

## **EFPF: European Connected Factory Platform for Agile Manufacturing**



### **WP4: EFPF Add-Ons**

#### **D4.13: Smart Factory Solutions in the EFPF Ecosystem - I**

**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:** 2020-06-30

**Dissemination:** Public

**Status:** <Draft | Consortium Approved | EU Approved>

##### **Short Abstract**

The deliverable provides a report of the Smart Factory Tools and Services available within the EFPF Ecosystem. It describes the interconnectivity and the deployment approach for Tools within EFPF, followed by detail of each tool: its purpose, how it is configured and what it provides to an end user. Finally a description of how the tools can be created and used together to satisfy solutions with a broader scope.

Grant Agreement:  
825075



## Document Status

<b>Deliverable Lead</b>	Simon Osborne, C2K
<b>Internal Reviewer 1</b>	Mathias Axling, CNET
<b>Internal Reviewer 2</b>	Ingo Martens, HAW
<b>Type</b>	Deliverable
<b>Work Package</b>	WP4: Development of EFPF Building Blocks
<b>ID</b>	D4.13: Smart Factory Solutions in the EFPF Ecosystem - I
<b>Due Date</b>	2020-06-30
<b>Delivery Date</b>	2020-06-30
<b>Status</b>	<Draft   Consortium Approved   EU Approved>

## History

See Annex B.

## Status

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

## Further Information

[www.efpf.org](http://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 D4.13 ‘Smart Factory Solutions in the EFPF Ecosystem’ 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.

This deliverable reports on the work being carried out in the following tasks of the EFPF project that correspond to the thematic topic of 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

The primary reason for combining the outcome of multiple tasks into one deliverable is that; the solutions developed in all of the above tasks are interrelated and, in most cases, rely on each other for data or functionality. In this respect, all the above tasks deliver solutions that can be offered under the broad categories of Smart Factory/Digital Manufacturing and/or Industry4.0 solutions, which are highly desired by the user manufacturing companies.

This deliverable provides a description of the different tools along with their implementation details and high-level explanation on how the tools are deployed or offered in the EFPF ecosystem. The range of tools described in this deliverable cover factory connectivity, productivity, collaborations, smart factory operations and software development. The deliverable also describes how the available tools can interface with one another and sources of data. Details of the different tools is provided to a level where the users are able to understand what the tools does, how it is configured and what are the expected results.

The deliverable also provides a detailed overview of the EFPF Software Development Kit that can be used to create applications either by combining different Tools and Services or by creating new applications that can be exposed in the EFPF ecosystem through the integrated Marketplace framework.

## Table of Contents

0	Introduction .....	7
1	Introduction to Smart Factory Tools in the EFPF Ecosystem.....	10
1.1	Architecture Overview .....	10
1.2	Architecture Considerations and Implications.....	13
2	Connectivity and Data Flow in the EFPF Ecosystem .....	16
2.1	Data Model Interoperability .....	16
2.1.1	Designing for interoperability .....	16
2.1.2	Tools for providing data model interoperability .....	17
2.2	Service Integration through Data Spine.....	18
2.2.1	Design-time Aspects and Activities.....	19
2.2.2	Run-time Aspects and Workflows.....	21
2.3	Communications Workflow .....	23
3	Deployment and Hosting.....	26
3.1	Deployment Process .....	26
3.2	Deployment Environments .....	27
4	Smart Factory Pilot Solution Requirements .....	29
4.1	Methodology .....	29
4.2	Aerospace .....	29
4.3	Furniture .....	31
4.4	Circular Economy .....	33
5	Factory Connectivity .....	37
5.1	Connectors and Gateways .....	39
5.1.1	Industreweb Collect .....	39
5.1.2	Symphony Hardware Abstraction Layer (HAL) .....	41
5.1.3	TSMATCH Gateway .....	42
5.2	Factory Connector Gateway Management Tool (FCGMT) .....	44
5.2.1	Configuration .....	45
5.2.2	Operation.....	46
6	Foundation Tools and Services .....	48
6.1	Introduction.....	48
6.2	Digital Tools for Smart Factory scenario.....	48
6.2.1	Symphony Event Reactor .....	48
6.2.2	Symphony Data Storage.....	52
6.2.3	Symphony Visualization App .....	53
6.2.1	Data Model Transformation Tool Suite .....	54
6.2.2	Secure Data Store Solution .....	55
6.2.3	The System Security Modeler.....	60
6.3	Collaboration Tools.....	63
6.3.1	Blockchain Framework .....	63
6.3.2	Distributed Workflow and Business Process Design and Execution.....	65
7	Productivity Tools.....	68
7.1	Introduction.....	68
7.2	Industry 4.0 Tools .....	68
7.2.1	Industreweb Global.....	68
7.2.2	Industreweb Visual Resource Monitoring Tool .....	73
7.2.3	Symphony Platform .....	75
7.3	Data Analysis.....	77

7.3.1	Anomaly Detection Solution.....	78
7.3.2	Visual Analytics Tool for Predictive Maintenance and Optimization of Supply Chain Planning Activities .....	88
7.3.3	Deep Learning Toolkit for Data Analytics.....	98
7.3.4	Customer Trend Analysis .....	102
7.3.5	Siemens Data Analytics Solutions .....	106
7.3.6	Risk Tool.....	110
7.4	Asset Management.....	115
7.4.1	Symphony Resource Catalogue .....	115
7.4.2	Catalogue Service .....	117
8	Smart Factory Solutions.....	120
8.1	Composite Applications .....	120
8.1.1	Case Study .....	122
8.2	Software Development Kit .....	125
9	Conclusion and Outlook.....	132
	Annexe A: History .....	133
	Annexe B: References .....	135
	Annexe C: Component Deployment .....	136

## 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: [www.efpf.org](http://www.efpf.org)

In order to foster the growth of a pan-European platform ecosystem that enables the transition from “analogue-first” mass production, to “digital twins” and lot-size-one manufacturing, the EFPF project will design, build and operate a federated digital manufacturing platform. The Platform will be bootstrapped by interlinking the four base platforms from FoF-11-2016 cluster funded by the European Commission, early on. This will set the foundation for the development of EFPF Data Spine and the associated toolsets to fully connect the existing platforms, toolsets and user communities of the 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 “D4.13: Smart Factory Solutions in the EFPF Ecosystem - I” 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 of what each Connector, Gateway, Tools and Services offer, how it is configured and what outcome is provided to the end-user.

