 running. If not, please install from http://www.java.com.

Verify with http://www.java.com/verify that you run version 7.

Go to http://jfed.iminds.be and click the green button `Quickstart jFed experimenter tool`.

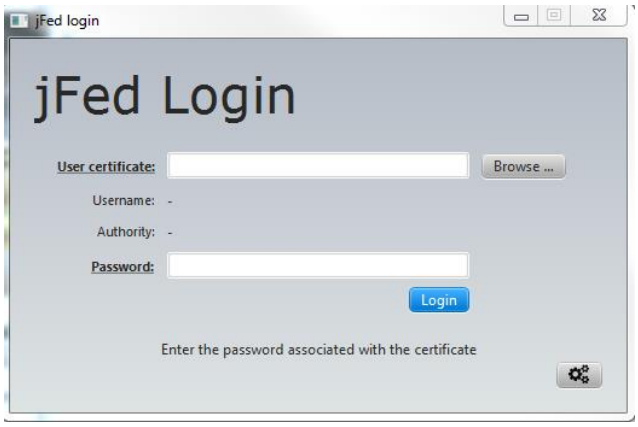
A dialog box will pop up saying that the Java application is signed by iMinds. Click `Run` and optionally you can tick the box 'Do not show this again'.



Run jFed for the first time

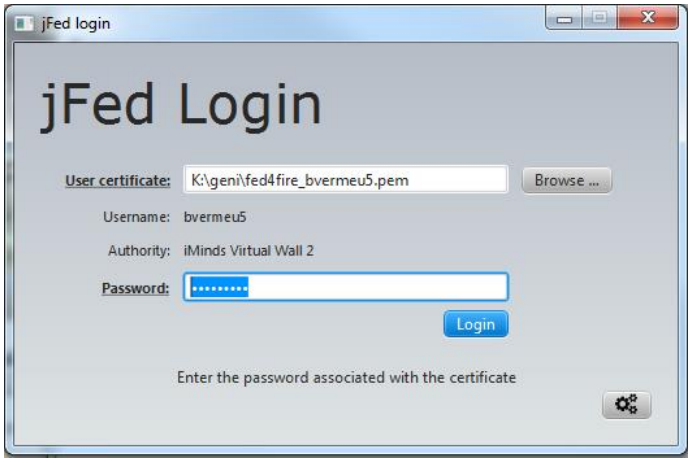
When jFed starts up for the first time, you will get the dialog box as shown below.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

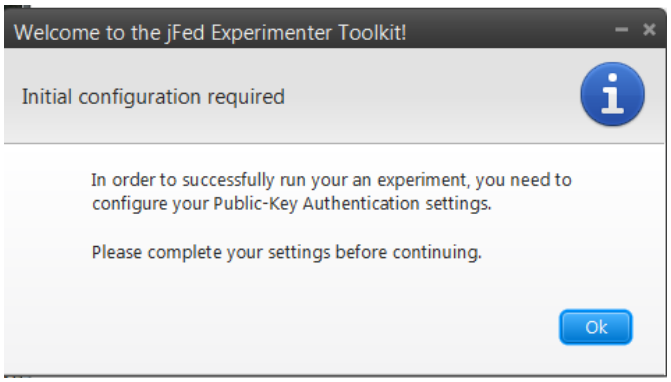


You should browse to the `pem` file that you downloaded from the authority portal or it can be pre-filled in (see ***Download your Fed4FIRE certificate***).

Now jFed should automatically show your `Username` and `Authority` as shown below.



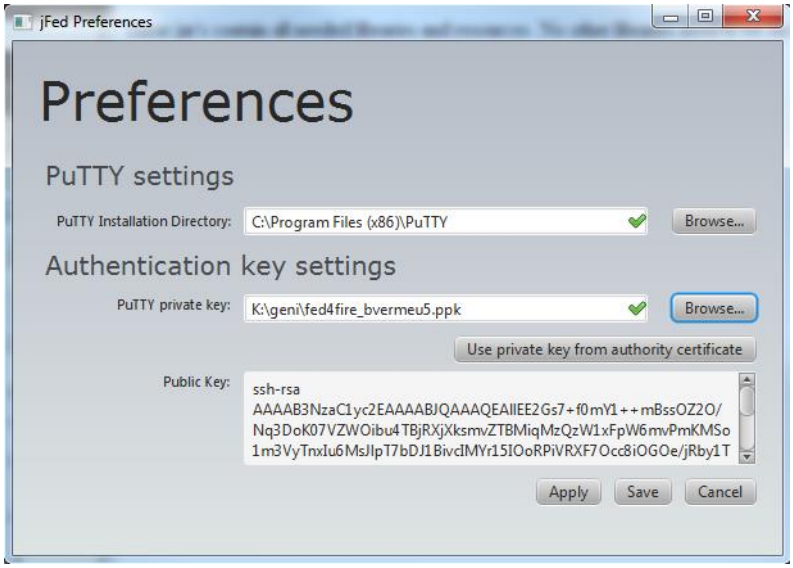
Type in your passphrase for this certificate and click `Login`. A dialog box will pop up to say that you have to configure jFed for this initial run.



