European Connected Factory Platform for Agile Manufacturing

European Factory Platform

WP4: Development of EFPF Building Blocks

D4.14: Smart Factory Solutions in the EFPF Ecosystem – Final Report

Vs: 1.9

Deliverable Lead and Editor: Simon Osborne, C2K

Contributing Partners: C2K, FOR, ISMB, NXW, AID, SIE, ICE, FIT, SRFG, CERTH, ISMB, ELN, ASC, UOS-ITI, A-D, ALM, CNET, SRDC, CMS

Date: 2022-12-31

Dissemination: Public

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

Short Abstract

The deliverable provides the final report of the Smart Factory Tools and Services available within the EFPF Ecosystem. It describes the functionality offered by each Tool or Service, the deployment approach within EFPF, and finally how they have been utilised and the new developments achieved since the last deliverable.

Grant Agreement: 825075

Document Status

Deliverable Lead	Simon Osborne, C2K
Internal Reviewer 1	Mathias Axling, CNET
Internal Reviewer 2	Gianluca Insolvibile, NXW
Type	Deliverable
Work Package	WP4: Development of EFPF Building Blocks
ID	D4.14: Smart Factory Solutions in the EFPF Ecosystem – Final Report
Due Date	2022-12-31
Delivery Date	2022-12-31
Status	<~~Draft~~ ¦ ~~Consortium Approved~~ ¦ EU Approved>

History

See Annexe 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 D4.14 ‘Smart Factory Solutions in the EFPF Ecosystem – Final Report’ describes the Tools and Services that are brought together by the project partners to satisfy the needs for Smart Factory solutions in the EFPF ecosystem.

T4.1 Factory Connectors and IoT Gateways

T4.2 Data Interoperability and Analytics

T4.3 Secure Data Store Solution

T4.4 Smart Factory Tools and Service Interfaces

T4.6 Distributed Workflow and Business Process Design and Execution

T5.5 Software Development Kit

This deliverable also provides a description of the new functionality that has been developed for the tools and services used in the original pilots, along with all new Tools and Services developed to offer greater diversity to the EFPF Smart Factory ecosystem. A description of development and refinement of the platform architecture and deployment process is also included from the perspective of the developers and users of Smart Factory Tools and Services.

The core objective of the work packages was to provide a diverse and comprehensive catalogue of Tools and services supported by an open and extensible platform architecture allowing users to develop solutions to today’s Industry 4.0 productivity challenges. This document describes the functionality offered by the Tools and Services delivered through the Tasks in this Work Package that demonstrate this objective has been achieved, and also highlights how they have been implemented and exploited in real world business scenarios and have been shown to offer tangible business benefits across a broad spectrum of manufacturing.

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 can be found at www.efpf.org.

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 four base platforms from FoF-11-2016 cluster funded by the European Commission, early on. This will inform the design of the EFPF Data Spine and the associated toolsets to fully connect the existing user communities of the four 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 “D4.14: Smart Factory Solutions in the EFPF Ecosystem – Final Report” is to describe the Connectors, Gateways, Tools and Services available within the EFPF Ecosystem and how they can be used individually or in combination to provide solutions to manufacturing challenges. This includes description how the tool or service has been used during the project, and how it has continued to develop since the last deliverable.

It is possible to treat this document as a reference to understand how solutions are implemented and as a guide to what offerings can be selected by end-users at this current time in the project.

0.3 Target Audience

This document aims primarily at the technical end-users from the manufacturing domain who wish to understand what offerings are made available through the EFPF Ecosystem to solve production issues or achieve productivity gains. The content of this deliverable is also of interest to technical audience from ICT, Engineering or Systems Integration organisations who are interested in either using or extending the existing offerings.

0.4 Deliverable Context

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

- **D2.3: Requirements of Embedded Pilot Scenarios:** Provide an overview of the pilot requirements on the federated EFPF platform

- **D3.1: EFPF Architecture-I:** Presents the baseline architecture of the EFPF ecosystem with focus on the EFPF platform and the Data Spine
- **D3.11: EFPF Data Spine Realisation - I:** presents an update to the architecture of EFPF, the design and realisation of the Interoperable Data Spine, interfaces for tools, systems, and platforms in the ecosystem and the data model interoperability layer at M18.
- **D3.12: EFPF Data Spine Realisation – Final Report:** presents an update to the architecture of EFPF, the design and realisation of the Interoperable Data Spine, interfaces for tools, systems, and platforms in the ecosystem and the data model interoperability layer at M42.
- **D4.13: Smart Factory Solutions in the EFPF Ecosystem - I:** Provides a report of the Tools and Services available within the EFPF Ecosystem that can be used to provide Smart Factory solutions
- **D6.1: EFPF Integration and Deployment - I:** Presents the development and deployment architecture of the EFPF ecosystem with focus on the EFPF platform and the Data Spine at M12.
- **D6.2: EFPF Integration and Deployment- Final Report**: Presents the development and deployment architecture of the EFPF ecosystem at M42.
- **D9.1: Implementation and Validation through Pilot-1**: Aerospace Pilot
- **D9.2: Implementation and Validation through Pilot-2**: Furniture Pilot
- **D9.3: Implementation and Validation through Pilot-3:** Circular Economy Pilot

0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 0: Introduction:** An introduction to this deliverable, including a general overview of the project, an outline of the purpose, scope, context, status, and target audience of the deliverable at hand.
- **Section 1: Evolution of Smart Factory Tools in the EFPF Ecosystem:** An introduction to User Requirements, followed by the Deployment Approach and the Service Integration and Communications Workflow.
- **Section 2: Smart Factory Tools and Services:** Describes the Tools and Services made available within the ecosystem that support the creation of Smart Factory solutions.
- **Section 3: Conclusions and Outlook:** Summary status and description of expectations of next steps.
- Annexes:
    - **Annexe A**: Document History
    - **Annexe B**: References
    - **Annexe C**: Component Deployment

0.6 Document Status

This document is listed in the Description of Action as "public".

0.7 Document Dependencies

This document is the second of the two deliverables that describe the technical status of Smart Factory solution with the EFPF Ecosystem. This first deliverable submitted at Month 48 of the EFPF project describes the final technical achievements and tools and services.

0.8 Glossary and Abbreviations

A definition of standard terms related to EFPF, as well as a list of abbreviations, is available at https://www.EFPF.org/glossary

0.9 External Annexes and Supporting Documents

Annexes and Supporting Documents:

- None

0.10 Reading Notes

- None

1 Evolution of Smart Factory Tools in the EFPF Ecosystem

The EFPF ecosystem supports the delivery and co-ordination of Smart Factory Tools, Services, Connectors and Gateways from multiple platforms involved in the project. The availability of such solutions addresses the diverse (e.g., digitalisation, shop-floor connectivity, analytics, etc.) needs of connected factories and lot-size-one manufacturing scenarios. Access to these solutions enables the manufacturing companies in the EFPF ecosystem to dynamically react to market opportunities, introduce efficiency and robustness in their production environments and use modern technologies to maximise business interests.

This deliverable describes the key functionalities and the approach on how these tools and services are hosted, connected, and used to share data in order to meet the end-user needs. It also explains the new functionality and innovations that have been implemented since the last deliverable, and describes how it has been exploited in real end user scenarios. It should act as a report of functionality available to end-users through the EFPF Ecosystem and provide guidance on how Tools and Services can be combined using the features of the EFPF Data Spine to solve diverse production issues.

1.1 Architecture Developments

EFPF Ecosystem Architecture

Figure 1 presents an overview of the high-level architecture of the EFPF ecosystem that consists of the Ecosystem Enablers and tools, services, and platforms from various providers. The Ecosystem Enablers are the core components that enable the creation and the functioning of the ecosystem. In addition, it consists of separate blocks for tools/services/data APIs indicating that the ecosystem also enables the integration of individual tools and services together with full-fledged platforms.

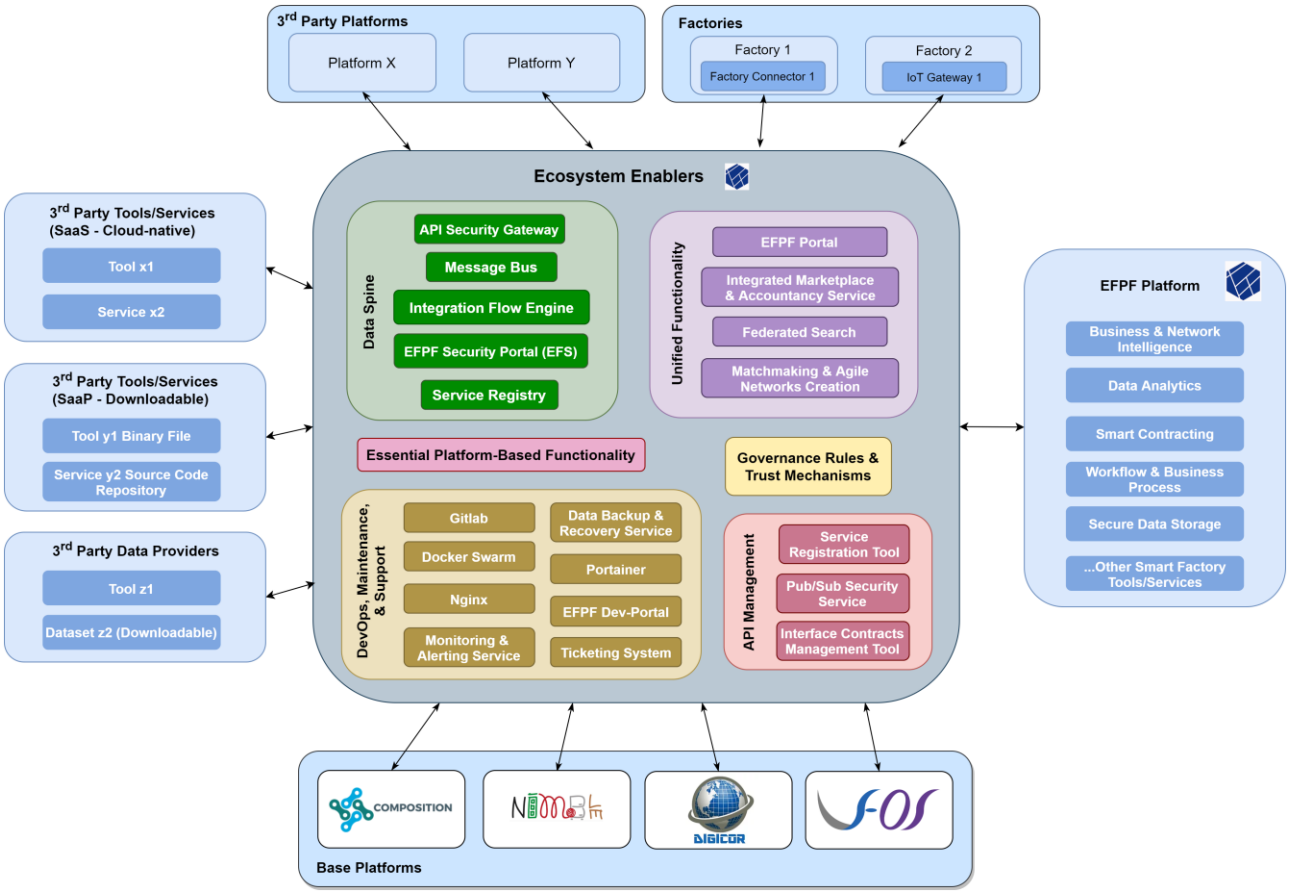


Figure 1. High-level Architecture of the EFPF Ecosystem

The EFPF platform follows the microservices architecture approach in which different functional modules implement individual functionalities that can be composed based on specific user needs. In order to implement this approach, all components in the EFPF ecosystem are prescribed to implement and publish open interfaces, preferably REST interfaces, allowing the exchange of data.

The EFPF ecosystem is designed considering the federation approach in mind where the distributed heterogeneous digital manufacturing platforms and the individual Smart Factory Tools and Services developed, provided, and managed by different independent entities permit the creation of added value within the ecosystem. To enable communication among them, an integration and communication layer, i.e., the Data Spine that acts as a translator/adapter between them is used. In addition, the rest of the Ecosystem Enablers provide the common core functionality and the digital infrastructure that is needed for the efficient operation of the ecosystem. Thus, the EFPF ecosystem as a whole follows Service-oriented architecture (SOA) style. The main elements in the EFPF federation are:

- **Ecosystem Enablers:** In the previous version of the EFPF ecosystem architecture, the Data Spine, that provides the interoperability infrastructure that interlinks and establishes interoperability between heterogeneous tools, services, and platforms and enables the creation of composite applications was illustrated as the only central core entity. In the latest version of architecture presented in this final report, the architectural vision was extended beyond interoperability and service composition to also include DevOps for easy deployment, clustering for high availability, automation for better usability, and components for an efficient infrastructure monitoring, management,

operations, etc. The Ecosystem Enablers are categorized into 6 types based on the functionality they offer:

1. Identity Federation, Cross-Platform Interoperability & Service Composition (Data Spine)
2. DevOps, Maintenance & Support
3. API Management
4. Unified Functionality
5. Essential Platform-Based Functionality
6. Governance Rules & Trust Mechanisms

- **Data Spine:** This Ecosystem Enabler is the central entity or gluing mechanism in the EFPF federation. The Data Spine provides the interoperability infrastructure that initially interlinks and establishes interoperability between the four base platforms: COMPOSITION, DIGICOR, NIMBLE and vf-OS (see D3.1 for more details). It adheres to common industry standards and follows a modular approach to enable the creation of a modular, flexible, and extensible ecosystem. Therefore, it can be easily extended beyond interconnecting the base platforms to "plug in" new 3rd party platforms and interlink them with the already connected platforms. Figure 1 also highlights the platform agnostic nature of the Data Spine, i.e., it is evident from the high-level architecture that as far as interactions with the Data Spine are concerned, there is no distinction between the EFPF platform and the base platforms or any other 3rd party platforms. Thus, the Data Spine would be independent from the rest of the EFPF platform. This hypothetically means that even if the EFPF platform were "switched-off" in the future, the Data Spine would not be affected and therefore would continue to support an interconnected ecosystem.
- **EFPF Platform:** This is a digital platform that provides unified access to dispersed (IoT, digital manufacturing, data analytics, blockchain, distributed workflow, business intelligence, matchmaking, etc.) tools and services through the Ecosystem Enabler called 'EFPF Portal' that acts as the single point of entry for the ecosystem. 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 from the EFPF project partners. The collection of enhanced versions of such tools and services from the base or 3rd party platforms deployed together as microservices would constitute the EFPF platform. These micro-services are made accessible through the EFPF Portal using the Single Sign-On (SSO) functionality offered by the Data Spine.
- **Base Platforms:** The EFPF ecosystem is created by initially interlinking the four digital manufacturing platforms from the European Factories-of-Future (FoF-11-2016) cluster focused on supply chains and logistics [EMC22]—namely NIMBLE [NIM22], COMPOSITION [COM22], DIGICOR [DIG22], and vf-OS [VFO22]. These are termed as the 'Base Platforms'. The base platforms provide functionality that is complementary to each other with minimum overlap and hence by interlinking them, the EFPF ecosystem is able to offer a comprehensive set of business functions.
- **3rd Party Platforms:** In addition to the four base platforms, the EFPF ecosystem enables interlinking of other 3rd party platforms that address the specific needs of connected smart factories. The examples of 3rd party platforms that joined the EFPF ecosystem include ValueChain's Network Portal platform [VLC22], Nextworks' Symphony platform [NXT22] and SMECluster's Industreweb platform [C2K22].

- **3rd Party Tools, Services, and Data:** The EFPF ecosystem can also be extended by connecting individual tools, services, and data APIs, etc., that do not belong to an existing platform.

Interaction between the Smart Factory Tools and the Data Spine

Figure 2 illustrates how different Smart Factory Tools, Services, Connectors and Gateways from multiple platforms can make use of the Data Spine to communicate with each other in general. The specific or exact interactions of the Smart Factory Tools with the components of the Data Spine depend upon specific use cases. The figure groups the Smart Factory Tools based on their domains or functionalities and shows the interactions of these groups with the Data Spine components for better readability.

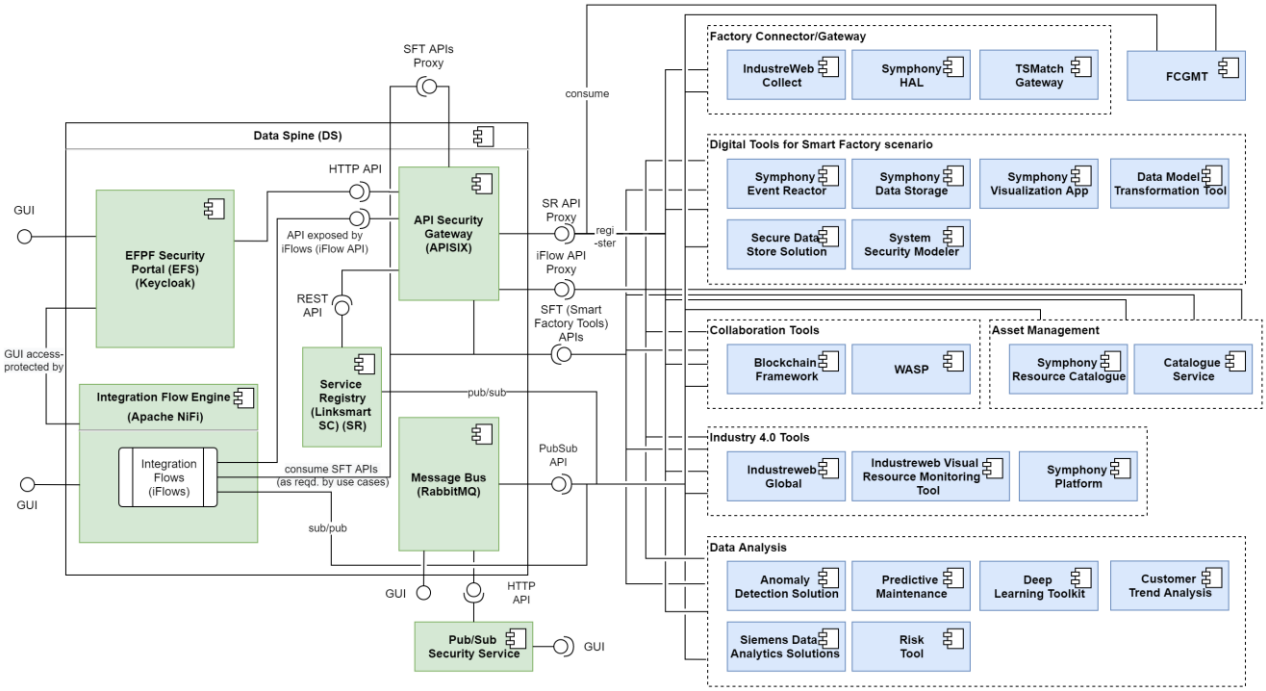


Figure 2: Interaction between the Smart Factory Tools and the Data Spine

In addition, Figure 2 specifies the technologies that were selected to realise the conceptual components of the Data Spine, and shows the APIs provided by these components, their interrelationships and how the Smart Factory Tools consume these APIs in order to interoperate and interact with each other through the Data Spine.

The EFS component of the Data Spine provides identity and access management functionality for the Smart Factory Tools and enables single sign-on (SSO) in the EFPF ecosystem.

The Smart Factory Tools make use of the Integration Flow Engine (Apache NiFi) to create integration flows which can be used to translate between heterogeneous protocols and data models as required by different use cases. The APIs of the Smart Factory Tools are secured by using the EFS. The HTTP-based APIs exposed by the integration flows are secured through the API Security Gateway (Apache APISIX). In Figure 2, the API 'iFlow API Proxy' exposed through the API Security Gateway is the secure proxy API for the 'iFlow API'. In addition, the API Security Gateway can also provide permanent reverse proxy endpoints (SFT APIs Proxy) for the Smart Factory Tools that do not have fixed/permanent API endpoints (SFT APIs).

The Smart Factory Tools use the Service Registry to register their services and to find the metadata of other services such as their API specifications that can be useful e.g., while creating the integration flows. The tools communicate with the Service Registry through its secure proxy API 'SR API Proxy' exposed through the API Security Gateway. After the iFlow APIs are registered to the Service Registry, the API Security Gateway automatically creates secure proxy APIs ('iFlow API proxy' in the figure) for them. The SFT APIs can be invoked directly by different tools/services or through the integration flows.

The Message Bus (RabbitMQ) component can be used for mediating the transfer of messages or data between the Smart Factory Tools that follow the asynchronous communication pattern. The Message Bus offers 'PubSub API' that can be consumed directly by the tools or through integration flows.

Finally, in an ecosystem as large as EFPF, there are multiple data providers and consumers. It is desirable that the data providers have direct control over who is authorized to access their APIs. In addition, the data consumers should be able to directly view the topics available through the Message Bus and directly ask the data providers for access permissions, without involving the administrator in the loop. The Pub/Sub Security Service offers an intuitive GUI to provide this access consent delegation functionality. The access to the PubSub API of the Message Bus is controlled through the Pub/Sub Security Service.

1.2 Architecture Considerations and Implications

This section presents a summary of architectural considerations and implications based on the requirements and from the perspectives of the providers and consumers of smart factory platforms, tools, services, and system integrator users concerning factors such as usability, multitenancy, platform integration efforts, etc.

- **Architectural design**: The EFPF ecosystem architecture classifies the EFPF components into two high-level categories:
  1. **Ecosystem Enablers**: The components that provide the core central functionality and tooling support to enable the creation and functioning of the ecosystem and operation of and communication among the other components. The Ecosystem Administrators are responsible for maintaining these components
  2. **Smart Factory Tools and Services**: These are the components from the connected platforms that provide use case- and domain-specific functionalities. The individual providers of these tools are responsible for their deployments and provisioning.

  This classification makes it easy for the Ecosystem Administrators to focus on maintaining the core central infrastructure, while the System Integrator users can make use of the Ecosystem Enablers to integrate the Smart Factory Tools and Services with the ecosystem.
- **Federated interoperability approach**: The EFPF ecosystem architecture follows a federation approach, where the interoperability between different tools/services is established "on-demand" i.e., when required by a use case through an integration flow. There is no common data model or API imposed at the ecosystem-level. Therefore, there is no overhead on the system administrators of maintaining such a complex canonical model and on the services to understand it and adhere to it.
- **Communication patterns**: The EFPF ecosystem supports two communication patterns that are widely used in the manufacturing domain:

1. Synchronous (request-response) pattern
2. Asynchronous (Pub/Sub) pattern

- **Interoperability**: The Data Spine bridges the interoperability gaps between services mainly at the following levels:
- **Protocol interoperability**: For the Synchronous (request-response) and the Asynchronous (Pub/Sub) pattern, the Data Spine supports standard application layer protocols that are widely used in the industry and employs an easily extensible mechanism for the supporting new protocols.
- **Data model interoperability**: The Data Spine provides the necessary digital infrastructure and tooling support to transform between the message formats, data structures and data models of different services thereby bridging the interoperability gaps for data transfer.
- **Security interoperability**: The Data Spine facilitates the federated security and SSO capability in the EFPF ecosystem.
- **Interaction approach interoperability**: The mismatch in the interaction approaches followed by different services can hinder communication among them. Such interoperability gaps between the services at the levels of “interaction approaches” can be bridged by using one/more components of the Data Spine.
- **Dependence on IoT Gateways**: To support lower layer protocols and other IoT networking technologies (e.g., ZigBee, ZWave, BLE, etc.), the Data Spine relies on Factory Connectors and IoT Gateways deployed at the edge.
- **Agility and flexibility**: The use of the Data Spine to establish interoperability allows the tools/services to be loosely coupled. This allows the tools/services to have neutral APIs that are not strongly tied to any specific implementation and provides the flexibility to different tools/services to evolve independently. The reliance on APIs as contracts between service providers and service consumers is a standard practice; however, successful collaboration depends upon the former adhering to the semantic versioning [Sem22] standard recommended by the EFPF ecosystem to version their APIs and conveying plans to deprecate/upgrade their APIs to the latter in advance.
- **Usability, multitenancy, and collaboration**: The Data Spine provides an intuitive, drag-and-drop style GUI to the system integrator users to create integration flows with minimal effort. The collaboration of work concerning a particular integration flow among different users is easy to manage as the Data Spine provides a Web-based GUI for creating integration flows. In addition, it provides a multi-tenant authorization capability that enables different groups of users to command, control, and observe different parts of the integration flows, with different levels of authorization. In addition, the Federated Search, Matchmaking and Agile Networks Creation services enable users to search for collaborators, negotiate, form teams, and work collaboratively.
- **Built-in functionality and tool/service integration effort**: The Data Spine provides connectors for standard communication protocols such as HTTP, MQTT, AMQP, etc., that are widely used in the industry. The Data Spine takes care of the boilerplate code and facilitates the system integrator users for integrating their services by configuring only the service-specific parts of the integration flows with minimal coding effort.
- **Platform integration effort**: The Ecosystem Enablers are cloud-native solutions, and therefore, no additional local deployments are needed to integrate platforms through it.

Integration of a platform with the EFPF ecosystem needs federating its Identity Provider with the EFS, registration of its services and integration with the unified ecosystem services such as the Integrated Marketplace.

- **API management**: The Data Spine provides a Service Registry component to store and retrieve service metadata including the API specifications, which ensures uniformity across and completeness of the API specifications. The Pub/Sub Security Service provides mechanisms that enable the data providers to have direct control over who accesses their data published to the Message Bus. The Interface Contracts Monitoring Tool (ICMT) notifies users in the case of any breaking changes to the APIs they are consuming.
- **Modularity and extensibility**: The architecture has been designed with modularity and extensibility in mind to meet the need for incorporating new tools, services, and platforms in the EFPF ecosystem with minimal effort. The Ecosystem Enablers are modular in nature and communicate with each other through standard interfaces and protocols. Support for new functionality such as protocols can be added by developing new processors/plugins. The Ecosystem Enablers adhere to common industry standards to enable the creation of a modular, flexible, and extensible ecosystem.
- **Performance, scalability, and availability**: The EFPF ecosystem makes it easy to integrate new tools/services and promotes reusability. To ensure high performance, high throughput and high availability, the performance critical Ecosystem Enablers such as the Data Spine have CD/CD pipelines configured for automated deployment and are deployed within a cluster using the Docker Swarm container orchestration technology.
- **Maintainability**: The loosely coupled and modular nature of the EFPF ecosystem helps significantly towards its maintainability. A high-quality user documentation of the Ecosystem Enablers and the smart factory services and tools in the EFPF ecosystem has been published on the EFPF Dev-Portal.
- **Interoperability approach**: The Data Spine's interoperability approach closely aligns with the Federated Interoperability Approach specified by the CEN/ISO 11354 Enterprise interoperability Framework [ISO11354].
- **Communication patterns and protocols**: Synchronous (Request-Response) and Asynchronous (Pub/Sub) patterns and standard application protocols such as HTTP, MQTT, AMQP, etc.
- **API specification**: OpenAPI and AsyncAPI specification standards
- **Pub/Sub topic naming convention**: Eclipse Sparkplug™
- **API versioning**: Semantic Versioning Standard - Semver (MAJOR.MINOR.PATCH)
- **Data models**: No common, canonical data model is enforced/prescribed at the ecosystem-level; however, the use of the following data models is recommended:
    - Industrial IoT, Industry 4.0: OGC SensorThings, W3C WoT TD, OPC UA Part 100
    - Supply Chain and Logistics: BITAS, GS1 – EPCIS
    - Platform Marketplace: UBL

1.3 Software Development Kit

The EFPF Software Development Toolkit (SDK) set of components can be used for developing applications that use the platforms and services defined in the EFPF scope.

The SDK itself exposes and centralises the APIs of EFPF services, making them available and reusable to other applications. This information can then be integrated using the SDK Studio, a full-fledged Integrated Development Environment (IDE), which includes the standard concepts of project definition, but also the creation of the application stack. The SDK Studio is able to configure the services APIs using a user-friendly interface, and is also able to embed other applications developed in EFPF, such as the EFPF WASP tool or the SDK Frontend editor.

This embedded WASP tool can be invoked by the SDK Studio, to allow developers to define the workflows and interactions of the proposed application to be developed. The resulting set of workflows can then be converted by the WASP tool's Process Designer service into code, that is properly accommodated in the developing environment. This code can then be customised, edited, changed, and improved using the SDK Studio's Code Editor, a powerful development environment which includes syntax highlighting, code folding, navigation, and is able to support multiple code languages.

