EFPF: European Connected Factory Platform for Agile Manufacturing

European Factory Platform

WP7: Operational Management, Maintenance and Experiment Support

D7.1: Planning, Operational Management and Technical Support for EFPF Platform - I Vs: 1.0

Deliverable Lead and Editor: Mathias Axling, CNET

Contributing Partners: CNET, CERTH, FIT, LINKS

Date: 2020-06-30

Dissemination: Confidential

Status: <~~Draft~~ ¦ Consortium Approved ¦ ~~EU Approved~~>

Short Abstract

This document presents the operational viewpoint of the EFPF Platform for developers, pilot activities open call experimenters and support and maintenance teams. It primarily documents the status of operational procedures drafted during the first 18 months of the project. The final results will be presented in the second part of this deliverable at M48 of the EFPF project.

Grant Agreement: 825075

Document Status

Deliverable Lead	Mathias Axling, CNET
Internal Reviewer 1	María José Núñez, AID
Internal Reviewer 2	Dieter Meinhard, BRM
Type	Deliverable
Work Package	WPWP7: Operational Management, Maintenance and Experiment Support
ID	DD7.1: Planning, Operational Management and Technical Support for EFPF Platform - I
Due Date	2020-06-30
Delivery Date	2020-06-30
Status	<~~Draft ¦~~ Consortium Approved~~ ¦ EU Approved~~>

History

See Annex A**.**

Status

This deliverable is subject to final acceptance by the European Commission.

Further Information

www.efpf.org

Disclaimer

The views represented in this document only reflect the views of the authors and not the views of the European Union. The European Union is not liable for any use that may be made of the information contained in this document.

Furthermore, the information is provided “as is” and no guarantee or warranty is given that the information is fit for any particular purpose. The user of the information uses it at its sole risk and liability.

Project Partners:

Executive Summary

This deliverable presents the operational viewpoint of the EFPF architecture based on the work carried out in the following three tasks of WP7 in the EFPF project:

T7.1: Operational Management and Maintenance of EFPF Platform

T7.2: Experiment Operational and Technical Support

T7.3: Cross-organisational Operations Planning and Execution

The deliverable provides the operational viewpoint of managing the EFPF platform along with the related concerns and the actions and design decisions that are taken to ensure smooth operations and management of the platform.

The deliverable describes the technical and organizational design to provide operational and administrative support during different activities in the project, including development, piloting, open call experiments and use of the platform by non-funded experiments. A support organization has been defined which initially will be staffed by the technical partners. A process for management of support issues by that organization has also been defined. During the first 18 months of the project, appropriate tools have been selected for alerting, monitoring and managing the support issues. This deliverable also describes the initial plan and actions for operational and technical support for managing the lifecycles of experiments, which includes possible integration of the experiments into the EFPF platform and the transfer of the concepts and methods developed within the experiment to the EFPF project. Finally, cross-organizational planning and execution is described, where the current set of policies and protocols for accessing and using the EFPF platform and providing support across organizations is presented. This is a snapshot of the living policy and procedure document, which at the time writing this document contains procedures for registration process, exposing or use of applications or services, use of data sources, adding new marketplaces and adding or purchasing products and services.

Table of Contents

0 Introduction

0.1 EFPF Project Overview

EFPF – European Connected Factory Platform for Agile Manufacturing – is a project funded by the H2020 Framework Programme of the European Commission under Grant Agreement 825075 and conducted from January 2019 until December 2022. It engages 30 partners (Users, Technology Providers, Consultants and Research Institutes) from 11 countries with a total budget of circa 16M€. Further information: efpf.org

In order to foster the growth of a pan-European platform ecosystem that enables the transition from "analogue-first" mass production, to "digital twins" and lot-size-one manufacturing, the EFPF project will design, build and operate a federated digital manufacturing platform. The Platform will be bootstrapped by interlinking the four base platforms from FoF-11-2016 cluster funded by the European Commission, early on. This will set the foundation for the development of EFPF Data Spine and the associated toolsets to fully connect the existing platforms, toolsets and user communities of the 4 base platforms. The federated EFPF platform will also be offered to new users through a unified Portal with value-added features such as single sign-on (SSO), user access management functionalities to hide the complexity of dealing with different platform and solution providers.

0.2 Deliverable Purpose and Scope

The purpose of this document, "D7.1 Planning, Operational Management and Technical Support for EFPF Platform", is to document the operational viewpoint of the EFPF architecture and the day-to-day operational management as well as maintenance of the EFPF platform. In the revised deliverable plan, this document replaces "D7.1 Operational Management and Maintenance of EFPF Platform", "D7.3 Experiment Operational and Technical Support" and "D7.5 Cross-organisational Operations Planning and Execution". It describes the technical and organizational design to provide operation, administration, and support for the pilots and open call experiments. controlled, managed, and monitored. It also describes operational and technical support for experiments and cross-organisational operations planning and execution.

0.3 Target Audience

This document aims primarily at project participants that will be part of the support organization and manage experiments but also to some extent at external entities that will integrate with the EFPF platform and need to comply with policies and protocols. In addition, this deliverable provides the European Commission (including appointed Independent experts) with an overview of operational management and technical support for the EFPF platform.

0.4 Deliverable Context

This document is one of the cornerstones for achieving the project results. Its relationship to other documents is as follows:

- **D3.2: EFPF Data Spine Realisation - I:** Provides an overview of the Data Spine architecture, interfaces and data model interoperability.
- **D4.1: Smart Factory Solutions in the EFPF Ecosystem - I:** Provides an overview of the services and tools available in the EFPF platform.

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 2: The EFPF Platform:** An introduction to the EFPF platform architecture and the operational viewpoint.
- **Section 2: Concerns:** Defines the main concerns that govern the design of the operational viewpoint.
- **Section 3: Operational Management and Technical Support Implementation:** Presents the approach to address the concerns in the previous section with regards to the organization of and tool support for the support staff. This section is related to Task 7.1 Operational Management and Maintenance of EFPF Platform.
- **Section 4: Experiment Lifecycle Operational and Technical Support:** Describes the integration of experiments into the EFPF platform, the technical and operational support during the experimentation period and methods for transfer of concepts and methods developed within the experiment to the EFPF project. This section is related to Task 7.2 Experiment Operational and Technical Support.
- **Section 5: Economical and Business Aspects**
- **The** EFF is going to be the owner and main exploitation partner for the EFPF platform and thus will have influence on the decisions made on tools for operational management. As the main commercial entity with the EFPF platform and its product, the considerations of EFF are expected to be different from the EFPF project.

In the current phase of the project, EFF has been setup as an independent non-profit association, registered in Austria. The EFF is currently composed of 12 EFPF partner organisations. All EFF members have shown commitment towards the development of a sustainable business model around the federated EFPF platform. The commitment is solidified by the pledge to support EFF activities e.g. the transfer of 1 person-month of project funding towards EFF. The members of the EFF mostly come from business and industrial backgrounds and therefore they bring the necessary experience of running a business. This experience will be valuable in the initiation phase of the EFF when the operating procedures, business models and organisation procedures are being defined.

With the setup of EFF, the question regarding the ownership of the EFPF platform has been answered. Since the EFF is composed of project partners and the association itself will be part of the project (pending ongoing DoA amendment), it will have the vital say in shaping up the operations, support and maintenance procedures concerning the EFPF platform.

When the business plans, economic priorities, customer communication management and service agreements for the EFF are more detailed in M48, this section in the upcoming D7.2 (M48) will provide an overview of the economic and business aspects of operational management of the EFPF platform after the project.

1 Conclusions and Outlook

This document has presented the current status of the technical support organization, tool support, experiment lifecycle support and policies and procedures for cross-organizational use of the platform. This will provide the necessary foundation for providing technical management and operational support during the pilots and open call experiments.

These activities will also provide an opportunity to evaluate and enhance these results based on the feedback from stakeholders. Some already planned future refinement and extensions are mentioned in this document: a more elaborate support issue process, further organization of and assignment of responsibility for the documentation and extended tool support, e.g. the API Monitoring Tool. Iterative refinement of the results presented will continue during the project lifetime and the final version, which should take into special consideration the exploitation of the EFPF platform and the business aspects of operational management and technical support, will be presented in D7.2 in month 48.

- **:** This chapter presents the activities for cross-organizations planning and execution and the governing policies and procedures. This section is related to Task 7.3 Cross-organisational Operations Planning and Execution.
- **Section 6: Economical and Business Aspects:** This chapter introduces the owner of the EFPF platform and provides an outlook on the final operational management and technical support aspects that will be detailed in the next deliverable D7.2.
- **Section 7: Conclusions and Outlook**
- **Annexes**:
  - **Annex A**: Document History
  - **Annex B**: References

1.1 Document Status

This document is listed in the amended Description of Action as “confidential” since it provides information for project-internal usage only.

1.2 Document Dependencies

This document is part of an iteration of living deliverables. It is the first of two documents where this document describes the outline and first version of the operational viewpoint. Some sections have only provisional work and are placeholders for the final iteration. “D7.2 Planning, Operational Management and Technical Support for EFPF Platform - Final Report” at Month 48 will provide a complete description and completes the document.

1.3 Glossary and Abbreviations

A definition of common terms related to EFPF, as well as a list of abbreviations, is available in the supplementary and separate document “EFPF Glossary and Abbreviations”.

Further information can be found at https://www.efpf.org/glossary

1.4 External Annexes and Supporting Documents

Annexes and Supporting Documents:

- None

1.5 Reading Notes

- None

2 The EFPF Platform

2.1 Architecture – Functional View

The EFPF platform is ecosystem-centric, which is supported through a federation model of many distributed platforms, tools and components. The EFPF platform follows the micro-service architecture approach, enabling different functional modules to implement individual functionalities that can be composed based on specific user needs. To implement such an approach, all components in the EFPF platform are required to publish their open interfaces, preferably REST interfaces (in case of synchronous communication), to allow for the exchange of data and avoid the lag-time introduced by interconnection buses.

