Grant Agreement No.: 732638
Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

# D3.5: Requirements and specifications for the 3rd cycle

| | |
|---|---|
| Work package | WP 3 |
| Task | Task 3.2 – 3.5 |
| Due date | 31/10/2021 |
| Submission date | 17/03/2022 |
| Deliverable lead | Imec |
| Version | 4 |
| Authors | Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU), Albert (Yiu Quan) Su (SU), Pauline Gaudet Chardonnet (SU), Dimitris Dechouniotis (NTUA), Costas Papadakis (NTUA), Aris Dadoukis (CERTH), Donatos Stavropoulos (CERTH), Ana Juan Ferrer (ATOS), Roman Sosa Gonzalez (ATOS), Joaquin Iranzo Yuste (ATOS), Rowshan Jahan Sathi (TUB), Alex Willner (TUB), Lucas Nussbaum (Inria), David Margery (Inria), Cedric Crettaz (MI) |
| Reviewers | Peter Van Daele (imec), Maria Chiara Campodonico (Martel) |
| Abstract | This deliverable gives an overview of the requirements for the developments in WP3 of the third cycle. WP2 are normal operations developments (add testbeds, fix bugs, small features, etc). WP3 is focusing on larger new functionality. |
| Keywords | Requirements third cycle, new functionality |

# DOCUMENT REVISION HISTORY

| Version | Date | Description of change | List of contributor(s) |
|---|---|---|---|
| V1 | 1/10/2020 | TOC | Brecht Vermeulen (imec) |
| V2 | 1/02/2021 | First version | Brecht Vermeulen (imec), Wim Van der Meerssche (imec), Thijs Walcarius (imec), Radomir Klacza (SU), Albert (Yiu Quan) Su (SU), Pauline Gaudet Chardonnet (SU), Dimitris Dechouniotis (NTUA), Costas Papadakis (NTUA), Aris Dadoukis (CERTH), Donatos Stavropoulos (CERTH), Ana Juan Ferrer (ATOS), Roman Sosa Gonzalez (ATOS), Joaquin Iranzo Yuste (ATOS), Rowshan Jahan Sathi (TUB), Alex Willner (TUB), Lucas Nussbaum (Inria), David Margery (Inria), Cedric Crettaz (MI) |
| V3 | 15/03/2022 | Version for submission | Brecht Vermeulen (imec) |
| V4 | 14/03/2022 | Submitted version | Peter Van Daele (imec), Maria Chiara Campodonico (Martel) |

## DISCLAIMER

## COPYRIGHT NOTICE

## ACKNOWLEDGMENT

Co-funded by the European Union

Co-funded by the Swiss Confederation

This deliverable has been written in the context of a Horizon 2020 European research project, which is co-funded by the European Commission and the Swiss State Secretariat for Education, Research and Innovation. The opinions expressed and arguments employed do not engage the supporting parties.

| Project co-funded by the European Commission in the H2020 Programme | | |
|---|---|---|
| **Nature of the deliverable:** | **R** | |
| **Dissemination Level** | | |
| **PU** | Public, fully open, e.g. web | **X** |
| **CL** | Classified, information as referred to in Commission Decision 2001/844/EC | |
| **CO** | Confidential to FED4FIRE+ project and Commission Services | |

** R: Document, report (excluding the periodic and final reports)*

*DEM: Demonstrator, pilot, prototype, plan designs*

*DEC: Websites, patents filing, press & media actions, videos, etc.*

*OTHER: Software, technical diagram, etc.*

# EXECUTIVE SUMMARY

Fed4FIRE+ testbeds are in constant change and Fed4FIRE+ partners are regularly adapting their testbeds to the latest requirements. Moreover, the whole federation needs constant upgrading and this deliverable is the 3rd and final in a series of 3 deliverables describing the requirements and specifications for the testbeds. While originally this was intended to be a systematic set of 3 cycles, based on series of Open Calls experiments providing feedback and requirements for adaptations, this turned out to evolve in a continuous way.

This deliverable provides an overview of the requirements for the developments in WP3 during the period 2020-2021 of the Fed4FIRE+ project. All normal operations developments (adding testbeds, fix bugs, adding small features, etc) are part of Work package WP2 while Work package WP3 is focusing on adding larger new functionalities to the federation and its testbeds.

