SOFIE - Secure Open Federation for Internet Everywhere

779984

DELIVERABLE D3.1

Integration Plan

Project title	SOFIE – Secure Open Federation for Internet Everywhere
Contract Number	H2020-IOT-2017-3 – 779984
Duration	1.1.2018 – 31.12.2020
Date of preparation	29.6.2018
Author(s)	Mikael Jaatinen (LMF)
Responsible person	Mikael Jaatinen (LMF), mikael.jaatinen@ericsson.com
Target Dissemination Level	Public
Status of the Document	Completed
Version	1.00
Project web-site	https://www.sofie-iot.eu/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 779984.

Table of Contents

1. Introduction

Fragmentation and lack of security are among the biggest problems of IoT platforms. Most IoT platforms are vertically oriented closed systems, dedicated to specific application areas. There are today hundreds of IoT platforms with no or very limited interoperability. The main goal of the SOFIE project is to enable diversified applications from various sectors to utilize heterogeneous IoT platforms and autonomous devices across technological, organizational and administrative borders.

Secure open federation is the key concept of the SOFIE approach, aiming to enable creation of business platforms, based on existing IoT platforms and distributed ledgers, without needing to negotiate with any gatekeeper (neither technology- nor business wise). SOFIE exercises security and data sovereignty by design.

The integration phase in the SOFIE project is responsible for integration of the single SOFIE reference platform and all the business platforms that will be used by the SOFIE pilot projects.

The purpose of this Integration Plan deliverable (D3.1) is to ensure that the SOFIE business platforms that are verified in WP4 and are used by the WP5 pilot projects are well integrated according to a well-defined & predictable time plan and with good software quality.

The functional growth for the integrated platforms will be done iteratively, with initial focus on the features and functions needed by the first main release.

The integration phase will also perform component interface testing, to ensure that the various system components interact and pass data across each other as expected and function together cohesively.

In line with the AGILE approach to software development, integration will be performed incrementally and continuously. Three main releases will be produced during the lifetime of the SOFIE project.

- Testbed release (version 0)
- Pilot evaluation release (version 1)
- Final release (version 2)

SOFIE will use Git as the Source Code Management System, Atlassian Jira as the Agile software development management tool and Jenkins as the Continuous Integration server. Developer documentation will be written. FOSS components may in addition also use public resources such as Travis for Continuous Integration.

1.1 Structure of the Deliverable

Chapter 2 discusses the strategy and methodology for integration. Chapter 3 gives an overview of the SOFIE consortium and the roles (in the context of integration) that the different partners have. Finally, Chapter 4 describes the actual integration plan.

1.2 List of Acronyms

AWS	Amazon Web Services public cloud
CI	Continuous Integration
BP	Business Platform (for pilots)
FOSS	Free Open Source Software
RP	Reference Platform
WP	Work Package