This EFPF SDK source code repository hosts the developed source code under version control enabling the code to be made available (open-source code) and disseminating it in the development community. Further details of the SDK can be found in section 2.7.

1.4 Revised Deployment Process

The EFPF ecosystem is composed by the above set of software components that need to be deployed and kept updated over the time. The centralized GitLab repository is used for project planning, source code management as well as continuous integration (CI), continuous delivery (CD) and monitoring and contains all the tools and configuration necessary to deploy or update a component to the corresponding environment (test and production). The continuous integration and testing environment are available to all EFPF base platform and service providers but mainly used for the Ecosystem Enablers. The Gitlab repository is being migrated to change ownership and management to EFF.

A high level overview of the EFPF CI/CD Pipeline system is shown in Figure 3:

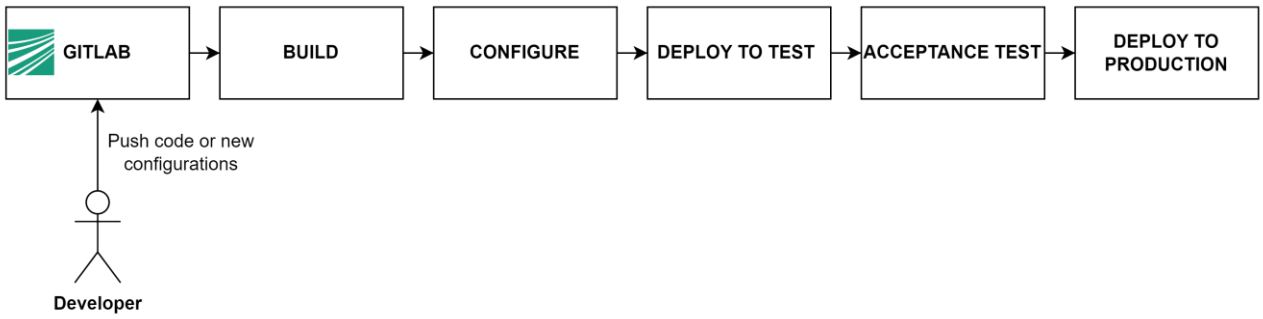


Figure 3. EFPF CI/CD Pipeline

The deployment process (shown in Figure 3) in EFPF is based on containerization using Docker. GitLab CI/CD can be used to build a new version and push versioned Docker

images to the EFPF registry (Gitlab Artifact Repository). This allows automated deployment and rollback process that is not dependent on component developers.

Most of the EFPF tools and services are configured and deployed through Ansible Playbooks, a tool that offers a repeatable and re-usable way to execute a set of instructions through an optimized approach (the current statement is executed only if needed). For instance: a “create directory” instruction is executed only if the directory itself has not already been created.

1.5 Deployment Environment Update

Starting with the Data Spine, a 3-tier deployment model has been used in the EFPF project. This model is composed of development, testing and production environments, which will have different rate and type of change.

Environment	Content	Provider
Development	Development version where new updates will be made.	FIT / SRFG
Testing	Test version to verify the integrated Data Spine. Elements passing quality gateway may be brought into Production.	C2K
Production	Stable live version used by all, including the open calls	C2K

The production environment was deployed in two major rollouts, after which the Ecosystem Enablers were in place.

1.6 Runtime platform

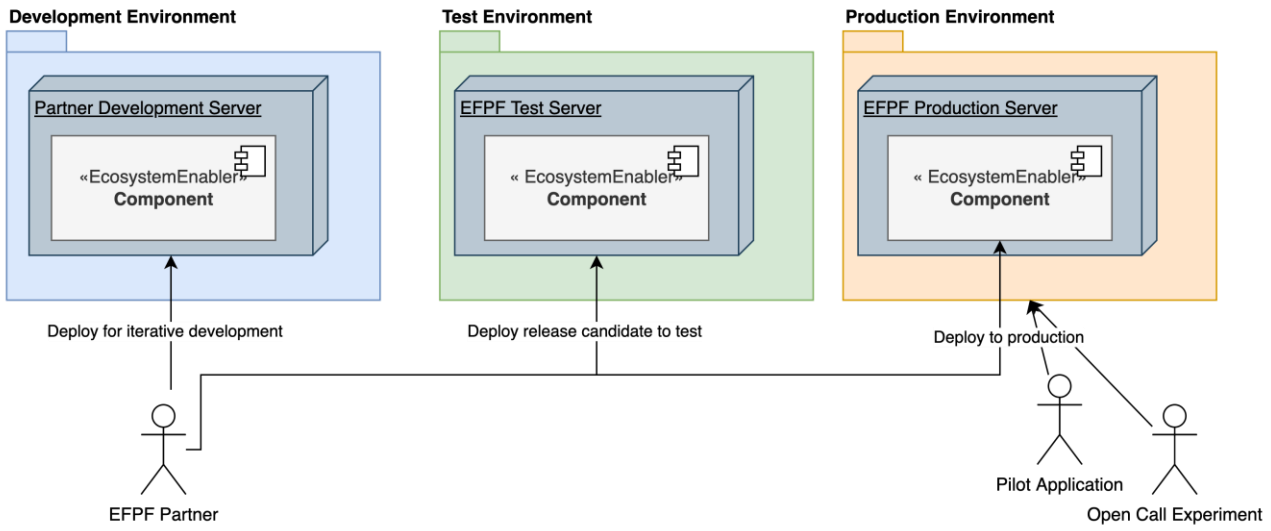


Figure 4. Runtime Platform Overview

The **Development Environment** was a less stable environment hosted on several distributed nodes. Development of the EFPF Ecosystem and integration of base platforms started at the same time as the specification of deployment pipeline and runtime

environment. The initial versions of the Data Spine, Portal and other central components were therefore distributed over the hosting resources of the technical partners developing the components. The development environment was discontinued, and the Data Spine services was shut down in September 2022.

To ensure reasonable resource requirements and guarantee stable operation, the **Test Environment** was reserved for deployment of the Ecosystem Enablers. The specified CI/CD pipeline must be used to deploy in the Test Environment. Test procedures are run and integration tests for dependent components can be made against the Test server versions. Release candidates that pass the tests can be deployed in the Production Environment. If a new version does not pass the quality gateway, it is rolled back, corrected, and re-deployed, not edited in place. The Test environment is currently used as a staging environment only by EFPF components but may in the future also be used for externally developed tools and services.

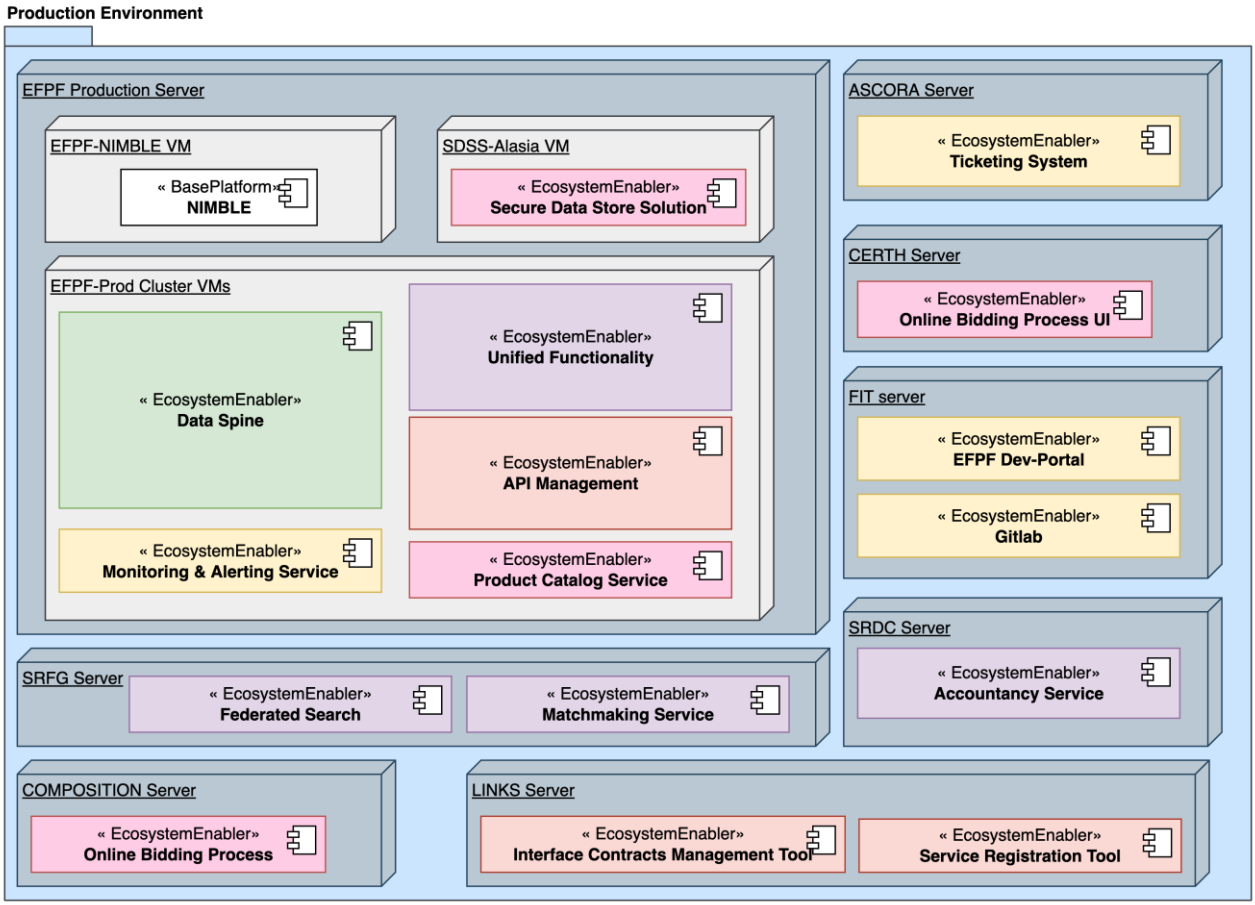


Figure 5. The Production Environment

Once the component passes the quality gateway, this version may be deployed to the **Production Environment**, which is hosted on a dedicated server. The decision was made to use on-premises hosting at C2K rather than relying on cloud resources, e.g., AWS or Azure Cloud. This was motivated by having predictable costs and a guaranteed continuous environment when transitioning to EFF.

The Production Environment hosts the stable versions of the Ecosystem Enablers. During the project, this primarily meant the Data Spine, Monitoring and Alerting Service and Portal. However, the migration of other Ecosystem Enablers is underway to prepare for EFF management of the platform. Figure 5 shows the deployment of components at M42.

Components can be hosted in the Production Environment if they implement the CI/CD pipeline and container technology. If they have high cohesion with the Data Spine and Portal and resource consumption is estimated to be low and stable, they may be hosted together with the Portal and Data Spine on the same nodes. Otherwise, a separate virtual machine is set up on the server. Performance efficiency and reliability for the Data Spine and Portal is the deciding factor when allowing other components to be deployed in the Production Environment.

2 Smart Factory Tools and Services

2.1 Factory Connectivity

In order to utilise the functionality of the tools and services data from the numerous data sources available within, the manufacturing environment needs to be made available. To achieve this, it is necessary to utilise a Factory Connector or IoT Gateway that can interface with the specific devices, sensors, and systems the user wishes to gather data from. The connectors and gateways then communicate with the EFPF Data Spine using a predefined data model that relates to the nature of the data and its application.

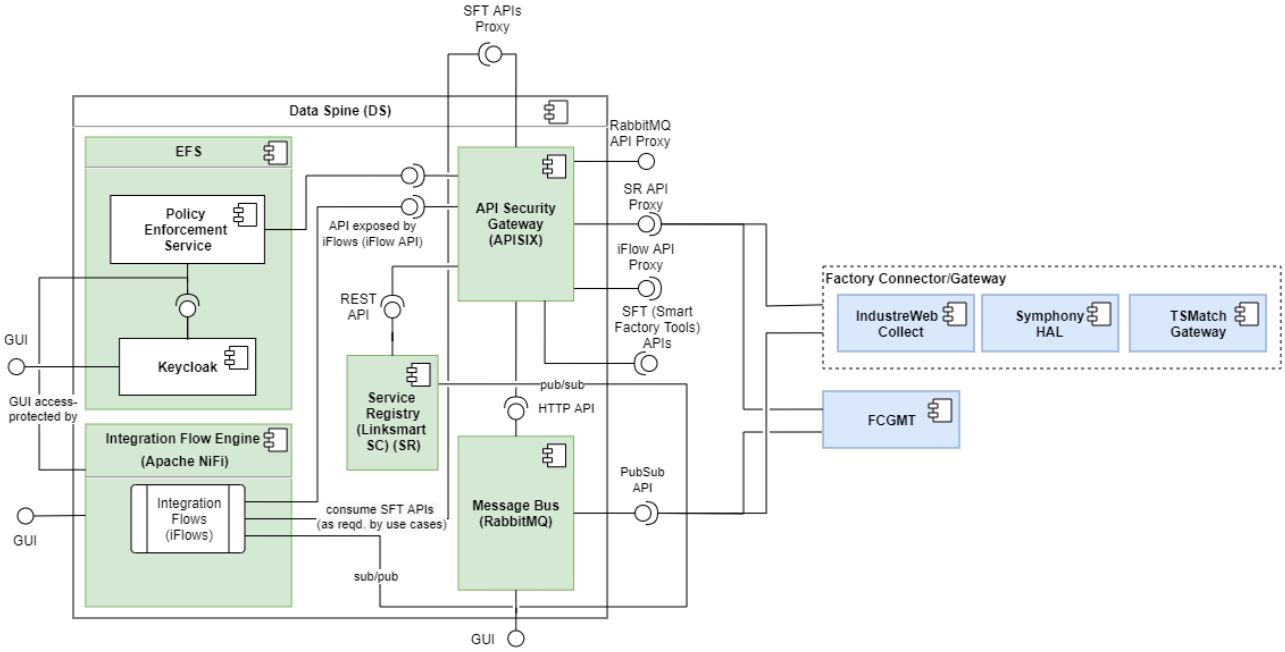


Figure 6: Interaction between the Factory Connectors and the EFPF Data Spine

In EFPF there are three implementations of Factory Connectors & Gateways, supporting between them the most widely used industry standards and systems (e.g., OPC UA, Modbus, Zigbee and propriety PLC protocols from Siemens, Rockwell, Omron, Schneider). The architecture, however, supports future compliant Connectors and Gateways to be supported.

Figure 7 shows the high-level architecture of Connectors / Gateways and their interaction with the EFPF Data Spine.

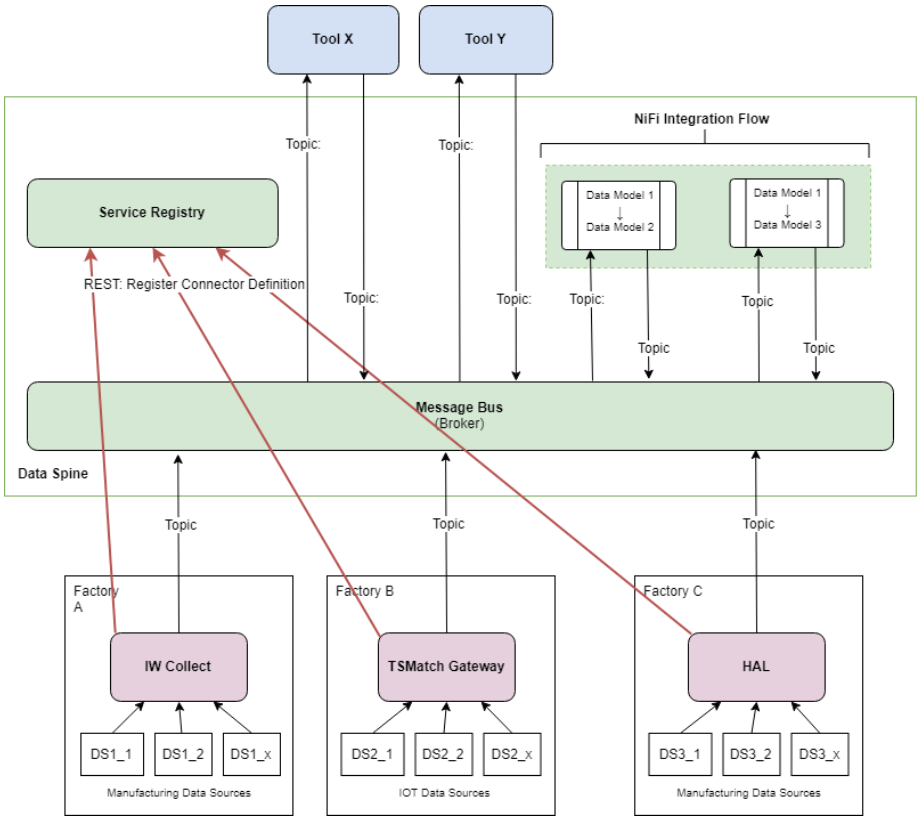


Figure 7: Factory Connectivity within EFPF

Each Connector or Gateway has its own functionality set supporting different connectivity options, protocols, and architecture, whilst offering different functions such as data thresholds the ability to make a calculation or scaling the data values. The interface with the EFPF Data Spine is a common interface, so in terms of the data presented to the tool, the versions of connector or gateway are not important.

As described in section 1.1, the communication with the Data Spine is achieved via MQTT Pub/Sub protocol which allows data to be published from the factory on a specific topic name when the data changes. The Data Spine Broker component then routes the data to any Tool or Service that has been granted access to subscribe to that data. This is more efficient than polling the Connector or Gateway at the interval and presents fewer firewall security concerns than accepting inbound polling requests.

The following sections will describe each of the existing Factory Connectors and Gateways and their functionality, followed by the management tool used to configure the Pub/Sub communications.

2.1.1 dIndustreweb Collect

2.1.1.1 Functional Description

Industreweb (IW) Collect is a high-speed data engine that interfaces with a range of systems and devices with the aim of extracting business critical data. The primary objective of IW Collect is to solve getting data from sources that may prove to be normally difficult or require a bespoke solution. All data sources are then transformed automatically to an in memory common data model to allow processing and event triggering.

Data sources can include industrial control systems, legacy equipment, wireless devices such as ZigBee, and industrial networks such as Profinet, Profibus, Modbus and AS-interface. It can also connect and interrogate databases such as MS SQL and MySQL, and flat file formats such as XML and JSON.

The architecture is based around the concept of connectors to enable monitoring of a diverse range of data sources by interfacing with the production systems and sensors. IW Collect runs on an embedded industrial computer platform that can be pre-installed on machines or retrofitted to existing machines.

2.1.1.2 Usage and Deployment

Industreweb Collect is deployed as an industrial edge device, located close to the production process. The Industreweb “Turn-Key” offering provided the Industreweb Collect Factory Connector with network switch and associated power supply in a quick to deploy industrial cabinet. Industreweb Collect is available on a commercial license basis.

The Industreweb Collect Factory Connector was utilised in four EFPF pilots within Furniture and Aerospace domains – its application in these pilots is described below:

Usage	Function	Description
Aerospace Pilot	Process control	Industreweb Collect was used as the core of the Spray Booth application to monitor the usage of the process via sensors installed, and to control the air supply valve and air extractor.
Aerospace Pilot	Alerting	Industreweb Collect was used within the Store Monitoring application to detect the quantity of products carried in companies’ production stores. A load cell sensor was used to detect the mass of products present and the value sent to the cloud to support Business Intelligence applications.
Furniture Pilot	Data logging	Industreweb Collect was used in the Predictive Maintenance application, where it was interfaced with 24 sensors recording air pressure, motor current, and motor temperature. This data was then published to the EFPF data spine for consumption by Analytics services.
Furniture Pilot	Error Proofing	Industreweb Collect was used to support Error Proofing by interfacing with a barcode scanner and the business ERP system. Parts passing the camera are scanned, and the string used to query the ERP, allowing the data returned to be passed to the Dashboarding system.

2.1.1.3 New Developments & Roadmap

Since the pilot phase the Industreweb Collect Factory Connector has undergone development in order to introduce many new features to simplify setup, provide new functionality and to improve performance:

Usage	Description
Strongly typed data point engine	Internally all data is now stored in data tags that relate to a specific DataType. This allows Collect to understand the nature of the data it is co-ordinating
Internal data cache to cater for data disconnection	Each data tag also contains a cache facility so that if connection is lost from the EFPF message bus, it can re-publish any data that has not already been sent.
Support for cloud-based admin tools	Support for parsing configuration data sent from the Industreweb Global admin tools.

The development of Industreweb Collect will continue with the roadmap and will be made available to end users via EFF marketplace.

2.1.2 Symphony Hardware Abstraction Layer (HAL)

2.1.2.1 Functional Description

The Symphony Hardware Abstraction Layer (HAL) is a tool that allows to interact with a hardware device at a general abstract level rather than at a detailed hardware level. It abstracts the low-level details of various heterogeneous fieldbus technologies and provides a common interface to its user.

HAL supports KNX, BACnet, Modbus-TCP, Modbus-RTU as well as several other proprietary control protocols. It can be extended by developing modules that can be plugged into its core. It can be interconnected with specific field buses either directly (via RS232/485 serial ports or GPIOs) or through the use of IP based gateways, such as KNX IP router and/or interface, Modbus/TCP gateways.

The HAL component provides access to any available resources (sensors and actuators) as datapoints. The datapoints are primitive objects with basic data type (int, float, boolean) but devoid of any semantic annotation (physical object type, measurement unit, …) and are presented according to the OGC SensorThings data format standard. The HAL supports access via REST and gRPC and furthermore enables publish/ subscribe features via MQTT. Figure 8 shows the internal architecture of the component and its external/internal interfaces: it exposes two configuration interfaces (GRPC and REST) and through the MQTT connector it is possible to stream the data flowing towards the fieldbus driver(s) after being internally pre-processed.

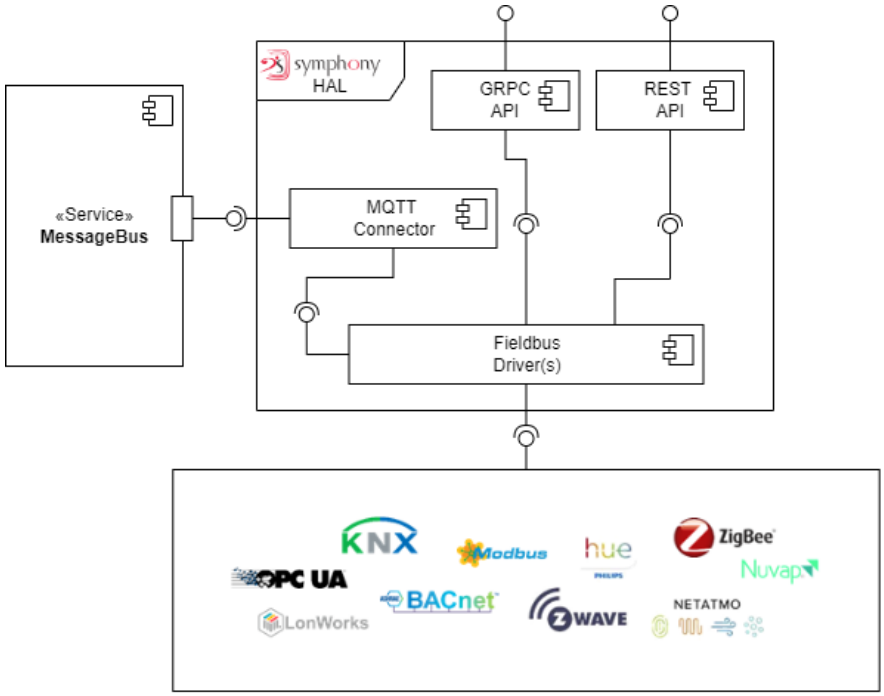


Figure 8 Symphony Hardware Abstraction Layer (HAL) architecture

2.1.2.2 Usage and Deployment

The Symphony Hardware Abstraction layer is released, installed, and configured (based on the customer requirements) by Nextworks.

It has been currently deployed as part of the Environmental Use Case pilot scenario with two running instance: a Cloud instance (hosted in the NXW's private cloud) to continuously monitor the status of a set of Edge sensors (lights and sirens) to display their status using the Symphony Visualization App; an Edge instance (hosted in the pilot premises) is used to directly interact with the sensors (read and write operations) and to post their status over the Data Spine.

2.1.2.3 New Developments & Roadmap

The Symphony Hardware Abstraction Layer is continuously updated and improved to be compatible with the latest connectivity protocols of the new state of the art sensors.

2.1.3 TSMatch Gateway

2.1.3.1 Functional Description

TSMatch [1]is a software-based solution that supports the semantic matchmaking of IoT data to services. The main goal of TSMatch is to automate the data supply between IoT data sources and services, while satisfying the service needs. For that, TSMatch v1.0 followed a semantic similarity matchmaking approach, where semantic descriptions of IoT devices are

[1] https://git.fortiss.org/iiot_external/tsmatch

matched to semantic descriptions of services, based on an ontological approach. TSMatch v2.0 relies on a *Natural Language Processing (NLP)* based approach with neural network model, to improve the matching between descriptions of IoT devices, and service descriptions.

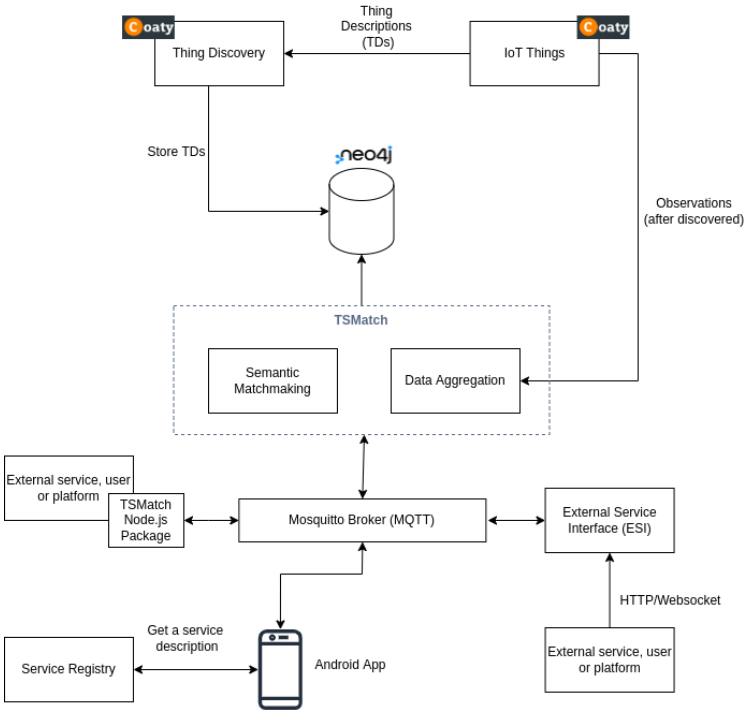


Figure 9: TSMatch high-level architecture and components.

Following a client-server approach, and as illustrated in Figure 9. TSMatch comprises a server-side, the **TSMatch engine**, and a **TSMatch client** (application for Android, to perform environmental monitoring on a shop-floor).

The TSMatch Engine is composed of 2 main functional blocks and several interfaces:

- **Semantic Matchmaking**: Performs semantic matchmaking between IoT Things descriptions (stored on a database) and Ontologies. The result is a set of enriched data nodes, which are also stored in a database.
- **Data Aggregation**: Sensor data aggregator.
- **Ontology Interface**: Provides support for ontologies to be imported into TSMatch.
- **Connectors**: Different connectors, e.g., Mosquitto to RabbitMQ; HTTP/REST, etc.
- **External Service Interface**: Interface based on OpenAPI to interconnect to an external service registry.

The input and output of the matchmaking and data aggregation processes are stored on a local Neo4J database. This database stores IoT Things descriptions, service descriptions, ontologies, and new data nodes (aggregated Things based on a category, e.g., temperature measurement).

Moreover, TSMatch relies on the following external components:

- An **MQTT broker**. TSMatch currently relies on an MQTT broker based on Mosquitto as message bus. The TSMatch client and engine interconnect to the MQTT broker.
- **Thing discovery**: IoT Thing discovery is supported via coaty.io[2]
- **Service Registry**: Holds a set of service descriptions. Currently holds environment monitoring service specification examples based on OWL and WSDL, which the user can select via the TSMatch client.