Figure 1 gives an overview of the high-level architecture of the EFPF platform, the Data Spine and the base and external platforms to be integrated into the EFPF ecosystem. It provides a formal split of key components of the EFPF federation and depicts the interaction between them. The Data Spine has a central role in the EFPF ecosystem and its internal composition and relationship with other components can be also seen in Figure 1. A majority of the EFPF components interact with the Data Spine.

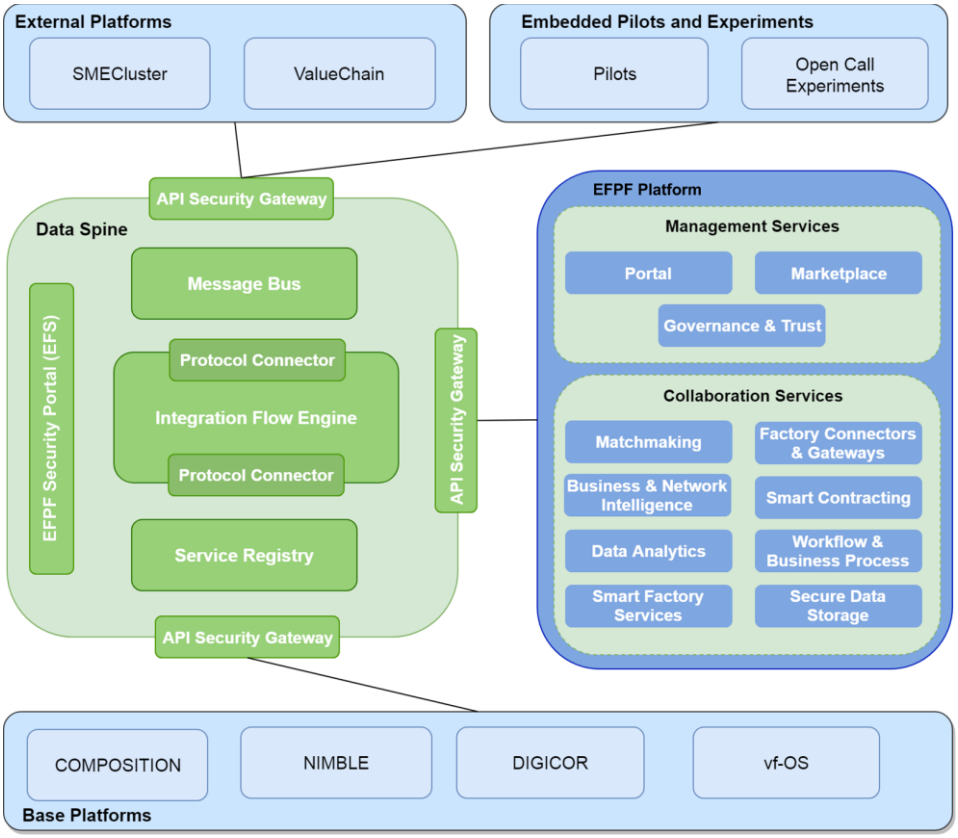

Figure 1: High-level Architecture of the EFPF Platform (from D3.1)

The main elements of the EFPF platform are:

- **Data Spine:** This is the central mechanism in the EFPF platform, which provides the interoperability infrastructure that initially interlinks between the four base platforms: COMPOSITION, DIGICOR, NIMBLE and vf-OS. It complies with the industry standards and follows the micro-services patterns to enable the creation of a modular platform. Therefore, it can be easily extended to 'plug-in' new external platforms and interlink them with the existing platforms, e.g. the SMECluster and iQluster

(ValueChain) platforms from EFPF partners. The Data Spine makes no distinction between the EFPF platform and the base platforms or any other platforms (external and third party) - the same services are available to all. Thus, the Data Spine is independent from the rest of the EFPF platform

- **EFPF Platform:** This is a digital platform that provides unified access to dispersed tools and services (IoT, digital manufacturing, data analytics, blockchain, distributed workflow, business intelligence, matchmaking, etc.) through a Web-based portal. The tools and services brought together in the EFPF platform are the market ready or reference implementations of the Smart Factory and Industry 4.0 tools brought to EFPF from the partners. The collection of enhanced versions of such tools and services from the base and/or external platforms deployed together as micro-services constitute the EFPF platform. The EFPF platform is comprised of the Management and Collaboration Services.
- **Base Platforms:** The four base platforms (COMPOSITION, DIGICOR, NIMBLE and vf-OS) in EFPF have been funded by the European Commission's Horizon 2020 program within the FoF-11-2016 (Digital Automation) topic. These base platforms are interlinked through the Data Spine.
- **External Platforms:** In addition to the four base platforms, the external platforms that joined the EFPF ecosystem at the beginning of the project are: ValueChain's iQluster[1] platform and SMECluster's Industreweb platform[2].
- **Pilots and Experiments:** Components and systems that will interact with the EFPF ecosystem through the EFPF platform, during the course of the project.

For the EFPF architecture, the study team refers to deliverable D3.1: EFPF Architecture-I.

2.2 Operational Viewpoint

This deliverable follows the same process and methodology for architectural description (AD) [Hil00] as outlined in deliverable D3.1, which is based on ISO/IEC/IEEE 42010:2011 "Systems and software engineering - Architecture description" [IEEE42010, 2011]. The viewpoint addressed is the Operational Viewpoint, as outlined in Rozanski and Woods [RW12].

The Operational Viewpoint addresses the operational concerns of the system's stakeholders and to identify solutions that address these. Examples of stakeholders are users, acquirers, system administrators, production engineers, developers, testers and communicators.

The major architectural concerns for this view are installation and upgrade, functional migration, data migration, operational monitoring and control, alerting, configuration management, performance monitoring, support, backup and restore and operation in third-party environments.

Models for the Operational Viewpoint are not as well-defined as for other architectural viewpoints [RW12]; descriptions in plain text are common. However, examples of models that can be used are installation models, migration models, configuration management models, administration models and support models. These models primarily describe the approach to address each concern, e.g. configuration groups and dependencies, rather than specific configuration values.

[1] https://valuechain.com/supply-chain-intelligence/iqluster
[2] https://www.industreweb.co.uk/

The Operational Viewpoint describes how operational concerns are managed; the specifics of e.g. the extent and response times of system support depends on contractual agreements.

3 Concerns

A concern [IEEE 42010, 2011] [RW12] about an architecture is a requirement, an objective, a constraint, an intention or an aspiration a stakeholder has for that architecture. The following sections list major architectural concerns of relevance to the operational view with a focus on the needs of the support staff, production engineers and system administrators [RW12] [HF10].

3.1 Stakeholders

The EFPF system has several stakeholders, whose interests and concerns are expressed in the documents governing the project, like the DOA. They can be categorized in groups for which the stakeholder classification from [ROZ, 2012] is used in the following sections. Stakeholders that do not participate in the technical development and maintenance of EFPF are:

- **Acquirers** that oversee the procurement of the system.
- **Users** that make use of the system and participate in defining the system functionality.

These stakeholder groups are the pilot partners, open call participants and future users of the system, whose concerns are mainly expressed in the requirements and use cases. D2.3 - Requirements of Embedded Pilot Scenarios captures the requirements and priorities of the pilot partners.

The other stakeholder groups are comprised of the technical partners in the project, commercial- and research-oriented. In the project phase of EFPF, these groups consist of the same people and maintainers will have a very small role. However, this is expected to change in exploitation and for clarity all of the will be used in this document.

- **Developers** design and deploy the system (EFPF) from specifications.
- **Production Engineers** design, deploy, and manage the hardware and software environments in which the system will be built, tested, and run.
- **Maintainers** manage the evolution of the system once it is operational.
- **System Administrators** run the system once it has been deployed.
- **Support Staff** provide support to users for the product or system when it is running.
- **Testers** test the system to ensure that it is suitable for use (for acceptance tests, pilot partners could be part of this group).
- **Communicators** explain the system to other stakeholders via its documentation and training materials.

It should be pointed out that since EFPF is a software platform, the User group will have developers from client organizations that use EFPF to build services and integrate software. The Developer group refers to developers of the EFPF platform only.

3.2 Support

The DOA states: “A Gitlab-based ticketing system will be setup to enable issue-reporting by the developers and users of the platform. The dedicated technical support team [support staff] will manage the ticketing system to provide instant support and carryout maintenance tasks.”. For operational management and maintenance of the EFPF platform, we should specify:

- The classes of stakeholders that will need support, and the type and level of support that should be provided.
- The classes and levels of incidents, which can be used to prioritize and assign support work.
- The support providers and the organization and responsibilities of different support providers. The processes and organization of the operational staff should be defined, and the operational staff must be committed.
- The escalation process, the path that a support issue should take in the support organization and when to which group of support providers an issue should be re-assigned when escalated. Some issues need to be handled by system administrators and others by production engineers or developers.
- The channels through which support should be provided.

3.3 Installation and Upgrade

An installation model should be developed for the platform that describes:

- Installation groups of components that should and can be installed together.
- Dependencies between different parts of the platform that introduce constraints on the installation process, e.g. require a specific ordering of installation.
- Constraints on e.g. data or service availability that require specific ordering of system start-up, running of custom tools, or restart of components.
- A backout plan that specifies the processes required to restore the system to its original or earlier state in case a deployment fails or must be removed because of defects. This should be available for both, the EFPF platform and connected systems.

In the early phases, during development and open call experiments, complete installation, migration and backup plans will still be in development as they still require inputs and experience from the developers and administrators of the platform.

3.4 Functional and Data Migration

Migration planning is not an immediate concern for the EFPF platform itself, but data and functional migration for other systems that join the EFPF ecosystem is a factor for acceptance and marketability of the platform. To new stakeholders joining the platform, it must be clear how to migrate users, data and function of existing systems to EFPF.