WP3 consists out of the following tasks, which are also the sequence of sections in this deliverable:

- Task 3.1 is focusing on SLA and reputation for testbed usage
- Task 3.2 is focusing on Experiment-as-a-Service (EaaS), data retention and reproducibility of experiments
- Task 3.3 is targeting Federation monitoring and interconnectivity
- Task 3.4 works on Service orchestration and brokering
- Task 3.5 researches ontologies for the federation of testbeds

## TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# 1 INTRODUCTION

Fed4FIRE+ testbeds are in constant change and Fed4FIRE+ partners are regularly adapting their testbeds to the latest requirements. Moreover, the whole federation needs constant upgrading and this deliverable is the 3rd and final in a series of 3 deliverables describing the requirements and specifications for the testbeds. While originally this was intended to be a systematic set of 3 cycles, based on series of Open Calls experiments providing feedback and requirements for adaptations, this turned out to evolve in a continuous way.

This deliverable provides an overview of the requirements for the developments in WP3 during the period 2020-2021 of the Fed4FIRE+ project. All normal operations developments (adding testbeds, fix bugs, adding small features, etc) are part of Work package WP2 while Work package WP3 is focusing on adding larger new functionalities to the federation and its testbeds.

WP3 consists out of the following tasks, which are also the sequence of sections in this deliverable:

- Task 3.1 is focusing on SLA and reputation for testbed usage
- Task 3.2 is focusing on Experiment-as-a-Service (EaaS), data retention and reproducibility of experiments
- Task 3.3 is targeting Federation monitoring and interconnectivity
- Task 3.4 works on Service orchestration and brokering
- Task 3.5 researches ontologies for the federation of testbeds

The sections of this document are linked to these tasks in the following order:

- Section 2 “SLA Reputation Service” is linked to the Task 3.1
- Section 3 “Reproducibility of Experiments” is linked to Task 3.2
- Section 4 “Central Broker” is linked to Task 3.4
- Section 5 “Resource Recommendation Service” is linked to Task 3.5
- Section 6 “Automated Open Stack Deployment” is linked to Task 3.5

# 2 SLA AND REPUTATION SERVICE

In the Fed4FIRE+ environment, the Service Level Agreement (SLA) and the Reputation Service provide the necessary tools and mechanisms for delivering to the users a quantitative view of the trustworthiness of the federated testbeds. This service facilitates the Fed4FIRE+ users to select the appropriate testbed in the federation according to their experiment's requirements.

The aim of adding SLA within Fed4FIRE+ is to enable testbed providers to create offerings that experimenters can accept establishing an agreement with the testbed owner. The agreement acts as a contract between the platform providers and the testbed users. Once the agreement has been created, its fulfillment must be verified. The information related to the execution of an experiment, i.e., if there is an agreement violation, will be sent to the other components using a notification / subscription pattern.

The Reputation Service of Fed4FIRE+ aims to enhance and extend the already-developed reputation service of Fed4Fire project. The updated service will leverage Quality of Service (QoS) metrics, such as Availability, Latency etc., Quality of Experience (QoE) metrics, e.g., Usability and Documentation Readability, and SLA data in order to compute the degree of confidence of both experimenters and testbed. At the end of an experiment, the users will be prompted to give their feedback for the reserved testbeds in order to update the reputation score of the testbed and the credibility score of the experimenter. This process mitigates the effect of abnormal or malicious evaluations and guarantees that the testbed's reputation score is fairly computed.

In the first cycle, the SLA and reputation services were developed. At the second cycle, these services were updated and are integrated with core federated testbeds. At the third slice, the SLA and reputation services will be integrated with the tools of federation, such as the Fed4FIRE Testbed Portal and the jFed tool for resource reservation. The benefits are twofold; a) the SLA service provides a real view of the performance of the utilized resources in an experiment based on agreed metrics, and b) the reputation service provides an objective view of the testbeds' performance that can facilitate future experimenters for the resource selection.

## 2.1 INTEGRATION OF THE SLA AQND REUTATION SERVICE WITH FEDERATION TOOLS

In the third-cycle, the already developed Reputation and SLA services will be integrated with the Fed4FIRE Testbed Portal (https://portal.fed4fire.eu/) in order to support various phases of the experiment lifecycle. The Reputation Service aims to provide a quantitative credibility measure on the provided resources/services of every testbed. The computation of the reputation score of each testbed is based on the performance of the utilized testbed's resources in the conducted experiments on the testbeds, as described in D3.02 and "Collaborative SLA and reputation-based trust management in cloud federations"[1]. As an indicative index, the reputation score facilitates the future experimenters to select the most appropriate testbeds and resources based on their features and their actual performance. For each testbed, this information will be available on the Fed4FIRE Testbed Portal. Furthermore, after the completion of an experiment, the user will be prompt to evaluate the prerformance of the involved testbed in order to update its reputation score. Figure 1 illustrates the workflow of the evaluation of an experiment and how the reputation score is recalculated. At the end of an experiment, an evaluation form is sent to the experimenter by the Portal. This form includes all the Quality of Service (QoS) and Quality of Experience (QoE) KPIs that must be evaluated by the experimenter. After the submission of the experimenter's response to the Portal, it is forwarded to the Reputation Service, more specifically to the Reputation Engine component, which is responsible for updating the reputation score of all involved testbs. Finally, the updated reputation scores are sent ot the Portal to be updated on the testbed description.