For Windows:

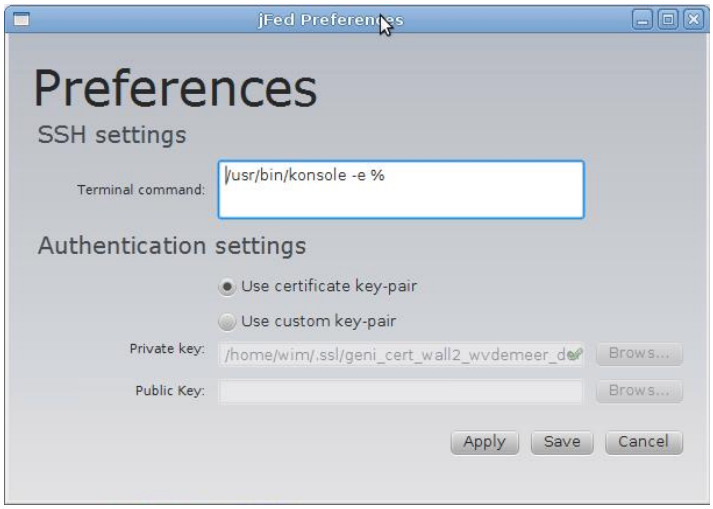
© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

In this preferences settings you should point jFed to the PuTTY installation directory (see ***Windows platform: PuTTY***). Secondly, you should point jFed to the ***private SSH key*** (`ppk` file) you saved on your PC. Automagically, jFed should show the 'Public key' below. Click `Save` at the bottom right to save these settings.



For Unix/Mac:

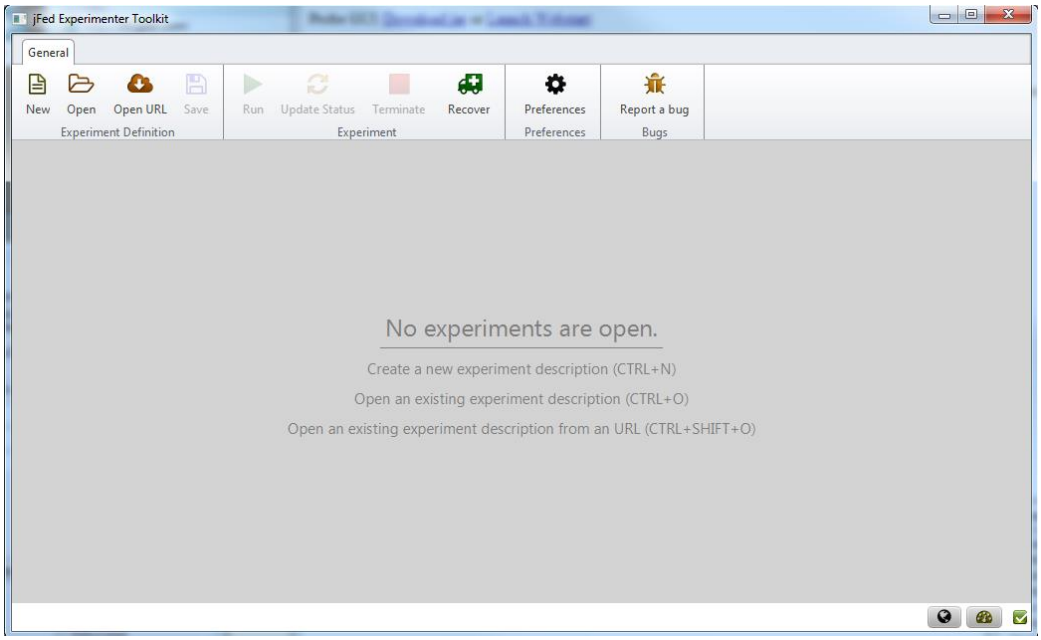
In this preferences settings, jFed should have a reasonable terminal configuration, so only change the default if it doesn't work when logging in. Secondly, you should click `Use custom key-pair` and point jFed to the ***private and public SSH key***you saved on your PC. Click `Save` at the bottom right to save these settings.