2.1.3.2 Usage and Deployment

TSMatch is open-source under an MIT license. It is available via the EFPF Website and via the FOR TSMatch public GitLab[3]. Documentation is available both via EFPF[4] and the TSMatch GitLab, to assist new users in the installation and experimentation of TSMatch.

TSMatch is being applied in the EFPF Aerospace pilot. The overall goal of this pilot scenario is to ensure that specific parameters in specific production machines and production environment are analysed in real-time to provide effective decision support. TSMatch supports, as illustrated in Figure 10, the data sensing workflow, where The TSMatch Factory Connector discovers IoT sensors on a shop-floor and connects them in an automated way, publishing their data into the Data Spine Message Bus to make it available to the other EFPF tools. The data is transformed using an integration flow in the IFE and made available to the Symphony platform's GUI integrated with the EFPF Portal. The GUI shows a real-time visualisation of the collected data to the users.

[2] https://coaty.io/

[3] https://git.fortiss.org/iiot_external/tsmatch

[4] https://docs.efpf.linksmart.eu/projects/factory-connectivity-smart-factory-tools/ds-tsmatch-gateway/

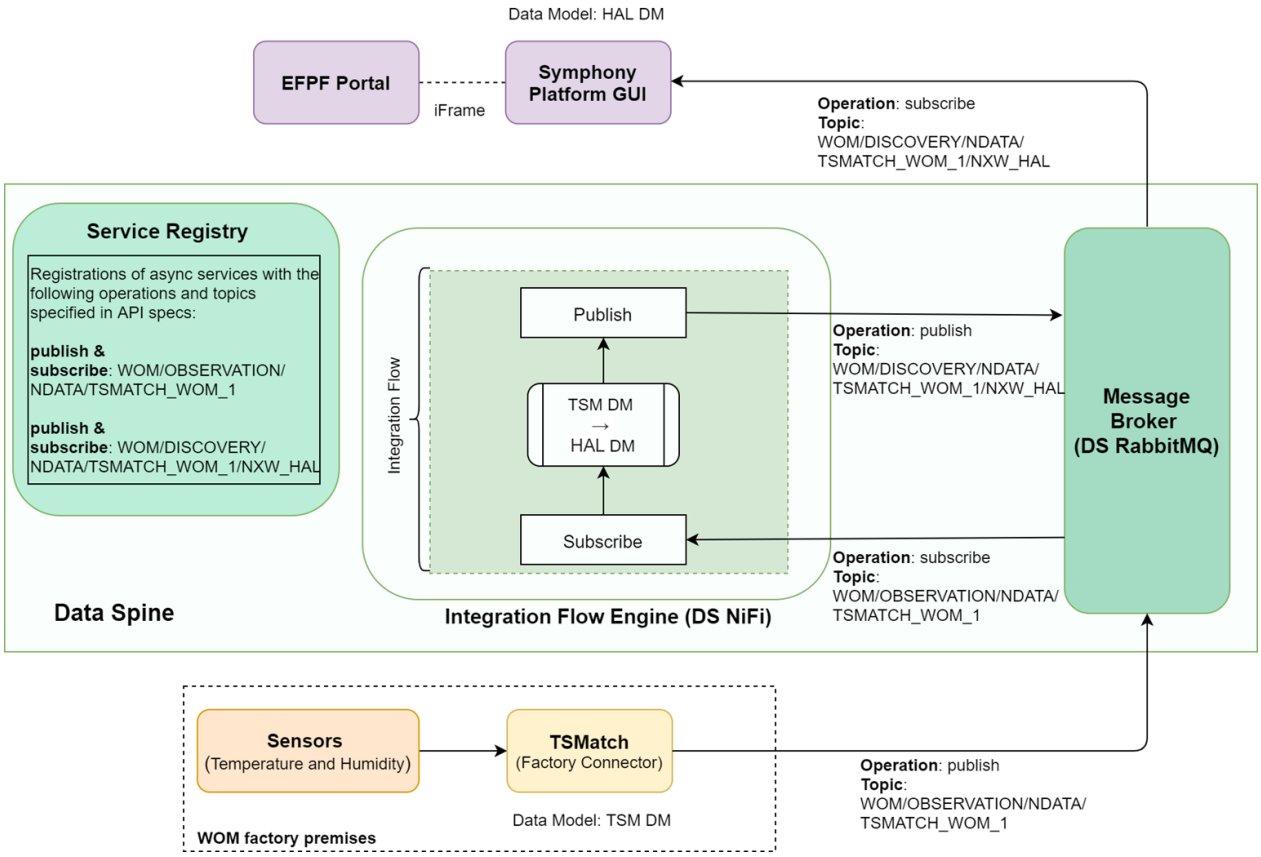


Figure 10 Workplace Environment Monitoring Dataflow 1: Sensing

TSMatch has also been used in the context of WP8, T8.4, unfunded experiments, as one of the tools that has been validated in the EU-IoT/EFPF Hackathon. The target community on this case related with students, and researchers, and the purpose was to assess the use of TSMatch.

FOR is also using TSMatch as an example of an Edge-based application that can be run in embedded devices on other projects, such as the C4AI Edge project[5] (cooperation with IBM), where fortiss is working on dynamic container orchestration.

2.1.3.3 New Developments & Roadmap

During this period, the following aspects have been developed:

- Design, implementation, and release of TSMatch v2.0 (semantic matchmaking based on ML).
- Revision of the whole code and interconnection with EFPF.
- Setup of TSMatch with support for MQTT Sparkplug.
- Revision of the TSMatch documentation in EFPF.
- Development of a series of tools, including a standalone TSMatch kit, to be transported to different demonstrations.
- Support on pilots, including installation of hardware and testing of TSMatch.

[5] https://www.fortiss.org/en/research/projects/detail/c4ai-edge

- Setup of a local TSMatch demonstrator on the fortiss IIoT Lab.

The TSMatch development phase has been successfully concluded as planned. Until M48, the following aspects will be worked:

- Interconnection of the fortiss IIoT Lab to EFPF via MQTT (WebSocket's).
- Adaptation of the application to allow remote monitoring of customer premises via the EFPF data spine.
- Code improvement, to allow for a more efficient use of TSMatch
- Support in running pilots.
- Dissemination, including demos in technical events.
- Performance validation (documented in scientific publications).

2.2 Digital Tools for Smart Factory scenario

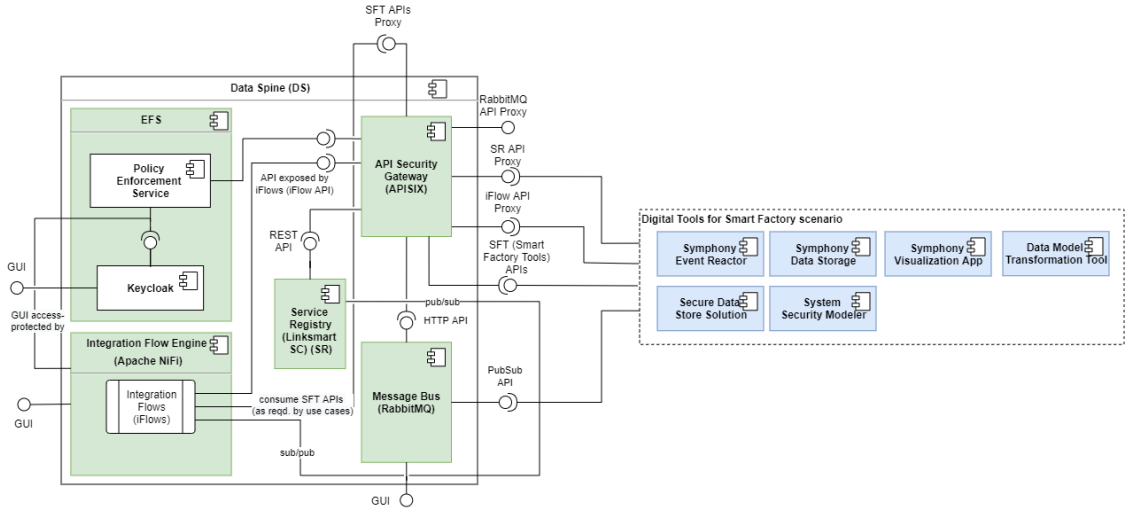


Figure 11: Interaction between the Digital Tools and the EFPF Data Spine

2.2.1 Symphony Event Reactor

2.2.1.1 Functional Description

The Symphony Event Reactor allows the integrator to define a behaviour in response to specific ingress events, by taking actions and potentially engaging an alarm lifecycle management process. Actions include pre-configured low-level procedures, custom code provided by the integrator, or generation of new events.

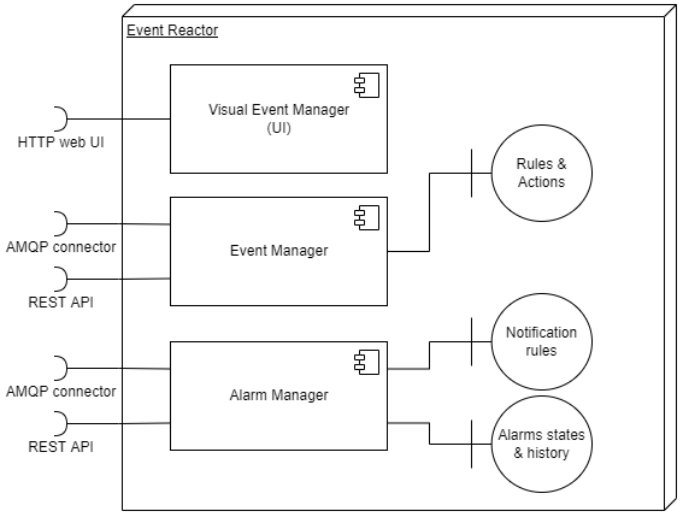


Figure 12: The Symphony Event Reactor internal architecture

From an architectural point of view, the Event Reactor is split into three separate independent services:

- **Event Manager** (EM): the component that executes custom rules, combining information coming from different sources (local sensors and device monitors, user

actions, CCTV cameras, intrusion detection systems, etc.) and message brokers (e.g., AMQP, MQTT) to determine actions to be taken. Actions may include field devices actuations, activation of scenarios, generation of events, notifications and alarms, and so on.

- **Alarm Manager** (AM): the component that reacts to *alarm conditions* to generate new alarms and update the status of existing ones. Correspondingly, it notifies users of such changes through a configurable priority based alert routing system to target a single, a group or mixed sets of users with SMS, emails, pop-ups, etc.
- **Visual Event Manager** (VEM): the component that exposes to the integrator an intuitive web UI based on visual block-level rule definition interface.

2.2.1.2 Usage and Deployment

The Event Reactor is released, installed, and configured (based on the customer requirements) by Nextworks.

It is currently involved as part of the Cloud deployment of the Environmental Use Case pilot scenario. It continuously monitors the status of a set of sensors connected to the TSMatch gateway[6] based on a set of rules configurable by the user and it reacts through the actuation of some sirens and lights connected to the Symphony HAL.

2.2.1.3 New Developments & Roadmap

The Symphony Event reactor is continuously improved and optimized by Nextworks to support new actuations and analytics functions on the sensors supported by the platform.

2.2.2 Symphony Data Storage

2.2.2.1 Functional Description

The Symphony Data Storage component provides a solution to store large amounts of data and perform basic aggregation, retention, and propagation operations.

Nextworks Data Storage is a highly scalable and high-performance data storage which is designed to handle large amount of AMQP/MQTT data. It offers aggregation, rate limiting and sub-sampling, configurable data retention policies and synchronization across multiple instances. It accepts AMQP and MQTT as data source input and provides REST as output.

[6] https://docs.efpf.linksmart.eu/projects/factory-connectivity-smart-factory-tools/ds-tsmatch-gateway/

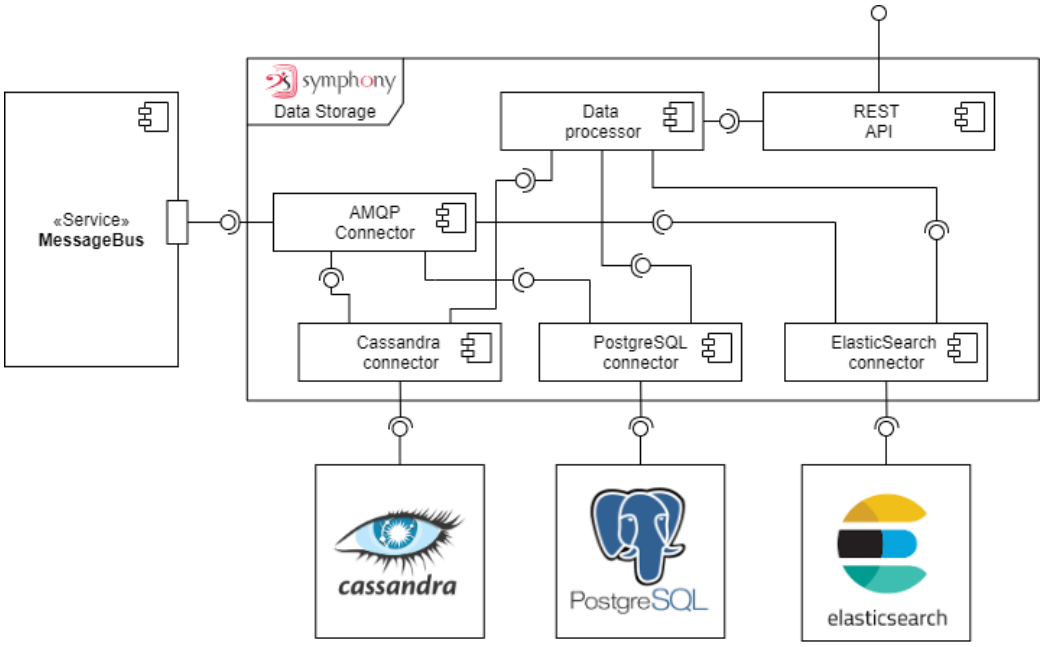


Figure 13: The Symphony Data Storage internal architecture

2.2.2.2 Usage and Deployment

The Symphony Data Storage will be provided both as a deployable component and as a managed service.

To configure the system, after running the docker container user can configure the Symphony Data Storage through its configuration shell. User can define back-ends (Apache Cassandra, PostgreSQL, and Elastic Search) and AMQP/MQTT data sources for subscribing. Also, the user should define sufficient resources based on the back-end(s) and the rate of incoming data.

After configuration, data immediately begin to flow to the Data Storage and user(s) can query data through REST interface.

It is released, installed, and configured (based on the customer requirements) by Nextworks.

2.2.2.3 New Developments & Roadmap

The Symphony Data Storage is continuously updated and evolved by Nextworks to improve its performances and to implement new cutting-edge features and keep it aligned with the state of the art.

2.2.3 Secure Data Store Solution

2.2.3.1 Functional Description

Secure Data Store Solution (SDSS) is intended to record timeseries data from Factory Connectors, and IoT Gateways, although it can be used for any utility that sends data over the Message Broker. Secure Data Store Solution is than able to provide that stored data is then available for later analysis tools. One key advantage of this is that analysis tools can be trained with historical data. Additionally, data can be queried from the SDSS for display on graphs for manual analysis.

SDSS uses the EFPF Keycloak instance to handle user verification.

SDSS acquires data by observing messages sent through the message broker. Users can use the SDSS web UI to configure which channels to monitor, which fields of messages to extract, and how to store the data as timeseries data. Logged in users can grant access to other EFPF users.

SDSS provides analysis tools the means to access data through an OGC SensorThings interface. It can additionally provide a simple JSON query interface for web apps, that need to be able to quickly get a direct, limited data set for on-demand display on devices.

2.2.3.2 Usage and Deployment

A centrally managed instance of the SDSS is currently hosted on the SME Cluster at https://sdss.tools.smecluster.com/. This allows for a reference service which can be accessed by partner's services. This instance is being used to record data from Lagrama and Innovint.

A core tenet of the SDSS is to be hosted onsite by organizations, allowing those organizations to maintain physical control over their data. To facilitate this, Docker images are provided, allowing for the rapid deployment of SDSS instances.

SDSS is open source, under the Apache License.

2.2.3.3 New Developments & Roadmap

Usage	Description
SSO Integration	SDSS has been adapted to utilize the EFPF SSO Server for user authentication and management.
Pub/Sub Security integration.	SDSS has been integrated with the Pub/Sub security service, allowing users to seamlessly access Message Broker Channels to which they have been given access.
Bulk JSON/CSV Export	Clients can make bulk data requests to extract data in JSON or CSV formats.
Simple pageable query API	Clients that require small amounts of data at a time in basic formats can use this API to retrieve the desired data.
OGC SensorThings export interface.	Clients that can extract data from the SDSS in the OGC SensorThings format.
Authorization for Database access to other EFPF users.	To share data, users can grant access to other users to access to their data.
Enhanced configuration dialog for Timeseries	Sample/store raw MQTT messages for configuration examples, allowing to highlight JSON path specified within message. The data stored can either be the value of the JSON path, or part of the keypath for instances when many attributes of the message may be interesting.
Deployable with variable paths, for deployments behind a reverse proxy	Administrators deploying the SDSS on site can specify the path the SDSS's single page application web UI should respond to.

Provide Docker image for standalone deployments	Integrated "standalone" deployment image, combining required databases, for simple deployment situations
Provide Docker image for demo mode	A "trial demo" variant of the standalone deployment, which can be directly launched with no persistent storage

Work is still ongoing to provide a standalone Docker image variant with integrated configuration for on-disc encryption, to provide a high level of confidence in the security of the data with minimal configuration.

2.2.4 The System Security Modeler

2.2.4.1 Functional Description

The SSM provides a graphical user interface where ICT managers, CTO and CEO can draw the system and evaluate the risks considering their impact on the business.

The main feature is the ability of assessing the knock on effect of a materialised threat on the system and on the business processes.

The tool follows the workflow defined in the ISO27005[7] standard. It performs design-time ICT based risk-analysis and compliance check. It performs knowledge-based reasoning and pattern identification over system models defined by the end-users (see Figure 14). The asset models represent the topology of the physical (the infrastructure), logical (the processes) and dataflow (the data) layers.

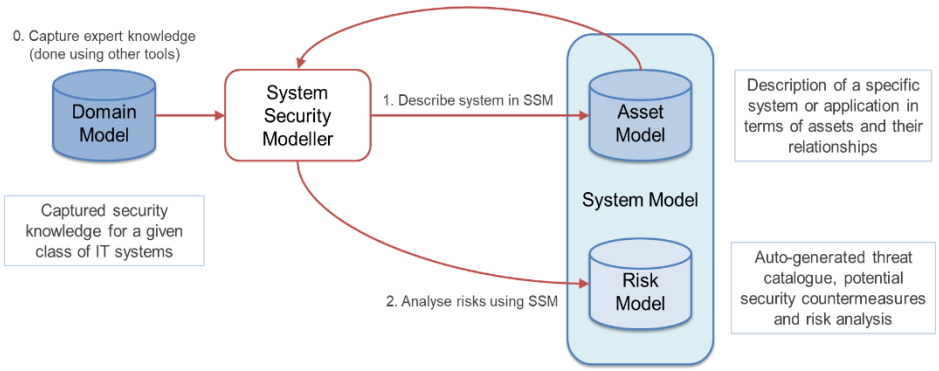


Figure 14: Overview of SSM Data Models

The main functions of the SSM tool are:

- System model management and system model construction.
- System analysis and threat identification. This includes potential regulatory compliance issues (compliance threats)
- Risk level analysis
- Navigation of attack paths and secondary effect cascades
- Compliance threats and modelling errors

[7] ISO/IEC 27005: 2018 – Information Security Risk Management https://www.iso.org/standard/75281.html

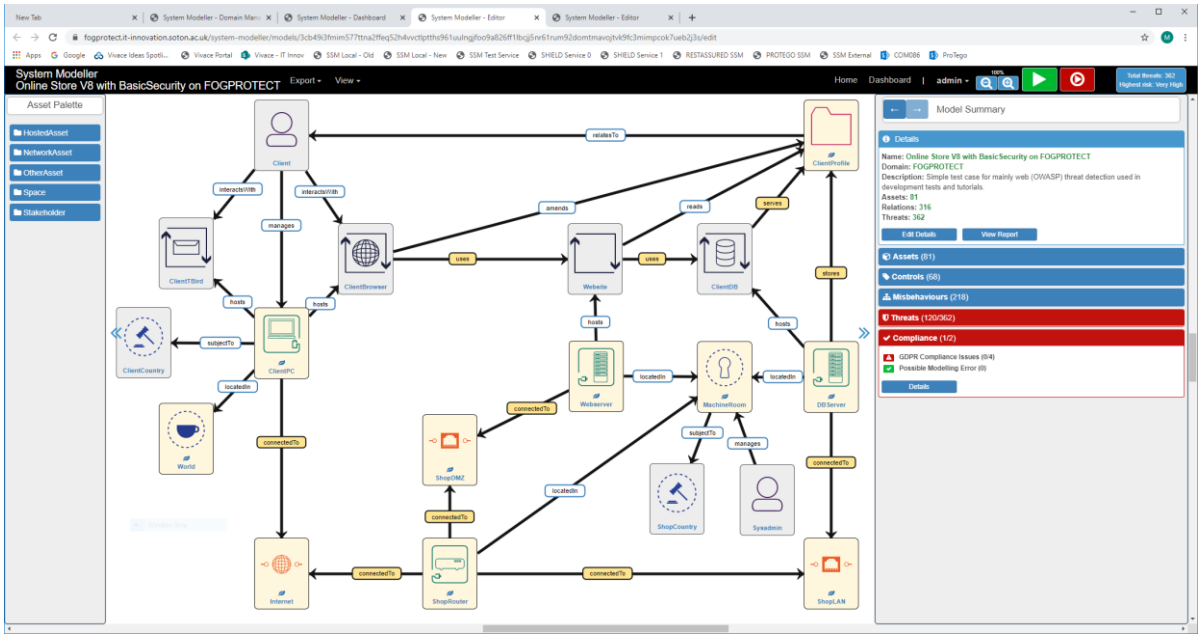


Figure 15: SSM screenshot

The tool will be used to assess the existing use case from an ICT risk level point of view and to assess the level of risk associated with the inclusion of new services in the existing Factory Ecosystem chains. The regulatory compliance check will be used to validate scenarios where the GDPR legislation is relevant and should be applied.

2.2.4.2 Usage and Deployment

The SSM has been used in the Circular Economy pilot in EFPF and to test the general EFPF core services.

It is offered as a service and reachable through the EFPF marketplace.

2.2.4.3 New Developments & Roadmap

The SSM has its own roadmap which goes beyond the EFPF project.

It includes a better integration and support for the business slayer (e.g., the importing of BPMN models to super-impose on the currently analysed data-flows). It also foreseen the improvement of the user interface for the definition of the so-called domain models (the knowledge base underpinning the risk assessment).

The SSM available in the EFPF portal covers the design time, but it is complemented by the ability of interact with monitoring systems for a run-time and near-real-time risk assessment. There are current plans to improve and evolve this part to provide a comprehensive tool that covers both the design time and run-time risk assessment.

2.2.5 Semantic Information Management Tool

2.2.5.1 Functional Description

The Data Spine follows Federated Interoperability approach specified by the CEN/ISO 11354 Framework for Enterprise Interoperability which means that there is no common

canonical data model imposed at the ecosystem level. The Smart Factory platforms and tools are free to choose any data model that suits their needs and still be a part of the ecosystem.

The interoperability among tools is enabled on-demand, when required by a use case, through data transformation. This is achieved by creating integration flows using the Data Spine's Integration Flow Engine (DS NiFi). A typical integration flow as illustrated in Figure 16 consists of a protocol connector (e.g., InvokeHTTP, ConsumeMQTT, etc) to consume Service Provider's API, a data transformation processor (e.g., JoltTransformJSON, TransformXml, etc.) that aligns the data model to the Service Consumer's data model and finally, another protocol connector that makes the data available to the Service Consumer.

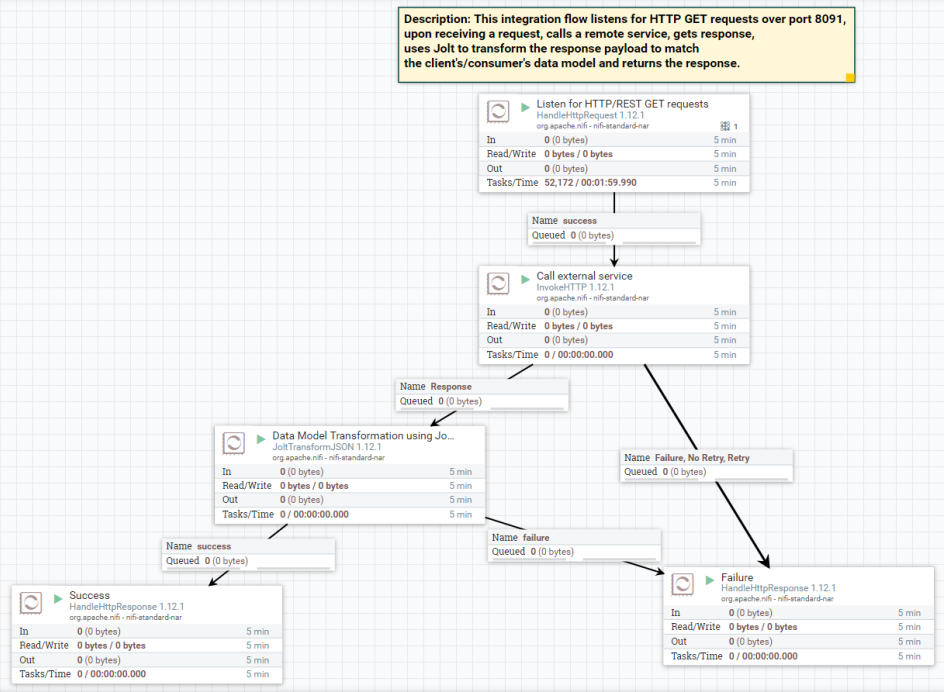


Figure 16. Illustration of an integration flow

In the current version of the Data Spine, although DS NiFi provides an intuitive GUI, service composition still largely remains a manual process performed at a lower abstraction level. I.e., the developers need to manually configure the processors to enter the protocol and endpoint specific details and write the data transformation logic (Figure 16). The Semantic Information Management (SIM) tool makes use of Semantic Web technologies to add embedded intelligence to the Integration Flow Engine in order to simplify the service composition process further. Figure 17 illustrates the high-level architecture of the SIM tool.

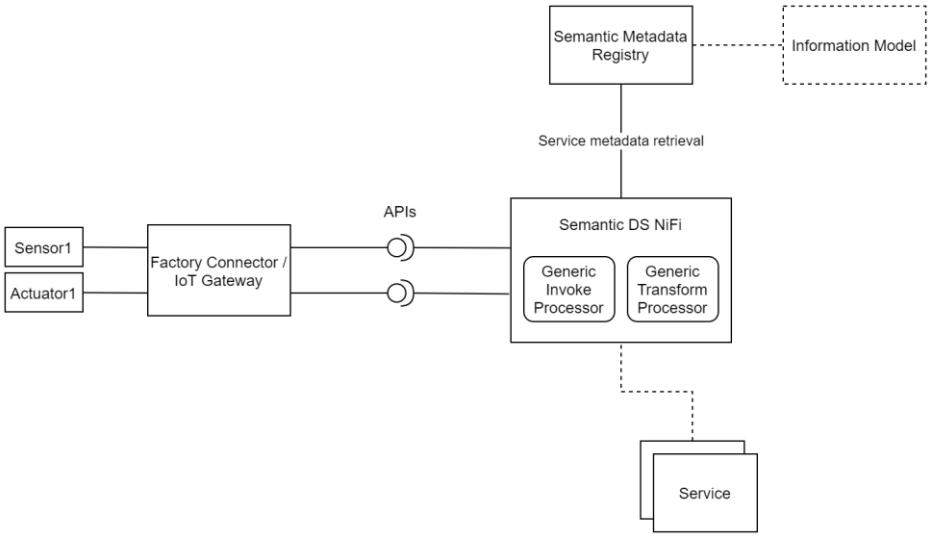


Figure 17. High-level architecture of the SIM tool

The proof-of-concept for the SIM tool makes use of the Web of Things (WoT) Thing Description (TD) standard to specify the metadata i.e., the TDs for the service APIs provided by tools such as the Factory Connectors. The Semantic Metadata Registry provides an API for the lifecycle management and discovery of the TDs. The SIM tool adds semantic support to DS NiFi in the form of two custom NiFi processors: (1) Generic Invoke Processor and (2) Generic Transform Processor. The Generic Invoke Processor provides the functionality to invoke an API and retrieve data by specifying its identity information such as its ID from the Thing Directory, thereby enabling the users to work at a higher abstraction level of services rather than dealing with the low-level details such as protocols and API endpoints. Secondly, the Generic Transform Processor can be used to automatically calculate the data transformation logic based on the service metadata captured in the form of the TDs. Thus, the SIM tool enables the developers to work at a higher level of abstraction, thereby simplifying the service composition process even further.