The pilots and open call experiments are not aimed at replacing existing systems. Functional and data migration are not critical concerns for this phase of EFPF. However, addressing these, and capturing experiences from the experiments, will be important for the operation of EFPF in the exploitation phase.

3.5 Configuration Management

If the system requires regular re-configuration, making coordinated changes to configuration parameters of a complex system, a configuration model may be needed. It involves describing groups of components that may be configured together, dependencies, configuration sets (e.g. different performance configurations) and production environment constraints. For less regular configuration, the deployment documentation may be sufficient. A dependency graph and deployment details are available in D6.1.

3.6 Operational Monitoring and Control

To provide the desired level of service to system stakeholders, dedicated monitoring and control of the system is necessary to ensure correct operation, and in case of system start-up, shutdown and similar tasks. Furthermore, it is likely that a set of maintenance procedures are needed after a time of stable deployment. This routine monitoring may be automated and handed over to an alerting system, and the extent of these tasks must be balanced to other resource demands. In any case, the necessary management tools, e.g. logging, must be available to support the work of the operational management and maintenance support staff and system administrators.

3.7 Alerting

Alerting is automatic and active monitoring and notification to the support organization of an unexpected event, typically some kind of failure. Alerts may be from infrastructure level, e.g. failing network connections or servers; system level, e.g. a failing component; or business level, e.g. invalid data entering a systems integration process. Depending on the hosting environment and whether the system is built on “bare metal” infrastructure, Infrastructure as a Service (IaaS), or Platform as a Service (PaaS), these alerts may already be provided. To set up alerting and filter relevant alerts, related information is needed from Users and/or Communicators on likely error conditions, procedures, as well as key performance and health indicators.

4 Operational Management and Technical Support Implementation

4.1 Overview

This section is related to Task 7.1 Operational Management and Maintenance of EFPF Platform, planning the day-to-day operational management and maintenance of the EFPF platform. The initial organization of and tools for support, installation and configuration groups, escalation process, support channels, operational monitoring and alerting are described.

4.2 Decisions to Address the Concerns

To resolve the concerns in section 3, a number of architectural decisions have been taken. Some of these relate to the Development Viewpoint. It was decided to use a common unit of deployment that enables automatic and scripted deployment, without manual configuration. Docker images will be used as unit of deployment and the running components will be Docker containers, that can be managed through a common tool and provide monitoring data, alerts and logs in a common way. Resource usage can be configured and monitored through the same tool. Deployment, redeployment, stopping and starting of components (docker containers) and to some extent also backout procedures can be performed in a uniform way.

A number of enabling tools compatible with Docker have been selected, as described in the sections below. The support team and system administrators will have a consistent way to manage the different parts of the EFPF platform and a limited number of tools that will be used. This addresses some of the concerns regarding installation and upgrade, operational monitoring and control and configuration management. Last in this section, the process and organization of the technical support team and strategies for refining the migration, installation and configuration models are outlined.

4.2.1 Installation and Configuration Groups

The installation and configuration groups will initially be identical. The Data Spine, EFPF Management Services and Secure Data Storage (framed in red in Figure 1Figure 2) have been specified in the Deployment Viewpoint and WP6 as “core components” of the EFPF platform which have common hosting, deployment and operational requirements.

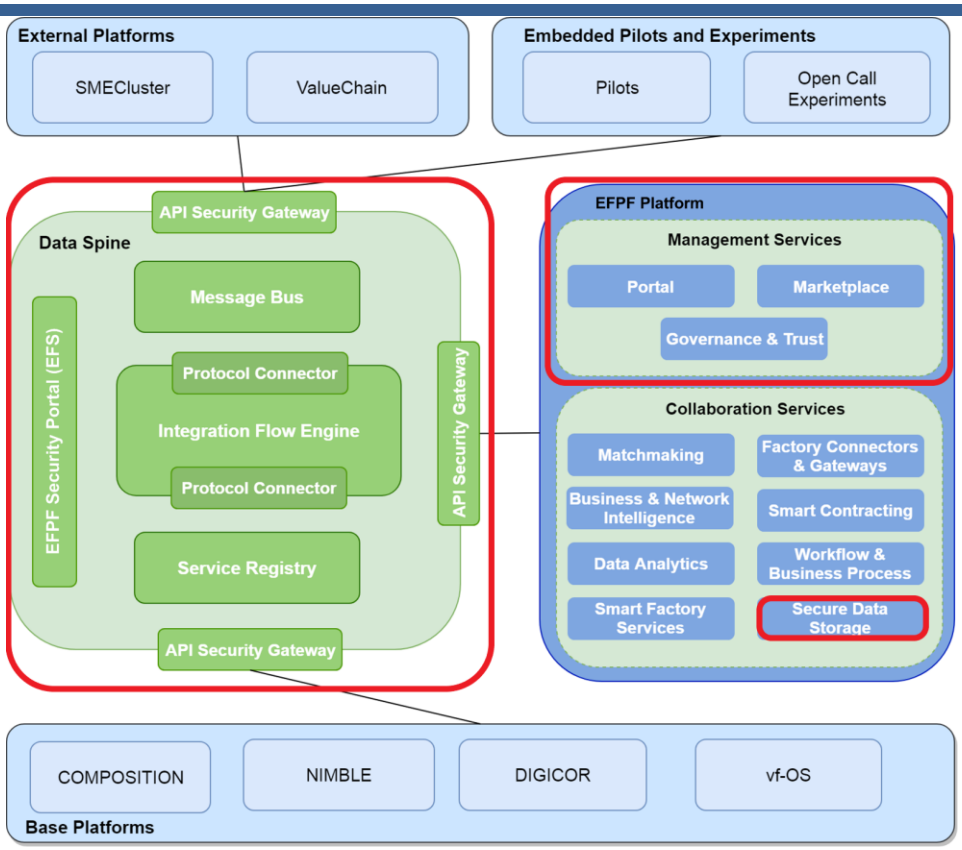

Figure 2: EFPF installation and configuration groups

Installation and configuration groups are based on this deployment organization:

- The Data Spine is installed and configured as one group
- The Management Services, the Portal and Marketplace components and the Secure Data Storage are installed and configured as one group
- The rest of Collaboration Services and Base platforms are individually installed and configured by the responsible partner (defined in D6.1 and D4.1)

4.3 Technical Support Team Organization

The support staff will be organized in two tiers: a first-line support group, and specialist groups organized by area of competence and installation and configuration groups (see Figure 3).

The chosen support issue system (ticketing system) must allow organization for support groups and different deployments, and for events restricted in time, e.g. experiments, pilots and large integration projects. These should relate to a specific deployment of the platform.

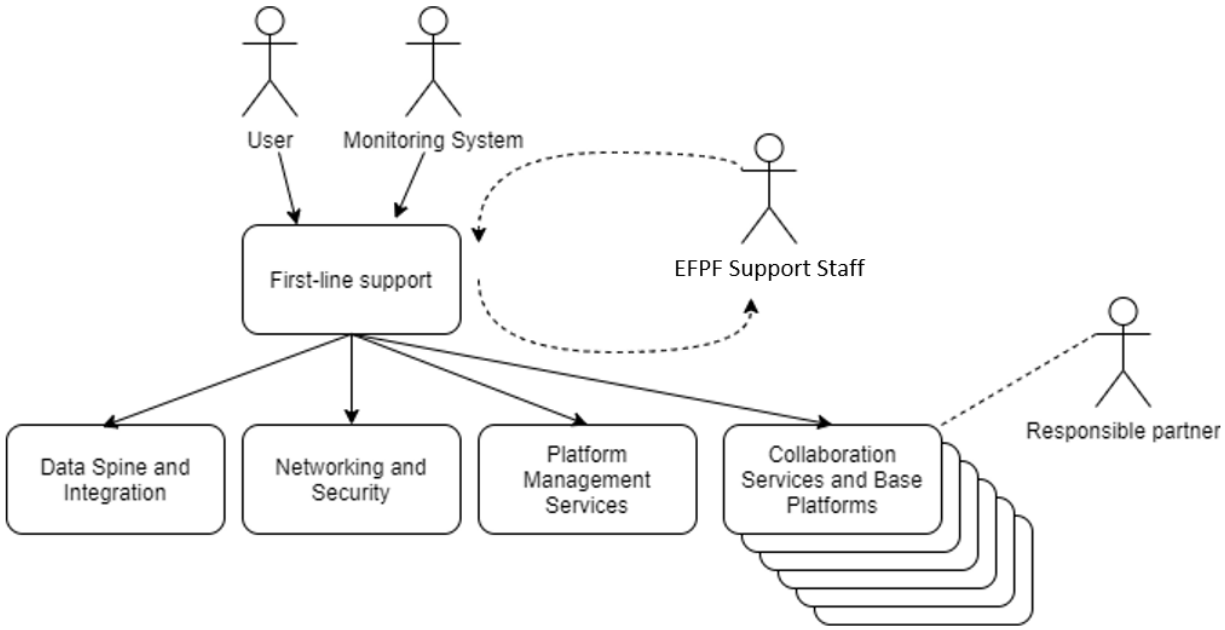

Figure 3: Technical Support Team Organization

4.3.1 Support to Clients

The clients of the support organization (classes of stakeholders that will need support) are EFPF technical and pilot partners, EFF members using the EFPF platform as well as users external to the EFPF project: open call participants and future commercial and research customers. Access to the ticketing system tool is needed for the support staff, but also for users who will be reporters of issues. The reporters typically do not have the need to login and do not have full access to the system but require ways to submit issues, e.g. using email or a web form.

4.3.2 Support Process

The figure below schematically shows the planned EFPF-related support process.