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. During the project, should new tools and services be developed or become available for inclusion in the EFPF Ecosystem, descriptions of each may be added for reference in the final version of the deliverable due at the end of 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:



- **D3.11 - EFPF Data Spine Realisation - I:** Provides details of the architecture used within the EFPF project to establish a federation of multiple platforms, tools and services.
- **D2.3 – Requirements of Embedded Pilot Scenarios:** Provides details of the user requirements captured in the EFPF project

## 0.5 Document Structure

This deliverable is broken down into the following sections:

- **Section 1: Introduction to Smart Factory tools in the EFPF ecosystem:** An introduction to this deliverable including an introduction to User Requirements, followed by the Deployment Approach and the Service Integration and Communications Workflow.
- **Section 2: Connectivity and Data Flow in the EFPF Ecosystem:** Provides a description of the approach connecting to data within the manufacturing environment, and explains the solutions available in the ecosystem that an end-user can utilise to make data available.
- **Section 3: Deployment and Hosting:** Describes the deployment hosting environments and solution provided in the EFPF project
- **Section 4: Smart Factory Pilot Solution Requirements:** Describes the user requirements captured in the project and the mapping of these requirements against the technical solutions
- **Section 5: Factory Connectivity:** Describes the factory connectivity solutions available in the EFPF federation. These solutions enable the extraction of data from manufacturing assets and utilisation of this data in smart factory solutions
- **Section 6: Foundation Tools and Services:** Describes the background services available in the EFPF federation. These services provide the connectivity, data modelling and security features in smart factory solutions
- **Section 7: Productivity Tools:** Describes the Tools and Services made available within the ecosystem that offer solutions to make productivity gains, improve quality, optimise processes, reduce waste and monitor or visualise KPI's.
- **Section 8: Smart Factory Solutions:** Describes the features and approach that an End User can take to create Smart Factory solutions by linking tools and services together to achieve a more comprehensive functionality.
- **Section 9: 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” since it provides information to a wider audience inside and outside the EFPF project.



## **0.7 Document Dependencies**

This document is the first of the two deliverables that describe the technical status of Smart Factory solution with the EFPF Ecosystem. This first deliverable submitted at Month 18 of the EFPF project describes the current technical achievements and tools and services. The second and final deliverable at Month 48 provides the final status of Smart Factory solutions and will provide documentation of how Smart Factory solutions have evolved to satisfy the needs of both project pilots and open call experimenters.

## **0.8 Glossary and Abbreviations**

A definition of common terms related to EFPF can be found at:

<https://www.efpf.org/glossary>

## **0.9 External Annexes and Supporting Documents**

None

## **0.10 Reading Notes**

None

# 1 Introduction to 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 address the diverse (e.g. digitalisation, shop-floor connectivity 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. The current list of tools and services available is described in this deliverable, outlining what is offered, how it is configured and what the resultant output from using the tools would be for an end-user. 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 Overview

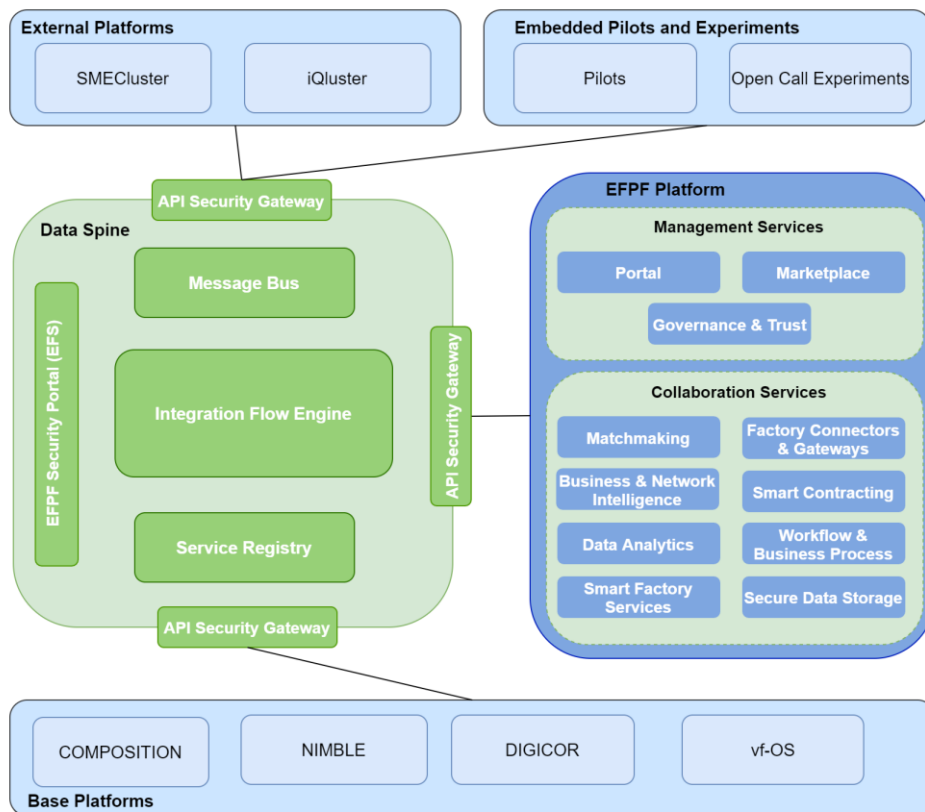


Figure 1: High-level Architecture of the EFPF Ecosystem

Figure 1 presents an overview of the high-level architecture of EFPF ecosystem. The EFPF ecosystem is based on a federation model, which consists of distributed platforms, tools and components provided by several partners and an integration and communications layer called 'Data Spine'. These individual components communicate through the central Data Spine and thus, the EFPF ecosystem follows Service-oriented architecture (SOA) style. The main elements in the EFPF federation are introduced below. The detailed description of the architecture can be found in D3.1: EFPF Architecture-I and D3.2: EFPF Data Spine Realisation – I.