2.2.5.2 Usage and Deployment

The current version of the Data Spine was used successfully to integrate heterogeneous Smart Factory platforms and tools from various independent providers into the EFPF ecosystem and interoperability was enabled using integration flows for pilot use cases as well as for Open Call experimentation scenarios. In the Data Spine Usage Surveys launched for the pilots as well as the Open Call experimenters, the usability of the Data Spine was rated between easy and moderate. The SIM tool was designed to simplify the service composition process even further by making use of the Semantic Web technologies.

In the proof-of-concept implementation of the SIM tool, the Semantic Metadata Registry is realised using the LinkSmart Thing Directory [LTD22] which is available as an Open-Source project on GitHub. It is an implementation of the W3C WoT Thing Description Directory (TDD) as specified in the WoT Discovery specification [WoTD22]. The Thing Directory provides a RESTful API for the provisioning and life cycle management of TDs. It also offers a Search API that supports multi-parameter queries. Figure 18 illustrates an example of a sample TD for a sensor ‘efpf:sensor1’ that provides the current ambient temperature at the factory ‘efpf:factory1’. It also describes the API endpoint to fetch the temperature information and the timeseries data model followed by sensor1. The SIM tool needs the data model attributes to be semantically annotated using the ‘@type’ JSON-LD keyword. Figure 18

illustrates a sample TD for an ‘efpf:dashboard1’ service that accepts incoming timeseries data that follows a different data model and provides a display service for this data.

```
{
    "@context": [
        "https://www.w3.org/2022/wot/td/v1.1", {
            "efpf": "https://efpf.org/dpe#",
            "saref": "https://saref.etsi.org/core/",
                            "bimerrw": "https://bimerr.iot.linkeddata.es/def/weather#",
                            "schema": "http://schema.org/"
        }
    ],
    "id": "efpf:sensor1",
    "schema:location": "efpf:factory1",
    "title": "Temperature sensor at factory1",
    "description": "Temperature sensor that provides the current ambient temperature at
factory1",
    "securityDefinitions": {
        "nosec_sc": {
            "scheme": "nosec"
        }
    },
    "security": "nosec_sc",
    "properties": {
        "temperature": {
            "type": "object",
            "readOnly": true,
            "properties": {
                "timestamp": {
                    "@type": "saref:hasTimestamp",
                    "type": "string"
                },
                "value": {
                    "@type": "bimerrw:Temperature",
                    "type": "integer"
                }
            },
            "forms": [{
                    "href": "https://efpf-sensor1-API-endpoint",
                    "contentType": "application/json",
                    "htv:methodName": "GET",
                    "op": "readproperty"
                }
```

```
            ]
        }
    }
}
```

Figure 18. Web of Things (WoT) Thing Description (TD) Example with Property Affordance

The Generic Invoke Processor and the Generic Transform Processor are implemented as custom NiFi processors 'SIMGenericInvokeProcessor' and 'SIMGenericTransformJoltProcessor' respectively and the source code is available in the EFPF Gitlab group. The processors can be built as maven projects and added to the '/lib' directory of NiFi deployment. The NiFi instance does not have to be rebuilt but it needs to be restarted. Once the NiFi instance restarts, the processors can be searched on the 'Add Processor' page as illustrated in Figure 19 and added to the NiFi canvas for creating integration flows.

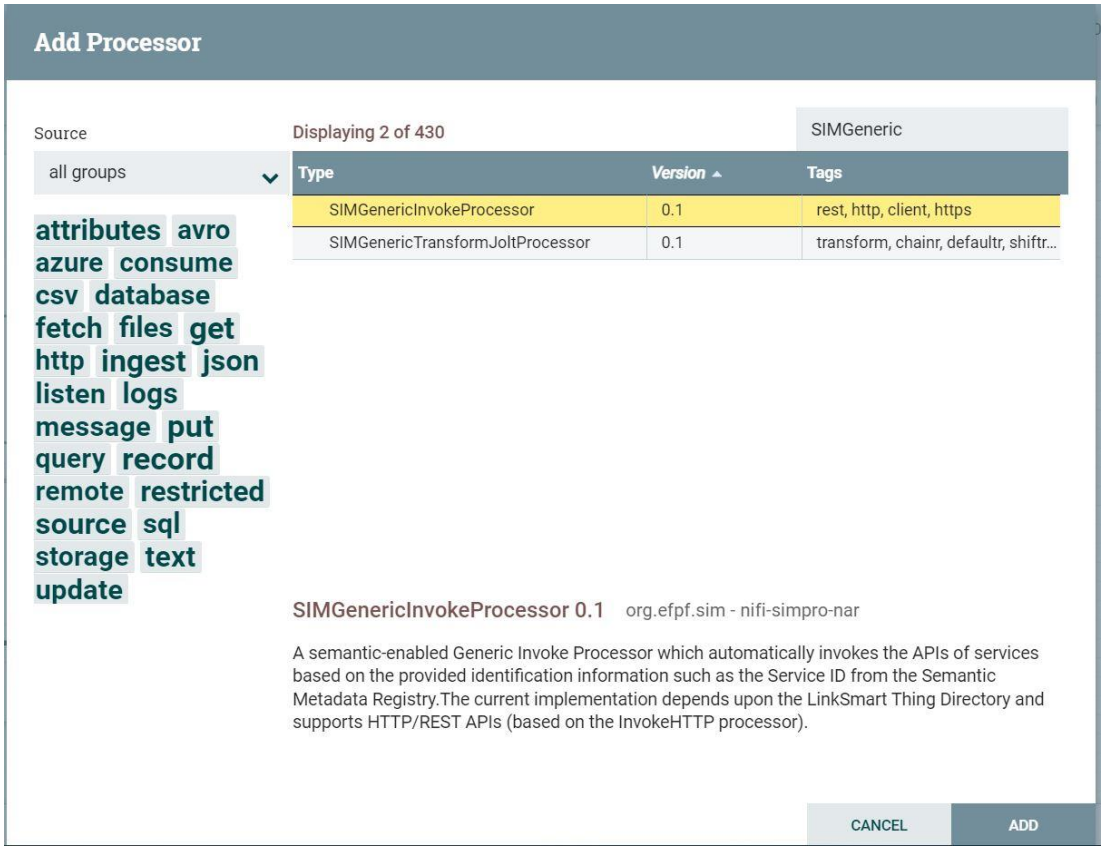


Figure 19: Searching the SIM Processors in NiFi

The SIMGenericInvokeProcessor takes 'Service Id', 'Affordance Type' and 'Affordance Name' as inputs from the developer. Based on this information, it queries the Thing Directory the service metadata that is needed for invoking the API endpoint. At run-time, it invokes the endpoint and makes the fetched data available to the subsequent processors in the integration flow. The current implementation is based on the InvokeHTTP processor and supports invocation of HTTP/REST APIs.

The SIMGenericTransformJoltProcessor, at design-time, accepts 'Service Id', 'Affordance Type' and 'Affordance Name' for the source (input) and destination (output) services as inputs from the developer. It then queries the Thing Directory to fetch the data models of these services. The processor makes use of the semantic annotations associated with the

data model attributes to automatically create the Jolt transformation spec and adds it as the value of the 'Jolt Specification' property of the processor. This value is accessible to the developer and can be updated, if required. The current implementation supports automated creation of Jolt transformation rules for simple data models that do not contain arrays and nested objects. At run-time, the processor accepts the input data model payload, applies the generated Jolt transformation rules using the JoltTransformJSON processor library in NiFi and makes the transformed data model available to the subsequent processors in the integration flow.

Figure 20 illustrates an integration flow created using the above-mentioned SIM processors. The first instance of the SIMGenericInvokeProcessor calls a service such as the 'efpf:sensor1' API (Figure 18) to fetch timeseries data, the instance of the SIMGenericTransformJoltProcessor in the middle automatically creates and applies the Jolt transformation rules to this data and transforms the data as per the destination services' data model, here 'efpf:dashboard1' (Figure 20). Finally, the second instance of the SIMGenericInvokeProcessor POSTs the data to the 'efpf:dashboard1' service which then displays it.

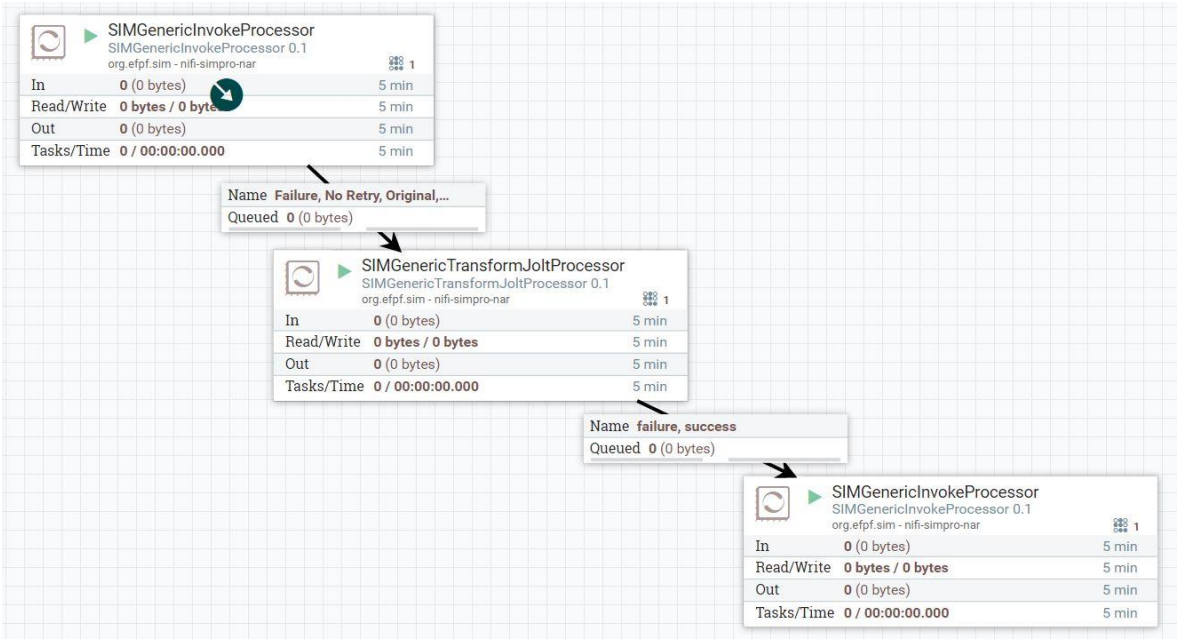


Figure 20. An integration flow created using SIM processors

2.2.5.3 New Developments & Roadmap

The SIM tool provides a roadmap for the further enhancement of the Data Spine using Semantic Web technologies. The proof-of-concept implementation of the SIM tool illustrates that the process of service composition can be simplified even further using semantically annotated metadata.

In addition, the use of semantics creates the possibility of enhancing the Data Spine further to capture service usage and pairing information which can be analysed to understand usage trends and to improve the functionalities such as recommendations. For instance, if a user is invoking a service API that provides sensor data, the Data Spine can be enhanced to recommend the usage of services such as timeseries storage, visualization dashboard or real-time analytics, etc.

The plans for the further development of the SIM tool are mentioned in the table below.

Usage	Description
SIM processors' property configuration	Enhancement of the SIM processors to automatically fetch the service identification information from the Thing Directory and provide autofill functionality to the developers.
Generic Invoke Processor	Add support for fetching Event Affordance data.
Generic Transform Processor	Add support for more complex data models that include nested objects.

2.3 Collaboration Tools

This section describes the digital tools in the EFPF ecosystem that are specifically designed to enable and support collaborations between different entities. These tools make use of different technologies with the aim to support secure interactions and data exchange between collaborative parties in a business network or supply chain.

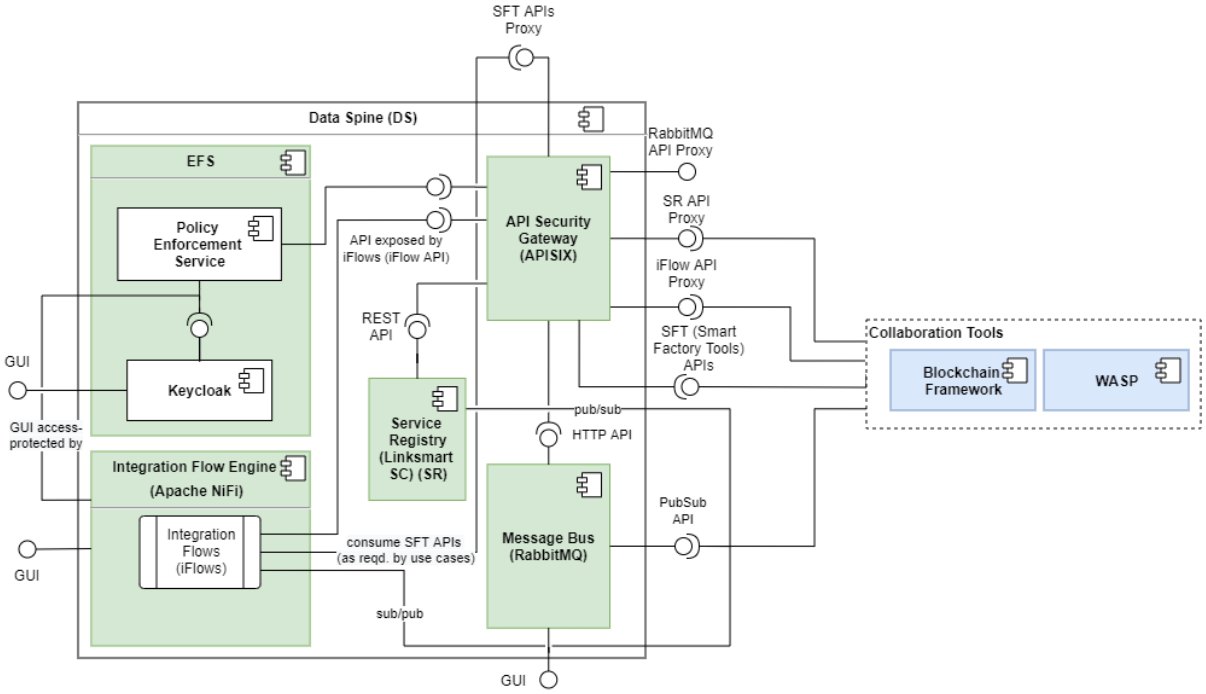


Figure 21: Interaction between the Collaboration Tools and the EFPF Data Spine

2.3.1 Blockchain Framework

2.3.1.1 Functional Description

The EFPF Blockchain and Smart Contracting services provide the means for applications in the EFPF Ecosystem to incorporate blockchains and smart contracts for audit trails for manufacturing and supply chain data, product data traceability and smart contracting agile networks. It offers integration of blockchains and smart contracting in three service levels:

- NIMBLE Blockchain: Integrated process traceability for users of the NIMBLE Platform. The NIMBLE Blockchain is not accessed from an API but integrated into the NIMBLE Platform.
- Blockchain As a Service (BaaS): A blockchain service API for managing identities, schemas, and data, providing the means to build audit trails for manufacturing and supply chain data, product passports, identity management and other immutable transaction logging applications.
- DAML Integration: to build systems with secure multi-party transactions and smart contracting agile networks, DAML applications can be defined and deployed for the EFPF Ecosystem, integrated with the EFPF Security Portal (EFS).

The Blockchain and Smart Contracting services hide the details of writing code for transaction families, dealing with cryptography and signing messages directly against the ledger. Hyperledger Sawtooth and Hyperledger Fabric provide the ledger implementations for the services. These are open source, private and permissioned distributed transaction ledgers [D5.2: EFPF Security and Governance] suitable for EFPF applications. The architecture is illustrated in Figure 22.

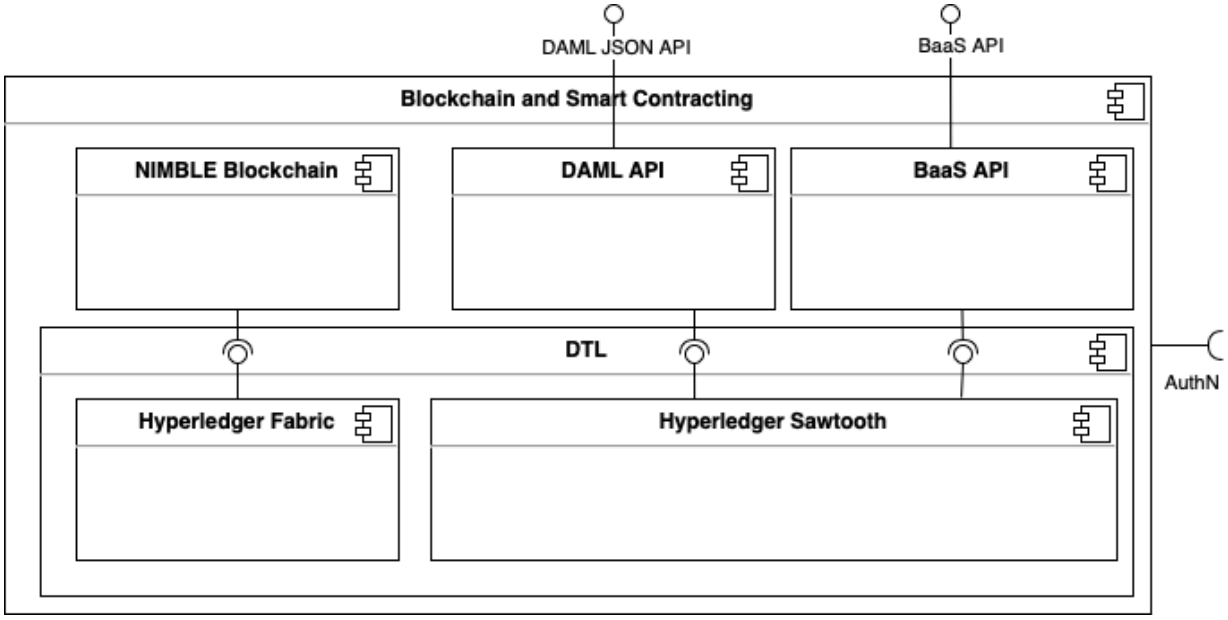


Figure 22. Architecture of Blockchain and Smart Contracting Services

2.3.1.2 Usage and Deployment

The usage model and deployment of the three service levels are different. The NIMBLE Blockchain is deployed and used as part of the NIMBLE platform. The BaaS APIs are deployed as standalone components, currently on CERTH servers. The DAML smart contract applications are deployed to Azure cloud servers hosted by CNET. The DAML API, EFS integration and ledger node are dockerized. Deployment of smart contract applications is done by uploading them to the DAML ledger via the API. The Pilots use separate servers, whereas the Open Call projects use one server - several DAML applications can be co-hosted and interoperate on one DAML server, and several DAML servers in different domains may be connected. This interoperability is the major design goal and feature of the DAML stack. The Hyperledger blockchain nodes are hosted by SRFG, CNET and CERTH.

The Blockchain and Smart Contracting Service was utilised in four EPFP pilots within Circular Economy, Furniture and Aerospace domains – its applications in these pilots are described below. It was also used in Open Call experiments. Hyperledger and DAML are available under Apache 2.0 license.

Usage	Function	Description
Aerospace Pilot	Material Track & Trace During Lifecycle	When a customer reports defect in a supplied product or a manufacturer of a material used in the product reports a defect, quality management employees can identify other affected products so that reliable track and tracing is possible and follow-up actions can be initiated accordingly (e.g., inform other customers). (DAML)
Aerospace Pilot	Secured Logistic Chain	Tamper-proof storage of information in a distributed ledger and smart contracts governing the transfer of freight/packages ensures compliance and proper monitoring of the relevant requirements. The goal is to ensure that wherever a transfer of freight/packages between persons/ companies takes place, that the freight/packages are the same and have not been manipulated and relevant persons are identified/authorized, and their data are recorded. (DAML)
Furniture Pilot	Delivery Process Monitoring	A POC DApp tool in the form of a mobile application that enables tracking the delivery process by implementing the Blockchain approach. The process workflow is defined through the backend, and the actors are defined to execute the process. (COMPOSITION BC)
Circular Economy Pilot	Blockchain Verification to Improve Transportation Security and Reliability	The EFPF solution related to the blockchain verification provides full supply chain visibility and improves transportation security and reliability through updated, secure, and reliable data that enables efficient decision making. (BaaS)
Circular Economy Pilot	Blockchain Traceability to Improve Closed Loop Supply Chain Management	The lack of capability to track and trace assets and transactions throughout the entire supply chain, is one of the key challenges that the three companies are facing towards circular economy. This lack of traceability is hindering companies to quickly adapt, plan, and manage their assets in an effective and optimised way. (BaaS)

Open Call Experiment	Digitanimal	BaaS Supply Chain as a Service API used to store farm-to-fork chain information for livestock and meat traceability. (BaaS)
Open Call Experiment	Digiotouch	DAML contracts for secure, interoperable, end-to-end service for lot-size-one furniture production. (DAML)
Open Call Experiment	DNET Labs	BaaS Supply Chain Logging Service used for storing predictive maintenance results in a transparent and tamper-evident way. (BaaS)
Open Call Experiment	Artificial Intelligence Talentum	BaaS Supply Chain Logging Service used in the validation stage for the results of the calculations. (BaaS)

2.3.1.3 New Developments & Roadmap

The Blockchain and Smart Contracting Service has developed new features of since the M18 deliverable:

Usage	Description
BaaS Supply Chain as a Service API	The BaaS provides three general-purpose smart contracts, or in Hyperledger Sawtooth terminology, transaction families, that allow an EFPF application developer to define customized blockchains for transaction logs, audit trails and transportation involving multiple organizations: the Schema Transaction Family, the Identity Management Transaction Family and the Track and Trace Transaction Family. The Supply Chain as a Service API is used in Pilots and Open Calls to build applications on top of these smart contracts.
EFS integration	Blockchain identities connected to the EFPF EFS identity management.
General smart contract development	Support for DAML smart contract language, dockerized DAML ledger deployment and API deployment of smart contracts.
Built in blockchain support for NIMBLE	The NIMBLE Logistic Contract provides audit trail, data provenance and verification for the trading & logistics data exchanged between trade partners in NIMBLE platform

The BaaS was functionally complete yearly in the project, with new features, APIs and UI improvements being added continuously. The NIMBLE blockchain is included with the NIMBLE platform and available as open source but has no feature roadmap at the time of writing. The DAML Pilot for Secured Logistic Chain is being further developed until the end of the project to be more feature complete and EFPF support for the DAML stack is planned to include DAML 2.0 and the ledger interoperability protocol Canton.

2.3.2 Distributed Workflow and Business Process Design and Execution

2.3.2.1 Functional Description

The Workflow and Service Automation Platform or WASP enables fast automation of processes allowing users to describe and execute business processes and compose services in a visual way, to model multiple manufacturing workflows to orchestrate the various assets available within a collaborative framework.

WASP is comprised of 2 logical components:

- Orchestration Designer: The Process Designer is a visual online reactive canvas allowing a business process designer to pull in existing BPMN models from a library representing the virtualised manufacturing assets, as well as integrating external services/APIs registered in the WASP Service Directory that facilitate the execution of production activities or the interconnectivity of supply chain operations. The Form Designer allows the user to build a form to enter process information. The forms can be a single use form or a form template that is attached to/associated with a User Task element in the BPMN process from within the Process Designer.
- Orchestration Runtime: This is based on the open source BPMN Camunda 'process-engine' and provides a Control Panel UI that allows for the management of process deployments and of individual process instances, as well as an Instance View UI that displays the real-time status of a process instance & process variables. A Tasks UI gives access to User Tasks assigned to the specific user, that may require simple verification of a process step, or the entering of specific process information (via a Form specifically built in the Form Designer) that is required for downstream process steps and which may be used to control the flow of the process.

2.3.2.2 Usage and Deployment

EFPF WASP is available through the EFPF Portal where it is integrated with the EFPF Keycloak security component that facilitates Single Sign On (SSO) to WASP as a EFPF Portal user. It is hosted on the ICE Production Server (https://efpf.icelab.cloud). WASP is not currently available on a commercial license basis.

The WASP component was utilised in the following projects:

- EFPF funded open-call sub-project - ebWASP

  The eBusiness eXpert project aims to provide an integration of the eBin platform, which focuses on document collaboration in supply chains, with the WASP platform which is provided by EFPF. The sub project aimed to demonstrate workflow automation of the eBin processes through WASP, whilst also allowing eBin functionality to be integrated within the WASP platform. As a proof of concept (POC), 2 processes were developed (Seller & Buyer) that integrated secure ebuisiness API services & demonstrated the potential for eBusiness eXpert to test alternative workflow technologies for use within the eBin platform.

- EFPF consortium partner pilot project - Innovint

  Innovint Aircraft Interior GmbH is developing and manufacturing cabin interiors and has established a leading position in the market with innovative products. IAI is participating in the EFPF project to experiment with the latest supply chain solutions and to potentially

adopt the innovations available in WASP to enhance their supply and production processes.

2.3.2.3 New Developments & Roadmap

There are ongoing discussions to make external/Cloud services/APIs available to WASP through the EFPF Marketplace & thereby provide a means to monetise the use of these services.

Work is currently on-going to deploy WASP as a standalone instance for example, on a company intranet or in the Cloud. This deployment will come in 2 flavours - either a dockerized WASP or a more robust Kubernetes solution. As part of these developments, there will be a new Keycloak component as the user management & authentication tool.

Discussions with both eBusiness eXpert and Innovint about using WASP beyond the lifetime of the EFPF project. These discussions have involved extending WASP functionality:

- Create a library of BPMN process templates that provide a range of collaborative workflows integrated with services/APIs available through the EFPF Marketplace.
- A dictionary component that replaces the technical terms in the default WASP instance UIs with words/phrases more recognisable/meaningful to the nature of the client business.

2.4 Industry 4.0 Tools

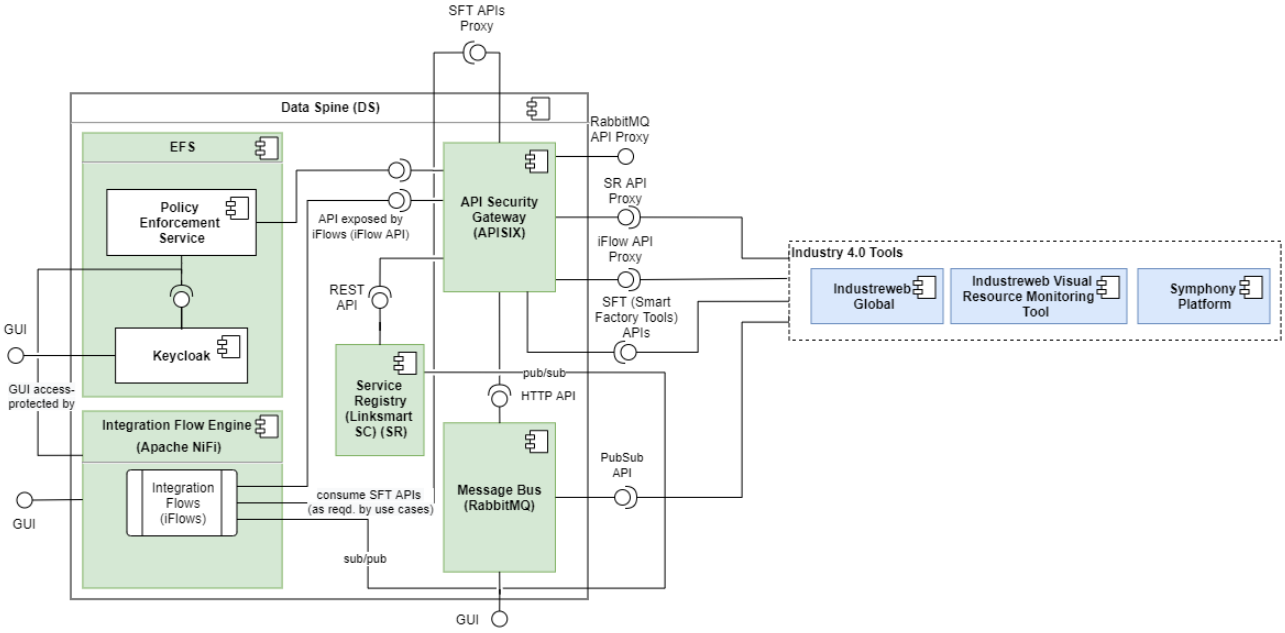


Figure 23: Interaction between the Industry 4.0 Tools and the EFPF Data Spine

2.4.1 Industreweb Global

2.4.1.1 Functional Description

Industreweb Global (IW Global) is a web framework that provides data visualisation, storage, workflow co-ordination, as well as Administration and Security Management tools for the

Industreweb Ecosystem. In Figure 24, the main elements and how they interact with Industreweb Display screens and Industreweb Collect Factory Connectors are shown. This section will explain what Industreweb is able to offer from a data visualisation and an application perspective

IW Display and IW Vactory are two UI tools within IW Global for visualising data which typically comes from data made available by IW Collect. Both Tools will be described below:

IW Display: IW Display is a browser-based visualisation tool that is designed to display production information and statistics from the Industreweb Collect server such as:

- Production Faults
- Production Performance
- Waste
- Historical Reports

Data in the Industreweb Collect Engine is subscribed to by tag enabling it to be updated in near real-time. This means that Dashboards or monitoring screens can be developed using responsive html templates in order to present these values on a wide range of devices.