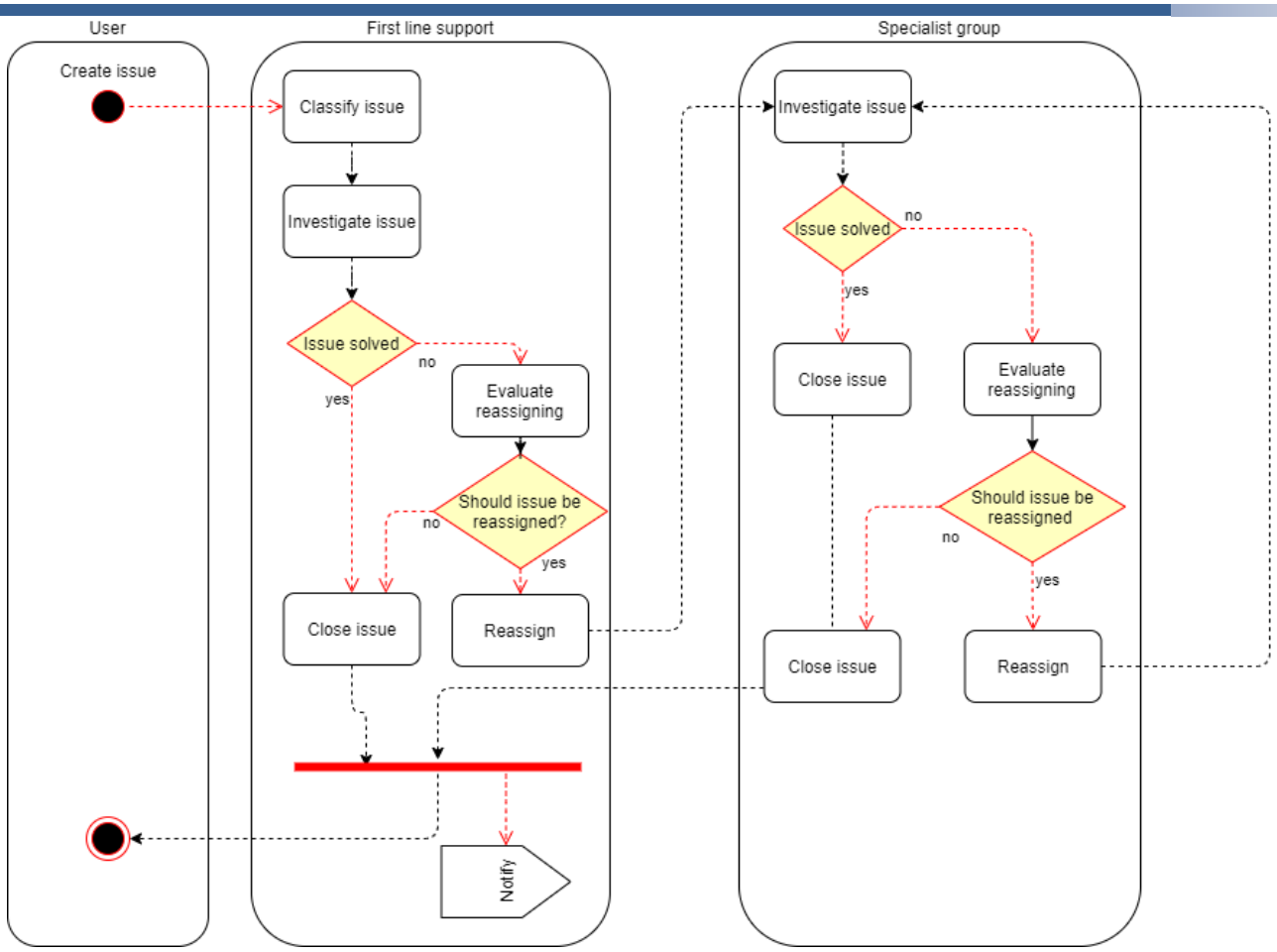

Figure 4: Support Process

A support issue (ticket) is entered into the ticketing system through a web form, by email or automatically through an alert. The exact technical capabilities depend on the chosen ticketing system, which at the time of writing is still being evaluated. If contact information is supplied or the reporter has a login in the ticketing system, the reporter of the support issue can be informed that the ticket is being handled. The issue is addressed by the first line support and classified by level of severity (in case of alerts it may already be classified). If the issue can be resolved by the first line support, it is closed, and a message is sent to the reporter. If the first line support cannot resolve the issue, it is escalated to the appropriate specialist group. The issue may be re-assigned between specialist groups.

If the issue was solved using documentation, a reference to this information should be included in the issue so that this is available if the problem reoccurs. If the documentation lacks the necessary information to resolve the issue, this should be noted in the documentation.

The description above summarises the initial, basic process. Depending on what the selected tool offers and what is found useful by the support staff, many different states may be used for the issue, and it could be re-opened, differentiate between closed and resolved, etc.

4.3.2.1 First-Line Support and Issue Prioritization

The responsibility of responding to support issues is shared between project partners on a rotating schedule. The responsibility of first-line support is to investigate and classify the

issue reported, and if it can be resolved by simple means such as restarting a container, to solve it. Issues and the related problems are classified by level of severity, from problems that need to be solved immediately to ones that can be put in the backlog or resolved immediately (will not be implemented). The initial severity classification (s) is the following:

- Highly severe - Issue Preventing Operation: An issue preventing operation exists when the use of the service / system is impossible or only possible under severe restrictions.
- Severe - Issue Interrupting Operation: An issue interrupting operation exists if the use of the system is not impossible, but the services or the functionalities are impaired in such a way that the use is only possible with clear restrictions. A malfunction is also present if slight errors lead to a not inconsiderable restriction of the use of the overall system.
- Minor Issue - A minor issue exists when the system can be used without or with insignificant restrictions.
- Insignificant – Issue considered neglectable, resolved immediately.

Issue prioritization should be based on a risk analysis. Following the eFMEA [BLI04] risk assessment method, a ranking of issue severity (s) and its effects, are combined with an estimated probability of occurrence (o) and detectability (d). These factors are then used establish an overall issue prioritization. Issues requiring deeper knowledge, will be delegated to one of the specialist teams.

4.3.2.2 Specialist Group

The specialist teams will be organized both by installation and configuration groups, and by areas of competence. The partner with the most knowledge of architecture and deploymemt of the components involved will organize the respective team:

- **Data Spine and Integration:** The group will provide support both with Data Spine operation and use of the data spine for service lookup, integration flows and data transformation. This team will be set up by FIT and C2K.
- **Networking and Security:** The networking and security group will handle issues regarding connections between components, resolve security incidents or implement the necessary changes in the access infrastructure such as access through firewalls and routing. This team will be set up by SRFG.
- **Platform Management Services:** The Platform Management Services group will handle issues regarding the Portal, Marketplace and Secure Data Store. This team will be set up by ASC.
- **Collaboration Services and Base Platforms:** Problems specific to one component or service in the EFPF platform other than the ones previously mentioned will be delegated to that group. For some components (see “D6.1 - EFPF Integration and Deployment I” or “D10.1 EFPF Exploitation, Sustainability and IPR Report – I”) access to hosting environments or source code is not provided and these will be managed by the responsible partner. This team will be set up by C2K.

4.4 Tools

To manage operational support tools have been evaluated and selected. The operational environment should provide a common installation, configuration, monitoring, alerting, documentation and issue management for the support staff, production engineers and developers. It is important that all core components can be managed and monitored in a

unified way and that anyone in the support team can deploy, monitor and manage any component. This is achieved by using Docker images as the unit of deployment, with scripted deployment, no manual configuration during deployment and using Docker compatible tools for administration and monitoring.

4.4.1 Application Containerization

The EFPF platform consists of applications which are developed using different programming languages and software libraries. The heterogenous nature of these applications makes it a challenge to offer consistent runtime across various operating systems when directly shipping software packages. Because of that, the project opts for a containerized delivery and deployment of all server applications. Containerization of the applications not only increases consistency of application runtime across different operating systems, but also adds high portability, isolation, and operational efficiency. This accelerates development, testing, and production cycles. Moreover, it simplifies deployment configuration and horizontal scaling. EFPF uses Docker and Portainer Community Edition to support containerized application management.

4.4.1.1 Docker as Containerization Technology

Docker is the most common technology for application containerization. It uses OS-level virtualization to facilitate lightweight containers that isolate applications from each other by bundling the software, libraries, and configurations. The containers can communicate with each other only through specific channels. Unlike virtual machines, Docker containers are managed by a single operating system kernel and make use of isolation features such as kernel namespaces to restrict application's access to host operating system, process tree, network, users, filesystem, and cgroups to limit computing resources. Such control is provided using a set of interfaces[3]; see Figure 5.