[1] Papadakis-Vlachopapadopoulos, K., González, R.S., Dimolitsas, I., Dechouniotis, D., Ferrer, A.J. and Papavassiliou, S., 2019. Collaborative SLA and reputation-based trust management in cloud federations. Future Generation Computer Systems, 100, pp.498-512.

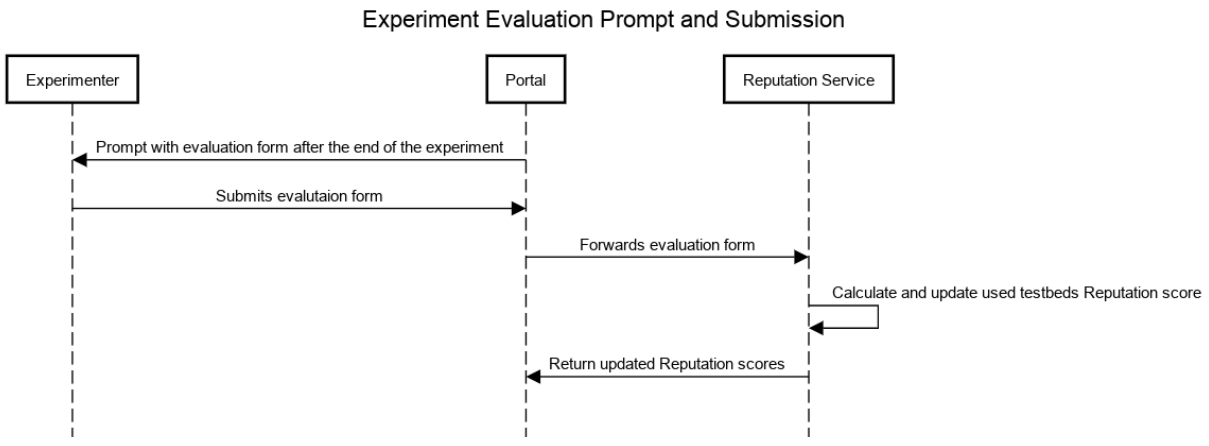
Experiment Evaluation Prompt and Submission
Experimenter
Portal
Reputation Service
Prompt with evaluation form after the end of the experiment
Submits evalutaion form
Forwards evaluation form
Calculate and update used testbeds Reputation score
Return updated Reputation scores

*Figure 1: Experiment Evaluation Prompt and Submission*

Furthermore, the objective of the complete SLA Service, implemented in the context of the Fed4FIRE+ project, has been to provide the capability to testbed providers to enforce QoS metrics of their infrastructure to experimenters.

The idea, as already described in previous version of the WP3 deliverables (in particular in the deliverable D3.03), has been that every time an experimenter initiates an experiment, a contract agreement, a Service Level Agreement (SLA), is established between the testbed provider and the experimenter – in general, stating the total availability of infrastructure resources during the total duration of the experiments.

The requirements for the development of this third cycle follow and consequently improve the requirements already presented on previous deliverables, mainly the deliverable D3.03 – “Requirements and specifications for the second cycle”.

During this 3rd project implementation cycle, we will be working around the improvement of the SLA service reliability as well as the capability to be agnostic to any kind of Portal used to implement the interaction infrastructure provider - experimenters.

In terms of software architecture, the SLA Service is composed of the following several components:

- SLA Core. Located in each of the testbeds, it is the main component of the SLA Service, and in charge of the management of entities and its evaluation.
- SLA Collector. It is located at federation level and it behaves as a façade to the SLA-Cores in the testbed and notifier.
- SLA Dashboard. It is the service that allows the federation users to inspect the status of their agreements. It is located at the federation level.

Some updates in these components have been developed since the release of D3.02.

The facade functionality of the SLA-Collector has been implemented. The SLA-Core of the testbeds must be registered on the Collector. The Collector exposes endpoints that are equal to the Core - adding a testbed-id parameter - so the Collector can forward the request to the appropriate testbed. The dashboard has been implemented, but it needs additional efforts to integrate it into the federation.

The sequence diagram below in Figure 2 shows the involved overall workflow with regards to the SLA service when a user reserves an experiment. The service, as already mentioned, is agnostic to the Portal.

It starts when a user decides to reserve infrastructure in one of the Fed4FIRE+ testbeds. According to the user’s needs, a number of testbeds can be available. Besides the infrastructure, the testbed offers guarantees for their resources (in principle, resources availability during the experiment, but additional metrics could be added). These guarantees are stored in a document called SLA Template, which can be viewed by the user.