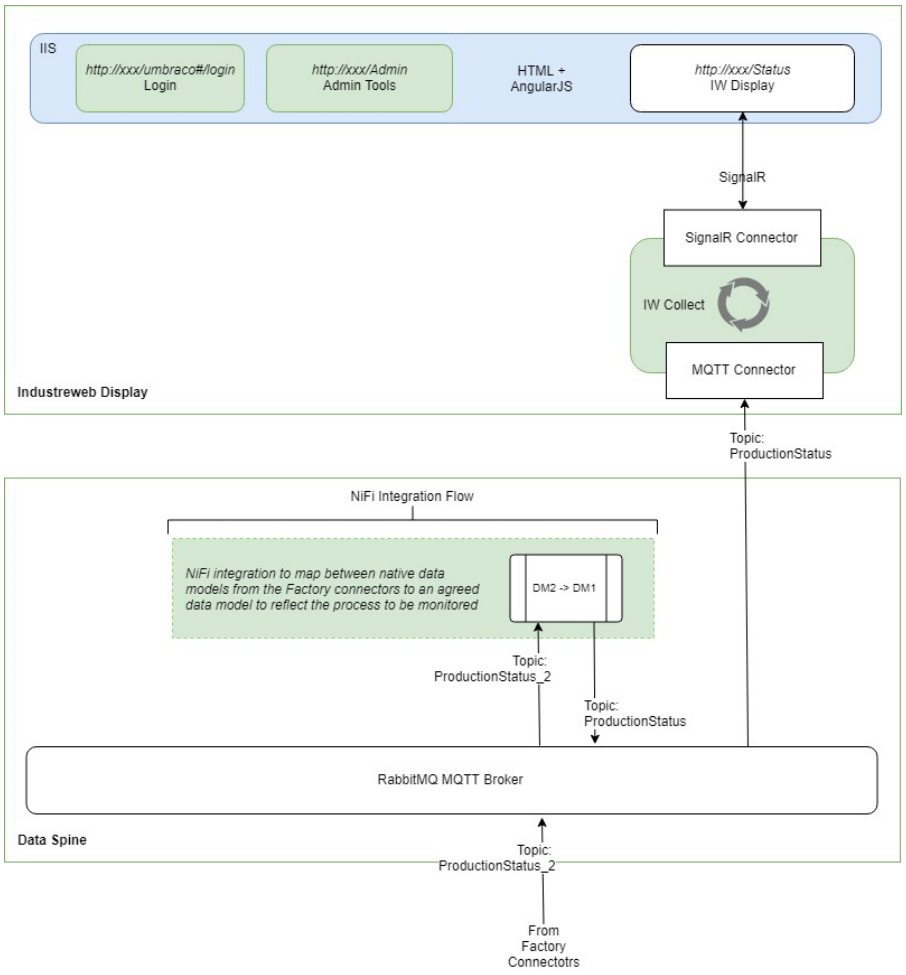


Figure 24: Industreweb Display Architecture

Industreweb Vactory: IW Vactory shares the same architecture as IW Display but makes the data available in a WebGL 3D UI that maps data points to a digital twin of the production

assets. Users can provide 3D models of production assets or use built in common models and map attributes of the model, such as the position or rotation of parts of the model or the colour of the model, to data points from IW Collect. IW Global can be used to monitor production processes in a more intuitive way, and also supports the use of historical data so that past events can be “re-played” in the digital twin.

2.4.1.2 Usage and Deployment

Industreweb Global is deployed as a web application in the cloud, on an on-premise server or on the Industreweb Factory connector edge device.

Industreweb Global was utilised in the EPFP Furniture pilots as described below:

Usage	Function	Description
Furniture Pilot	Error Proofing	Industreweb Global was used to create a Display dashboard to visualise the information relating to the product being produced. The parts barcode is scanned, and the barcode string used to query the business ERP system. The returned information is then shown in an operator friendly format at the machine.

2.4.1.3 New Developments & Roadmap

Since the pilot phase the Industreweb Global has undergone development in order to introduce many new features as described below:

Usage	Description
Cloud Deployment	By deploying Industreweb in the cloud, Industreweb Collect Factory connectors can be securely commissioned and managed from a single Admin UI across multiple locations.
Feature rich dashboards	A broad palette if dynamic UI components have been developed to allow users to render both simple and complex data on the screen to visualise data on the shopfloor.
Audit trail	All user edits made with Industreweb Global are saved automatically as part of an audit trail. This enables tracking of changes and the rollback of user mistakes.

2.4.2 Industreweb Visual Resource Monitoring Tool

2.4.2.1 Functional Description

The Visual Detection and Alerting system uses a Monitoring Box component running in the business premises or manufacturing facility to monitor using a camera and to recognise objects within its field of vision. It uses an Intel Visual Processing AI Unit to detect objects that it recognises from a pre-learnt AI model.

IW Collect running on the monitoring box then detects these events and based on a set of rules determines what Actions to perform. This could be to notify by email or SMS, to sound

a siren, to light a warning lamp, push data to the EFPF cloud or display a message on a screen or dashboard.

Three scenarios are proposed for this technology in EFPF:

- Detection and alerting of users not wearing mandatory PPE
- User in an area where they should not be or where there is a health and safety risk they should be aware of (e.g., near a running machine or heavy loads are being moved)
- Detection of staff not wearing mandatory COVID-19 protection

When the system is executed it loads in its AI model and Actions and Events and starts it continual scanning. When an object is detected a probability of match is made, which if greater than a predefined user threshold triggers the execution of the associated Actions as shown in Figure 25.

Figure 25: Object detection example using the AI framework

The technology can be applied to any scenario where the system should detect a specific object or objects and trigger actions whether preventative, warning or confirmative Actions need to be undertaken.

2.4.2.2 Usage and Deployment

The Visual Detection and Alerting system is deployed on an Industreweb Edge Device. Within EFPF it has been implemented successfully within the following use cases:

Usage	Function	Description
Aerospace Pilot	Spray Booth Monitoring	When the operator of the spray booth needs to use the system to spray the aerospace components, they must wear the appropriate PPE spray mask. The AI system detects if the user is wearing the mask and if so, engages the air supply for the spray gun, and the air extractor. At the end of the process the operator removes the mask which is detected, and this then turns off the air.

2.4.2.3 New Developments & Roadmap

Since the pilot phase the Industreweb Collect Factory Connector has undergone development in order to introduce many new:

Usage	Description
Air pressure sensor	By monitoring the air pressure valuable business intelligence can be gained regarding the relationship between production quality and the air pressure used during spraying.
Air Quality sensor	Through the use of an air particle sensor, it is planned to detect when there are high levels of dust particles in the air as this can affect the quality of the surface finish of the spray paint.

2.4.3 Symphony Platform

2.4.3.1 Functional Description

Symphony is a cloud-based software suite capable to incorporate a set management tools to create a complete Building Management System (BMS). The Symphony M2M platform can communicate with any automation controls, both standard protocols and proprietary systems. It is capable to integrate different protocols in order to monitor and control diverse building automation system with an open and modular approach.

The main functionalities of the Symphony platform include:

- Support for major automation standards (e.g., KNX, BACnet, LonWorks, OPC-UA, Modbus), allowing seamless interconnection of heterogeneous systems
- Physical objects abstractions (e.g., a virtual sensor that is a composition of physical ones) Groups of (heterogeneous) objects accepting the same commands
- Semantic information model aligned with OGC SensorThings, ETSI oneM2M and OPC-UA models
- User defined scenarios
- Pervasive environment sensing
- Energy monitoring and power management with customizable user defined policies
- Rule-based event management
- Notification engine
- Cloud platform for remote management and control
- Data collection and storage to enable analytics
- Hooks to attach external business intelligence services (e.g., to optimize production workflows)

The system integrator is able to define control logics and system behaviours using Symphony as a single system or as part of a global distributed deployment controlled by a cloud-base instance of the tool. The generic development framework of Symphony can define reactions to events which are fed to potentially complex processing rules ranging from simple algorithms to more complex decision systems, which result in actions performed by the system.

Events are generated by objects (e.g., motion/presence detectors, open/close contacts), simple threshold comparators (e.g., lux sensor, temperature sensor) or processing rules themselves. Actions include specific operations on (groups of) objects, notifications, activation of scenarios, etc.

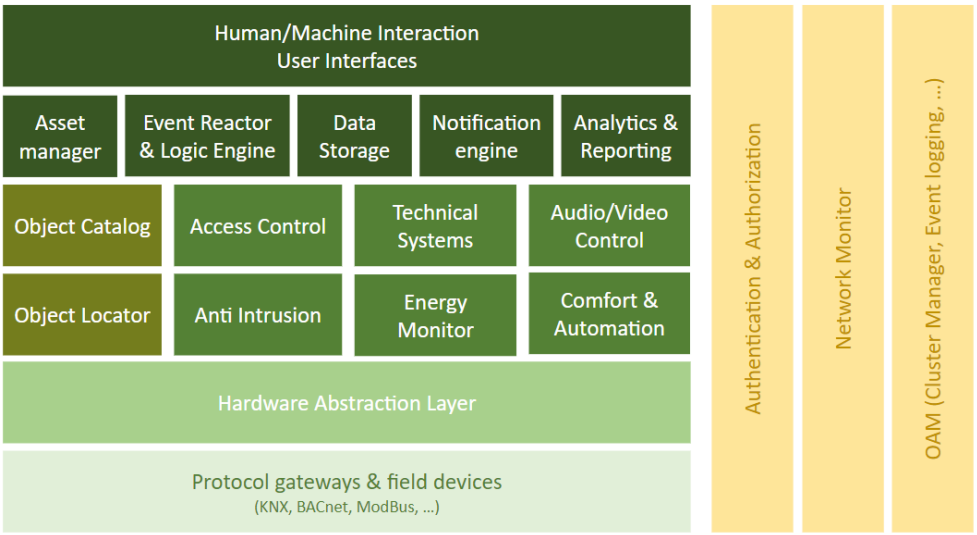


Figure 26: The Symphony Platform architecture

2.4.3.2 Usage and Deployment

The Symphony Hardware Abstraction layer is released, installed, and configured (based on the customer requirements) by Nextworks. The tool is involved on the realization of the Environmental Monitoring use case scenario to remote monitor the status of a set of sensors connected to the TSMatch gateway through the combination of multiple user-defined rules.

2.4.3.3 New Developments & Roadmap

Nextworks is actively working on the improvement of the Symphony platform to extend its functionalities, to support new sensors to be monitored, improve the efficiency of the overall system and to adapt the tool for new scenarios.

2.5 Data Analysis

Digitisation has enabled manufacturing companies to create a digital thread that unifies very aspects of their value chain. Based on the availability of data points from different aspects of the manufacturing activities, from raw material provenance through production steps, work in process, yield rates, quality conformance, equipment effectiveness, and so forth, all the way through supply-chain planning and logistics; the manufacturers are not only able to better understand the problems in their activities but also optimise where possible.

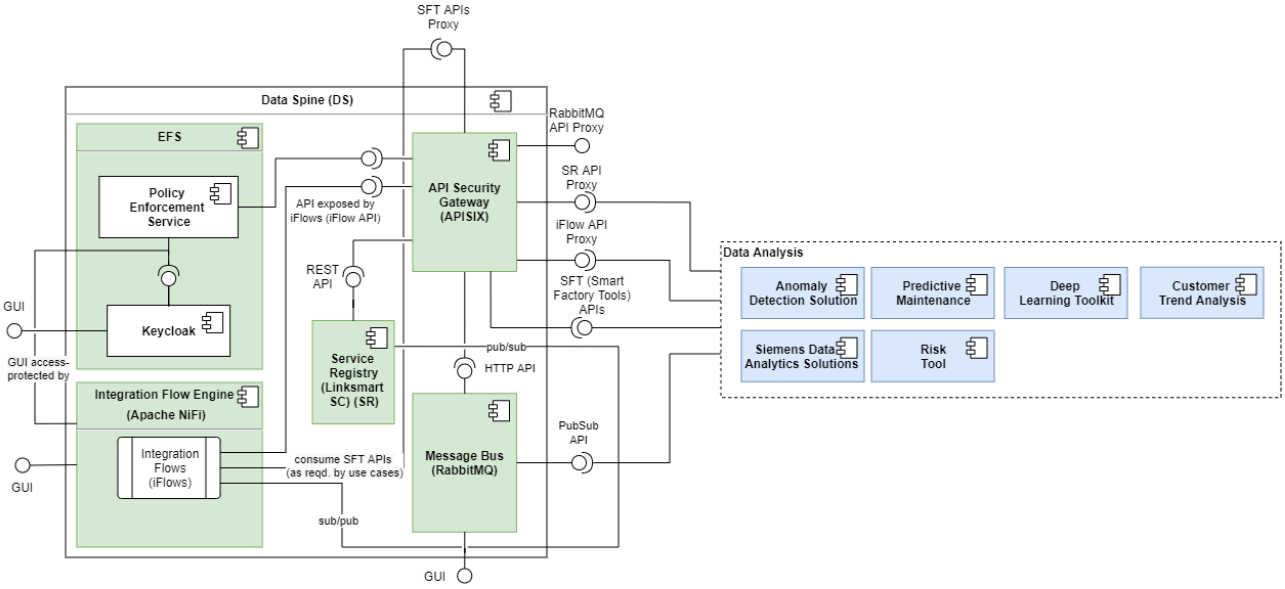


Figure 27: Interaction between the Data Analysis Tools and the EFPF Data Spine

The majority of manufacturers are relying on analytics to improve their activities and make decisions more accurately, quicker and with lower potential costs and risks. Based on this background, the EFPF platform brings together the advance analytic components that serve a variety of purposes and needs of the manufacturing companies. The overview of these components is provided in the following sub-sections:

2.5.1 Visual Analytics Tool for Predictive Maintenance and Optimization of Supply Chain Planning Activities

2.5.1.1 Functional Description

Visual and Data Analytics tool from CERTH is a web-based tool that supports real world needs and provides analytics methodologies for both production and supply chain domains. The front-end part's development is based on Angular and libraries like Chart.js. The algorithms regarding data analysis and forecasting were developed using Python.

In particular, the tool provides the following main functionalities:

- An analytics and monitoring dashboard for machine's anomaly detection cases applicable to both real-time and historical data. The solution was applied and validated for edge-banding machines (temperature and current values were used) and polishing machines (vibration data is used).

- An analytics and monitoring dashboard for bins' fill level. The tool provides real time monitoring of actual fill level and trend analysis results. The trend estimation enables the optimization of planning activities of waste management companies can give higher priority to companies/clients with the most aggressive trend in their bins.
- An analytic solution for tonnage forecasting. The tool provides to the end-user different methodologies to get predictions/estimations related to future quantity of a raw material based on historical data. By using this tool, the purchasing manager of a company can optimize the planning for ordering this material or reserve resources to handle this material.
- An administration form where the user can add information about new orders in order to add them to the rest historical data available for materials, tonnages, prices, and customers.
- Real-time integration with a Deep Learning Toolkit that provides price forecasting for different type of materials in order to further support and optimize the planning activities for end user (e.g., purchasing managers).
- Different type of visualizations based on different type of analytics and data in order to provide the output results to the view that better fits with the user's preferences.
- Option to end user to deploy/load historical data in .csv format in order to get analysis and visualization outcomes.

The Visual Analytics tool has the following main interactions with other components:

- EFPF Data Spine Portal: the portal provides the secure entry point to the tool's interface
- EFPF Data Spine Security Framework in order to use role management capabilities for providing different views to different users (Keycloak Role Management is used)
- Factory connectors provided by NXW and C2K in order to access sensors data from factories' premises (in alternative to this, the tool support IDS connectivity based on IDS Trusted Connector)
- Deep Learning Toolkit from LINKS for providing a visual interface to this tool and give an integrated solutions to the end-users that are related to planning activities

2.5.1.2 Usage and Deployment

CE and Furniture Pilot partners currently use the tool. The Visual Analytics tool provides solutions for both production processes and supply chain as well.

2.5.1.2.1 Circular Economy Pilot

Services related to Predictive Maintenance:

The tool enables anomalies detection towards predictive maintenance for KLE polishing machines. Two methodologies are available to the KLE maintenance operators:

- **Machine Vibration Diagnosis Profile (MVDP):** A solution based on real-time data coming from deployed vibration sensors. The method creates a profile of machines normal operation and detects abnormal vibrations by detecting the time point(s) when abnormal vibrations occur from the profile of the eigenvalues' sums. The basic

assumption of MVDP is that significant eigenvalue sums with simultaneous significant variations could point out to abnormal vibrations

- **Standardized Mahalanobis Distance (SMD):** technique is based on calculating the distance of a point P from a distribution T. Then, in order to detect anomalies, this algorithm was initially trained with samples belonging in normal condition in order to define threshold for highlighting a sample as normal or as an outlier. The outliers are visually presented to the user as points over a threshold:

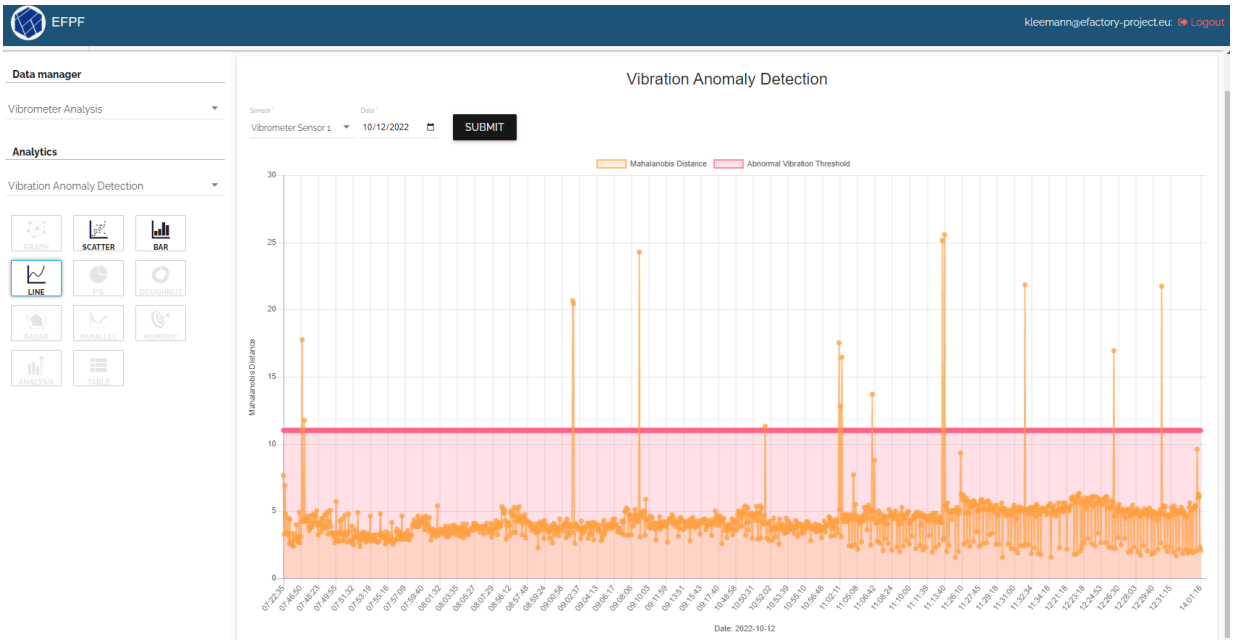


Figure 28 Screenshot of SMD Anomaly Detection at KLE

Services Related to Supply-Chain Management

- A **Monitoring Dashboard** for Fill Level monitoring is available to pilot partners KLE and ELDIA
- **Trend Analysis** for bins' fill level is available to ELDIA pilot partner. The solution is based on Slop Statistic Profile methodology
- **Tonnage Forecasting** solutions regarding wastes transportation are available to partners ELDIA and MILOIL. The solutions are based on various methodologies such as Auto-regression, Moving Average and Markov Chain models
- **Price Forecasting** is available through Visual Analytics Tool UIs to MILOIL and ELDIA partners. The service is based on Deep Learning Tool integration by partner LINKS which has applied LSTM architecture to available historical data for various waste materials' prices.

2.5.1.2.2 Furniture Pilot

- A **Monitoring Dashboard** for Fill Level monitoring is available to Lagrama pilot partner
- **Trend Analysis** for bins' fill level is available to Lagrama as well. The solution is based on Slop Statistic Profile methodology
- A **Machine Anomalies' Detection** solution based on Majority Voting method. The method was applied over widely used ML methods such as k-means, k-nearest

neighbours, one-class SVM, Local Outlier Factor and Density-based Spatial Clustering of Applications with Noise. It is applied in Lagrama's Edge-banding machine for temperature and current values coming from the corresponding sensors. The solution's architecture and an indicative screenshot are available in the next figures:

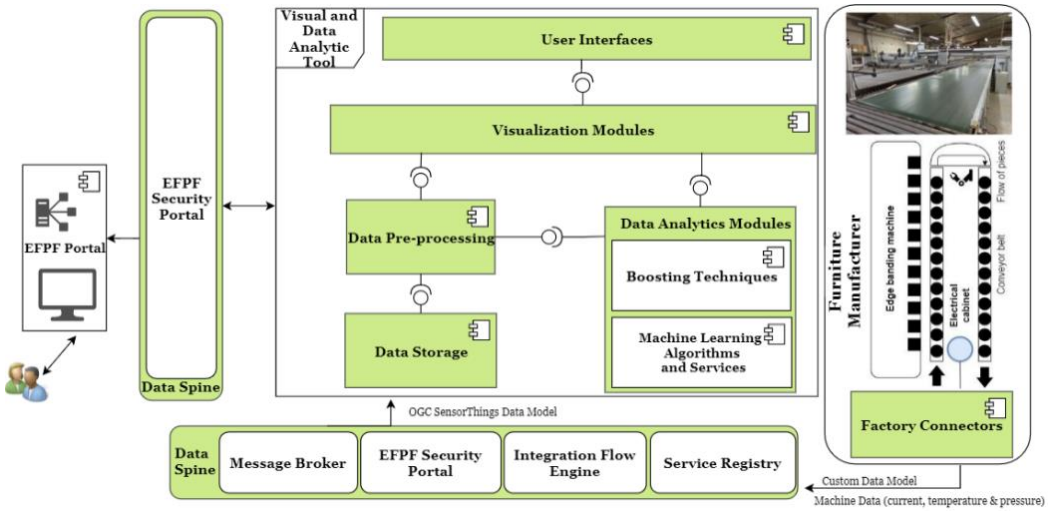


Figure 29 Architecture of Anomalies Detection Solution for Edge-banding Machine based on Visual Analytics Tool

Figure 30 Screenshot of Anomaly Detection UI

2.5.1.3 Tool's Availability beyond EFPF Pilots

The tool services regarding supply chain management are also available at platform level to all registered to EFPF portal users. The tool is available as a step wizard that a user can initially select an algorithm/solution and after that to import data, to configure the algorithm and select visualization graph, to explore the configuration details and finally to run the

configured algorithm and get the corresponding analytics outcome. An example for steps and interfaces regarding Trend Analysis solution is available in the next figure:

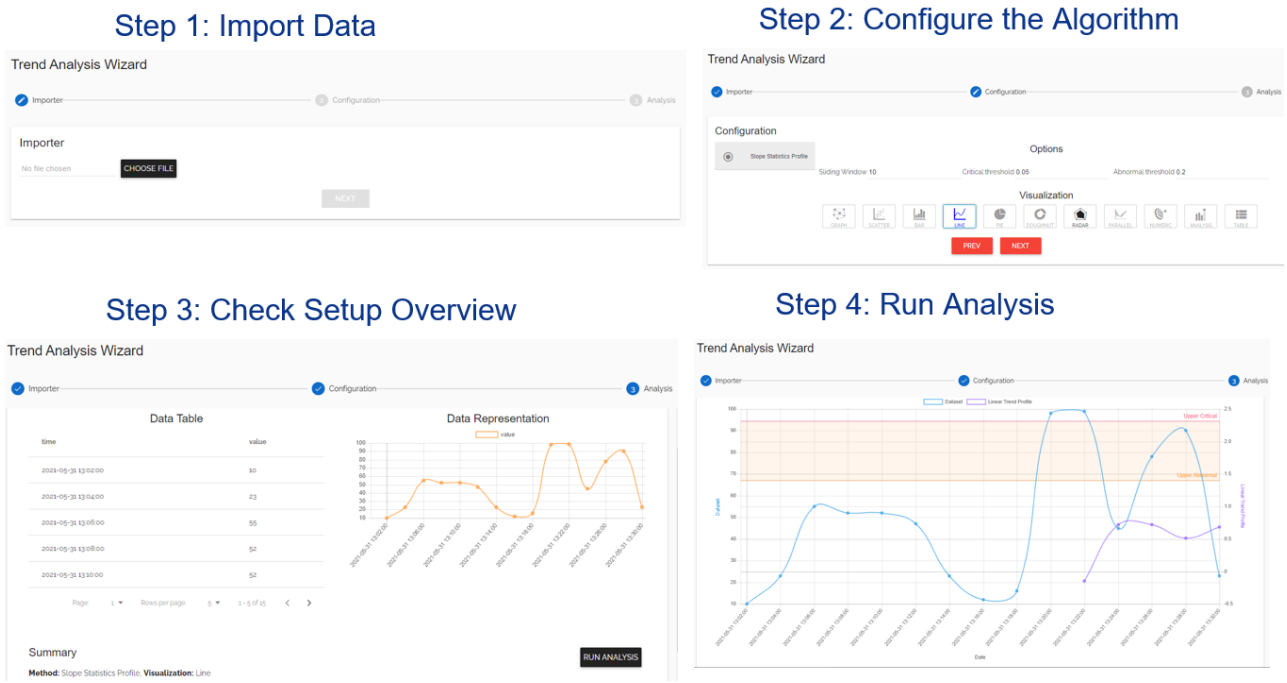


Figure 31: Steps for Trend Analysis Solution

2.5.1.4 Tool's Usage in EFPF Open Call Sub-projects

Visual Analytics tool was also used and extended through EFPF experimentation. During the Atlantis Engineering sub-project an Explainable AI (XAI) Fusion Service (FS) for Predictive Maintenance has been developed. The Visual Analytics tool was connected to this service and provided the front end part enabling analytics and XAI visualization.

XAI-FS project aimed at designing and implementing a solution that combines detection and prediction maintenance models within an intelligent fusion engine so to provide more accurate prediction/detection outputs and improved understandability on AI-based predictive maintenance solutions. The solutions inherited capacity to provide explainable AI capabilities (during the whole process) offering the user understanding and controlling of the process and enhanced trust to the solution and the result. Regarding the fusion baseline models were used and fused (kNN, SVM, XGBoost, etc.). The XAI-FS service was applied in KLE pilot partner's vibration data coming from its polishing machine. The solution was added to Visual Analytics tool dashboard, and it is available to KLE maintenance operators alongside the aforementioned solution, Machine Vibration Diagnosis Profile (MVDP) and Standardized Mahalanobis Distance (SMD).