2. Integration Strategy and Methodology

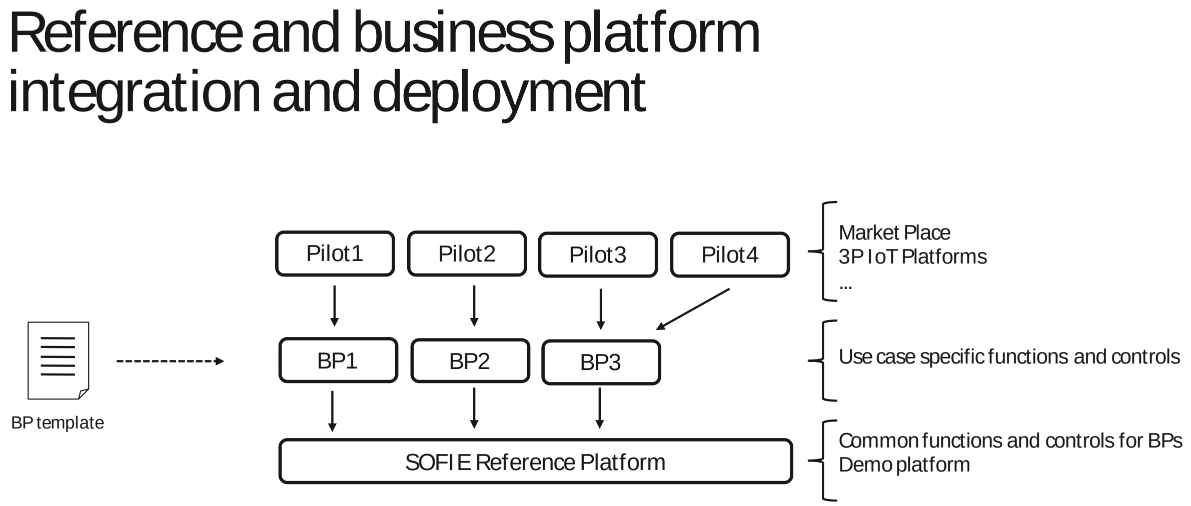
2.1 Strategy for Integration

The SOFIE project is committed to deliver four live pilots in three sectors (gaming, energy and food chain) to demonstrate the benefits that can be achieved with a federated inter-ledger platform approach.

Common functions shared and needed by (most of) the different pilots will be packaged into a reference platform module. It will be used stand-alone for small-scale demonstrations and as a testbed for initial evaluation.

Business Platform (BP) templates are used to define the BPs that are built on top of the reference platform by adding pilot-specific controls, DLTs, other needed third-party IoT platforms with needed adapters and a marketplace and/or consumable service APIs. This means that the BPs will include both FOSS and non-FOSS licensed software components. Multiple pilots can share a common business platform, according to the following conceptual diagram.

The different platform components will also include adapters and interfaces to external systems such as public blockchains.



Reference and business platform integration and deployment
Pilot1
Pilot2
Pilot3
Pilot4
Market Place
3P IoT Platforms
...
BP template
BP1
BP2
BP3
Use case specific functions and controls
SOFIE Reference Platform
Common functions and controls for BPs
Demo platform

Figure 1: Reference and business platform integration and deployment

The functional growth for the integrated platforms will be done iteratively. Initially, the focus is on the features and functions needed by the first main release, which is the testbed release (version 0).

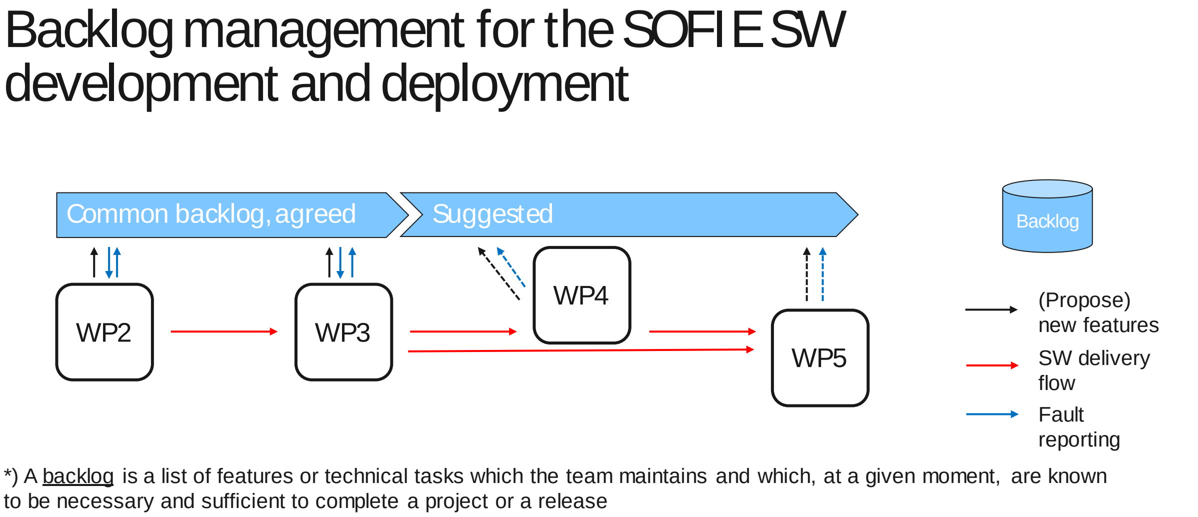
2.2 Methodology

2.2.1 Introduction

Modern Agile Development and Continuous Integration methodologies will be applied in WP3. Sprint cycles will initially be 4 weeks long (during the project, the Sprint cycle length can be reconsidered if necessary). Weekly team meetings will be arranged.