When the user selects a testbed and reserves the experiment, an agreement between the user and the testbed is created automatically by the system. This agreement, depending on the guarantees, is evaluated during the experiment or when the experiment ends.

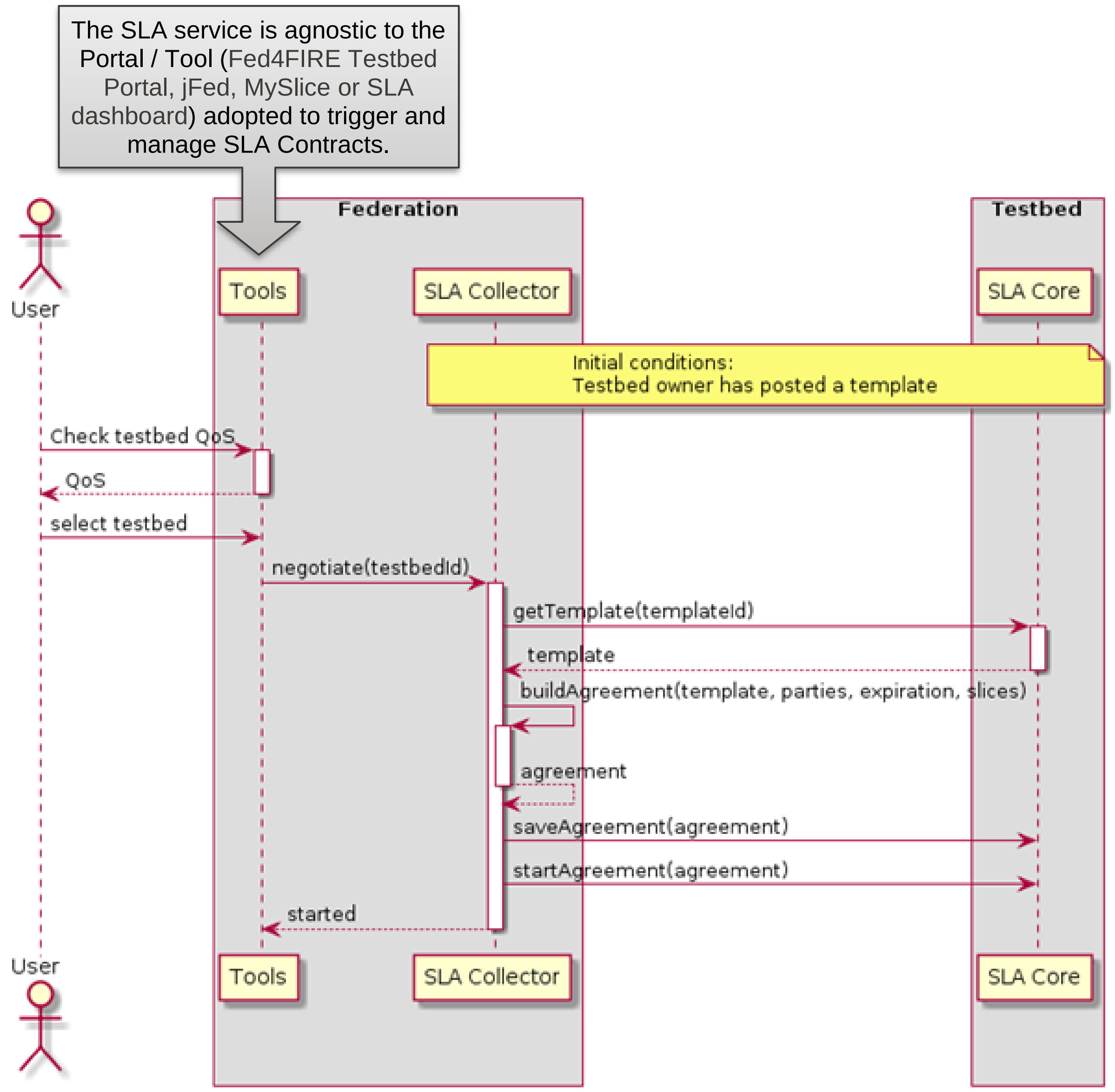
The SLA service is agnostic to the Portal / Tool (Fed4FIRE Testbed Portal, jFed, MySlice or SLA dashboard) adopted to trigger and manage SLA Contracts.
Federation
Testbed
User
Tools
SLA Collector
SLA Core
Initial conditions:
Testbed owner has posted a template
Check testbed QoS
QoS
select testbed
negotiate(testbedId)
getTemplate(templateId)
template
buildAgreement(template, parties, expiration, slices)
agreement
saveAgreement(agreement)
startAgreement(agreement)
started
User
Tools
SLA Collector
SLA Core