[3] https://en.wikipedia.org/wiki/Docker_(software)

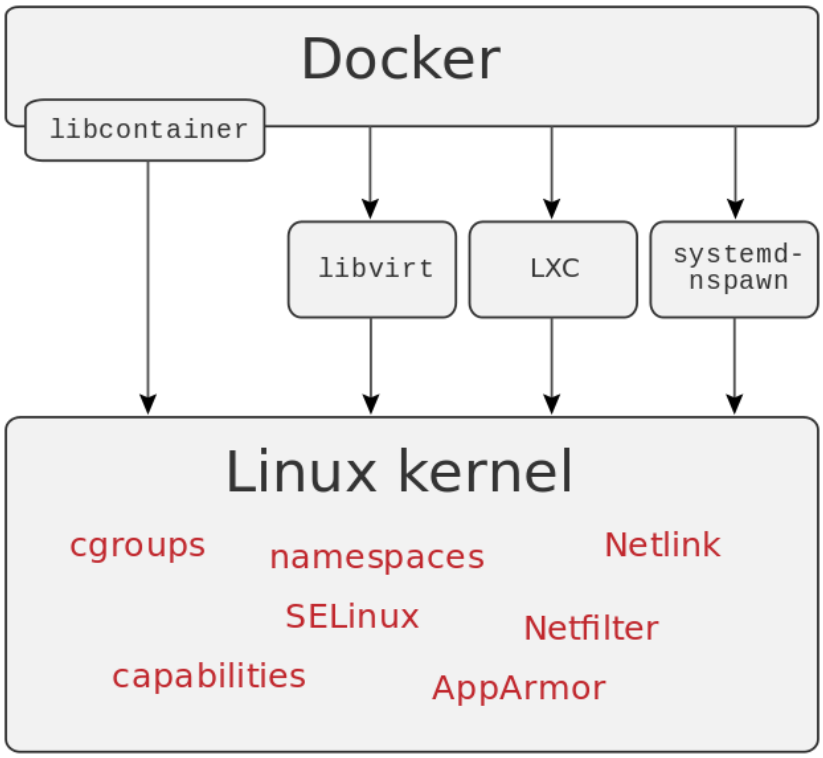

Figure 5. Docker uses various interfaces to access the virtualization features of the kernel

Docker consists of the following open source components and tools[4]:

- **Engine:** The Docker daemon is the process which manages container objects. Users interact with the engine using the Docker CLI or Docker Engine API
- **Objects:** These are the building blocks of a Docker application:
  - Containers are the standard, isolated environments to run applications
  - Images are built containers in a format which is ready to store and ship
  - Services allow scaling of containers across multiple Docker engines
- **Registry:** An online repository for docker images with an API to allow uploading (push) and downloading (pull) of images. The registry may be private or public.
- **Docker Compose:** A tool for configuring and orchestrating the lifecycle of multi-container applications
- **Docker Swarm:** Native Docker clustering functionality as part of the Docker Engine which can overlay a group of Docker Engines and treat them as a single Docker Engine. This enables cross-machine deployment and networking of containers as well as scaling and load balancing

4.4.1.2 Portainer for Graphical Container Management

The Docker Command Line Interface (CLI) and Docker API provide management functionalities to all Docker features. However, the use of these interfaces requires specific technical knowledge and is prone to errors. This gets especially hard and tedious when a single Docker Engine is set to host a large number of containers, managed by several

[4] https://en.wikipedia.org/wiki/Docker_(software)

administrators. The EFPF project envisions the use of test and production servers for deployment of various containerized applications by multiple independent stakeholders.

Portainer Community Edition[5] (CE) is an open source graphical management user interface for Docker. Portainer CE provides various graphical features which translate user requests into equivalent Docker API calls. This coupled with container status and health indicators as well as multiple Docker Engine support significantly increases the usability of containerized application management. Moreover, it provides graph-based representation of resource usage by each container for quick debugging and analysis. Most importantly, Portainer CE adds user access control to Docker API. The access control features allow creation of users and teams with different access to containers.

4.4.2 Alerting/Monitoring System

The purpose of the monitoring tools is to identify functional faults in different services and components in a deployment. The tools are also used for identification of security breaches, monitoring and visualization of the resource usage by the different services, visualization and logging of service usage statistics, and visualization and monitoring of network traffic. The monitoring tools ease debugging and optimization process by logging and providing relevant information.

There are many monitoring solutions available in the market. Main criteria for a monitoring tool in the current project include the following:

- The monitoring tool should be able to fetch information from different components with minimal effort. These components might include the services developed within the project or off the shelf.
- The monitoring tool should be able to provide visualization.
- The monitoring tool should be able to support distributed monitoring wherein the components can reside in different locations including factory premises.
- It should be able to provide querying, searching and alerting capabilities for the stored metrics.

4.4.2.1 Prometheus and Grafana

Prometheus[6] with Grafana[7] shall be used as a monitoring and visualization tool for EFPF. Prometheus with Grafana addresses the aforementioned requirements in the following way.

4.4.2.1.1 Component Integrations

Any service which is developed as part of the project, when needs to be monitored, can add implementation to their code via client libraries implementing different kinds of Prometheus metric types.

Any third-party systems on the other hand could be monitored with the help of third-party exporters. There are many official and unofficial exporters available for Prometheus which can be readily used for the third-party systems.

4.4.2.1.2 Visualization, Alerting

[5] https://www.portainer.io/products-services/portainer-community-edition/
[6] https://prometheus.io/
[7] https://grafana.com/

Grafana provides a rich set of panels for visualization. Grafana also can integrate to multiple types of data sources with the help of readily available plugins.

4.4.2.1.3 Distributed Monitoring

Distributed monitoring in Prometheus is achieved by using Prometheus federation, wherein one Prometheus server can scrape from another Prometheus server running in a completely different cluster.

4.4.2.1.4 Querying

Prometheus Query Language (PromQL) is a query language provided by Prometheus that lets clients select and aggregate the time series data in real time. This can be used for both fetching the data for report or user interface presentations and to set alerting rules.

4.4.2.2 A Typical Deployment

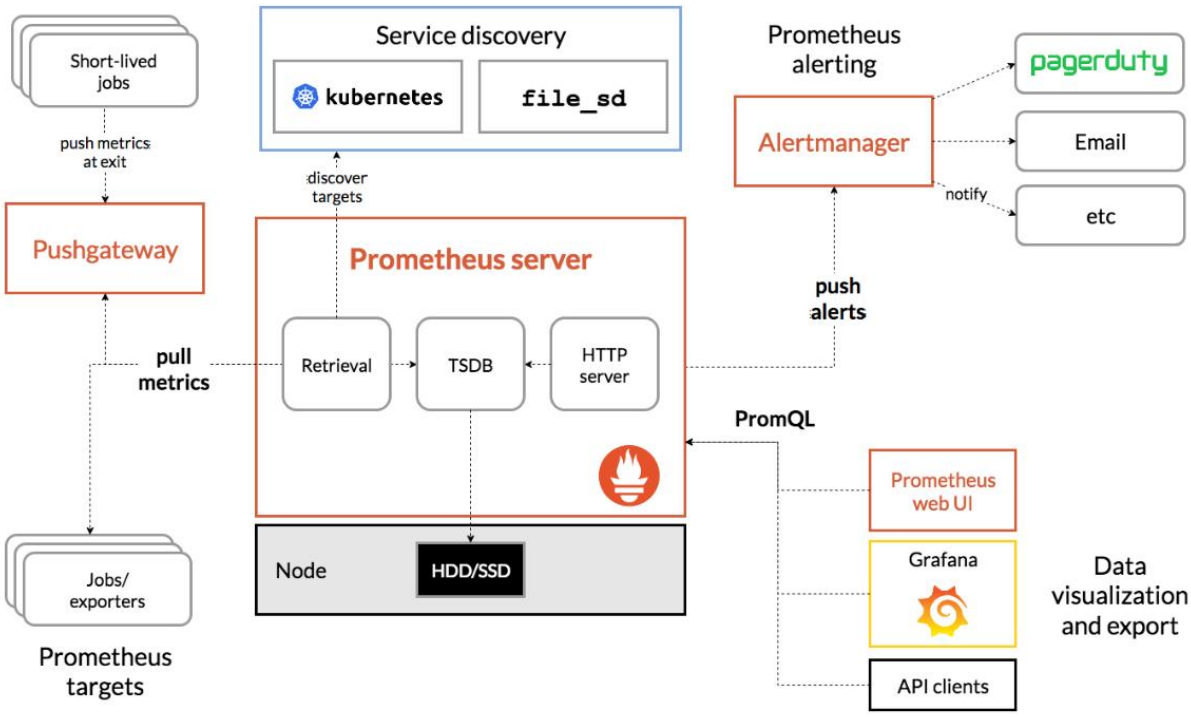

Figure 6. Components in a Prometheus deployment and the call flow (source: Prometheus docs[8])

Figure 6**Error! Reference source not found.** shows the architecture of Prometheus and the different components that are monitored by it. Prometheus scrapes different metrics from multiple jobs, databases, services and platforms. Once retrieved, these metrics are stored in Time series database or are used for real-time alerting and visualization. Prometheus can discover different targets for monitoring using methods such as service discovery (e.g. via Consul), configuration files or using DNS-SRV records (Domain Name System Service Records).

[8] https://prometheus.io/docs/introduction/overview/

4.4.3 Ticketing System

The DOA states that a Gitlab-based ticketing system will be set up to enable issue-reporting by the developers and users of the platform. The ticketing system will be used by the support staff to distribute, process, and escalate the reported issues. Issues can be captured in various ways: direct manual entry by a user, email, web form, automatic detection or communication by user with support staff that make a direct manual entry in the support system. The users do not necessarily need to have login credentials for the ticketing system. The system needs to have ways to assign issues to different support staff and groups in the support staff and categorizing issues by severity and installation group.

Three ticketing systems have been considered: (i) Tikki, a dedicated support ticket system, (ii) the EFPF Gitlab instance used for development and (iii) Engagement Hub from vf-OS. The final choice is still to be made. Tikki is a mature and full-featured tool for support ticket management, while Engagement Hub has a free open source licence model and can serve both support staff and customers in development and operational support. The development Gitlab instance is already in place but has a commercial license.

4.4.3.1 Tikki

Tikki is a specialized ticketing system which supports asynchronous updates between users with a hierarchical role and escalation system for tickets. Tickets can be entered via email to an IMAP inbox. From the inbox, tickets are processed manually and assigned to queues. Queues are the main way to organize the support work, e.g. per product, installation, or configuration group. Support staff can be assigned to queues, and tickets can be assigned to queues and users. There are various features to enhance the support process: tickets can change the status, be flagged for follow up, commented upon, have a history and support staff can receive notifications via email or push. The UI is multilingual.