All documentation will be written in English.

Continuous Integration (CI) will be adopted from the beginning of the project. The software build process will be automated, including automated tests to ensure that critical software faults are detected immediately and can be fixed as fast as possible. One key enabler for this is that WP2 and WP3 intend to work from a common backlog, as explained in the following figure.



Backlog management for the SOFIE SW development and deployment

*) A backlog is a list of features or technical tasks which the team maintains and which, at a given moment, are known to be necessary and sufficient to complete a project or a release

Figure 2: Backlog management

The aim is to establish a single source repository for all FOSS software produced in SOFIE. For commercial software and existing reused FOSS components, a software component (system) database will be created and maintained for the reference platform and for all business platforms, in order to be able to integrate and deploy them. The database will specify details such as the vendor, download link, software version and known major vulnerabilities.

2.2.2 Continuous Integration

Continuous Integration is a development practice that requires developers to integrate code into a shared repository at least daily, but typically several times a day. Each check-in is then verified by an automated build, allowing teams to detect and locate problems early.

When automating the build process, automated tests are introduced into the process, so as to identify the major areas where things go wrong and get automated tests to expose those failures. Examples of such failures are software components failing to start up and incompatibilities in APIs between software components. When a build fails, the key priority of the development and integration team is to repair the build rather than adding new functionality on a broken base.

Continuous Integration leads to significantly reduced integration problems and enables transparency and predictability of the software development progress.

In WP3, continuous integration will initially be run with daily or every second day builds. WP2 FOSS components may in addition also use public resources such as Travis[1] for CI.

2.2.3 Managing the Integration Environment

Docker[2] containers are preferred for as many SOFIE software components as possible.

The plan is to set up one integration/testing account and one staging account in AWS. WP3 will own and administer both accounts, including access management for SOFIE users.

The integration/testing account will be used for the continuous integration and automated build process.

[1] https://travis-ci.org/

[2] https://www.docker.com

The staging account will be used for hosting parts of the business platforms, such as the reference platform and a private Ethereum network. The pilot projects will be able to leverage on the hosted services rather than having to (re)deploy everything.