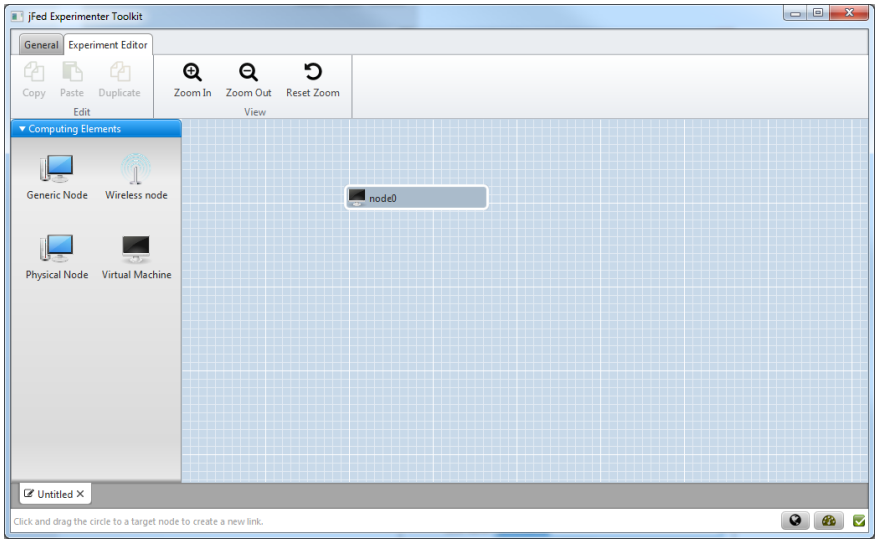
Create your first experiment

When you have logged in, and filled in your preferences, you see jFed with no experiments loaded.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



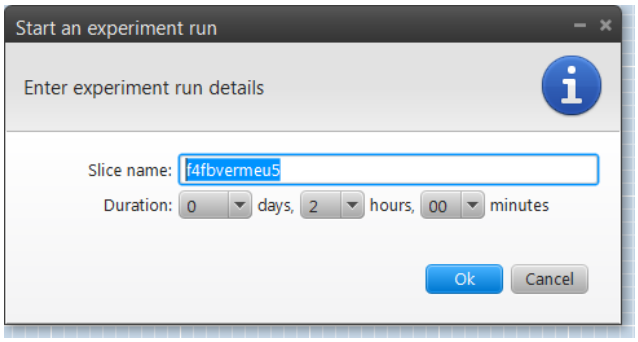
When you click `New`, you get a blank canvas where you can draw your experiment. Let's drag in a `Generic node` from the left side to the canvas. For more specific experiments you can right click and configure the node, but for now let it in the default settings.



Run the experiment

Let's run this experiment, by clicking the tab `General` at the top, and then the `Run` button. We will now have to choose a name for the experiment (= `slice name`) and choose a maximum duration.

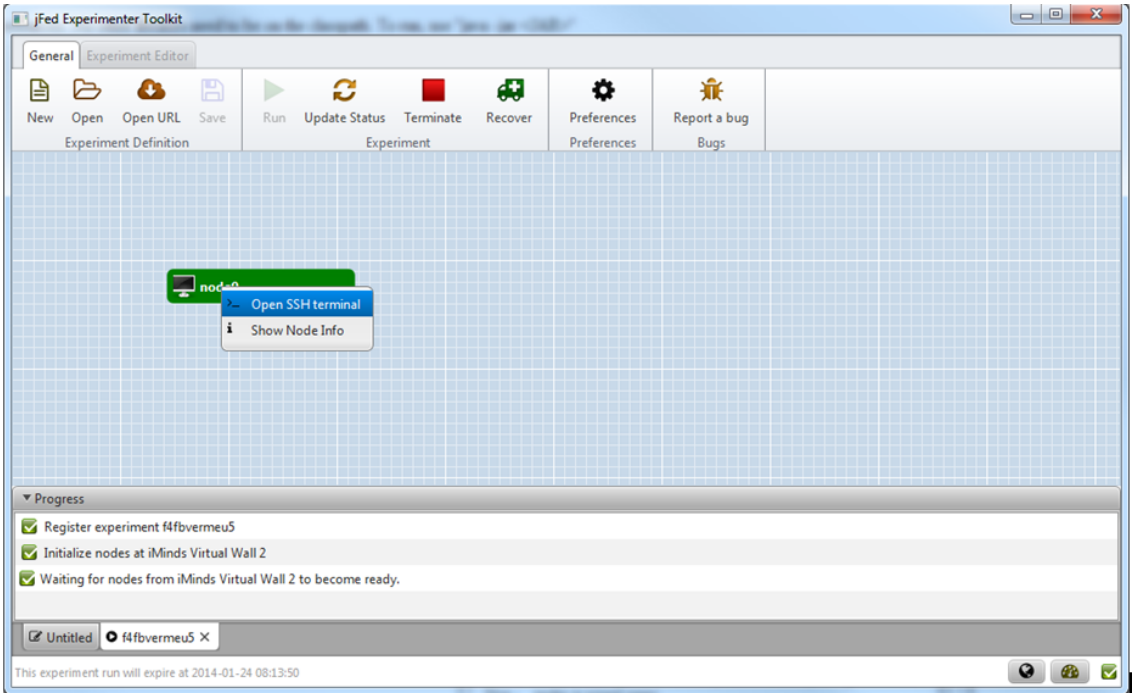
© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