*Figure 2: Sequence diagram showing the involved overall workflow with regards to the SLA service when a user reserves an experiment.*

## 2.2 SLA AND REPUTATION SERVICE REQUIREMENTS

The activities to be developed during the 3rd cycle must fulfil the functional requirements listed below. The first group of requirements refer to the Reputation Service, while the second ones refer to the SLA service

These are the type of requirements that we envision (for this cycle only FUNC, ENV, DATA and USE requirements are defined):

| | | |
|---|---|---|
| **Functional** | Functional | FUNC |
| | Data | DATA |
| **Non-functional:** | Look and Feel Requirements | L&F |
| | Usability Requirements | USE |
| | Performance Requirements | PERF |
| | Operational - Environmental Requirements | ENV |
| | Maintainability and Support Requirements | SUP |

*Table 1: Overview of type of requirements*

### 2.2.1 SLA Functional Requirements:

The activities to be developed during the 3rd cycle must fulfil the functional requirements listed below.

The below tables summarize all service requirements to be fulfilled (implementing new code or improve development done during past project cycles).

| | |
|---|---|
| **ID** | SLA_01 |
| **Title** | **SLA solution must cover the whole lifecycle specified in WS-Agreement through interacting with the Fed4FIRE Testbed Portal (https://portal.fed4fire.eu/)** |
| **Short description** | The solution must cover the SLA lifecycle:<br>• Generation of WS-Agreement templates and agreements<br>• Provisioning of the agreements.<br>• Management of SLA related entities: templates, agreements, providers, violations and penalties<br>• Assessment of Service Level Objectives (SLOs) and generation of corresponding penalties when an SLO is violated<br>• Notification of detected violations and incurred penalties to the SLA Collector in order to handle it with the subscription service.<br>• Stop the agreements and their monitoring |
| **Additional Information:** | A platform owner through the Portal, but no other, must be able to create an offering. Technically this is implemented by creating an SLA template and including measurable terms. The information of the violations must be generated throughout an experiment lifetime. |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 2: SLA Functional Requirement – SLA_01*

| | |
|---|---|
| **ID** | SLA_02 |
| **Title** | **SLA solution REST interface** |
| **Short description** | The SLA solution needs to be agnostic, thus it will provide the same REST interface implemented during the 2nd implementation cycle in order to enable third-party applications to interact with it also through the Fed4FIRE Testbed Portal (https://portal.fed4fire.eu/). The third-party software must be able to retrieve the details about an offering, template or enforce (start the execution) of an agreement. The result of the execution of an agreement must be also available via the REST interface. The message format must be in XML or JSON. |
| **Additional Information:** | - |
| **Type** | FUNC / ENV |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 3: SLA Functional Requirement – SLA_02*

| | |
|---|---|
| **ID** | SLA_03 |
| **Title** | **SLA solution Subscription mechanism** |
| **Short description** | SLA must provide a subscription mechanism in order to allow third-party software to receive the information of the violations that are occurring in a specific agreement while the agreement is enforced or at the end of the agreement enforcement. The subscription mechanism must enable filtering the messages based on the content. |
| **Additional Information:** | - |
| **Type** | FUNC |
| **Priority Level** | Medium |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 4: SLA Functional Requirement – SLA_03*

| | |
|---|---|
| **ID** | SLA_04 |
| **Title** | **SLA solution multitenant** |
| **Short description** | SLA must support the recording of offerings and agreements from different organizations in such a way that the organizations cannot interfere with each other. |
| **Additional Information:** | As the architecture is not centralized, every testbed will have different data bases and data between them will not be shared. |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 5: SLA Functional Requirement – SLA_04*

| | |
|---|---|
| **ID** | SLA_05 |
| **Title** | **SLA Management Dashboard** |
| **Short description** | The SLA Solution must provide also an independent SLA Management Dashboard GUI to simplify the task of testbed providers of creating new offerings and to check the agreements that have been created based on its offerings. Detailed information associated to the offerings and agreements like the terms to be fulfilled or the violations that have occurred must be also identifiable with this GUI.<br><br>Moreover, experimenters will be able follow up the agreements created outside the dashboard, since the dashboard is not responsible to create new agreements. |
| **Additional Info:** | - |
| **Type** | USE |
| **Priority Level** | Medium |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 6: SLA Functional Requirement – SLA_05*

| | |
|---|---|
| **ID** | SLA_06 |
| **Title** | **Agreement creation and enactment** |
| **Short description** | The agreement creation will be done with jFed or MySlice as well as the Fed4FIRE Testbed Portal tools. It must be always based to an offering created by a testbed provider. This agreement must always include the terms that must be fulfilled, the expiration time and the id of the offering it is based on.<br><br>The SLA solution must allow creating an agreement between platform provider (testbed owner) and testbed client (experimenter) based on an offering |
| **Additional Information:** | The testbed client must be able to check the existing offering in order to find out if there is an interesting one. |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 7: SLA Functional Requirement – SLA_06*