- **Data Spine:** The Data Spine provides the interoperability infrastructure that can be used to interlink and bridge the interoperability gaps between the tools and services of different platforms. It adheres to common industry standards and follows a modular approach to enable the creation of a modular, flexible and extensible ecosystem. The interoperability requirements of the tools and services from different platforms were surveyed and based on the results, the conceptual components of the Data Spine were defined as shown in Figure 1. The details of this survey can be found in D3.1: EFPF Architecture-I. The Integration Flow Engine component is intended to bridge the interoperability gaps between heterogeneous tools at protocol and data model level while EFS (EFPF Security Portal) together with API Security Gateway facilitates security interoperability. The Message Bus enables asynchronous communication in the EFPF ecosystem, and the Service Registry facilitates the lifecycle management of service metadata including their API specifications.
- **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 a Web-based portal. The tools and services brought together in the EFPF platform are the market ready or reference implementations of the Smart Factory and Industry 4.0 tools from project partners.
- **Base Platforms:** The four base platforms (COMPOSITION, DIGICOR, NIMBLE and vf-OS) in EFPF are funded by the European Commission's Horizon 2020 program within the Collaborative Manufacturing and Logistic Cluster (FoF-11-2016).
- **External Platforms:** In addition to the four base platforms, the EFPF ecosystem enables interlinking of other platforms and open-source tools that address the specific needs of connected smart factories. The external platforms that joined the EFPF ecosystem at the beginning of the project are: ValueChain's iQcluster platform<sup>1</sup>, Nextworks' Symphony platform<sup>2</sup>, and SMECluster's Industreweb platform<sup>3</sup>
- **Pilots and Experiments:** These are the components and systems that will interact with the EFPF ecosystem (including the EFPF platform and the Data Spine) during the course of the project

<sup>1</sup> <https://valuechain.com/supply-chain-intelligence/iqcluster>

<sup>2</sup> <https://www.nextworks.it/en/products/brands/symphony>

<sup>3</sup> <https://www.industreweb.co.uk/>

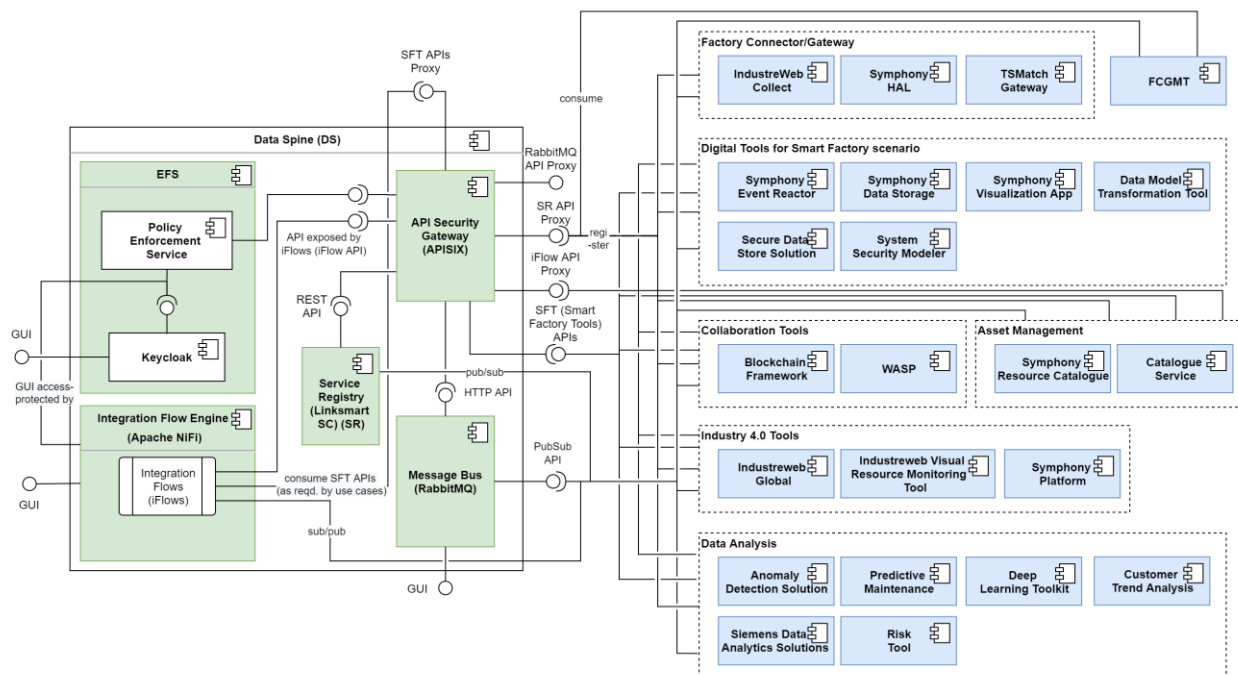


Figure 2: Interaction between the Smart Factory Tools and the Data Spine

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.

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 in the EFPF ecosystem. The API Security Gateway (APISIX), acting as the policy enforcement point, intercepts all the traffic to the Data Spine and invokes the EFS for authentication and authorization decisions. Thus, EFS together with APISIX are responsible for providing an SSO facility across 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 and their proxy APIs resulting from the integration flows are secured by the EFS through the API Security Gateway. In the figure, the APIs 'SFT APIs Proxy' and 'iFlow API Proxy' exposed through the API Security Gateway are the secure proxy APIs for 'SFT APIs' (the APIs offered by different Smart Factory Tools) and 'iFlow API' respectively.

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 Smart Factory Tools' APIs (SFT APIs) are registered to the Service Registry, the API Security Gateway automatically creates secure proxy APIs ('SFT Proxy APIs' in the figure) for them. The secure SFT Proxy APIs can be invoked directly by different tools/services or through the integration flows.