It will now take a couple of minutes to get the node prepared

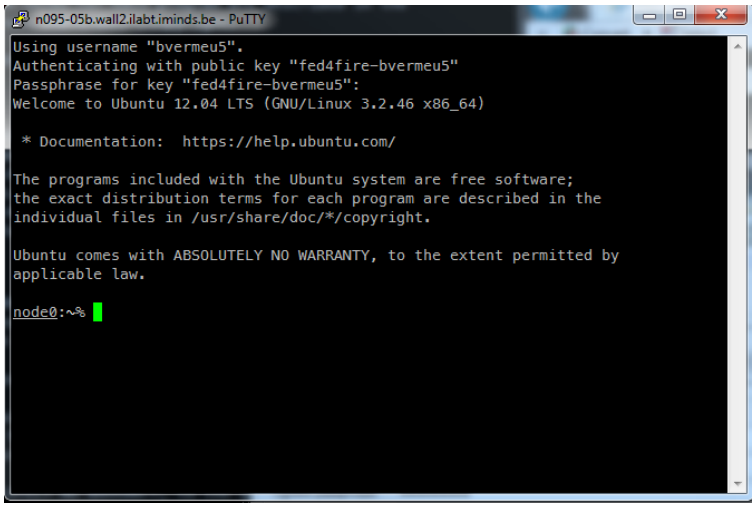
Login on a node of the experiment

When the node becomes green, we can right click on the node, and click `Open SSH terminal`.



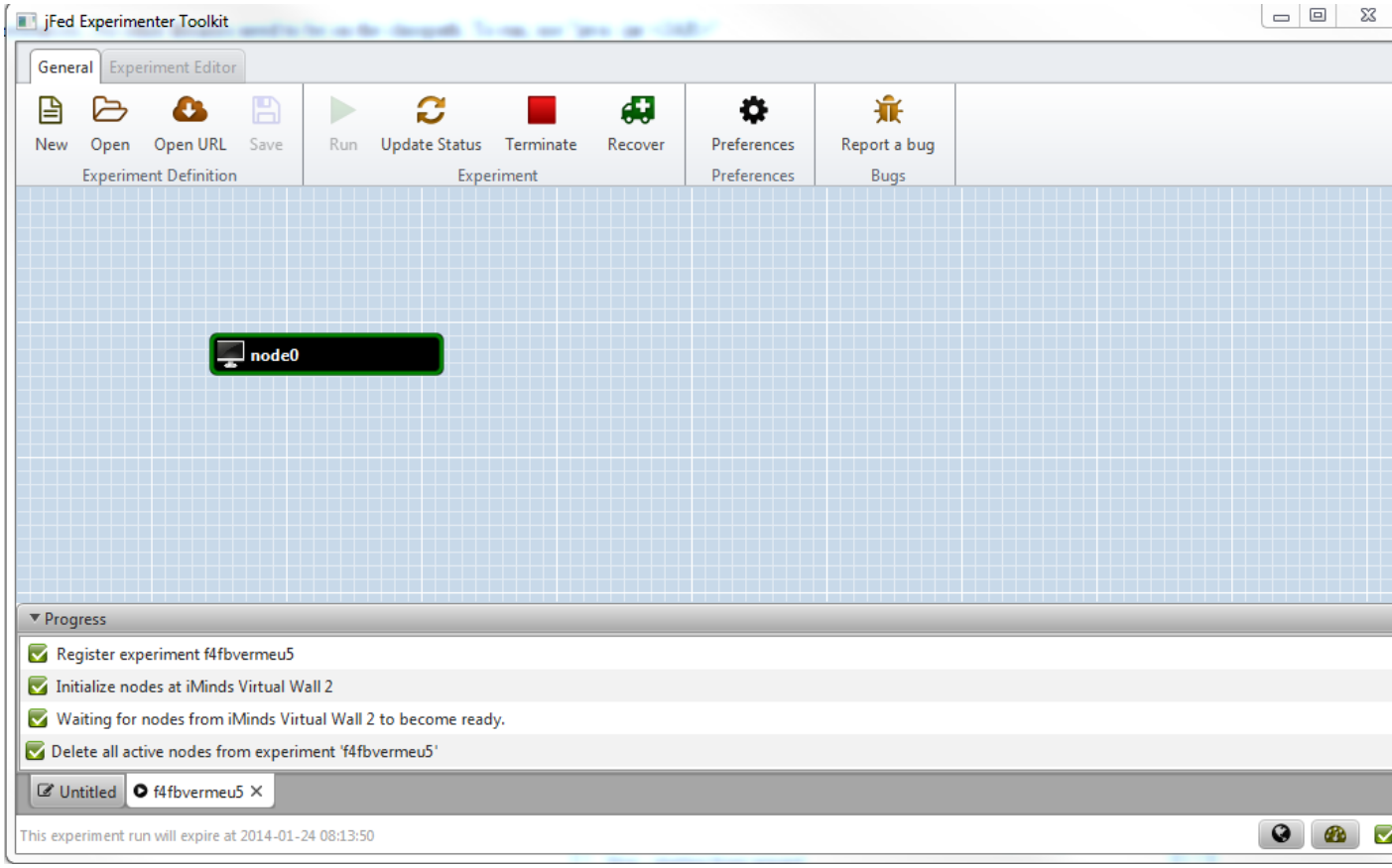
And then it will ask the passphrase on your ssh key. If the node says `Key refused` or another error it means something has gone wrong. See ***Note on connectivity***.