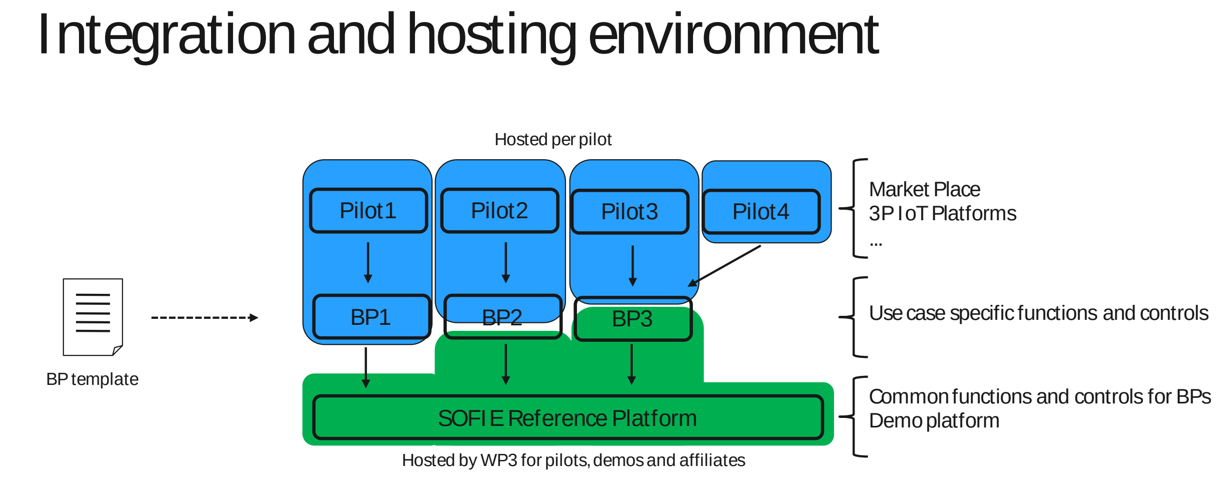

Integration and hosting environment
Hosted per pilot
Pilot1
Pilot2
Pilot3
Pilot4
BP1
BP2
BP3
BP template
SOFIE Reference Platform
Hosted by WP3 for pilots, demos and affiliates
Market Place
3P IoT Platforms
...
Use case specific functions and controls
Common functions and controls for BPs
Demo platform

Figure 3: Integration and hosting environment

2.2.4 Tools and Documentation

SOFIE will use Git as the Source Code Management System. Atlassian Jira[3] will be used as the Agile software development management tool with a common backlog for WP2 and WP3.

WP3 will use Jenkins[4] as the Continuous Integration server. WP2 FOSS components may in addition also use public resources such as Travis for CI.

Developer documentation will be written.

[3] https://www.atlassian.com/software/jira

[4] https://jenkins.io/

3. Actors and Roles for Integration in SOFIE

Table 1: Actors and roles for integration in SOFIE

Actor	Role
AALTO	Produces and delivers SOFIE software for integration
ASM TERNI	Contributes to BP integration (Italian energy pilot)
AUEB	Contributes to the SOFIE software design, and is also a receiver of the integrated platforms for WP4 evaluation
EMOTION	Contributes to BP integration (Italian energy pilot)
ENGINEERING	Contributes to BP integration (Italian energy pilot)
LMF (ERICSSON)	Responsible for integration of the RP and BPs
GUARDTIME	Contributes to BP integration (Estonian energy pilot)
OPTIMUM	Contributes to BP integration (Food chain pilot)
ROVIO	Contributes to BP integration (Mobile Gaming pilot)
SYN	Contributes to BP integration (Food chain pilot)

4. Integration

4.1 Testbed Release (version 0)

4.1.1 Features and Requirements

The integration plan will initially focus on the features and requirements of the first SOFIE main release, which is the testbed release (version 0).

The testbed release will follow the integration strategy with separation of functionality into a reference platform part, business platform parts and interfaces to external systems such as public blockchains. In the testbed release, there will be a deviation from the integration strategy. All functions will be packaged into one platform that provides minimum test and demonstration capabilities for WP4 and WP5 in a testbed environment. In the next main release, the reference and business platform parts will be separated.

The following requirements and indicative and may be subject to change during the Testbed release development.

4.1.1.1 General Requirements

- Use of Docker containers by default
- Per node, the standard requirements are (any deviations to these are specified):
    - RedHat or Centos Linux
    - Network connectivity (external and between test nodes)
    - SSH privileges
    - Minimum 10Gb of storage for test data, with the option to upgrade

4.1.1.2 Partner and Pilot Project Requirements

LMF (Integration Environment and hosted services)

- Two Amazon Web Services accounts will be created (European region)
    - One testing and integration account for the purpose of integration and testing activities
    - One staging account that will be used for hosting services for WP4 and WP5 pilots
- One administration (VPN/SSH) node for hosted services
    - Monitoring
    - Logging
    - Minimum 50Gb storage space
- One Jenkins node for continuous integration, integration testing nodes (1-N, dynamically scaled on spot instances)
- Status update integration with Slack (push build information)
- Ethereum (including needed network connections)
    - Private network (linked to AUEB and Aalto)
    - Rinkeby public test network
- Hyperledger Fabric 1.0 (linked to AUEB and Aalto)
- Guardtime KSI®Blockchain

Aalto (Development)

- 1-2 nodes (2-5 containers) for student project integrations
- Ethereum (including needed network connections)
    - Private network (linked to AUEB, Ericsson and Aalto's miners)