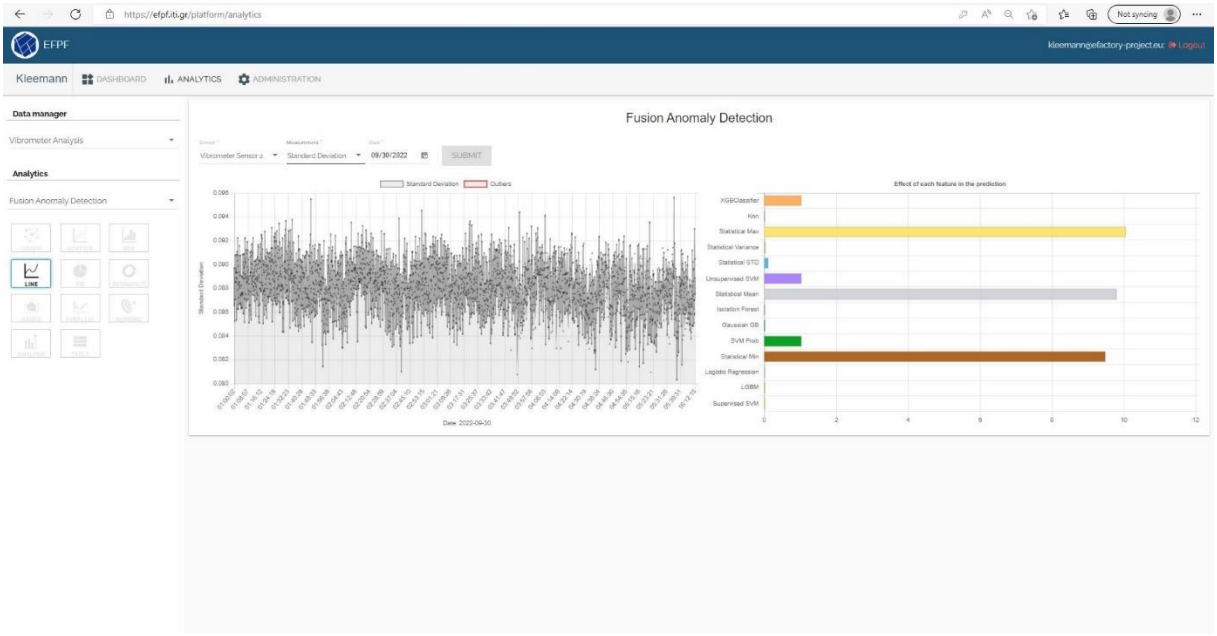


Figure 32: Screenshot of Open Call XAI-FS Service in Visual Analytics Dashboard

Besides this Open Call sub-project, the tool was used and validated in other sub-projects as well like the Predictive Maintenance approach in Premar case (CTAG/Cabomar) and Pressious Arvanitidis for fill level trend analysis regarding the inks level for printing machines.

2.5.1.5 Visual Analytics Tool Deployment & IPR

The tool is currently hosted by CERTH regarding the front-end part and the back-end/analytic is deployed on FIT instance of EFPF. Regarding IPR the tool is owned by CERTH, and some services are open available to EFPF portal. They are services regarding trend analysis and algorithms & UIs for tonnage forecasting. However, the complete tool included solutions for machines' anomalies detection is available only after bilateral agreements with CERTH.

2.5.1.6 New Developments & Roadmap

New developments regarding Visual Analytics tool with comparison to the previous version of this tool are summarized in the following bullet points:

- Front-end Development regarding Standardized Mahalanobis Distance (SMD) and tool's availability to KLE pilot partner for machine anomalies detection
- Updates and model's retraining regarding SMD after pilot's validation and feedback
- Front-end Development regarding tonnage and price forecasting for MIL pilot partner and tool's availability for testing and validation by the pilot
- Back-end and front-end development of both anomaly detection and fill level monitoring services for Furniture pilot partner, LAG. After that the tool was available for validation and testing by pilot that introduced some updates regarding fill level notifications etc.
- Development of a new adaptive window rolling median methodology for time series anomaly detection

- Implementation of a new instance of the tool to be available at platform level through the portal to all registered users and not only to EFPF Pilot. Trend analysis and tonnage forecasting analytics and visualizations are available in a form of a step wizard to end-users
- Tool's connection with new Production Server deployment to support SSO
- Interfaces extension to support XAI-FS service from EFPF Open Call
- Improvements based on Open Call feedback

Besides the aforementioned updates, new developments and improvements special focus was also given in scientific dissemination of the tool and its functionalities. Three scientific papers were presented in conferences: (a) *Fault Detection on Bearings and Rotating Machines based on Vibration Sensors Data* in 2021 IEEE International Conference on Progress in Informatics and Computing (PIC), (b) *Application of a Visual and Data Analytics Platform for Industry 4.0 enabled by the Interoperable Data Spine: A Real-world Paradigm for Anomaly Detection in the Furniture Domain* in I-ESA'22 Interoperability for Enterprise Systems and Applications and (c) *Utilizing an adaptive window rolling median methodology for time series anomaly detection* in 4th International Conference of Industry 4.0 & Smart Manufacturing (ISM 2022).

The next steps regarding the Visual Analytics tool are to further proceed with Open Call experimentation feedback and update any issues that were detected related to no error messages, no handling of bad input etc. in order to provide an updated version to EFPF Portal and its users.

2.5.2 Deep Learning Toolkit for Data Analytics

2.5.2.1 Functional Description

The Deep Learning toolkit is solution which provides an easy way to develop and integrate Deep Learning models with the EFPF platform. The default LSTM deep learning model provided allows the user to quick start their predictive maintenance experimentation. The tool composed of 5 different subcomponents:

- The Subscribe Adapter is in charge of fetching data from the provided Message Bus topic and to convert it to make it usable for ingestion to the Core DLT component. In order to be consumed the time series data has to be encoded in a JSON formatted OGC-Sensor Things data model.
- The Redis queue used in the Deep Learning Toolkit allows async communication between the different components of the tool. Being not tightly coupled allows the different tool's component to be deployed in a distributed manner, making it easy to get the most performance from the hardware available to the user.
- The Learning Component's role is to run the Long Short-Term Memory Neural Network Engine. LSTM networks are well-suited to classifying, processing, and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. This component includes both the functions for definition of the network's architecture and the function for performing the training and inference operations.
- The Publish Adapter is the component in charge of publishing the Deep Learning Toolkit predictions to the provided Message Bus topic. These predictions are formatted using a

JSON encoded proprietary simple data model and contain the value predicted, a timestamp and the confidence on the prediction.

- The Deep Learning Toolkit also a component in charge of providing a set of REST HTTP API. Using these APIs, the user can automate most of the operations available with the tool. This APIs allow in fact to perform CRUD operations on models and on the tool's configuration files. Starting and stopping the Learning Component is possible as well.

2.5.2.2 Usage and Deployment

The Deep Learning Toolkit is deployed as a Cloud/Edge service, depending on the requirements. Since this tool is distributed both as source code or Docker image it can be deployed both on machines that have Docker installed or on specifically built environments.

The Deep Learning Toolkit has been used in one EFPF Pilot within the Furniture domain and in two Open Call Projects. The applications are briefly described below:

Usage	Function	Description
Furniture Pilot	Predictive Maintenance	The DLT uses data coming from the Factory Connector at Lagrama to make predictions about possible failures in their Edge Bending Machine. The tool in this pilot has been deployed on the cloud.
Octavic Open Call Project	Predictive Maintenance	The DLT is used as a data source of Octavic's Predictive Maintenance Dashboard. The tool in this project has been deployed on the edge.
DNET Labs Open Call Project	Predictive Maintenance	DNET Labs aggregate different sensors monitoring environmental and machine variables to create a PdM solution. An edge deployment was required due to the high data volumes this project produced.

The Deep Learning Toolkit is distributed as open-source software, using the Eclipse Public License v2.0. Everyone can test and modify the provided source code and no warranty is provided. LINKS provides support and ML models under commercial agreements.

2.5.2.3 New Developments & Roadmap

Since the pilot phase the tool has received different updates according to the feedback received from the pilot partners and from the open call partners as well.

Usage	Description
Offline training	The users have now the option to train a model locally on batches of historical data.
Custom LSTM architecture definition	The users are not forced to use the LSTM network that comes with the tool but can define their own architecture.
Data integrity checks	The tool now perform integrity checks and warns the user if the data provided has not the correct format or if it presents gaps.

REST API for controlling the tool	The tool can now be started and stopped remotely, the option to change the configuration files through an API has been added as well.

2.5.3 Customer Trend Analysis

2.5.3.1 Functional Description

Customer trend analysis was designed to comprise different models that will allow companies to analyse their customer data and to predict customer behaviour. The first and only model that has been built is a churn prediction model which classifies users as (1) likely to churn, that is to stop being a customer, or as (2) unlikely to churn, that is to remain a customer.

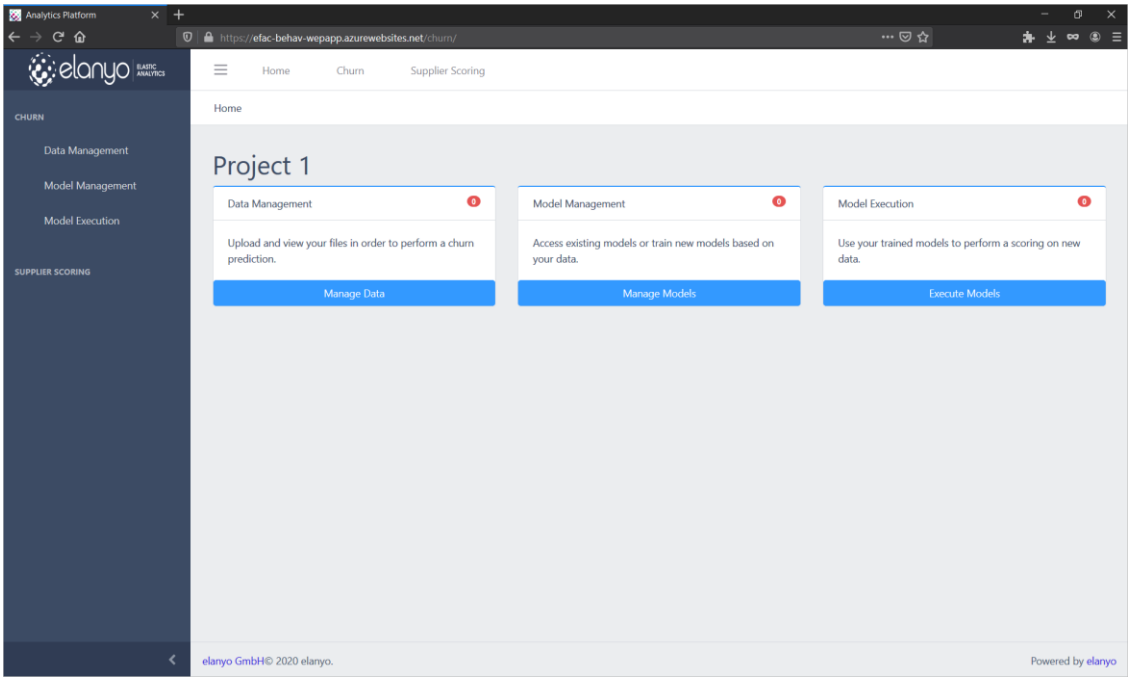


Figure 33 Customer Trend Analysis dashboard

This churn prediction model works as follows: customer data is divided into two sets according to its timestamp. Data before a certain date is used to create a customer profile, while data after a certain date is used as a target variable.

The first data set, the customer profile, could include a long list of features, such as:

- userActivity
  - absolute
  - percentage: morning, noon, evening
- boughtProduct
  - absolute
  - percentage: morning, noon, evening
- activatedProgram

- customerCart
- Newsletter Clickrate (reactionToNewsletter / openedNewsletter)
- Newsletter:
  - openedNewsletter
  - reactionToNewsletter
- unsubscriptionFromNewsletter

The more independent feature, the better the customer profile.

The second data set will only calculate one target variable and assign a 1 if this variable is positive or a 0 if this variable is negative. In case of churn prediction, this means that a one could indicate that a customer has churned (and is no longer a customer), while a zero could indicate that a customer has not churned (still a customer).

Following this step, the entire dataset (including the features and the target variable) is split into train and evaluation data, e.g., 80 percent for training and 20 percent for evaluating. Then, the training data is used to train a model, and, following that, the evaluation data is used to evaluate if the model performs well or not.

Once everything is set up, new data on customers can be given to the model which will then return a value for each customer: 1 if the customer will churn according to the model and 0 if the customer won't churn according to the model.

Description of tabs/sides in the tool:

On the dashboard, users are provided with an overview of all their models and data. They see the data they uploaded, the models they trained and the data they scored. They can either directly click on the respective buttons or navigate through the menus on the top or on the left to fulfil a certain task or move to a certain process step.

On the data management tab, users can upload data on their users. Right now, it only works for data of a given structure, but functionality to add generic data will be added.

On the model management tab, users can train models (with the step they have previously uploaded). Right now, the only model that can be trained is the churn prediction model and this one is based on a logistic regression in Keras.

Once the model has been trained, several figures and graphs are displayed that allow the user to assess the model's predictive power, including loss, accuracy, false positives, false negatives, and the ROC curve.

On the final tab (model execution), users can upload scoring data to run it through the trained models. Right now, users can only upload customer data and score it with the churn prediction model. The result is a list of customers that may churn and of customers that may not churn.

2.5.3.2 Usage and Deployment

The tool was used by partner Ascora. It is deployed with Azure.

2.5.3.3 New Developments & Roadmap

The tool hasn't been changed since the last update. No future changes are planned.

2.5.4 Analytics Integrator Platform Tool

2.5.4.1 Functional Description

The Analytics Integrator Platform, developed by Siemens, is a batch-oriented analytics workflow creator which allows the EFPF's users to create their own custom analytical pipelines containing various modules used in order to parse data, extract meaningful observations which can be then published in to the main EFPF communication services for others to use in further domain specific activities. The Analytics Integrator Platform provides the following functionalities to fulfil these requirements:

- Batch-oriented
- Scheduling and manual triggering
- Docker operator and containers
- Data Spine sink/source interaction

The Analytics Integration Platform (AIP) is an open-source tool that facilitates integration of diverse analytics services available within EFPF amongst themselves and also with the EFPF databus – Data Spine.

The paradigm followed is mini batch execution, each step is triggered when sufficient data is accumulated on its input. This allows for reactive execution which although not real-time can satisfy use-cases requiring minute-level latency. The SDSS (Secure Data Service) is used to adapt the message-based data sources to a more archive type system that can be consumed in a periodic fashion.

The coordinator for the analytic flow is responsible in triggering each step and serializing/deserializing data to/from each connected step. Analytic steps can either consume from certain data sources, push data to output datasinks, or perform analytics either internally or via delegation to external systems.

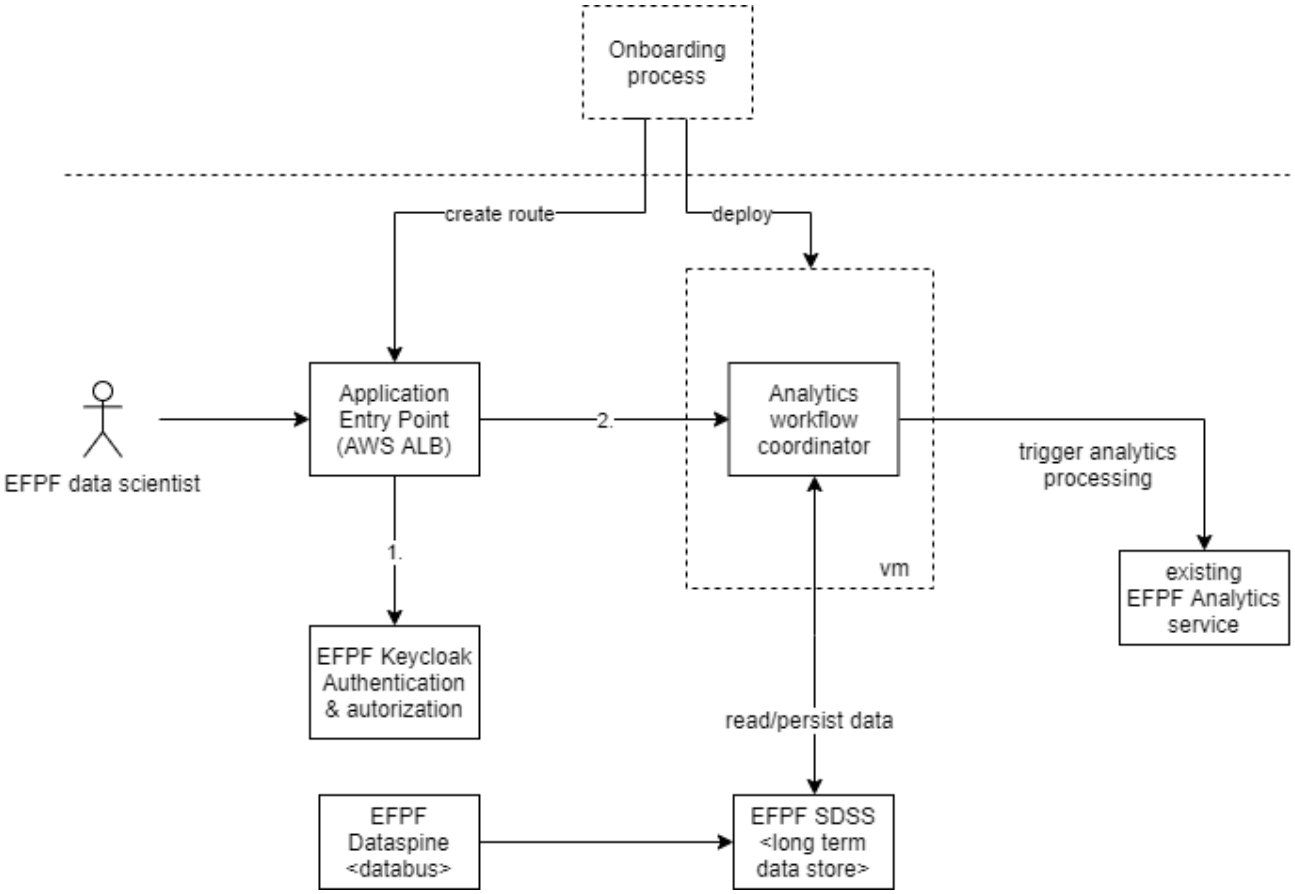


Figure 34: AIP access flow

2.5.4.2 Usage and Deployment

The AIP source-code, with all components, is currently deployed on the EFPF server (https://gitlab.fit.fraunhofer.de/) and the AIP services are hosted on AWS Cloud. The Analytics Integrator Platform deployment flow is depicted in Fig. 12.

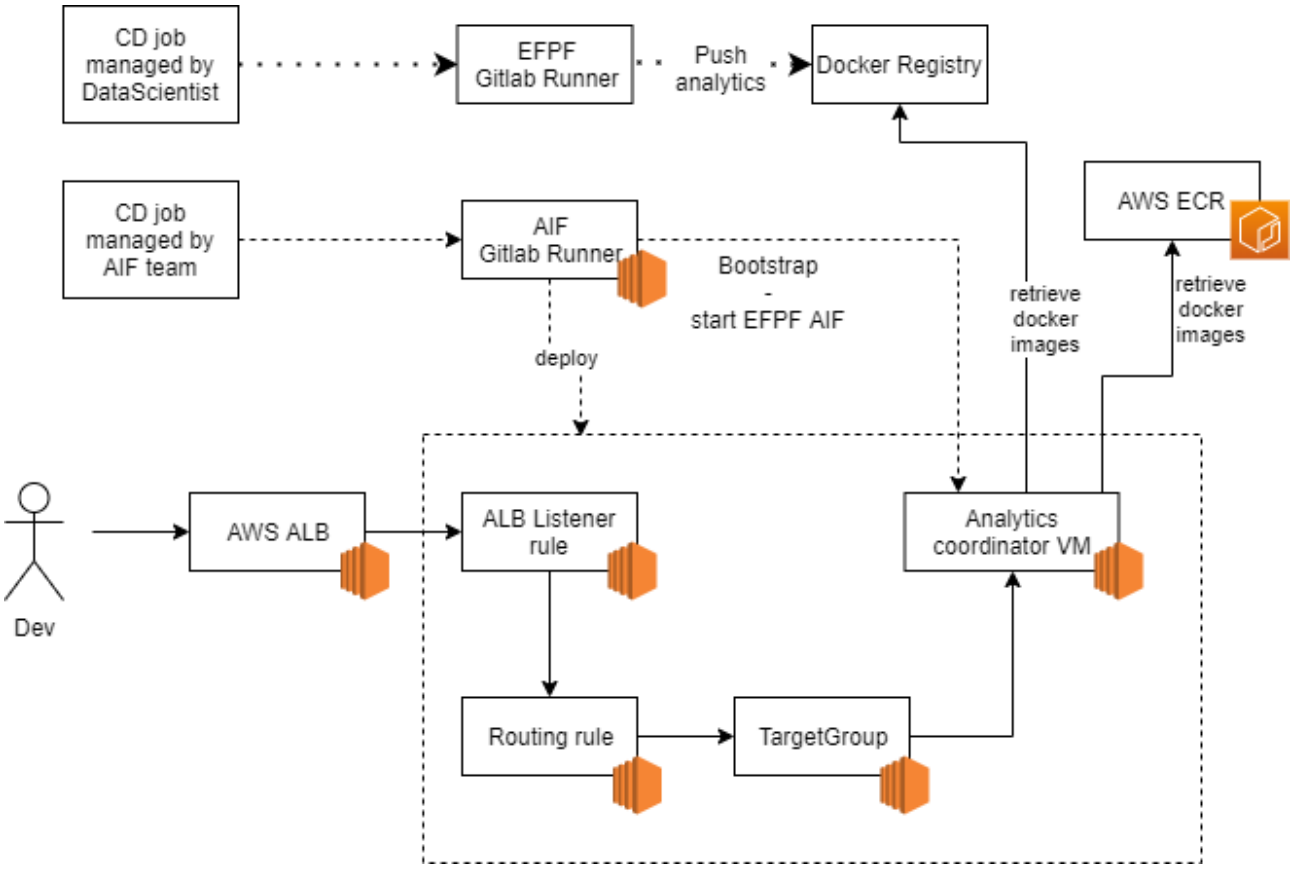


Figure 35: AIP deployment flow

2.5.4.2.1 Service orchestration

This process is managed by an Orchestration tool (Apache Airflow). This tool integrates user defined pre-existing analytical modules and existing EFPF analytics modules.

Apache Airflow platform enable workflows creation, scheduling, and monitoring. Workflows can be described as Direct Acyclic Graphs (DAGs) of tasks (e.g., fetching data, running analysis, triggering other systems etc.), which can be executed based on specified order and dependency. Through the user interface, the execution pipelines (configured as Python code) can be visualized, monitored, and debugged.

Defining workflows as code, allows a version capable and collaborative environment, while the Airflow modular architecture and use of a message queue make it easy scalable. DAGs are parametrizable (execution date + other optional parameters), and they can be run in parallel. Airflow load DAGs from Python source files (looks for DAG_FOLDER) and they can be run when triggered (manually or via the API) or on a defined schedule (part of the DAG).

There are 3 common types of tasks:

- Operators (predefined tasks that can be string together)
- Sensors (special subclass of Operators which wait for an external event to happen)
- TaskFlow-decorated @task (custom Python function, TaskFlow takes care of moving inputs and outputs between the Tasks, as well as automatically calculating dependencies).

Task's dependencies are specified with ">>" (downstream) and "<<" (upstream) operators. By default, a DAG will only run a Task when all the Tasks it depends on are successful. Tasks don't pass information to each other by default and run entirely independently. However, there are mechanisms (like XComs) which provide information exchange from one Task to another.

For the current EFPF use-case, the Airflow orchestration consists in:

- Airflow platform, configured to run on each tenant machine
- Airflow related env variables are added within the VARIABLES file. They are needed for the Airflow environment configuration (https://airflow.apache.org/docs/apache-airflow/stable/start/docker.html#initializing-environment). Alternatively, the .env file can be manually created in the same folder where the airflow docker-compose.yml is placed) and add the airflow variables here.
- The DAG implementation is defined in the /airflow/dags/ folder: LagramaTADUpDown.py. In this file the analytics python functions are put together, and the Airflow libraries are imported (DAG, operators etc). Then, the DAG is instantiated with the default arguments (described below):

```
default_args = {
      #owner (str) – the owner of the task, using the unix username is
      #recommended
    'owner': 'EFPF_ANALYTICS',

     #the previous task instance does not needs to have succeeded
    'depends_on_past': False,

     #the 'to' email address(es) used in email alerts
    'email': ['vladut.dinu.ext@siemens.com '],

    #indicates whether email alerts should be sent when a task
    #failed/is retried
    'email_on_failure': False,
    'email_on_retry': False,

    #number of retries that should be performed before failing the
    #task
    'retries': 3,
    #delay between retries
    'retry_delay': timedelta(minutes=2),
}
with DAG(
    'Lagrama_Predictive_Maintenance_TAD_Remodeling',
    default_args=default_args,
    description='Predictive maintenance using Twitter s anomaly     detection
with Rule engine on Lagrama Data',
```

```
    #schedule_interval[8] is defined as a DAG argument, which can be
    #passed as a "cron"[9] expression
    schedule_interval='*/20 * * * *',

    #determines the execution_date for the first task instance
    #the best practice is to have the start_date rounded to the
    #DAG's schedule_interval[10]
    start_date=days_ago(0),

    #In order to filter DAGs, you can add tags in each dag
    tags=['Predictive'],

    #catchup concept[11] turned off
    catchup=False
) as dag:
```

The PythonOperator is used for calling the defined functions. Next, the Airflow dependency operator (>>) enforces the desired execution order:

```
getData= PythonOperator(
        task_id='Data_Acquisition',
        python_callable=DataAcquisition,
        )
...
   kaplan_meier = PythonOperator(
        task_id='kaplan_meier',
        python_callable=kaplan_meier,
        )

  getData >> remodelingData >> anomalies >> asso_rule_generator >>
kaplan_meier
```

The web Interface is available at: `efpf.daiconro.eu/<tenant_name>/`. The default account has the `login: airflow` and the `password: airflow`.

Understanding Apache Airflow features

Airflow's environment assembles into a platform used to create, schedule, and monitor different workflows. This ecosystem's strong points are represented by a scalable

[8] https://airflow.apache.org/docs/apache-airflow/stable/dag-run.html

[9] https://en.wikipedia.org/wiki/Cron#CRON_expression

[10] https://airflow.apache.org/docs/apache-airflow/1.10.10/faq.html#what-s-the-deal-with-start-date

[11] https://airflow.apache.org/docs/apache-airflow/stable/dag-run.html#catchup

architecture, the dynamic generation of workflows created using Python, a powerful UI that show everything you want to know about anything, it is easy to use and, also, it is open source. On top of this, we have integrated a plug-in that offers an IDE inside its UI that can keep track of the workflow's versions by connecting to a GIT repository.

A workflow is represented by a DAG. This is the core concept of Airflow, collecting and organizing the tasks creating relationships between them.

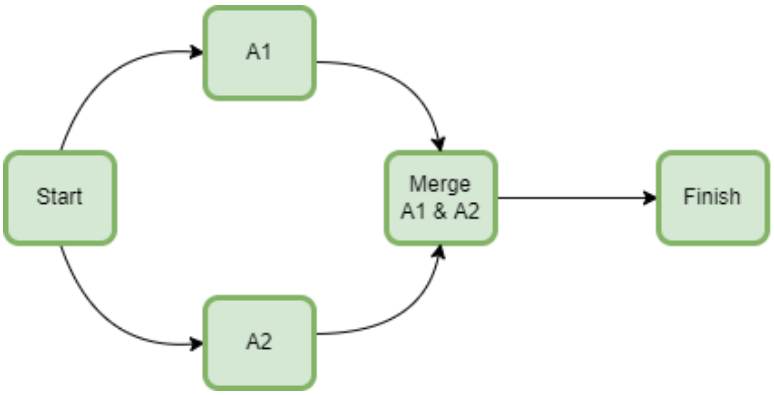


Figure 36: DAG representation

The DAG shown above has a starting process, then 2 actions are parallelized, their results are then merged and sent to the finish stage.