Finally, 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. The secure proxy of the Message Bus' HTTP API 'RabbitMQ API Proxy' is used by administrators to configure access policies for different users and topics.

## 1.2 Architecture Considerations and Implications

Figure 1 shows the conceptual components of Data Spine, the central gluing mechanism in the EFPF ecosystem and Figure 2 shows the technologies used to realise these conceptual components and interactions between these components. The reasoning behind the identification of these conceptual components, the methodology followed for the selection of technologies to realise these components and the quality assessment of the architecture are explained in detail in D3.1: eFactory\* Architecture-I and D3.2: eFactory\* Data Spine Realisation – I. The summary of architectural considerations and implications in general and also from the perspectives of Smart Factory Tools and system integrator users is presented below:

*\*After the publishing of these deliverables, the eFactory acronym is replaced by EFPF*

- **Federation model:** The EFPF ecosystem architecture is based on a federation model, meaning that interoperability between different tools/services needs to be established "on-demand" i.e. when required by a use case through an integration flow. As there is no common data model or format imposed, 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. But, on the downside, a pair of services need to create an integration flow to interoperate and communicate with each other.
- **Agility and flexibility:** The use of Data Spine and the integration flows in particular to establish interoperability allows the tools/services to be loosely coupled. This allows the tools/services to have neutral APIs 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 services is a standard practice; however, successful collaboration depends upon service providers adhering to the semantic versioning<sup>4</sup> standard recommended by the EFPF ecosystem to version their APIs and conveying plans to deprecate/upgrade their APIs to the service consumers in advance. To enable service consumers to monitor changes to the APIs they consume, a new tool 'Interface Contracts Monitoring Tool' is being developed.
- **Usability:** 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

<sup>4</sup> <https://semver.org/>

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 and a multi-tenant authorization capability that enables different groups of users to command, control, and observe different parts of the dataflow, with different levels of authorization.

- **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. In addition, to transform between different data models, it provides three data transformation processors: JoltTransformJSON, TransformXml and ExecuteScript. Thus, 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.
- **API Management:** The system integrator users need to refer to the API specifications of services to create integration flows. The Data Spine provides Service Registry component to store and retrieve service metadata including the API specifications. To ensure uniformity across and completeness of the API specifications, the EFPF project recommends the use of OpenAPI Specification<sup>5</sup> standard for specifying the APIs of services that follow synchronous request-response communication pattern and AsyncAPI Specification<sup>6</sup> standard for specifying APIs of services that follow asynchronous publish-subscribe communication pattern. Thus, this implies that the service providers need to register their services to the Service Registry and follow the proposed standards to specify the APIs of their services.
- **Modularity and extensibility:** The architecture of the EFPF Data Spine and platform has been designed with modularity and extensibility in mind to meet the need for incorporating new tools in the EFPF platform and external platforms in the EFPF ecosystem, with minimum effort. The components of the Data Spine communicate with each other through standard interfaces and protocols and are modular in nature. Support for new functionality such as protocols can be added by developing new processors/plugins. In this way, the Data Spine adheres to common industry standards and follows a modular approach to enable the creation of a modular, flexible, and extensible ecosystem.
- **Performance and scalability:** The EFPF ecosystem makes it easy to integrate new tools/services through the use of Data Spine and promotes reusability. As the Data Spine is a central entity, it should be highly performant and should support high throughput. The performance critical components of the Data Spine have the capability to operate within a cluster and will be deployed within a cluster in the Production environment of the EFPF ecosystem.
- **Availability and monitoring:** The failure of central components such as the Data Spine could impact a significant part of the EFPF ecosystem. While the clustered deployment of such crucial components mitigates such risks, it becomes necessary to monitor failures, unusual behaviour such as sudden spikes in resource consumption (CPU, memory, etc.) to ensure high availability. A new 'Monitoring/Alerting System' is being

---

<sup>5</sup> <https://swagger.io/specification/>

<sup>6</sup> <https://www.asyncapi.com/docs/specifications/2.0.0>

developed to monitor the availability and resource consumption of the components of EFPF ecosystem and generate alerts if such anomalies are detected.

- **Maintainability:** The loosely coupled and modular nature of the EFPF ecosystem helps significantly towards its maintainability. A high-quality documentation of the Data Spine and the Smart Factory Tools is being published in the ‘EFPF Documentation Portal’<sup>7</sup>.

---

<sup>7</sup> <https://docs.efpf.linksmart.eu/>



## 2 Connectivity and Data Flow in the EFPF Ecosystem

This Section describes service integrations and data flow in the in the EFPF ecosystem. It highlights the integration aspects from the perspectives of Service Providers and Service Consumers.

### 2.1 Data Model Interoperability

#### 2.1.1 Designing for interoperability

The objective of the data model interoperability layer is supporting information exchange and business processes that spread across two or more of the existing EFPF platforms.

Two options are available to provide interoperability between incompatible data models.

The first option would be to use a set of standard data models on the Data Spine which can be interpreted as using a one-to-one approach when dealing with data model incompatibilities as shown in Figure 3. When using this approach, data coming from a given tool/service using a proprietary data model is converted to a standard data model appropriate for the tool/service use case. A set of considered standard data models is available in D3.2 “EFPF Data Spine Realisation”. This data can be then directly consumed by tools adopting the given standard data model or can be converted back to different proprietary data models.

This first approach can be advised in a scenario in which the data producers want to attract many different data consumers (e.g. shop floor data that can be consumed by different analytics services). In fact, this approach allows for more modularity and change resilience at cost of a little more complexity due to the higher number of data model translations with respect to the following option.