  - Rinkeby public test network
- Hyperledger Fabric (linked to AUEB and Ericsson testbed)
- Access to Guardtime KSI®Blockchain
  - Interworking with FIWARE, Ethereum and Hyperledger Fabric
- Connectivity with Aalto building automation IoT systems
  - ACRE API gateway or BioTope project O-MI/O-DF testbed
- Connectivity with local gateways (COAP and MQTT devices)
- HTLC between Hyperledger Fabric and Ethereum

AUEB (Evaluation)

- Ethereum (including needed network connections)
  - Private network (linked to Aalto and Ericsson)
  - Rinkeby public test network
- Hyperledger Fabric 1.0 (linked to Aalto and Ericsson testbed)
- Access to Guardtime KSI®Blockchain
- A virtual server for running FIWARE
- Connectivity with local IoT nodes (Raspberry PIs, running web of things)
- No VPN required (public IP addresses available)

Guardtime (Hosted Services)

- Hosting access to Guardtime KSI®Blockchain network
  - At least one aggregator
  - At least one extender

Estonian Energy Pilot (Guardtime)

- 3 nodes for the business platform and 3P IoT platforms
- Option to configure a firewall
- Public IP/host name

Italian Energy Pilot (ASM, Emotion, Engineering)

- 1 node for the business platform
- 1 node for EV/EVSE IoT platform
- Private Ethereum
- Access to Guardtime KSI®Blockchain

Food Chain Pilot (Optimum, Synelixis)

- 1 node for the business platform
- 1 node for transportation IoT platform
- Static public IP addresses / domain
- Private Ethereum
- Hyperledger Fabric 1.0
- Access to Guardtime KSI®Blockchain

Mobile Gaming Pilot (Rovio)

- 1 node for the business platform
- 1 node for Rovio proprietary platform
- Ethereum
  - Private network
  - Rinkeby public test network

4.1.2 SOFIE Reference Platform

The purpose of the SOFIE reference platform is to provide a minimal environment that provides a set of common basic controls for all business platforms.

In the testbed release, the reference platform will include the following features and technologies that are common for the different partners' needs:

- Ethereum
  - Private network
  - Rinkeby public test network
- Hyperledger Fabric 1.0
- Guardtime KSI®Blockchain
- Demo application to test Ethereum, Hyperledger Fabric 1.0 and KSI®Blockchain
- Adapters to the third-party IoT platforms that are part of the pilot projects

4.1.3 SOFIE Business Platforms

No separate business platform packages will be provided in the testbed release yet. The minimum functionality by the pilots will be included in the reference platform.

4.1.4 Testbed Deployment

The testbed will be setup in the AWS staging account by Ericsson for minimum demonstration purposes. Access to partners will be arranged. Partners may also choose to install the testbed fully in own hosted environment.

The private SOFIE Ethereum network will decentralized across (at least) Aalto, AUEB and LMF. Other partners (pilots) may choose to join it or to setup their own private Ethereum networks.

There will also be a decentralized Hyperledger Fabric 1.0 ledger across (at least) Aalto, AUEB and LMF. Other partners (pilots) may choose to join it or setup their own private Hyperledger Fabric deployment e.g. as side chains for Ethereum.

Guardtime will fully host the KSI service components and provide consumable API access to the service.

4.2 Sprint Schedule

The monthly sprint schedule for 2018 is the following:

Sprint schedule (2018)	Main purpose
2018-07	Integration environment preparations
2018-08	Integration environment preparations and account creation
2018-09	Integration environment build

2018-10	Integration environment build and CI build for the testbed release
2018-11	Integration of the testbed release
2018-12	Integration testing and supporting testbed verification

4.3 Future Releases

The testbed release (version 0) will, according to plan, be made available in November or December 2018.

Two additional main releases, Pilot evaluation (version 1) and final release (version 2) will be made available during 2019-2020. The integration plan will be updated to reflect the content of these releases as the scope stabilizes. The actual feature growth for these main releases will be done incrementally by following the strategy and methodology presented in previous chapters.