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

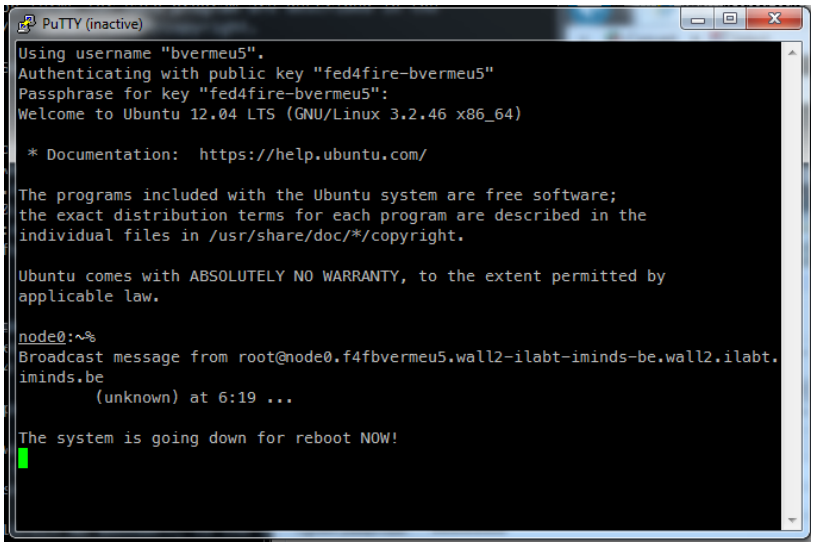


Ending the experiment

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.



© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



Note on connectivity

As in Europe public IPv4 addresses are scarce, we have the following problems for getting connected to the nodes:

- testbeds as Virtual Wall or w-iLab.t are only accessible through IPv6
- testbeds as BonFIRE have only a limited number of public IPv4 addresses, which is minimal in relation to the number of virtual machines they run.
- other testbeds run only on private IPv4 address and should be accessed through a gateway node.