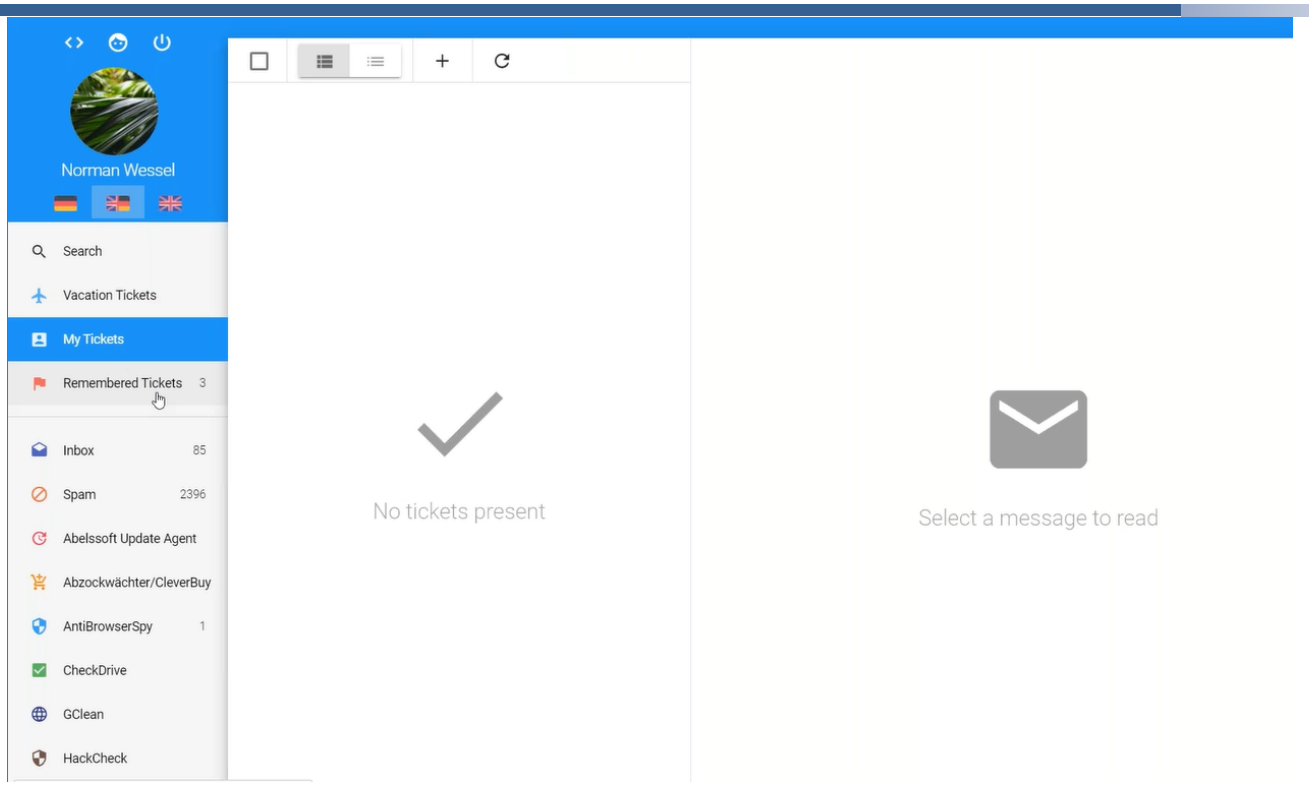

Figure 7: Tikki user interface

Tikki features Keycloak integration and can be set up with the EFPF SSO.

The Tikki license is closed source: it will be free to use during the project but will be subject to fee after the project ends. First-time configuration, setup and hosting will be by partner ASC. After the setup, the system will be provided as-is.

Tikki provides many useful features and is purpose built for ticket management. The UI is easily accessible and intuitive.

4.4.3.2 Gitlab-based ticketing system

The Gitlab system used for EFPF project management, development and code repository can also be used to manage support issues. The necessary features to organize the support tickets are there:

- Gitlab Issues for tickets
- A group can be used to separate support from development
- Projects can be used for installation and configuration groups, corresponding to specialist teams
- Milestones can be used for specific events or projects restricted in time, e.g. experiments or pilots. These may relate to a specific deployment of the platform

Since all potential members of the support staff are already registered in the system, which is also used for documentation and development, there are some integration benefits from using the existing Gitlab system. However, the licensing is not open source and running the system after the end of project will be subject to fees and re-deployment.

4.4.3.3 Engagement Hub

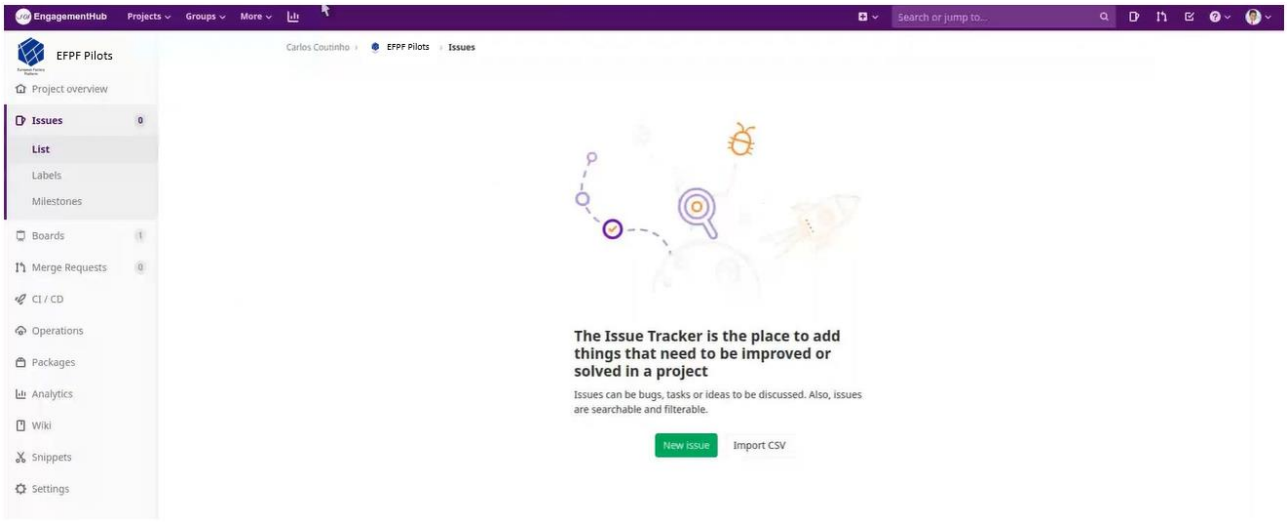

Figure 8: Engagement Hub User Interface

The Engagement Hub is provided by the vf-OS base platform that is now an EFPF tool. It is targeted towards the users of the EFPF platform that will develop solutions and integrate system using EFPF platform. Users will have logins for the Engagement Hub, and issues can be captured through direct entry in the web user interface.

The Engagement Hub is based on Gitlab CE and includes all features of Gitlab issue management as well as a chat feature. It is open source and free to use.

4.4.4 Interface Management Tool

Monitoring the evolution of the interfaces exposed from tools and services is a simple technique that focuses on documenting the rights and responsibilities of software modules to ensure that software which are depending one on the other will be able to continue working correctly in case of an update. This methodology guarantees that developers making use of software interface are informed about their evolution and can act upon it.

The Interface Contract Management Tool (ICMT) is a component under development which will be used to track changes of the interfaces of the tools and services in the EFPF platform. This tool is independent from the Service Registry (SR) and it serves as an extension of it, helping developers keeping track of the software interfaces they make use of.

Based on the ICMT usage, developers on the EFPF platform will be able to select the interfaces they are interested in monitoring and will be subscribed to alerts about their updates. The alerts will make use of the semantic versioning polices used by tools on the platform. This ensures that breaking and incremental changes to interfaces are properly handled, minimizing the risk of breaking data flows crucial for the correct operations of tools and services on the platform.

4.5 Documentation

To manage support issues, installation and migration, the production engineers, system administrators and support staff need documentation of components, configurations, and procedures. Documentation will be kept in ownCloud and in wiki form, preferably available in the same tool used for support issue management (the ticketing system). Resolved issues

could link directly to relevant documentation. Information about component and API versions should be included and tagging used to improve search.

The documentation should be organized by installation and configuration groups. The Data Spine is one group, the Management Services components are another one, and the other components are grouped per responsible partner.

Documentation will be of three types:

- **System Documentation:** Comprising architectural views, configuration information, and installation instructions. Targets developers, production engineers, support staff and maintainers of the EFPF platform.
- **Tutorials:** Describing use of the components from a production engineer or user perspective. Targets production engineers, open call users and EFPF customers building applications using the EFPF platform.
- **Design Guidelines**: Best practices in the use of the EFPF platform and polices that components must adhere to in the EFPF ecosystem Targets developers and production engineers for base platforms and EFPF components as well as users.

During development, installation and support, system documentation should be captured by developers and system engineers. Responsibility for documentation currently lies with the developing partner, an overview of this will be made later in the project to ensure relevant and current information.

4.6 Other Tools

Configuration Management Database (CMDB) tools have been considered to help the support team to manage the complexity of the EFPF infrastructure. These can help to provide an overview and the ability to estimate the impact of a malfunctioning component or a change on the various services and APIs. Tools that have been considered are iTop[9], CMDBuild[10], and OneCMDB[11]. However, the introduction of such a tool will be delayed to a later stage due to the fact that all developers (i.e. the current support staff) are very familiar with the configuration and the set up and learning curve will add to much time at this stage. When the support staff has some experience of managing issues and the requirements are clearer the decision will be reconsidered.

An API Monitoring Tool has been proposed in the project for monitoring the key performance indicators for services (e.g. number of requests per hour, successful and failed calls, response time). This could be valuable for users in integration design and procurement of services. It can be used for three cases:

- **Docker container monitoring:** Used for the services published by docker containers in the Data Spine and Management Services components in the EFPF platform.
- **Distributed Service Monitoring:** Follow key performance indicators across all services published in the EFPF Platform.
- **Distributed Logging:** Provide the ability to collect logs from all services in the EFPF platform for better technical support.