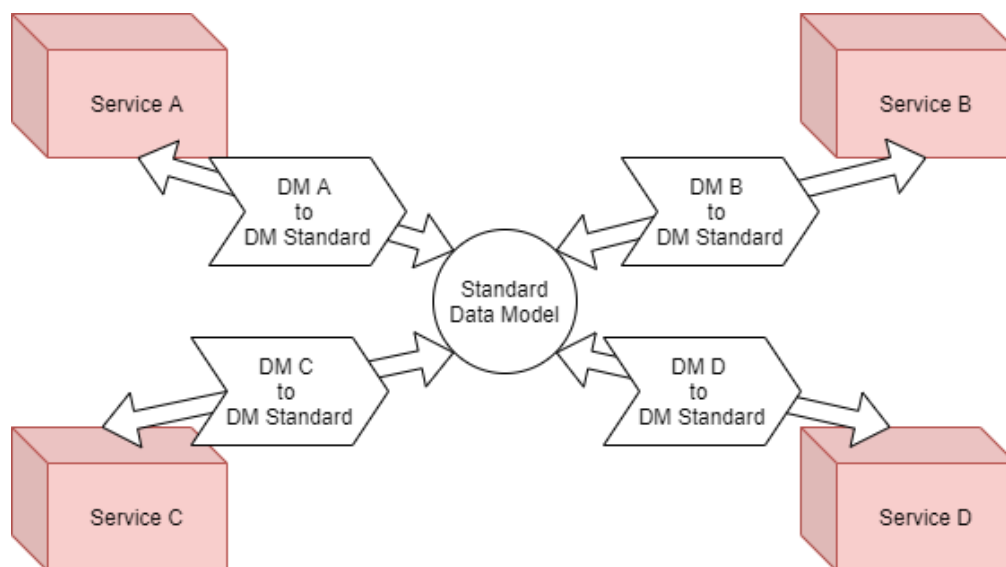


Figure 3: Use of a standard data model

A second approach would be to use a one-to-many translation strategy in which each tool/service has to directly deal with all the data models used from the other tools/services as shown in Figure 4. With this approach, the data models are directly converted from the proprietary data model used by the tool/service producing the data to the data model used by the tool/service consuming the data.

This second approach is advised if, differently from the first scenario, there is the need to connect a few tools /services together. The benefits of the first approach are greater in a scenario with many interconnected tools/services. In this second scenario, the developers do not have to deal with the complexity of a central standard data model at the cost of less interoperability with future service/tools additions compared to the first scenario.

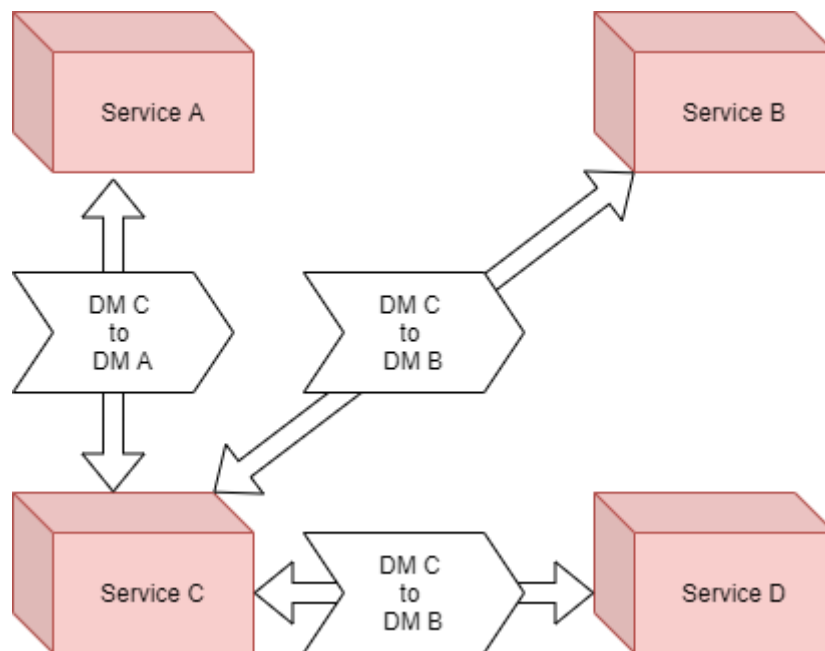


Figure 4: Direct conversion between proprietary data models

### 2.1.2 Tools for providing data model interoperability

The EFPF ecosystem makes use of the Data Spine to ensure that data model interoperability is achieved. By using the EFPF Data Spine these transformation tools can easily have access to all of its components such as the Message Broker, the API Security Gateway, or the service registry this makes obtaining and giving access to transformed data easy and secure.

Regardless of the chosen integration strategy, different technical solutions are provided to developers to easily integrate their tools and services by transforming their data models. This data model transformation process is transparent from the point of view of the tools/services and can happen both on the Integration Flow Engine and outside of it on dedicated infrastructure. Some of the options provided are:

- **JOLT:** To perform JSON to JSON data model conversion, Apache NiFi, the dataflow management platform that is used to realize the Integration Flow Engine of the Data

Spine contains a processor capable of running Jolt scripts. Therefore, it can be easily added to an integration flow to perform data transformation. JOLT (JsOn Language for Transform) is a JSON to JSON transformation library. The Jolt specifications for performing the transformation is also written in the form of a JSON document. Among the different transform option, the shift transform option thanks to the inclusion of several 'wildcards' is the most powerful and easy transformation provided by JOLT, it enables the user to perform about 80% of the typical transformation work.

- **XSLT:** If the need arises to transform data encoded as XML, an option is to use the TransformXml processor in NiFi which can transform an XML payload obtained from a flow file using a provided XSLT configuration file. XSLT, a WC3 standard, is a language for transforming XML documents into other XML documents. It is used in conjunction with XPath 2.0 to write rules to match parts of a parsed XML document and compile a new XML document.
- **Ad-Hoc microservices:** In case the developers do not want to use any of the options provided above an option is to use a dedicated microservice. This option fetches the data from the Integration Flow Engine in the Data Spine towards a custom developed and maintained component which needs to be hosted outside from the Data Spine. There the data can be transformed from a custom to a standard data model. This process can be performed using a framework of choice from the developers without the need for it to be supported by the Integration Flow engine. Once transformed the data can be then pushed back to the Data Spine to be used from other services with the options provided by the EFPF ecosystem.