We are currently working around this in several ways, but for this first testrun now:

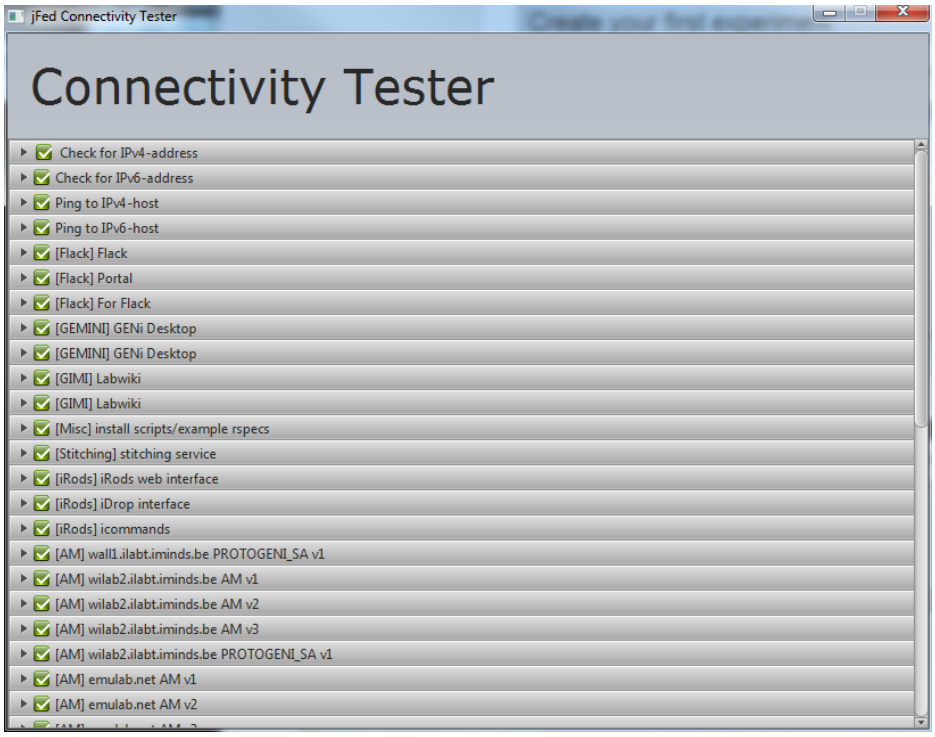
- the default node that was selected is at the Virtual Wall testbed (which is only accessible through IPv6).
- if you have IPv6 all will be okay and you will be able to login on the node.
- If the node becomes green, it means everything is allright with your account and certificate setup. You can stop the experiment now without login.
- if you don't have IPv6, register for an IPv6 address and tunnel, e.g. at Sixxs (choose AYIYA tunnel) and install Aiccu. This will take a couple of days to get approved, so `Terminate` this experiment with the button on top in jFed if you have one running.
- instead of a Generic node, you can draw a Virtual Machine on your canvas, and you will get a BonFIRE Virtual Machine with a public IPv4 address. for BonFIRE, you need an additional account (see ***BonFIRE***).

Test connectivity

You can test your connectivity with jFed by clicking the small button on the bottom:

And you will see a connectivity report:

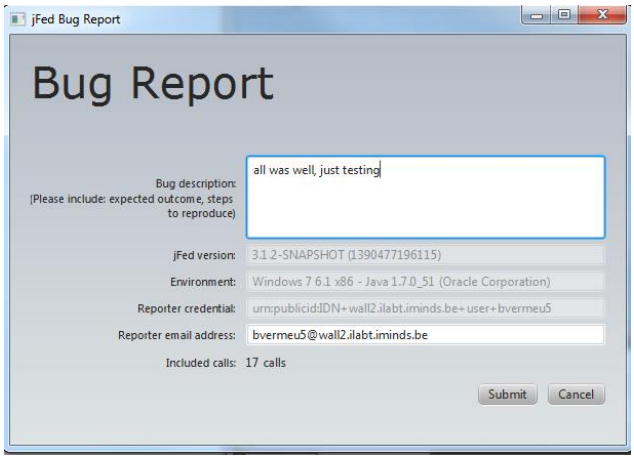
© Copyright iMinds and other members of the Fed4FIRE consortium, 2015



Bug report with jFed

In case you cannot get a green node on your canvas (e.g. at the bottom of jFed you see problems passing by), click the`Report a bug` button in jFed and fill in a bug report. This will send all relevant information on calls and connectivity to jFed staff, so they can investigate the problem and report back to you.

The reporter email address is standard filled in with your certificate email address and this is forwarded by the authority to your own email address, so you don't have to change this.



© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix J: Enable apache to authorize SSL certificates

Install apache on Ubuntu with SSL

```
apt-get install php5
apt-get install ssl-cert   (for self signed certificate on server)
a2ensite default-ssl
a2enmod ssl
service apache2 reload
```

you should be able to browse https://.... to your site now

Enabling certificate required login

(you can of course run this on a different port, different part of your website, etc, standard apache stuff)
pico -w /etc/apache2/sites-enabled/default-ssl

add:

```
                SSLRequireSSL
                SSLOptions +StdEnvVars +ExportCertData
```

in

```
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride None
                Order allow,deny
                allow from all
                SSLRequireSSL
                SSLOptions +StdEnvVars +ExportCertData
         </Directory>
```

at the end of /etc/apache2/sites-enabled/default-ssl, add

```
        SSLVerifyClient require
        SSLVerifyDepth  10

        # A bundle of trusted sites.
        SSLCACertificateFile /etc/apache2/genica.bundle
        # Another bundle of CRLs (certificate revocation lists)
        # SSLCARevocationFile /etc/apache2/genicrl.bundle
```

(just before </virtualhost>)

copy the content of http://users.atlantis.ugent.be/bvermeul/wall2.pem which is the root certificate of the iMinds authority.
in /etc/apache2/genica.bundle
(you can put other root certificates also there then)

service apache2 reload
now your browser should ask for a certificate, and you will see the same content.

Extract information from the certificate

If we now want to extract information from the certificate at server side, use e.g. this php:

```
<?php
$ssl=openssl_x509_parse($_SERVER["SSL_CLIENT_CERT"]);
// use this to see other stuff in the array
//print_r(array_values($ssl));
$start=strstr($ssl["extensions"]["subjectAltName"], "urn:publicid:");
$end=strpos($start, ",");
$urn=substr($start, 0,$end);
$end2=strpos($start, ",",$end+1);
$email=substr($start,$end,$end2-$end);
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

```
$end3=strpos($email,":");
$email=substr($email,$end3+1);
echo "<html><body>";
echo "User URN:".$urn."<br><br>";
echo "User email:".$email."<br><br>";
$urnExpl = explode("+", $urn);
$pid = str_replace('.','-', $urnExpl[1]);
if (strpos($pid,'ple:') !== false) {
    $pid='ple';
}
$user = $urnExpl[3];

echo "User: ".$user;
echo "</body>";
echo "</html>";
?>
```

and this will print:

```
User URN:urn:publicid:IDN+wall2.ilabt.iminds.be+user+bvermeul
User email:bvermeul@wall2.ilabt.iminds.be
User: bvermeul
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

Appendix K: How to provide facility monitoring for the First Level Support dashboard

See 3.4.3 for the architecture side.
The further part of this appendix will describe details on the OML part and aggregating the status of the internal testbed monitoring.

Extracting information from Zabbix and inserting into OML

This section shows an example on how to query a Zabbix server through the Zabbix API and insert the results into an OML database. This script can be run on any machine, as long as it has connectivity to both Zabbix and OML server. (it should be run by the testbed provider)

- Install guide :
  - Ruby packages: ruby1.9.1, ruby1.9.1-dev, rubygems
  - OML client library: liboml2 (detailed instructions at http://oml.mytestbed.net/projects/oml/wiki/Installation)
  - Rubygems: zabbixapi, oml4r (install with 'gem install <gemName>')
- Fed4Fire FLS OML Server details :
  - Version: OML 2.10 with PostgreSQL back-end
  - Connection: flsmonitor.ilabt.iminds.be port 3003
- The example script below(original template https://github.com/mytestbed/oml4r/blob/master/examples/oml4r-zabbix.rb) shows how to query the Zabbix API to check if the SSH service is running.
  - In the *oml_opts* hash, please change the *:domain* and *:nodeID* values to reflect your own testbed.
  - Change the *zabbix_opts* to connect to your own Zabbix server.
  - Define your own measurement point. An example measurement point is given to check the SSH service. Multiple measurement points can be defined (e.g. ICMP, HTTP, SSH).
  - Add more Zabbix queries if needed. Change the *:name* value to query different measurement data from the Zabbix server.
  - Send the Zabbix data to the monitoring server by calling *<mpName>.inject(<dbFields>).*
  - Add the host names as command line arguments and start the script (-i specifies the query interval :

ruby oml4r-zabbix.rb <hostname1 hostname2 …>

Aggregating the status to red/green/amber

In the aggr-status.rb script an example implementation (ruby) is given to calculate the aggregated testbed status. In this example script for the Virtual Wall, the script first checks if the three core servers of the Virtual Wall can be pinged, after which it will check if it is possible to SSH to the user server and verify if the webserver of the testbed is up and running.
The following states can be shown:

© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

- 0 (green): everything is OK, testbed is up and running.
- 1 (amber): some servers or switches cannot be reached, but part of the testbed may still be up.
- 2 (red): one or more of the core servers/switches is down, the testbed cannot be used.