The implementation mechanism to be used and whether it should use push or pull to gather data is still to be determined. A possible implementation is to use Prometheus for this tool.

[9] https://www.combodo.com/itop-193
[10] https://www.cmdbuild.org/en/products/cmdbuild
[11] https://www.capterra.com/p/163428/OneCMDB/

5 Experiment Lifecycle Operational and Technical Support

5.1 Overview

This section is related to Task 7.2 Experiment Operational and Technical Support. The task aims to support the overall lifecycle of the experiments. This comprises the integration of the experiments into the EFPF platform in the cases that it is needed, the technical and operational support during experimentation period and the possible transfer of the concepts and methods developed within the experiment to the EFPF project. The work that will be performed in this task is connected to all technical packages. In particular, the task is strongly correlated with the rest of tasks in WP7 and with the activities in WP8, which is focused on Open Call Experimentation. Furthermore, the task is related with the activities in WP3 (Architecture and Data Spine) and WP6 (Integration and Deployment). Of course, the platform components that will be implemented in WP4 and WP5 are related as well with operational and technical support during experimentation as well.

As the task officially starts on M30 (alongside with experiments kick off) in this report we provide a series of early actions that have been performed and some facts that are proofs of how we are working in order to provide effective support during the experimentation period. The combination of these facts and actions reflects the status related to experiments operational and technical support and can be translated as an initial plan for these support activities. The detailed plan and the activities/procedures will be followed to support the upcoming experiments will be documented by the end of experiments in D8.3: Support for Experimentation on the EFPF Platform on M42. In this document we will provide information related to how the EFPF platform was used and user experiences during the experiment, how the support activities were used, and problems were solved, evaluation about chosen design patterns, configuration, and choice of tools. Any transfer of the concepts and methods developed within the experiment to the EFPF project will be documented as well.

5.2 Initial Plan for Experiments Operational and Technical Support

The planning for experiments lifecycle operational and technical support is comprised by two main type of actions:

1. Actions related to general project's design and structure
2. Actions related to activities that focus on operational and technical support and they are also connected with the experimental period

5.2.1 Actions related to general project's design and structure

- Creation of a WP related to operational management and support with the addition of a task (T7.2) devoted to the experiments support in order to create all the required procedures to provide this type of services.
- All technical tasks from WP4 and WP5 are designed with an extensive duration and ends in M48 despite the fact that pilot tasks end on M24. This will enable all the technical/research partners that are involved in these tasks to provide support during experimentation period for the corresponding components and functionalities.

Moreover, the aforementioned partners will be able to transfer concepts and methods developed within the experiment to the EFPF project by the end of the project.

- Almost all the deliverables related to the project platform design and the implemented components (WP3, WP4, WP5) will be publicly available in order to support the provided documentation to the experiment's participants.
- A generative and scalable approach during the project architecture's design has been followed. The realization of the interoperable Data Spine and the Service Catalogue of the project's platform will support the effortlessly integration of experiments infrastructure to EFPF or the use of the platform core services by experiments infrastructure. By adopting this design approach, the project targets to support the experiments implementation by providing to them a platform that is implemented with the focus to connect services and tools.

5.2.2 Actions related to activities that focus on operational and technical support

- A specialist/support team has been set up in T7.1 of the project. This team of specialists is going to contribute on operational and technical support activities during the experimentation period as well.
- A dedicated email line for the technical support of the experiments will be available to the participants.
- Webinars related to the technical aspects of the project's platform and the available tools and services will be available prior the experiments kick off.
- Documentation about platform architecture and tool/services of the project will be available to open call applicants.
- An internal survey that has been conducted for the different platform components and relevant IPR issues contained a dedicated section for the Open Calls period. In this section all the partners describe any issue related to use and support activities related to their components, any limitations or possible costs etc. This part of the survey was necessary in order to define if we can provide technical and operational support for all the involved components during experiments phase of the project.
- Other supporting tools and infrastructure that are used during the implementation phase of the project are planned to be provided during experiments as well. Tools that documented in chapter 3 of this document (i.e. Alerting/Monitoring System, Interface Management tool, Gitlab etc.) can be provided in order to technically support the experiments.

6 Cross-organisational Operations Planning and Execution

6.1 Overview

This chapter is related with the activities focussing on cross-organizational planning and execution. The chapter's information is derived from T7.3 however; this task is strongly connected with various tasks and their activities. As the task aims to define a set of policies and protocols related to various aspects of the EFPF platform, such as access to the platform, to distributed infrastructures, resources etc. it is perceived that the task is correlated with activities related to the data spine, portal, security and governance, deployment, technical and operational support etc. Besides this, task 7.3 is also related with the establishment of the support infrastructure across organizations etc. so it is also connected with chapters 3 and 4 of this report. The infrastructure for operational and technical support in platform level and during experimentation period is documented there in details so in this chapter, we will not repeat this information but we will focus on the first aspect related to policies and protocols.

6.2 Methodology for Collecting a Set of Polices/Protocols

In order to gather the different needs, rules, procedures and policies that are demanded on cross-organizational level in order we provide access and support, a 'living' document has been set up. In this document, some protocols that address a specific major topic/activity related to the platform have been collected by corresponding partners. The main topics that initially were defined are:

- Registration process.
- Exposing or use applications or services.
- Use of data sources.
- Adding new marketplaces.
- Add or purchase product and services.

Each protocol specifies a set of actions or guidelines that will be followed by the platform providers (EFPF project partners or other entities in the EFPF ecosystem) or users while carrying out specific activities. This 'living' document is going to be updated during the project's duration until the experiments kick off. The final policies/protocols will be presented in the second iteration of this document on M42 and D8.3 - Support for Experimentation on the EFPF Platform. Furthermore, it is worth mentioning that the presented protocols focus on operational and execution point of view. Other protocols and policies related to security, governance and general legal issues are described in corresponding deliverables.

6.3 Current Policies and Protocols

The next table (Figure 9) presents all the policies/protocols that have been gathered from various partners leading the corresponding activities and components' development. Each protocol specifies a set of actions or guidelines that will be followed by the platform providers (EFPF project partners or other entities in the EFPF ecosystem) or users while carrying out specific activities. As mentioned in the previous section, the following table demonstrates

the outcome of a 'living' document and it will be continuously updated and extended until the end of the project.

ID	Protocol	Description
1	Process should be followed for user registration on the EFPF Portal	In order to register and access the platform a user should follow the next steps: • Open Register here: https://efpf-portal.ascora.eu/auth/register • Enter all required information to the registration form • Accept Terms and Conditions (https://efpf-portal.ascora.eu/terms) • Wait for activation from EFPF admin • Start email validation process by logging in for the first time. Based on user's email, the admin will verify the identity of the user using KYC (Know Your Customer) check and assign the required roles to the user • Log in again for completed registration.
2	Protocol for users' roles assignment - A role should be selected/assigned	A new user of the platform should have/select one of the following roles: • Basic role is assigned to the user enabling the access to the marketplace and federated search of platform's products and services (i.e. role: EFPF_basic) • Administration role is assigned to privileged users with the responsibility for functionalities of the overall platform (i.e. role: EFPF_admin) • Data analytics role is assigned to those users who have access to analytics services on the platform (i.e. role: EFPF_analytics) • Factory connectors data analytics role is assigned to those users who have access to specific Factory Connector data analytics on the platform (i.e. role: EFPF_fc) • COMPOSITION data analytics role is assigned to those users with the access to COMPOSTION's data analytics services. The user will be assigned relevant roles based on the initial KYC verification. If the user wants to access additional services then the platform admin must be contacted to approve each required update of the user roles.
3	The user's data are stored and are accessible by specific users/entities	The user should know and agree that his/her data are stored and accessible. The user registration data are stored in the EFS (i.e. Keycloak Identity Server). Keycloak uses hashing techniques to store the data securely. Internally Keycloak uses the Postgres database for storage. In addition, the Keycloak in EFPF is stored on Hetzner Online server, which requires to complete a Data Processing Agreement (DPA) that incorporates GDPR requirements (see: https://wiki.hetzner.de/index.php/Datenschutz-FAQ/en#How_should_I_fill_out_the_Data_Processing_Agreement_.28DPA.29.3F) Furthermore, the user registration data are available for the administration of the platform. The administrator has a privilege to temporarily block or delete or reset credentials or manage system roles for the users. The administrator does not have any access to the users credentials as they are hashed and safely stored in the Keycloak identity server.

4	Protocol for users coming from base platform (from registration point of view)	Users that are coming from base platforms (Nimble, COMPOSITION, SMECluster etc.) that are already connected to EFPF must be registered to EFPF platform as well. However, after the registration process, the user's credentials are re-used for the log in to other platforms in the EFPF ecosystem.
5	Protocol for exposing an application, service or platform in the federated EFPF ecosystem	Two types of communication are supported. Based on its type a user should follow the corresponding procedures: Case 1 - Synchronous (request-response) communication: 1. The service provider needs to get an EFPF account and the necessary access permissions required to register services to the Data Spine Service Registry. 2. The service provider needs to register his/her service to the Service Registry with an appropriate service 'type' (preferably in the format "<platform_name>.<service_type>"). (Additional steps needed, but not to be performed by the service provider: • An EFPF administrator user 'admin1' needs to define/configure the access permissions for accessing that service's endpoints and operations permitted on those in the EFS. • The API Security Gateway (ASG), which checks for new service registrations/updates to services in the Service Registry periodically, creates a proxy endpoint/routes for endpoints of the new service in ASG – this could be used to invoke the endpoint directly without creating an integration flow in case if protocol translation and/or data transformation isn't needed.) 3. The consumers can now either consume the proxy endpoint in ASG for service provider's service directly (in the case of protocol translation and/or data transformation isn't needed) or create an integration flow in the Integration Flow Engine of Data Spine to consume the original endpoint(s) of the service provider's service, perform protocol translation and/or data transformation as needed in the integration flow. Case 2 - Asynchronous (PubSub) communication: 1. The publisher (service provider) needs to get an EFPF account and the necessary access permissions required to register services to the Data Spine Service Registry. 2. If the publisher plans to publish data to the Data Spine Message Bus using an EFPF Factory Connector/Gateway (FCG), he/she needs to log in to the EFPF Marketplace and purchase this Factory Connector there; else, this step is skipped. 3. Upon purchase, the publisher gets a root topic name which is specific to that publisher's company. 4. The publisher needs to logs in to the Factory Connector/Gateway Management Tool (FCGMT)