| | |
|---|---|
| **ID** | SLA_07 |
| **Title** | **SLA terms quantizable** |
| **Short description** | SLA offerings and agreement will contain terms that must be guaranteed. These terms must be quantizable and comparable. |
| **Additional Information:** | This is a functional requirement needed by the SLA. The testbed providers must indicate terms that must be quantizable and comparable, otherwise the SLA solution won't be able to calculate if the terms are being fulfilled or not. |
| **Type** | FUNC |
| **Priority Level** | Medium |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 8: SLA Functional Requirement – SLA_07*

| | |
|---|---|
| **ID** | SLA_08 |
| **Title** | **SLA access to monitoring data** |
| **Short description** | SLA solution must have access to the monitoring data and it must be able to retrieve it using terms that specified by the testbed provider in the guarantee terms. Once an agreement has been created, it must be able to monitor it and calculate if violation occurs or not |
| **Additional Information:** | In order to calculate if a violation occurs or not, the SLA solution must access to the monitoring data. The agreement will contain different equation with terms and comparison that must be fulfilled (guarantee term). The monitoring data must be retrieved based on the names of the terms. The values that are retrieved must be comparable with the expression used in the equation to be fulfilled. |
| **Type** | DATA |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 9: SLA Functional Requirement – SLA_08*

| | |
|---|---|
| **ID** | SLA_09 |
| **Title** | **Distributed federation architecture.** |
| **Short description** | A decision taken in Fed4Fire is that the SLA solution has to be distributed, and this is a requirement that will remain in Fed4FIRE+. |
| **Additional Information:** | The SLA solution will allow the integration with other components in order to manage the agreements exposing interfaces. |
| **Type** | ENV |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 10: SLA Functional Requirement – SLA_09*

| | |
|---|---|
| **ID** | SLA_10 |
| **Title** | **SLA solution software dependencies** |
| **Short description** | The different SLA components can have different technologies and they must expose REST APIs to communicate each other. They have to be modular and decoupled. |
| **Additional Information:** | Technologies used Python and Java based on the results of Fed4Fire. |
| **Type** | ENV |
| **Priority Level** | High |
| **Identified by Partner(s)** | ATOS |
| **Status** | Work in progress |

*Table 11: SLA Functional Requirement – SLA_10*

| | |
|---|---|
| **ID** | SLA_11 |
| **Title** | **Include node information in violation** |
| **Short description** | A violation must contain the information about the node that made the violation occur. |
| **Additional Information:** | |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | NTUA |
| **Status** | Work in progress |

*Table 12: SLA Functional Requirement – SLA_11*

### 2.2.2 Reputation Functional Requirements:

| | |
|---|---|
| **ID** | PORTAL_REPUTATION_01 |
| **Title** | **Portal Interconnection with Reputation Service REST interface** |
| **Short description** | The Fed4Fire+ Portal will connect to the Reputation Service's REST API to retrieve reputation scores of all testbeds and submit new evaluations through the REST API after the completion of an experiment. |
| **Additional information** | The message format must be in JSON. |
| **Type** | FUNC / ENV |
| **Priority Level** | High |
| **Identified by Partner(s)** | Imec / NTUA |
| **Status** | Work in progress |

*Table 13: SLA Reputation Functional Requirement – PORTAL_REPUTATION_01*

| | |
|---|---|
| **ID** | PORTAL_REPUTATION_02 |
| **Title** | **Display Reputation Scores on Portal** |
| **Short description** | The Portal will present to experimenters through its GUI, the reputation score of each testbed in the experiment creation and testbed selection process. The reputation scores will be retrieved on demand from the Reputation Service's REST API. |
| **Additional information** | - |
| **Type** | FUNC / ENV / DATA |
| **Priority Level** | HIGH |
| **Identified by Partner(s)** | Imec / NTUA |
| **Status** | Work in progress |

*Table 14: SLA Reputation Functional Requirement – PORTAL_REPUTATION_02*

| | |
|---|---|
| **ID** | PORTAL_REPUTATION_03 |
| **Title** | **Experiment Evaluation Prompt** |
| **Short description** | After the completion of an experiment, the Portal will prompt the user to evaluate the Testbeds used with an appropriate rating form through the GUI. |
| **Additional information** | - |
| **Type** | FUNC / ENV |
| **Priority Level** | High |
| **Identified by Partner(s)** | Imec |
| **Status** | Work in Progress |