The script updates the aggregate status on the FLS montoring database every minute and is run by iMinds at the FLS servers (one script per testbed which provides monitoring information over OML).

Oml4r-zabbix.rb for monitoring ssh: to be adapted and run by testbed provider

```
#!/usr/bin/env ruby
# example script for FLS Testbed monitoring with Zabbix
# make sure you install these gems

require "zabbixapi"
require "oml4r"
require "date"

# # Zabbix node names
# nodes = ["10.129.16.11", "10.129.16.12", "10.129.16.13"]

# Define your own Measurement Point
class SSH_MP < OML4R::MPBase
  name :ssh
  param :insert_time, :type => :string
  param :node, :type => :string
  param :up, :type => :double
  param :last_check, :type => :string
end

# Initialise the OML4R module for your application
oml_opts = {
  :appName => 'zabbix',
  :domain => 'iMinds',
  :nodeID => 'iMindsTestbeds',
  :collect => 'tcp:flsmonitor.ilabt.iminds.be:3003'
}

zabbix_opts = {
  :url => 'https://xxxxx/api_jsonrpc.php',
  :user => 'xxxxx',
  :password => 'xxxxx'
}

interval = 10

nodes = OML4R::init(ARGV, oml_opts) do |op|
  op.banner = "Usage: #{$0} [options] host1 host2 ...\n"
  op.on( '-i', '--interval SEC', "Query interval in seconds [#{interval}]" ) do |i|
    interval = i.to_i
  end
  op.on( '-s', '--service-url URL', "Zabbix service url [#{zabbix_opts[:url]}]" )
do |u|
    zabbix_opts[:url] = p
  end
  op.on( '-p', '--password PW', "Zabbix password [#{zabbix_opts[:password]}]" ) do
|p|
    zabbix_opts[:password] = p
  end

  op.on( '-u', '--user USER', "Zabbix user name [#{zabbix_opts[:user]}]" ) do |u|
    zabbix_opts[:user] = u
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

```
  end
end

if nodes.empty?
  OML4R.logger.error "Missing host list"
  OML4R::close()
  exit(-1)
end



# connect to Zabbix JSON API
zbx = ZabbixApi.connect(zabbix_opts)
# catch CTRL-C
exit_requested = false
Kernel.trap( "INT" ) { exit_requested = true }

# poll Zabbix API
while !exit_requested
  nodes.each{|n|
    results = zbx.query(
      :method => "item.get",
      :params => {

        :output => "extend",
        :host => "#{n}",
        # only interested in SSH
        :search => {
          :name => "SSH service is running"
        }
      }
    )

    unless results.empty?
      up = results[0]["lastvalue"]
      # injecting measurements into OML
      SSH_MP.inject(Time.now.to_s, n, up,
Time.at(results[0]["lastclock"].to_i).to_s)
    else
      OML4R.logger.warn "Empty result usually means misspelled host address"
    end
  }

  sleep interval
end


OML4R::close()
puts "Exiting"
```

Aggregation script: example for virtual wall, scipt run by iMinds based on information provided by testbed provider

```
#!/usr/bin/ruby
require "pg"
#open DB connections
flsmon = PGconn.open('dbname=flsmonitoring')
#query iMinds testbeds
iminds = PGconn.open('dbname=iMinds')

#wall3 - aggregated testbed info
res = iminds.exec("select node,up,max(last_check) as last_check from zabbix_icmp
where node='boss.wall3.test.ibbt.be' or node='ops.wall3.test.ibbt.be' \
                  or node='fs.wall3.test.ibbt.be' group by node,up ;")
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015

```
$up=res[0]['up'].to_i==1 && res[1]['up'].to_i==1 && res[2]['up'].to_i==1

#only if ping tests succeed -> test SSH and HTTP
if $up
  sshres = iminds.exec("select up from zabbix_ssh where
node='ops.wall3.test.ibbt.be' order by insert_time desc limit 1;")
  httpres = iminds.exec("select up from zabbix_http where
node='boss.wall3.test.ibbt.be' order by insert_time desc limit 1;")
  $up = $up && sshres[0]['up'].to_i && httpres[0]['up'].to_i
end

if $up==1 then $result=0 else $result=2 end
#if something failed, insert 2 (=testbed is not usable!)
update = flsmon.exec("update flstestbeds set aggregatetestbedstate=#{$result} ,
last_check='#{res[0]['last_check']}' where testbedid=1 ; ")
iminds.close()
flsmon.close()
```


© Copyright iMinds and other members of the Fed4FIRE consortium, 2015