		and create a sub-topic under the root topic to publish to and get a key for publishing to that sub-topic. 5. The publisher needs to configure his/her tool/service that publishes data to publish to Data Spine Message Bus over the newly created sub-topic using the received key. 6. The publisher needs to register his/her service to the Service Registry with an appropriate service 'type' (preferably in the format "<platform_name>.<service_type>"). 7. The publisher's tool/service can now start publishing data to Message Bus over the created sub-topic using the given key.
6	Standard procedures for versioning, monitoring and deployment updates	There is no standard procedure applicable specifically to the services provided through the Data Spine for versioning and deployment updates However, a new 'Interface Contracts Monitoring Tool' is being developed and it is envisioned to be used to monitor changes to APIs and provide those updates to services consumers so that they can update their integration flows accordingly.
7	Protocol for adding a new Marketplace in EFPF ecosystem	Adding a marketplace is a manual process, which requires both technical effort and legal agreements. Therefore, it will not be possible for all users to add a marketplace. Any added external marketplace will be searchable via the EFPF Marketplace however all the business transactions will be between the buyer and the external marketplace. The EFPF marketplace will only be an agent which makes it easy to search for products at one location. External marketplace have to provide monthly (or shorter) details on transactions, which have EFPF as an origin. Any commission fee for EFPF isn't a fixed one yet
8	Protocol/procedures for creating a new Agent in EFPF ecosystem for Bidding Process	A user from EFPF platform should register/set up his/her virtual agent in order to be able to participate in online bidding service for specific goods. The following procedures should be followed: • The user should be registered and logged in to the EFPF platform • The user should select a role for his/her agent (supplier or requester) • The user should fill out some basic information about the company that the agent will represent • The user should wait for the agent's automated deployment Finally, the user will be able to use the agent for purchasing goods through automated bidding process or to initialize a bidding process in order to obtain a service/good
9	Protocol for making use of the Data Spine	Based on the two supported communication types a user should follow the corresponding procedures: Case 1 - Synchronous (request-response) communication: 1. The user needs to get an EFPF account and the necessary access permissions required to create integration flows in the Integration Flow Engine (IFE - NiFi) and to register services to the Data Spine Service Registry.

		2. The user would be given a 'Process Group' where he/she could create integration flow(s). This Process Group would be intended to contain all the integration flows related to a particular project/composite application developed by the employees of a particular company, for example. 3. The user can now log in to the GUI of the IFE and create a new integration flow inside that particular Process Group. 4. The user needs to get the technical metadata for the service to be consumed from the Service Registry. 5. The user needs to request for and acquire the necessary access permissions to invoke the endpoint(s) of the service to be consumed and to perform the required operations. 6. The user needs to create an integration flow using the GUI of the Integration Flow Engine to invoke the required endpoints, perform data transformation and finally create an 'interoperability-proxy' endpoints for the endpoints of the consumed service in the integration flow. 7. The user now needs to register these new endpoints to the Service Registry. 8. ASG automatically creates a proxy endpoints for these newly registered endpoints after a certain period. 9. The user now needs to request for and acquire the necessary access permissions to invoke these proxy endpoints in ASG and to perform the required operations. 10. The process is now complete and the user can now start invoking the service to be consumed through the IFE by invoking the proxy endpoint(s) in ASG. Case 2 - Asynchronous (PubSub) communication: 1. The user needs to get an EFPF account and the necessary access permissions required to create integration flows in the Integration Flow Engine (IFE - NiFi) and to register services to the Data Spine Service Registry. 2. The user would be given a 'Process Group' where he/she could create integration flow(s). This Process Group would be intended to contain all the integration flows related to a particular project/composite application developed by the employees of a particular company, for example. 3. The user can now log in to the GUI of the IFE and create a new integration flow inside that particular Process Group. 4. The user needs to get the technical metadata for the service to be consumed i.e. topic to subscribe to, Message Bus endpoint, etc., from the Service Registry. 5. The user needs to request for and acquire the necessary access permissions i.e. the key from the Message Bus to subscribe to the required topic.

		6. The user needs to create an integration flow using the GUI of the Integration Flow Engine to subscribe to the 'topic' of interest using the obtained key, perform data transformation and finally publish the resulting data to the Message Bus over a new topic. In order to be able to publish data to the Message Bus, the user would first need to get a root topic and a key for a sub-topic under this root topic. 7. The user now should register this new topic to the Service Registry. 8. The process is now complete and the user can now subscribe to this new topic he/she created using the obtained key and start receiving data.
10	Protocol for making use of the EFPF Data Broker	In order a user to be able to access and use the EFPF Data Broker should be follow the procedures that described Asynchronous (PubSub) communication in protocols #5 and #10. In a brief description, the user after creating a workflow in NiFI, have to index the service in the Service Registry (SR). Upon indexing this service, the API Security Gateway (ASG) automatically creates a route for the dedicated service. After this, the user should call the ASG endpoint with the required token to access the new workflow created by the developer.
11	Protocol for purchasing products and services from the EFPF Marketplace	Products and services will be bought not from the EFPF marketplace but from the external marketplaces. Therefore, the users should search for corresponding protocols, terms and conditions to the external marketplaces that are linked to EFPF. In the case that purchases directly from EFPF Marketplaces will be enabled by the end of the project, the corresponding details will be added in this list as well.
12	Protocol for uploading products and services from the EFPF Marketplace	At this time, the upload functionality for products and services is in the planning phase. The goal is to use the external marketplaces as destinations for the uploaded products and services as the EFPF Marketplace itself will not host any products and services. Therefore, the procedures that should be followed in this case will be driven by the approaches that are followed by each one of the external marketplace.

Figure 9: Policies and Protocols

7 Economical and Business Aspects

The EFF is going to be the owner and main exploitation partner for the EFPF platform and thus will have influence on the decisions made on tools for operational management. As the main commercial entity with the EFPF platform and its product, the considerations of EFF are expected to be different from the EFPF project.

In the current phase of the project, EFF has been setup as an independent non-profit association, registered in Austria. The EFF is currently composed of 12 EFPF partner organisations. All EFF members have shown commitment towards the development of a sustainable business model around the federated EFPF platform. The commitment is solidified by the pledge to support EFF activities e.g. the transfer of 1 person-month of project funding towards EFF. The members of the EFF mostly come from business and industrial backgrounds and therefore they bring the necessary experience of running a business. This experience will be valuable in the initiation phase of the EFF when the operating procedures, business models and organisation procedures are being defined.

With the setup of EFF, the question regarding the ownership of the EFPF platform has been answered. Since the EFF is composed of project partners and the association itself will be part of the project (pending ongoing DoA amendment), it will have the vital say in shaping up the operations, support and maintenance procedures concerning the EFPF platform.

When the business plans, economic priorities, customer communication management and service agreements for the EFF are more detailed in M48, this section in the upcoming D7.2 (M48) will provide an overview of the economic and business aspects of operational management of the EFPF platform after the project.

8 Conclusions and Outlook

This document has presented the current status of the technical support organization, tool support, experiment lifecycle support and policies and procedures for cross-organizational use of the platform. This will provide the necessary foundation for providing technical management and operational support during the pilots and open call experiments.

These activities will also provide an opportunity to evaluate and enhance these results based on the feedback from stakeholders. Some already planned future refinement and extensions are mentioned in this document: a more elaborate support issue process, further organization of and assignment of responsibility for the documentation and extended tool support, e.g. the API Monitoring Tool. Iterative refinement of the results presented will continue during the project lifetime and the final version, which should take into special consideration the exploitation of the EFPF platform and the business aspects of operational management and technical support, will be presented in D7.2 in month 48.

Annex A: History

Document History	
Versions	V0.0.1 ● Initial structure and draft. Used to capture design decisions in T7.1. V0.1.0 ● Initial concerns and design decisions V0.1.2 ● Updated concerns and design decisions V0.1.3 ● Additional content V0.2.0 ● CERTH content integrated V0.2.1 ● FIT content integrated V0.2.2 ● LINKS content integrated V0.3.0 ● Additional content, diagrams V0.3.1 ● Documenting tools V0.3.2 ● Additional content, ISMB logo replaced V0.4.0 ● Final editing V0.5.0 ● Ready for internal review V1.0.0 ● Review comments addressed
Contributions	CNET: ● Mathias Axling ● Matts Ahlsen CERTH ● Alexandros Nizamis ● Thomas Dimakis ● Dimostenis Ioannidis FIT ● Farshid Tavakolizadeh LINKS ● Edoardo Pristeri

Annex B: References

[BLI04] Blivband, Z. Grabov, P. and Nakar, O., "Expanded FMEA (EFMEA)," Annual Symposium Reliability and Maintainability, 2004 - RAMS, Los Angeles, CA, USA, 2004, pp. 31-36, doi: 10.1109/RAMS.2004.1285419.

[HF10] Humble, J., Farley, D (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation, Addison-Wesley Professional.

[IEEE 42010, 2011] May, I. S. O. Systems and software engineering–architecture description. Technical Report. ISO/IEC/IEEE 42010, 2011.

[KRU04] Kruchten, P. (2004). The Rational Unified Process: An Introduction. Addison-Wesley Professional.

[RW12] Rozanski, Nick, and Eoin Woods. "Software Systems Architecture: Viewpoint Oriented System Development." (2012).

European Factory Platform

www.efpf.org