*Table 15: SLA Reputation Functional Requirement – PORTAL_REPUTATION_03*

| | |
|---|---|
| **ID** | PORTAL_REPUTATION_04 |
| **Title** | **Experiment Evaluation Submission** |
| **Short description** | When a user fills and submits the rating form described in PORTAL_REPUTATION_03 requirement, the Portal will submit it to the Reputation Service's REST API along with information for the Testbeds and resources used. The Reputation Service's API will return to the Portal the updated reputation values of the evaluated testbeds. |
| **Additional information** | |
| **Type** | FUNC / ENV / DATA |
| **Priority Level** | High |
| **Identified by Partner(s)** | Imec / NTUA |
| **Status** | Work in progress |

*Table 16: SLA Reputation Functional Requirement – PORTAL_REPUTATION_04*

# 3 REPRODUCIBILITY OF EXPERIMENTS

There is an increased demand for features enabling reproducibility of experiments. In previous cycles we developed the Experiment Specification (ESpec) as a new standard for setting up experiments. It combines various existing industry standards and leverages them to make it easier to fully setup an experiment: from requesting and provisioning the necessary testbed resources to installing software, doing the configuration management and the application deployment. This is also documented towards the users at https://jfed.ilabt.imec.be/espec .

This way, the ESpec can be used as a base for creating "Experiments-as-a-Service", where we provide experimenters with fully automated experiments that provide an excellent starting point for doing their scientific research or education activities, see e.g. automated openstack deployment.

This ESpec helps in provisioning an experiment. Nevertheless,it is not ideal/meant/usable for the experimentation orchestration itself, so in cycle 2 we also developed a lightweight tool ExpO:

(https://gitlab.ilabt.imec.be/ilabt/expo) for Experiment Orchestration.

For this cycle we took in following requests:

- On the Fed4FIRE+ testbeds it is possible to large scale networking experiments, but they are restricted to the physical limits (e.g. only up till 11 network cards on nodes), so it's not infinite scale. Mininet on the other hand is a tool for network simulation but is limited to a single machine. If we use the power of the testbeds with a lot of nodes and deploy a distributed version of mininet on this, that would scale to really huge networks. This tool will be called Distrinet and should allow huge networking experiments while also being sure that the emulation results can be trusted.
- More and more researchers are using jupyter notebooks to combine python code with notes (as a notebook in a classical lab session). GPULab already supported jupyter notebooks (https://doc.ilabt.imec.be/ilabt/jupyter/index.html) and now Grid'5000 also will support jupyter notebooks in cycle 3.
- For reproducing experiments, it is important to know all details about an experiment (e.g. hardware description of resources used in an experiment, software versions, monitoring data). It takes a lot of time to collect all this manually (without forgetting anything). A demand has raised to develop a tool ('experiment metadata bundler') for doing this.
- The jFed tool for experimenters has a GUI version and also a command line version (CLI). However, this CLI had a more complex workflow than the GUI (it was leaning more towards the API calls) and it does not yet support the use of ESpecs (enabling full automation through command line tools). In this cycle we envision jFed CLI 2 which has an easier workflow and supports ESpecs.

# 4 CENTRAL BROKER

In cycle 1 and 2, the Central Broker was developed as an overarching service that can be utilized by the experimenters to discover resources that span the federation and fulfil their experimentation requirements.

In cycle 2, we developed a testbed selector for the new Portal (https://portal.fed4fire.eu/explore/discover) which helps in finding the right testbed for an experimenter based on high-level properties of the testbeds.

In cycle 3, we are planning to add a combination of the testbed selector with the detailed resource overview of the central broker. Also, if experimenters are searching for specific resources (e.g. specific cpu, gpu, wireless card, …) they can find the testbed that serves those resources.

# 5 RESOURCE RECOMMENDATION SERVICE (RRS)

The resource recommendation service (RRS) will be offered to end users to help them select the most appropriate resources (across the available FED4FIRE+ testbeds) for performing an experiment, while exploiting its capabilities at full potential through the use of the federation as a whole (and not in a fragmented manner through the use of specific testbeds). Specifically, the development of the RRS aims to a) increase functionality of the federated testbeds, and b) improve the user experience, while also contributing to a) increasing visibility of the federation, and b) supporting a sustainable solution for the federation in particular beyond the project duration. The recommendation service will be offered prior to the resources' reservation process, aiming to overcome related entry-point barriers. This way, the users will be intelligently navigated through a user-friendly interface towards the examination and recommendation of the potential resources that can host their experiment, based on their needs, while significantly improve the end user experience - especially for newcomers - and reduce the learning curve for using the Fed4FIRE+ services.