The party responsible for the data model integration can be the data producer, data consumer, or integration company. To achieve data model interoperability, various resources and documentation on how to use the different data model transformation tools are provided on the EFPF Documentation Portal<sup>8</sup>. Information provided includes example transformations using the different technologies described for translating standard data models and blueprints for adapting the provided examples to different use cases. More details and an in depth description of some example transformation can be found D3.2 ("EFPF Data Spine Realisation").

## 2.2 Service Integration through Data Spine

Data Spine is the central entity or gluing mechanism that interlinks and establishes interoperability between the services of different platforms in the EFPF ecosystem. In order for a pair of heterogeneous services in the EFPF ecosystem to communicate with each other, they need to be integrated through the Data Spine at first. Figure 5 depicts the Data Spine along with its internal components and also shows its relation to the platforms, services and tool such as the Factory Connectors, etc. A detailed description of the Data Spine can be found in the deliverable D3.11 EFPF Data Spine Realisation - I.

The subsequent Sections elaborate the steps to be followed by services for integration through the Data Spine and describe the service-to-service communication that happens at run-time through the Data Spine in the form of workflows once they are integrated.

<sup>8</sup> <https://docs.efpf.linksmart.eu/>

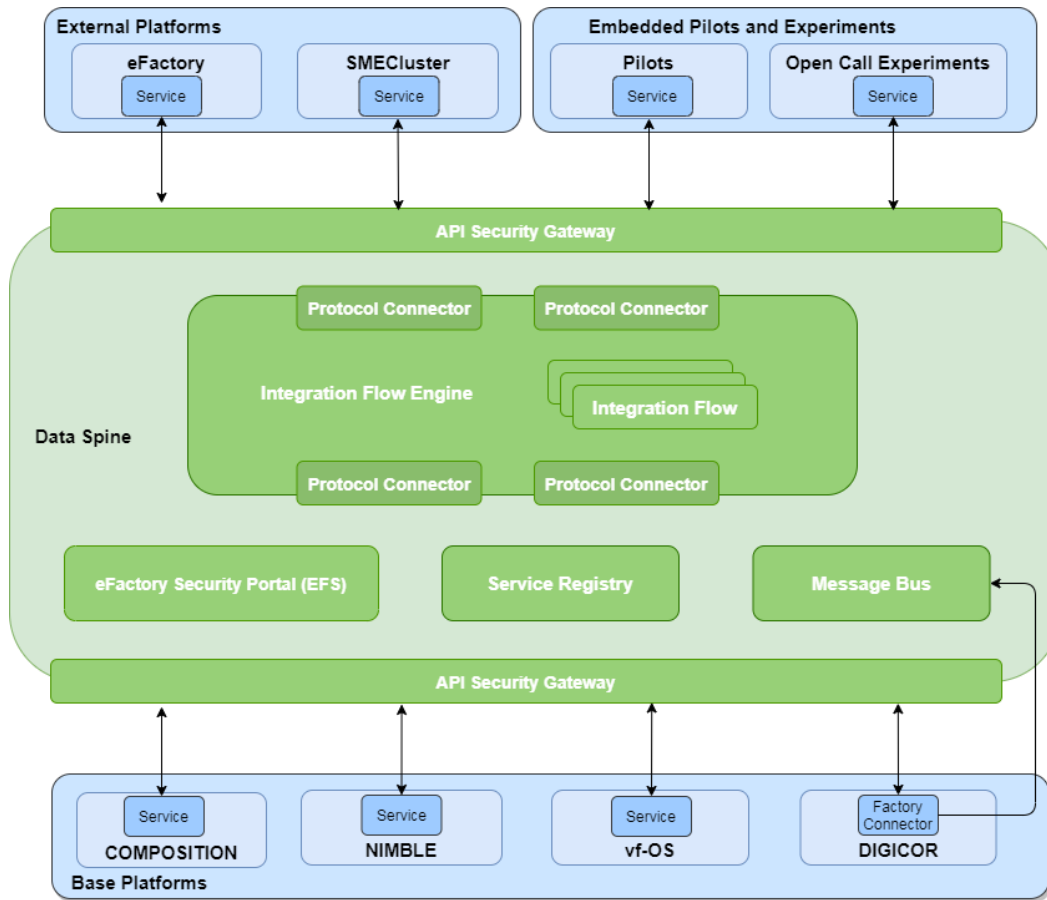


Figure 5: High-level Architecture of the Data Spine

### 2.2.1 Design-time Aspects and Activities

This Section lists the activities that the service providers need to do in order to provide their services through the Data Spine and the activities the service consumers need to do in order to consume the services provided through the Data Spine with the help of examples. These design-time aspects become the prerequisites to enabling communication through the Data Spine. This section is further divided into 'Synchronous Communication' and 'Asynchronous Communication' as the activities differ for both.

#### Synchronous Communication

Prerequisites:

The service provider 'Provider1' and service consumer 'Consumer1' are both EFPF users. Provider1 and Consumer1 have the necessary access permissions required to register services to the Service Registry.

Design-time activities:

1. Provider1 registers their service 'Service1' to the Data Spine Service Registry with an appropriate service 'type'. For simplicity, let us assume that Service1 has only one API endpoint 'EP1'.
2. An EFPF administrator user 'Admin1' defines/configures the access permissions for accessing EP1 in the EFS.

3. The API Security Gateway (ASG), which checks for new service registrations/updates to services in the Service Registry periodically, creates a proxy endpoint/route 'EP1<sub>P</sub>' for EP1 in ASG – this could be used to invoke the endpoint directly without creating an integration flow in the case if protocol translation and/or data transformation isn't needed (here it is assumed that data transformation is needed).
4. Consumer1 decides to consume Service1 and gets the technical metadata for service1 including its API spec from the Service Registry.
5. Consumer1 requests for and acquires the necessary access permissions to invoke EP1.
6. Consumer1 creates an integration flow using the GUI of the Integration Flow Engine to invoke EP1, perform data transformation and finally create an 'interoperability-proxy' endpoint EP1-C for EP1 in the integration flow.
7. Consumer1 registers this new EP1-C endpoint to the Service Registry.
8. ASG creates a proxy endpoint EP1-C<sub>P</sub> for EP1-C.
9. Consumer1 requests for and acquires the necessary access permissions to invoke EP1-C<sub>P</sub>.
10. Provider1's service and Consumer1's service are now integrated through the Data Spine and can start communicating with each other.