Airflow allows the user to set different parameters for a DAG which, for example, specifies how many times to retry either a process or the entire DAG. Being based on pure Python, the DAG's comportment will be exactly like a script. If an error shows up, there will be the exact syntax that appears if a script fails to run.

Setting the dependencies between processes, the following symbols are used: **>>** and **<<**. Based on the example shown above, the dependencies are written like this:

Start >> [A1, A2] >> MergeA1&A2 >> Finish

The first dependencies says that after the "Start" process has successfully finished, run both "A1" and "A2" processes simultaneously. Keep note that a list of processes will run all of them at the same time.

Our Airflow use-case demonstration

To implement our analysis pipeline mentioned in the previous sections of this document, Airflow was used to orchestrate each step. Every 20 minutes the DAG is triggered to start and evaluate the gathered data. If there was enough data, the DAG continues to run in order to process the data. Either way, the DAG is stopped from further running.

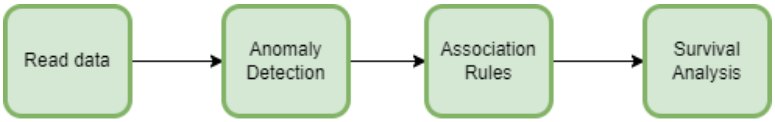


Figure 37: Implementation workflow

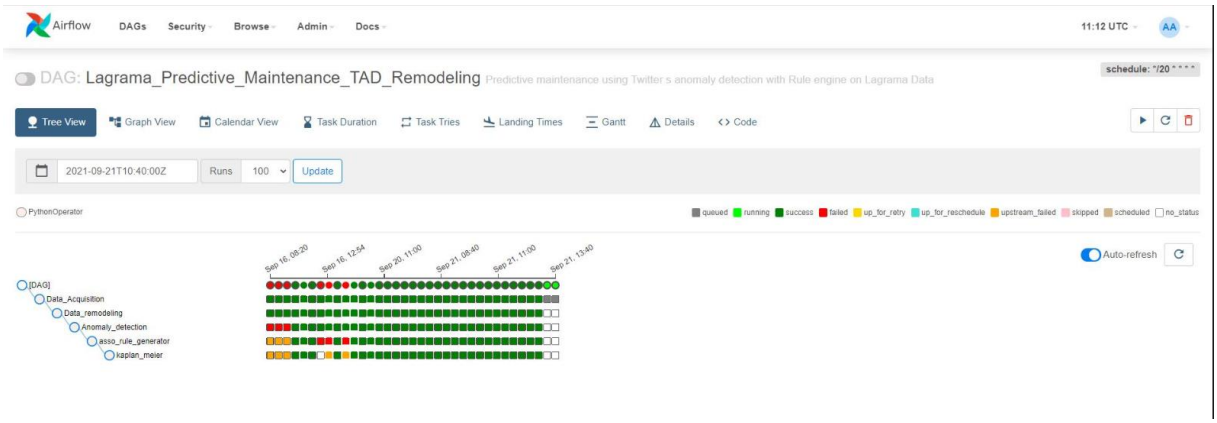


Figure 38: DAG Tree View Dashboard

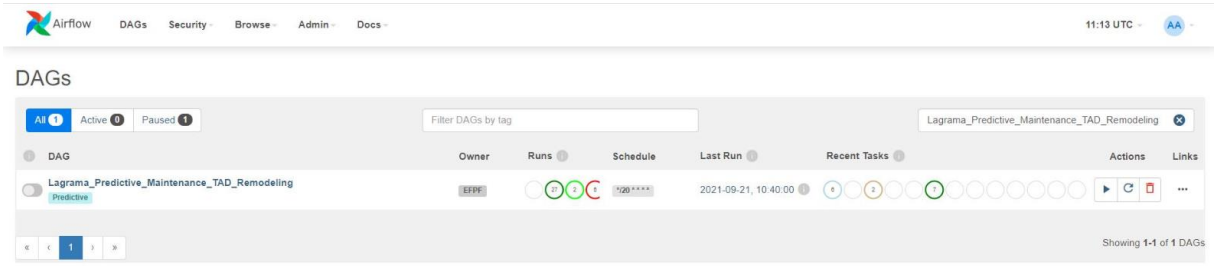


Figure 39: Airflow dashboard

Utilizing Airflow

Code Editor

At the first sight, the client will be presented with an empty Airflow Dashboard.

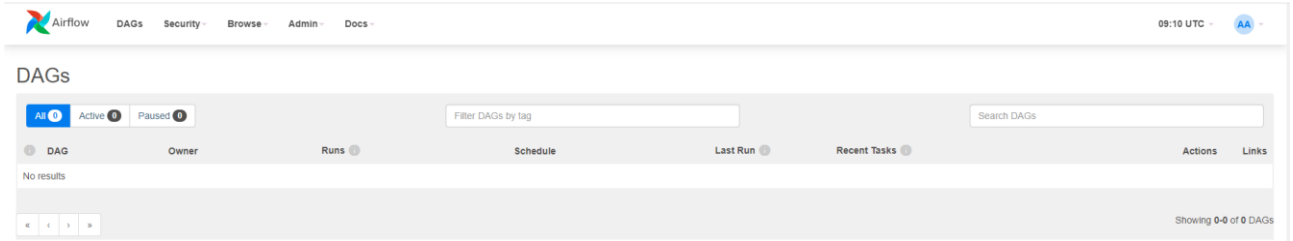


Figure 40: Airflow empty dashboard

In order to start the development using Airflow, the client has to go to the Admin tab and then to the Dags Code Editor tab. There, the editor plugin will be used to create the analytical flows.

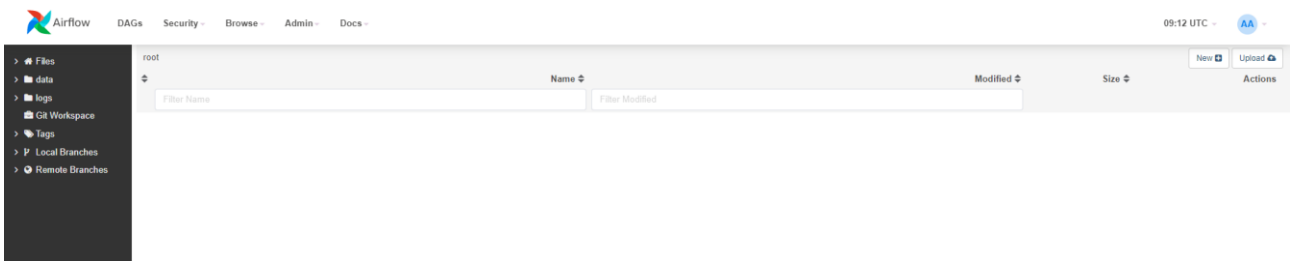


Figure 41: Code editor dashboard

The code editor tab will be empty with a local git repository instantiated. To create files the client has to go the Files tab (which is the default tab selected), press the *New* button on the top right corner. The UI will change into the editing mode:

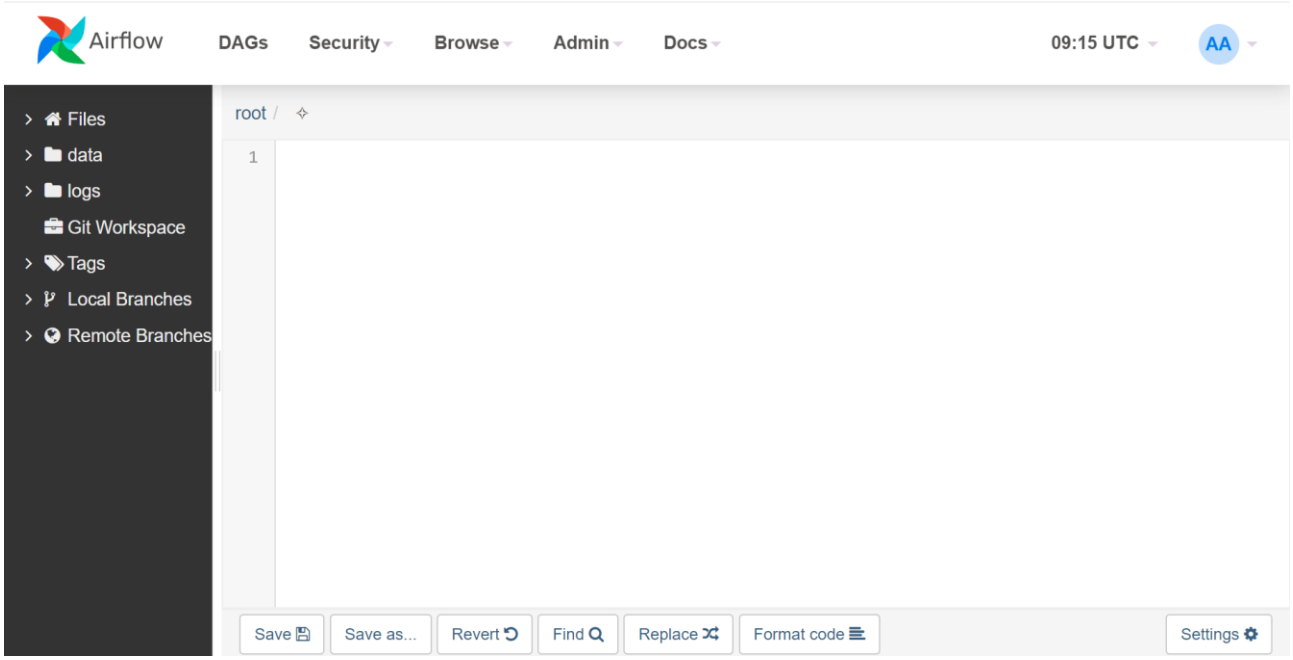


Figure 42: File editor view

Here, the client can write its Python script in order to use the Federated services, or the ones developed by himself. In order to save the file, press the *Save as..* or *Save* button placed under the text editing UI.

After saving the file, the Airflow's scheduler runs in background and scans the *Dags* folder for new files. If the file created by the user has syntax errors or the pip packages required are not found on the machine, Airflow will signal this on the *Dashboard*.

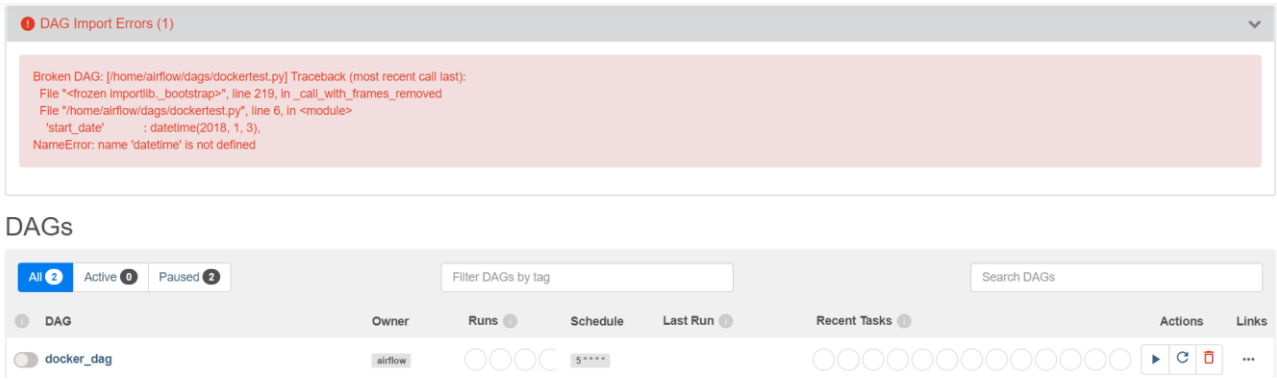


Figure 43: Error occurred on DAG import

Analytical pipelines and workflow versioning system

In order to version the DAGs or the flows created, the client has to go to the *Git Workspace* tab. There, a UI similar to others Git Versioning Tools will be shown. To commit the changes made to a file, select the file from the bottom left list called *Working Copy*, and then press the *Stage* button which is next to the list name.

In the *Message* tab, the client has to write a short text specifying the changes done to the files selected. To save the files, press the *Commit* button.

The *Staging Area* will show all the files selected to be saved.

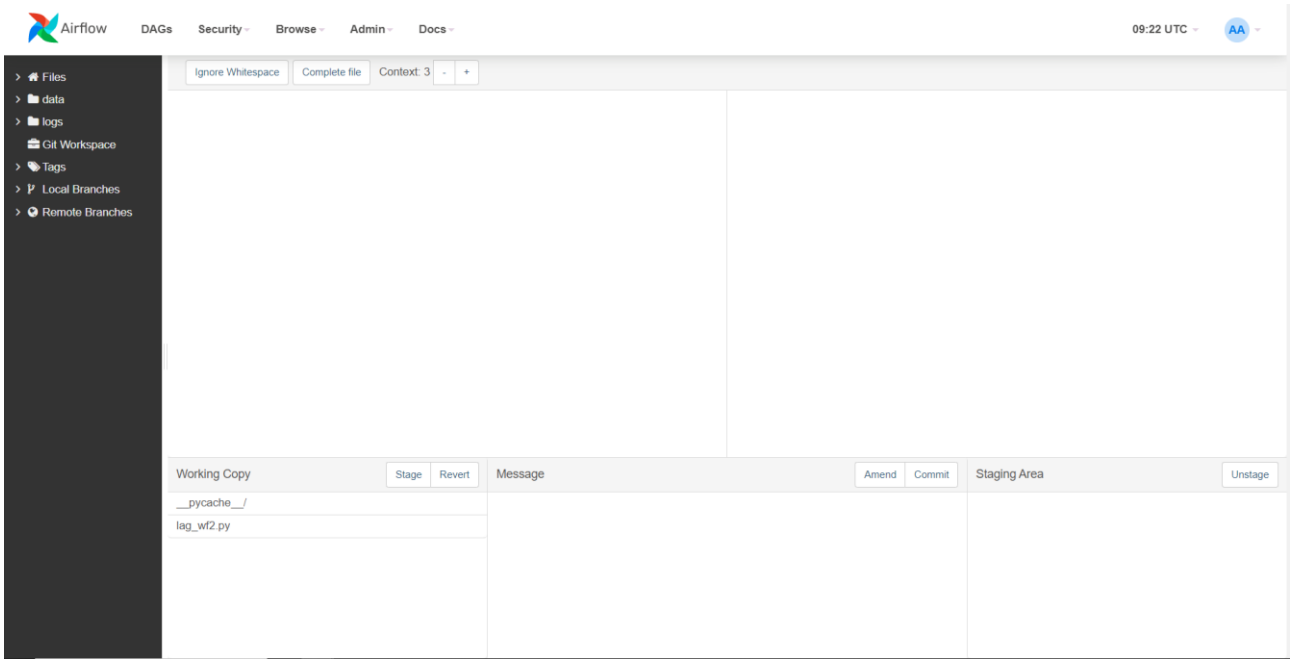


Figure 44: Code staging dashboard

Whenever a file from the *Staging Area* is selected, the tabs above will contain the changes done to the selected file.

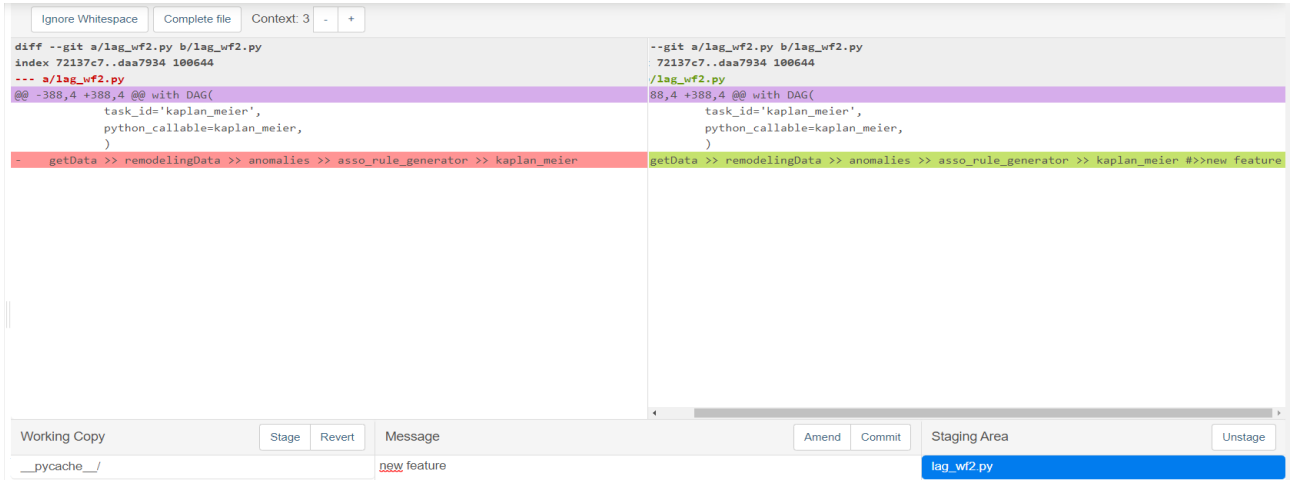


Figure 45: Differences between commits

The commits are found in the *Local Branches* tab.

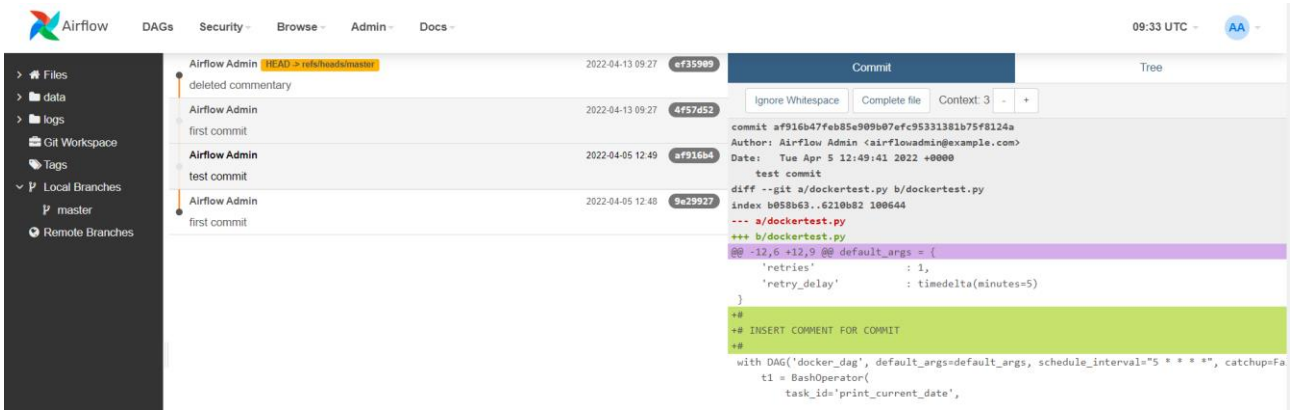


Figure 46: Branch commits

As shown in the figure, the client can select any commit in order to see the changes. These will appear in the right part of the UI. In order to see all the files from the local repository, select the *Tree* tab which is next to the *Commit* tab.

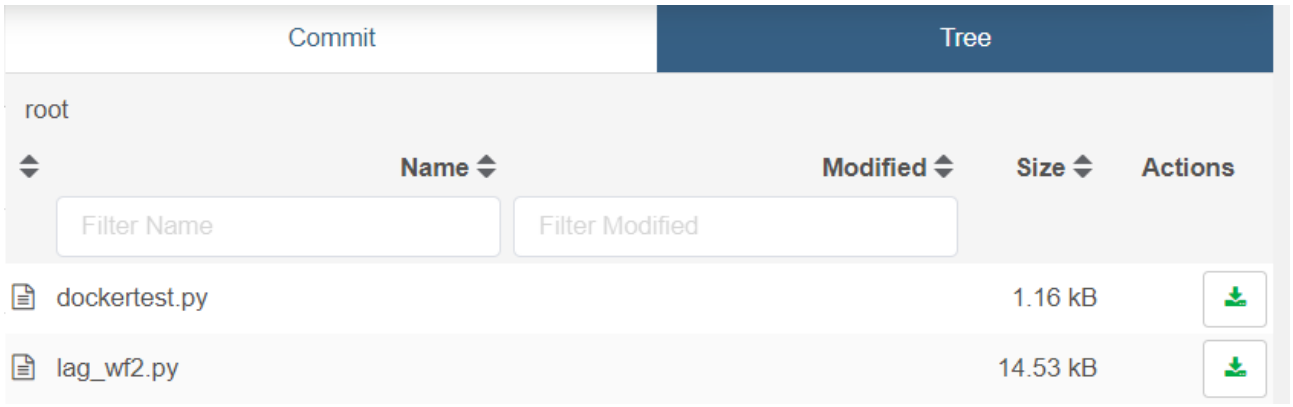


Figure 47: Files within the repository

2.5.4.2.2 Service integration

Dockerized workflow

Our vision within the provided integration platform is that every analytical module/service should be a Docker container with a service that is used to communicate with the exterior. Basically, Airflow provides a DockerOperator used to create Docker containers with the specified configuration, like the analytical module image, and then proceed with the workflow by writing Python functions that are communicating with the container's service in order to send and retrieve the results.

As shown in the example below, a basic HelloWorld is performed using the DockerOperator withing a DAG workflow. There are also BashOperators that are used for some basic commands, such as "date" or for echoing a message.

The workflow prints the current date to the terminal, creates an empty Docker container and then the last operator prints "hello world".

```
from airflow import DAG
from airflow.operators.bash_operator import BashOperator
from datetime import datetime, timedelta
from airflow.operators.docker_operator import DockerOperator
default_args = {
    'owner': 'airflow',
    'description': 'Use of the DockerOperator',
    'depend_on_past': False,
    'start_date': datetime(2018, 1, 3),
    'email_on_failure': False,
    'email_on_retry': False,
    'retries': 1,
    'retry_delay': timedelta(minutes=5)
}
with DAG('docker_dag', default_args=default_args, schedule_interval="5 * * * *", catchup=False) as dag:
    t1 = BashOperator(
        task_id='print_current_date',
        bash_command='date'
    )
    t2 = DockerOperator(
        task_id='docker_command',
        image='centos:latest',
        api_version='auto',
        auto_remove=True,
        command="/bin/sleep 30",
        docker_url="unix://var/run/docker.sock",
        network_mode="bridge"
    )
    t3 = BashOperator(
        task_id='print_hello',
        bash_command='echo "hello world"'
    )
    t1 >> t2 >> t3
```

Figure 48: Integration with docker containers workflow

Any analytical module with the requirements mentioned above can be used and deployed within the platform in order to create complex processing workflows using the DockerOperator.

2.5.4.3 New Developments & Roadmap

The Analytics Integrator Platform has been developed since the last deliverable and continues to undergo functional improvements.

2.5.5 ROAM Tool

2.5.5.1 Functional Description

The ROAM Tool allows users to perform operations such as data transformation, analysis, and monitoring on their data. Users do this by providing data through the EFPF Message Bus via MQTT, and then configuring the ROAM Tool to feed this data through a workflow. Workflows comprise a set of customizable recipes that each perform a single operation of varying complexity. Such operations include, but are not limited to, transforming data to a certain format suitable for other recipes, sending email and push notifications based on some criterion, and calculating statistics, such as calculating averages and variances, or analysing trends to determine expected hitting times and data distribution. The ROAM Tool includes recipes that inherently contain a cache, but will soon also allow for integration with the SDSS (Secure Data Store Solution) to provide the workflow with either their own or the ROAM Tool workflows historic data.

The ROAM Tool comprises a backend, but also a frontend webapp. In this webapp, users can more easily authenticate and view all their available workflows and their own recipes.

They can also see all predefined customizable recipes. There, users can create, edit, remove, and copy their workflow and recipes, view details such as outputs and usage metrics of their workflows, and view the last workflow output and input for monitoring purposes. Moreover, depending on the functionality pertaining to the constituent recipes of a workflow, users may enable webpush notifications for their current browser and cleanse their cache.

To enable users to use their own defined input and output topics, the ROAM Tool is integrated with the Pub/sub security service. This allows users to supply their own input topic, and create their own output topics on the ROAM Tool vhost. The ROAM Tool will then handle granting access itself access to their input, using the permission of the user, and granting the user access to ROAM Tool vhost output topics to remove overhead on both sides.

2.5.5.2 Usage and Deployment

The ROAM Tool is centrally hosted by Almende, and is deployed using Docker containers. The users can access this instance of the tool via the EFPF portal, and here users can only see any recipes and workflows pertaining to themselves. If necessary, the containers could be provided to be hosted locally with local environment files, but users would need to create their own Keycloak clients to be able to integrate their instance of the ROAM Tool with the rest of the EFPF platform.

The ROAM Tool was used in two EFPF pilots, and was planned for usage in an open call pilot, but the latter (last one in the table below) was cancelled due to time constraints.

Usage	Function	Description
Stores Monitoring Pilot	Stock Depletion Alerts	The ROAM Tool sends email and push notifications whenever the weight sensors measuring wood plank storage subceeds a certain threshold.
Furniture Pilot	Predictive Maintenance	The ROAM Tool sends email and push notifications whenever current, pressure, or temperature sensor measurements exceed certain thresholds.
CTAG/Cabomar Pilot	Anomaly Detection Alerts	The ROAM Tool sends email and push notifications whenever the machines are demonstrating anomalous behaviour (which is inferred from data obtained through another tool). Workflows were created to accommodate for a real time situation, and also for a proof-of-concept daily summary message for a scenario in which no real-time data could be pushed, but rather a person would manually push a data collection every day.

2.5.5.3 New Developments & Roadmap

Since the pilot phase, the ROAM Tool has accommodated to the changes made in the EFPF Data Spine, and to a couple of new use cases:

Feature	Description

New email and push notification recipes	Users wanted some form of notification/alert functionality, which we now accommodate for, and it turns out that this is now a much requested functionality.
Authentication using EFPF Keycloak	Users can now only see their own recipes and workflows, eventually any workflows that they have been given external access to, and possibly workflows of a company that they are an admin of.
Other new generalized recipes	New recipes for data transformation, trend analysis, extrapolation, and expected hitting times.
Integration with Pub/sub security service	Users should be able to view their workflow output. To that end, as the ROAM Tool is fully automatically configurable, we had to develop a way for users to automatically grant us access to their input topics and for us to create output and error topics, and automatically grant users access to those.
Privacy & fair use policies	As we store some user data, a privacy policy had to be formulated. As the tool is usable by arbitrarily many people, a fair use policy had to be written to ensure that people do not abuse the tool.

There are still a couple of things that should be implemented and improved:

Feature	Description
Integration with the Secure Data Store Solution	This is already underway. The code has been written, but still needs automated access functionality and more rigorous testing. There is also an integrated recipe that inherently uses the SDSS to use process history easier.
Workflow example page	A panel in the frontend that showcases how different recipes can be used in different workflows. This also functions as a tutorial.
More rigorous testing suite for the backend	This testing suite is integrated with GitLab CI/CD, such that the entire tool is tested before deployment.
Eventual marketplace advertisement	The ROAM Tool can be hosted locally, and we should sell a license for this via the marketplace. We also want to offer a consultancy service to help using the ROAM Tool. Users may request us to create new recipe functionalities specifically for them too.
Refurbishment of old recipes	Some old recipes were very tailored to the furniture pilot, so these should be generalized and thoroughly tested.
Improved resource monitoring	The tool should include a more thorough monitoring and alerting functionality, to ensure users adhere to fair use policies, and to ensure that the available resources remain sufficient.

2.6 Asset Management

This section describes the tools that are specifically designed to support business and operations management activities. Currently two tools are listed in this category that address the user requirements for resource and product management.