Through the declaration of a set of requirements and preferences, the user will receive suggestions for reserving resources in the candidate testbeds that fulfil these requirements. Following, a set of filters will be designed and applied, where the user will be able to eliminate the suggestions for the suitable resources, upon the provision of requirements for their experiments (e.g., need for specific wireless nodes, need for IoT nodes, need for computational power). The recommendation service is going to be made available through a web portal. This portal can complement the existing "Getting Started" process in particular for the new experimenters/ or non-experts. The recommendation service is going also to be interlinked with the reputation service that provides reputation values for all the involved testbeds.

## 5.1 REQUIREMENTS FOR THE RESOURCE RECOMMENDATION SERVICE

The activities to be developed must fulfil the functional requirements listed below.

| | |
|---|---|
| **ID** | RRS_01 |
| **Title** | **Recommender System** |
| **Short description** | The solution must be able to provide recommendations for the testbed that can host the requested experiment, based on the provided requirements. |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 17: Resource Recommendation Service Requirement – RRS_01*

| | |
|---|---|
| **ID** | RRS_02 |
| **Title** | **Suggestion based on federated resources** |
| **Short description** | The solution must be able to suggest resources that may be provided by more than one testbed. |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 18: Resource Recommendation Service Requirement – RRS_02*

| | |
|---|---|
| **ID** | RRS_03 |
| **Title** | **Interoperability with the reputation service** |
| **Short description** | The solution must be able to consider the values provided through the reputation service towards the provision of recommendations. |
| **Type** | FUNC |
| **Priority Level** | High |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 19: Resource Recommendation Service Requirement – RRS_03*

| | |
|---|---|
| **ID** | RRS_04 |
| **Title** | **User Experience** |
| **Short description** | The solution must be easily adoptable through a web interface by the end users. Filtering and navigation functionalities must be straightforward. |
| **Type** | “Usability” |
| **Priority Level** | High |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 20: Resource Recommendation Service Requirement – RRS_04*

| | |
|---|---|
| **ID** | RRS_05 |
| **Title** | **User-friendly Installation** |
| **Short description** | Minimal and easy installation of the developed software. |
| **Type** | “Usability” |
| **Priority Level** | Mandatory |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 21: Resource Recommendation Service Requirement – RRS_05*

| | |
|---|---|
| **ID** | RRS_06 |
| **Title** | Comprehensive Platform and API Documentation |
| **Short description** | The semantic software and API documentation should be available, comprehensive and consistent with current functionality. |
| **Type** | “Usability” |
| **Priority Level** | Mandatory |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 22: Resource Recommendation Service Requirement – RRS_06*

| | |
|---|---|
| **ID** | RRS_07 |
| **Title** | **Reliable Service Infrastructure** |
| **Short description** | Ensure that the recommendation service is available at all times. |
| **Type** | “Usability” |
| **Priority Level** | Mandatory |
| **Identified by Partner(s)** | NTUA |
| **Status** | In progress. |

*Table 23: Resource Recommendation Service Requirement – RRS_07*

# 6 AUTOMATED OPENSTACK DEPLOYMENT

Following a high user demand for an easy set-up of Openstack environment, this requirement was added for the 2nd cycle. In cycle 2, we developed the necessary scripts and documentation (ESpec, https://doc.ilabt.imec.be/ilabt/virtualwall/tutorials/openstack.html) to deploy automatically a flexible Openstack environment in the Fed4FIRE+ field. This is based on the existing frameworks of as EnOS (http://beyondtheclouds.github.io/ ).

The feedback was positive . Nevertheless, after a while, it turned to be outdated because some component (ubuntu, openstack, EnOS, …) was updated and the whole scripting did not work anymore. Thus, there is the need to do automatic nightly testing in order to detect problems early on.

# 7 CONCLUSIONS

In this document, we described the requirements and updated requirements for the 3rd cycle of developments in WP3. These requirements are defined along the different tasks which make up the activities in WP3:

- Task 3.1 SLA and reputation for testbed usage
- Task 3.2 Experiment-as-a-Service (EaaS), data retention and reproducibility of experiments
- Task 3.3 Federation monitoring and interconnectivity
- Task 3.4 Service orchestration and brokering
- Task 3.5 Ontologies for the federation of testbeds

For the SLA and reputation service, (status) updates of the requirements identified in cycle 1 were given and new requirements were listed to integrate with the Fed4FIRE+ tools jFed and the Portal.

For the reproducibility of experiments, we had demand for multiple new tools such as a distributed version of mininet for large scale experimentation, a jFed CLI version that is easier to use and supports ESpecs and a tool for easily bundling all metadata of an experiment.

The opportunity we saw for the central broker is to integrate it with the testbed selector tool in order to facilitate the search of new experimenters for the right testbed through a hardware selection process.

Finally, a user demand for making the automatic setup of Openstack environments more robust was identified and will be added to the Federation Monitor as a nightly test.