### Asynchronous Communication

Prerequisites:

The publisher 'Publisher1' and subscriber 'Subscriber1' are both EFPF users. Publisher1 and Subscriber1 have the necessary access permissions required to register services to the Service Registry.

Let us assume that publisher1's entity that publishes/intends to publish to the Data Spine Message Bus is 'fc1'.

Design-time activities:

1. If fc1 is a Factory Connector/Gateway (FCG), Publisher1 logs in to the EFPF Marketplace and purchases this Factory Connector fc1 there; else, step 1 is skipped.
2. Publisher1 gets a root topic name (for Publisher1's company) 'p1'.
3. Publisher1 logs in to the Factory Connector/Gateway Management Tool (FCGMT) and creates a sub-topic 'Topic1' under p1 to publish to and gets a key 'pk' for publishing to that topic 'p1/Topic1'.
4. Publisher1 configures fc1 to publish to Data Spine Message Bus over the topic 'p1/Topic1' using the key pk.
5. Publisher1 registers its service 'Service1' that consists of an API containing this publication information (topic 'p1/Topic1', associated data model, etc.) to the Service Registry.
6. Subscriber1 decides to subscribe to Service1's topic 'p1/Topic1' and gets the technical metadata for Service1 including its API spec from the Service Registry.
7. Subscriber1 requests for access permissions to subscribe to p1/Topic and gets the key pk for the same.

8. Subscriber1 creates an integration flow using the GUI of the Integration Flow Engine to subscribe to p1/Topic, perform data transformation and finally to publish the resulting data to Message Bus over the topic 's1/topic1' using the key sk ('s1' is Subscriber1's root topic and he/she obtains the same along with the key 'sk' following the same procedure as publisher1).
9. Subscriber1 registers his/her service with the APIs containing its subscription and publication information to the Service Registry.
10. Publisher1's service and Consumer1's service are now integrated through the Data Spine and can start communicating with each other.

## 2.2.2 Run-time Aspects and Workflows

This Section describes the service-to-service communication that happens at run-time through the Data Spine in the form of workflows. In order to enable such a communication, the participant services need to be integrated through the Data Spine first by following the design-time activities enlisted in the previous section. Therefore, in a way, the workflows described in this Section are a continuation of the activities mentioned in the previous Section. This section is further divided into 'Synchronous Communication Workflow' and 'Asynchronous Communication Workflow' as the workflows differ for both.

### Synchronous Communication Workflow

1. Consumer1's service invokes EP1-C<sub>P</sub> endpoint/route of ASG.
2. ASG, acting as a reverse proxy, checks with EFS to ensure that Consumer1 has the necessary permissions to access EP1-C<sub>P</sub> and to perform the requested operation.
3. ASG, upon receiving a positive reply from EFS, invokes EP1-C exposed by the integration flow.
4. Integration Flow Engine does the necessary request (payload, query parameter, path parameter and/or header) transformation as defined by the integration flow and finally, invokes the external endpoint EP1.
5. Upon receiving the response, the Integration Flow Engine performs payload data transformation as defined by the integration flow and finally, sends the resulting payload as a response to ASG.
6. ASG sends the received response to Consumer1's service.