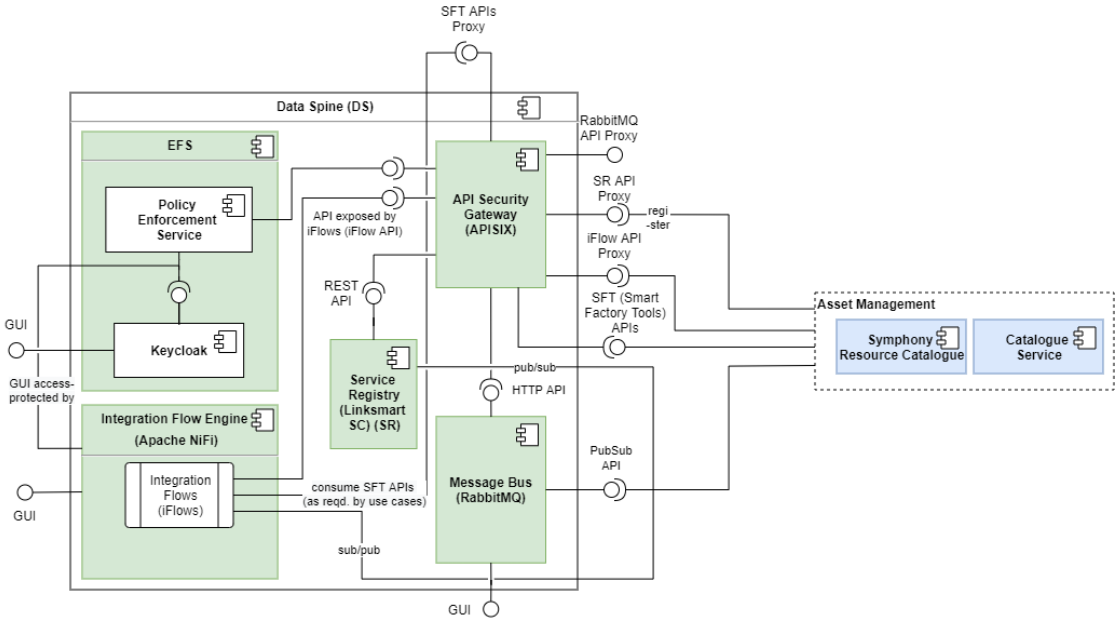


Figure 49: Interaction between the Asset Management Tools and the EFPF Data Spine

2.6.1 Catalogue Service

2.6.1.1 Functional Description

Catalogue Service is a platform for product / service publishing, and it is the main enabler of the partner discovery phase as it allows companies to introduce themselves to the EFPF platform with the products they supply and the services they provide.

To enable users to find what they are looking for quickly, Catalogue Service offers publishing products with semantically relevant annotations. It makes use of generic and sector-specific taxonomies as knowledge bases from which relevant annotations can be obtained automatically given a product category. The main taxonomy used in Catalogue Service is eClass which is an ISO/IEC compliant industry standard for cross-industry product and service classification[12]. Further, available taxonomies can be extended with domain-specific taxonomies such as Furniture Taxonomy and Textile Taxonomy as well.

Catalogue Service makes use of Universal Business Language (UBL), a world-wide standard providing a royalty-free library of standard electronic XML business documents that are commonly used in supply chain operations, as the common data model since it contains appropriate data elements for catalogue/product management such as catalogues, products, product properties and so on. Moreover, products and services as well as catalogues are persisted on a UBL-compliant relational database.

[12] https://eclass.eu/en

Structure of the product data might differ from sector to sector or from company to company; therefore, raw data, which could have varying formats, are kept in disparate repositories while metadata are kept in a global registry. Maintaining all the metadata in a single repository enables querying on products having heterogeneous structures initially. Once a product is identified, its complete, structured definition can be fetched from the respective repository. Finally, Catalogue Service provides several REST APIs for management of products and catalogues through CRUD (Create-Read-Update-Delete) functionalities.

2.6.1.2 Usage and Deployment

Catalogue Service is the marketplace component of the NIMBLE Platform bringing sellers and buyers together to negotiate on physical products for different terms (e.g., price, quantity, etc.) and complete checkouts. Similar to the all the components of the NIMBLE platform, Catalogue Service is open source under Apache License 2.0, and it is free to use the service (and hence all the NIMBLE Platform) for commercial purposes. The source code can be found on https://github.com/nimble-platform and there is a dedicated repository containing the deployment information based on Docker containers on https://github.com/nimble-platform/docker_setup.

Catalogue Service is currently deployed on EFPF Production Environment, and it has been used in 2 EFPF pilots within Aerospace and Furniture domains. Additionally, Catalogue Service was also utilised in one of the Open Call projects. The applications of the Catalogue Service in the pilots as well as the Open Call project are described below:

Usage	Function	Description
Open Call	Base platform for scrap items	The Catalogue Service has been extended to account for scrap items (defective products, old inventory, scraps, and end-of-life components)
Aerospace Pilot	Product offering	The Catalogue Service has been enhanced to be a part of the EFPF Integrated Marketplace Framework to increase sales efficiency because of better visibility for companies' complete product portfolio in the federated EFPF ecosystem.
Aerospace Pilot	Searching	The Catalogue Service is used to search for searching suppliers for different products and services by using various parameters in order for companies find best products (price / lead-time / quality) and suppliers (reliability / communication / quality
Furniture Pilot	Order management	The Catalogue Service is utilised send and receive orders for different products to demonstrate easy and comprehensive management of the orders placed to suppliers as well as those received from other companies
Furniture Pilot	Catalogue exchange	The Catalogue Service is used to share catalogues with different parties for improved management of catalogues of external companies through an additional implementation of local import mechanism

2.6.1.3 New Developments & Roadmap

The Catalogue Service has been extended in many ways in order to fulfil the requirements from Aerospace and Furniture partners. Following table summarizes the efforts undertaken for new developments:

Usage	Description
Taxonomy for aerospace domain	Development of a new taxonomy for each new domain is necessary in Catalogue Service; therefore, a new taxonomy was developed in order to categorize products and service in a semantic way for the aviation industry.
ATA Specification support	The Catalogue Service has been enhanced to perform all order related communications (order administration / payment etc.) based on ATA Spec2000 standard and data model has been updated accordingly for ATA Spec2000.
Offering products / catalogues for a predefined user group	The Catalogue Service has been customized to place products and services in a catalogue, so that potential customers (predefined user groups like A/C OEM, Airlines, MRO, 1st-tier suppliers) can search for them easily, which leads to a better visibility of our capabilities to the respective customer groups.
Whitelist/blacklisting catalogues	The Catalogue Service has been updated to allow users to be added into whitelists / blacklists for certain catalogues of products and services
Catalogue-specific contract generation for parties	The Catalogue Service has been enhanced to generate contracts between buyers and sellers additional to system-wide generic contracts.
Catalogue exchange functionality	The Catalogue Service has gained the ability to share catalogues to different companies and predefined user groups.
Ability to hide prices for certain catalogues	The Catalogue Service has been customized to allow hiding certain information, such as price, for certain catalogues so that only whitelisted users can access that information.

For the moment, there is no planned development activities for the Catalogue Service; however, taxonomies for new domains can be developed upon request.

2.7 Application Development

This section describes the tools that are specifically designed to support the development of applications that make use of the EFPF framework of services. Currently the tools that are listed in this category are all comprised in the EFPF Software Development Toolkit (SDK) set of components, which are described as follows.

2.7.1 EFPF SDK

2.7.1.1 Functional Description

The EFPF Software Development Toolkit (SDK) has the purpose of supporting developers with services that aim to define, design, develop, compose, and orchestrate solutions and manufacturing services, based on the services provided by EFPF. This SDK comprises an API framework that provides developers with the means to easily generate applications and services. It includes programming tools to provide value-added service compositions, and a service API which offers methods to access functionalities of all technical components that the service developers need. The basics for the development of the SDK is to provide a component or framework of elements that foster and improve the development of solutions. The SDK provides a user interface (named Studio) and other application-development components, the resources and services that they require. The following schema (Figure 50) represents the internal structure and the basic elements of the EFPF SDK for the development of the project solutions.

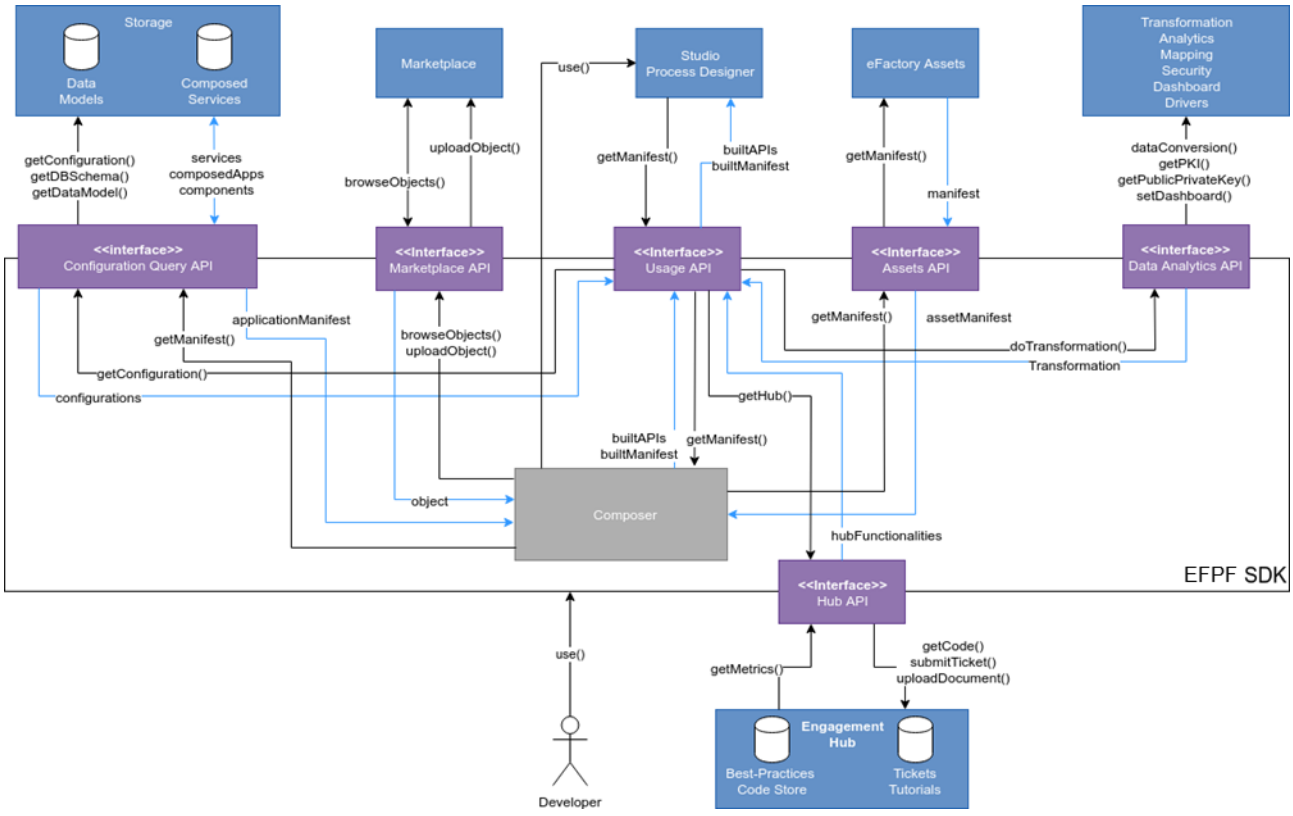


Figure 50: High-level Architecture of the SDK Component

2.7.1.2 Usage and Deployment

Despite this tool is currently not directly used in any pilot, it has been widely promoted in the EFPF Open Calls and was used for the development of three internal applications of the EFPF project to support the pilots (Shop Floor Intelligence App, Spray Booth Intelligence App, Spray Booth Air Quality App).

The EFPF SDK is a JavaScript wrapper, which is released as open-source code using the Apache 2.0 Licence, available in the EFPF Developer Engagement Hub, in the links:

- https://engagementhub.caixamagica.pt/efpf/sdk
- https://engagementhub.caixamagica.pt/efpf/internal-sdk-libraries

2.7.1.3 New Developments & Roadmap

Developments on the SDK include:

- Inclusion of the EFPF DataSpine AMQP and MQTT brokers
- Centralisation of the EFPF Database services
- Integration of the WASP Process Designer views

Future work includes additional integrations with other EFPF services and APIs, namely the integration with the DataSpine Secure Database Services.

2.7.2 EFPF SDK Studio

2.7.2.1 Functional Description

The Application Development Studio consists of an Integrated Development Environment (IDE) that allows applications to be seamlessly developed using a Graphical User Interface that allows the simplification of the developer work. This IDE includes functionalities such as a code editor with drag and drop of elements, build automation, code completion, a compiler, a debugger, and a testing environment. It is a holistic GUI supporting developers to easily implement applications by integrating and orchestrating services, APIs, and connectors developed in EFPF. The Studio provides necessary tools and means to develop and deploy applications on end user devices to application developers. The developer will be supported by step-by-step procedures which cover the entire development process including the registration of new applications (or updates) on the EFPF Marketplace.

It has a standard interface for accessing the resources made available by the EFPF SDK, which allows their control and monitoring, their modification and customisation, and their provisioning to the target applications.

2.7.2.2 Usage and Deployment

Despite this tool is currently not directly used in any pilot, it has been widely promoted in the EFPF Open Calls and was used for the development of three internal applications of the EFPF project to support the pilots (Shop Floor Intelligence App, Spray Booth Intelligence App, Spray Booth Air Quality App).

The EFPF SDK Studio is a tool developed on top of the Eclipse CHE platform, which is released as open-source code using the Eclipse Public License 2.0 Licence, available in the EFPF Developer Engagement Hub, in the link:

- https://engagementhub.caixamagica.pt/efpf/sdk-studio

2.7.2.3 New Developments & Roadmap

Developments on the SDK Studio include:

- Accessing the Studio with HTTPS security features
- Integration of the Studio with Keycloak authentication
- Integration of the Studio with the WASP Process Designer
- Embedding the Frontend Editor
- Deployment of the Applications with the EFPF Marketplace

Future work includes additional integrations with services and tools, embedding of other tools in the Studio views.

2.7.3 EFPF SDK Frontend Editor

2.7.3.1 Functional Description

The EFPF SDK Frontend Editor is designed to support the development of custom applications initiated with the EFPF Software Development Kit (SDK) based on the services provided by the EFPF platform. The Frontend main functionality is to provide developers with a graphical user interface (GUI) editor for prototyping, to integrate and customize applications built with the SDK. Developers can combine all microservices based on implementations of the SDK integrated functionalities.

The Frontend can be accessed by the SDK Studio interface using a plain browser. The solution is based on predefined templates that stand for themselves (e.g., customized GUI elements) and can bind data sources from EFPF that are orchestrated by Application Development Studio. Application developers have a high degree of flexibility and power by combining the predefined templates and visual elements that can be used inline or nested. This approach results in a multitude of possible applications designs.

The Frontend UI offers additional guidance and allows developers to speed up the process of rapid prototyping. Any design strategy is supported, and a broad range of applications can result from mixing single-page, multi-page, and progressive web application designs. The workflows are highly configurable translating business process models of the use-case scenarios into functional maintainable applications.

Figure 51 shows an overview for the Front-end module along with its integration with the EFPF SDK.

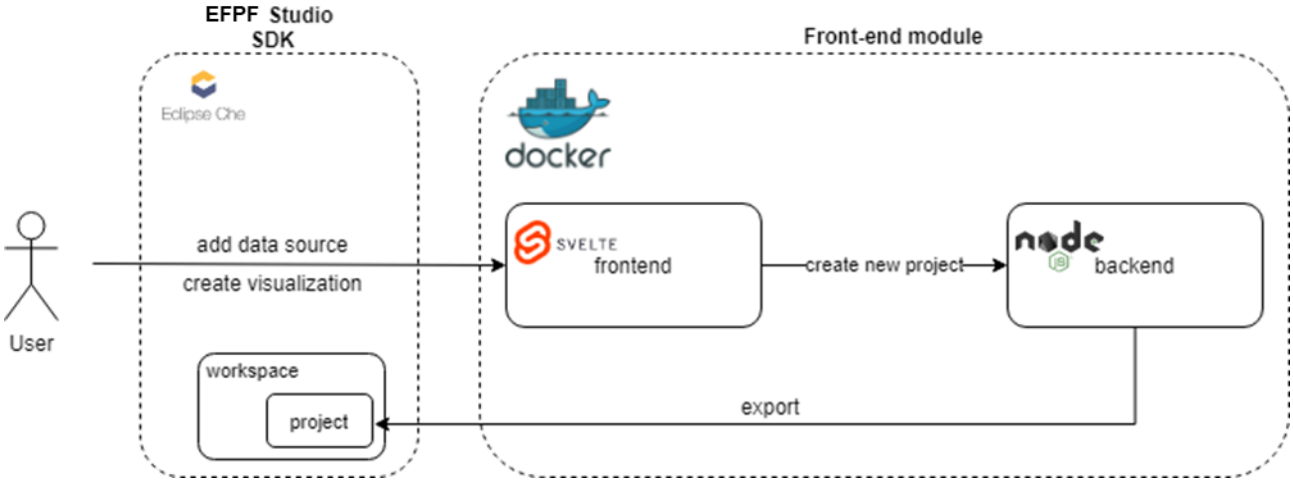


Figure 51: Architecture of the SDK Frontend and interaction with the SDK Studio

2.7.3.2 Usage and Deployment

Despite this tool is currently not directly used in any pilot, it has been widely promoted in the EFPF Open Calls and was used for the development of three internal applications of the EFPF project to support the pilots (Shop Floor Intelligence App, Spray Booth Intelligence App, Spray Booth Air Quality App).

2.7.3.3 New Developments & Roadmap

New developments regarding SDK Frontend Editor tool summarized in the following bullet points:

- Back-end and front-end of a web-based UI design environment that allows the design, and development of intuitive user interfaces, webforms and analytic visualisations with a high degree of flexibility
- Revision of the whole code and interconnection with EFPF
- Dissemination, including demos in technical events
- Improvements based on feedback gathered from demos and pilot usage
- SSO integration: The Frontend Editor has been adapted to utilize the EFPF SSO Server for user authentication and management
- Deployment of Frontend Editor tool on EFPF Portal
- Development and deployment of specific use-cases applications that show users what they can achieve by using the Frontend Editor:
    - Shopfloor Intelligence: allows users to view various Tools that can support the analysis of machine condition data
    - Lagrama Predictive Maintenance: consists in a set of visualization tools provided by Almende
    - Spray Booth: provides users with the ability to see data regarding paint job number and number of particles in the air captured by a particle sensor installed in the paint booth
- Deployment of samples applications on Marketplace.

No future changes are planned.

2.7.4 EFPF SDK Developer Engagement Hub

2.7.4.1 Functional Description

The EFPF Developer Engagement Hub has the purpose to define and develop a suite of tools that fit together and consist of a platform to support developer collaboration between developers, customers, and communities. It is a framework available from one single platform location (web-based), which not only supports the development of tools, but it also involves the development community, fosters their active contributions in the shape of tests, comments, suggestions, and new requests in the form of change requests in existing applications, e.g., to support other platforms, trends, needs, or extensions. It allows the community to download/reuse/fork the existing code from the Studio (if published) and actually use/test it on new conditions or scenarios. The outcomes of these tests will always

come in the form of issues, reports, comments, or suggestions. It also promotes the usage and creation of standards, methodologies, and best practices. This framework includes mechanisms such as wikis, issue trackers, forums, and blogs. Other tools such as configuration management, business continuity, and business process design and management are initially conceived out of scope of this component and relevant to other components such as the SDK/Studio. This component will be implemented by a web server hosting information that fosters the creation and support of communities related to the manufacturing business.

2.7.4.2 Usage and Deployment

Despite this tool is currently not directly used in any pilot, it has been widely promoted in the EFPF Open Calls and was used for the development of three internal applications of the EFPF project to support the pilots (Shop Floor Intelligence App, Spray Booth Intelligence App, Spray Booth Air Quality App).

The EFPF Engagement Hub is a Web Portal, developed on top of the tool GitLab CE, which is released as open-source code using the MIT Licence, available in the EFPF Developer Engagement Hub, in the links:

- https://engagementhub.caixamagica.pt/efpf/sdk-engagement-hub

2.7.4.3 New Developments & Roadmap

Developments on the SDK Developers Engagement Hub include:

- Integration of the Engagement Hub with the EFPF Studio
- Integration of the Engagement Hub with chat services
- Support of OpenCalls.

No further developments foreseen.

3 Conclusion and Outlook

The objective of this work package was to develop, enhance and integrate the building blocks (systems, tools, services, methods) that represent the core offerings of the eFactory platform. This is aligned with the overall vision that the eFactory platform is not providing just one smart factory tool or data analytics service, but it is providing the open architecture that can integrate multiple smart factory tools and services from different providers. As such, Tasks in this work package generally did not focus not on a single tool/approach, rather the focus was on delivering a category of tools/approaches that developed and enhanced for integration within the eFactory platform.

All partners in this work package worked to develop and adapt Tools and Services to provide them in a form that makes them ready for integration and deployment on eFactory platform, resulting in a comprehensive and diverse catalogue of Tools and Services available to users of the EFPF ecosystem. Enabled by the federated approach to the base platforms, Users are able to select the Tool or Service appropriate to their needs from the various base platform marketplaces.

Traditionally, the isolated nature of Smart Factory tools has restricted Users from implementing solutions to tackle broad scope production issues, but with the EFPF core Data Spine components, the development of composite solutions to satisfy the needs of today's manufacturing Industry 4.0 environments is possible. Furthermore, Users can be reassured that when new tools become available as the ecosystem naturally matures, they can be integrated into existing applications or used to develop new solutions. Should the required functionality be unavailable on the EFPF Tool catalogue, Users can develop their own Smart Factory solutions through the integrated SDK framework. User defined tools such as these can then form part of a larger composite solution with other "off the shelf" tools, or be written specifically to provide all functionality necessary to deal with an entire application.

The flexibility and choice offered by the EFPF Ecosystem enables both approaches dependent on users' needs, technical ability, financial considerations, and attitude to using 3rd party tools. The execution of the EFPF pilots and the subsequent Open Call experimentation validated the success of this approach and has led to a "continuous" development process of the Tools and Services, bringing new features and functionality driven by user feedback and requirements. The ongoing sustainability and use of these Smart Factory solutions by end users is looking very positive and the maturity of the Tools and Services will allow the EFF to exploit the outcomes from the work package.

Overall, not only have Tools and Services been developed but they have also been implemented and exploited in real world business scenarios and have been shown to offer tangible business benefits across a broad spectrum of manufacturing.

Annex A: History

Document History	
Versions	V0.1: • Document set-up and draft Table of Contents V1.0: • First draft version ready for partner contributions V1.1-1.5: Partner contribution V1.6-1.9: Final version for internal review
Contributions	C2K • Simon Osborne NXW • Gabriele Scivoletto FOR • Rute Sofia CNET • Mathias Axling ASC • Brian Clark FIT • Rohit Deshmukh ICE: • Rob Woodfin CERTH: • Georgios Alexandros Nizamis Michailidis • Athanasios Vafeiadis • Panagiotis Gkekas • Dimitrios Ioannis Iakovidis Dimoudis ALM: • Carlos Hermans • Carolyn Langen SRDC: • Senan Postaci CMS • Carlos Coutinho

Annex B: References

[EMC22] European Collaborative Manufacturing and Logistics Cluster. Available online: https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/fof-11-2016 (accessed on 1 June 2022).

[NIM22] Nimble (Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe) Project. Available online: https://www.nimble-project.org/ (accessed on 1 June 2022).

[COM22] COMPOSITION (Ecosystem for Collaborative Manufacturing Processes) Project. Available online: https://www.composition-project.eu/ (accessed on 1 June 2022).

[DIG22] DIGICOR (Decentralised Agile Coordination Across Supply Chains) Project. Available online: https://www.digicor-project.eu (accessed on 1 June 2022).

[VFO22] vf-OS (Virtual Factory Operating System) Project. Available online: https://www.vf-os.eu (accessed on 1 June 2022).

[VLC22] ValueChain's Network Portal platform. Available online: https://valuechain.com/products/network-portal/ (accessed 1 June 2022).

[NXW22] Nextworks' Symphony platform. Available online: https://www.nextworks.it/en/products/symphony (accessed 1 June 2022).

[C2K22] SMECluster's IndustreWeb platform. Available online: https://www.industreweb.co.uk/ (accessed 1 June 2022).

[Sem22] Preston-Werner, T. Semantic Versioning 2.0.0. Available online: https://semver.org/spec/v2.0.0.html (accessed on 15 June 2022).

[NDoc22] Apache NiFi Documentation. Available online: http://nifi.apache.org/docs.html (accessed on 15 June 2022).

[ISO11354] ISO 11354-1:2011 Advanced Automation Technologies and Their Applications—Requirements for Establishing Manufacturing Enterprise Process Interoperability—Part 1: Framework for Enterprise Interoperability. Available online: https://www.iso.org/standard/50417.html (accessed on 6 January 2021).

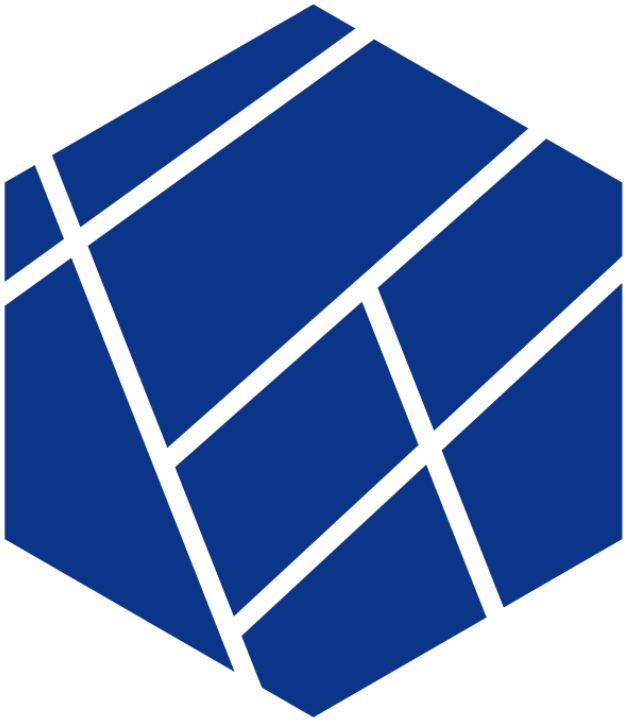
[LTD22] LinkSmart Thing Directory. Available online: https://github.com/linksmart/thing-directory (accessed on 1 November 2022).

[WoTD22] Cimmino, Andrea, et al. "Web of Things (WoT) Discovery, W3C First Public Working Draft 24 November 2020. W3C First Public Working Draft." World Wide Web Consortium (W3C). http://www. w3. org/TR/2020/WD-wot-discovery-20201124 (2020).

Annexe C: Component Deployment

Component	Comment	Production Hosting	Runtime environment	Unit of deployment
Industreweb Collect	Factory Connector component that can be used in a production facility	Industreweb Server within the production facility	Windows, .Net	
Symphony HAL	Factory Connector component that can be used in a production facility	Cloud	Cloud	
Dynamic Factory Connectivity Service	Factory Connector component that can be used in a production facility	Local in factory	Windows, Linux	
TSMatch Gateway				
Factory Connector Gateway Management Tool		Integrated with EFPF portal/alternative standalone deployment	Web application	
Symphony Event Reactor	Standalone application for triggering events and generating alarms	Cloud	Cloud	Linux executable
Symphony Data Storage	Data Storage standalone application	Cloud	Docker	Docker Image
Symphony Visualization App	HMI standalone application	Cloud	Cloud	Linux executable
Data Model Transformation Tool Suite	Foundation tool providing support to EFPF system integrators/developers	Cloud	Cloud	Docker image
Secure Data Store Solution	Store collected sensor data for later analysis	EFPF production server, local in factory	Docker	Docker Image
The System Security Modeler	Tool for identifying security risks in data pipelines	UoS-ITI infrastructure	Service	JAR
Blockchain Framework		Distributed on consortium nodes	Docker	Docker image
Distributed Workflow and Business	WASP process design and execution platform, together with integrated service marketplace	ICE server	Docker	Docker images

Process Design and Execution				
Industreweb Global	Administration and Application delivery framework for Industreweb Platform	Cloud	IIS, .Net	Docker image
Industreweb Visual Resource Monitoring Tool	AI Visual Detection solution for manufacturing environment	Industreweb Server within the production facility	Windows, .Net	Windows executable
Symphony Platform	Complete BMS Platform	Cloud	Cloud	Linux executable
Analytics Tool	Visual and data analytic tool that provides both predictive maintenance and supply chain optimization solutions	CERTH Server	Linux	Flask/Python
Anomaly Detection Tool	Data analytic tool with integrated machine learning algorithms and GUI	ICE Server	Docker	Docker images
Deep Learning Toolkit	Tools for data analytics	LINKS Server	Docker	Docker Image
Risk Tool	Tool for risk identification and assessment	ALM Server	Docker	Docker images
Symphony Resource Catalogue	Resource catalogue standalone application	Cloud	Cloud	Linux executable
Catalogue Service		SRFG Server	Service	
Software Development Kit		Standalone tools		



European Factory Platform

www.efpf.org