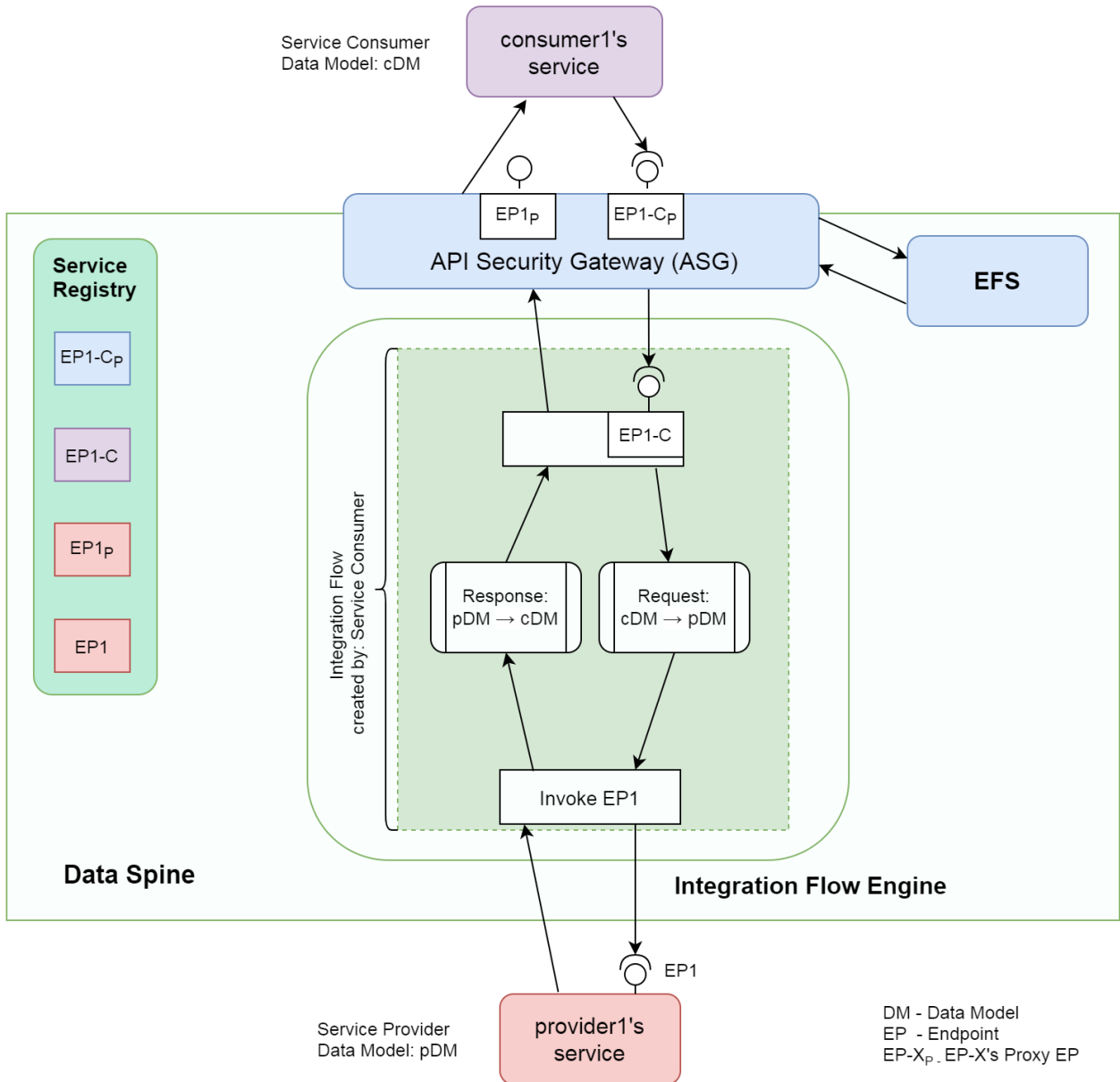


Figure 6: Example of Synchronous Communication Workflow

### Asynchronous Communication Workflow

1. Publisher1's service publishes data to the Data Spine Message Bus over topic p1/Topic1 with the key pk.
2. The integration flow created by Subscriber1 subscribes to this topic p1/Topic1 using the key pk, transforms the payload data from Publisher1's data model pDM to its own data model sDM and finally, publishes this transformed payload data to the Message Bus over topic s1/Topic1 with the key sk.
3. Subscriber1's service subscribes to the topic s1/Topic offered through the Message Bus using key sk and starts receiving the data.



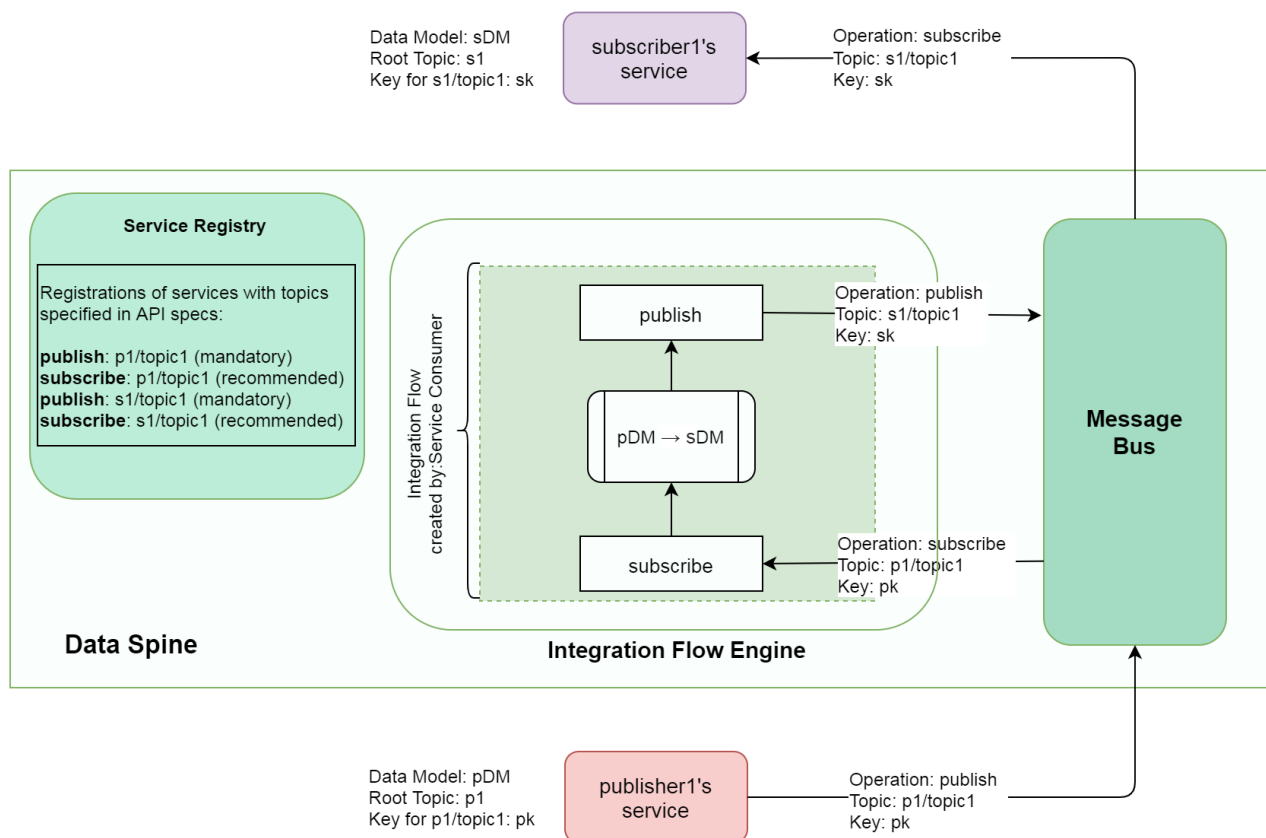


Figure 7: Example of Asynchronous Communication Workflow

## 2.3 Communications Workflow

The API service security is handled by the API Security Gateway (ASG). The ASG acts as a border gateway ahead of all API calls to the Data Spine. Its role is to enforce security policies on the service calls. In EFPF, ASG is implemented using Apache APISIX. The following diagram (Figure 8) shows the basic communication flow of services to the EFPF Data Spine.

The ASG automatically creates the routes for services via reading the data spine's service registry. Any routes which are not exposed in the ASG will result in a 404 Not found response. The API Gateway has two custom plugins for security enforcement.

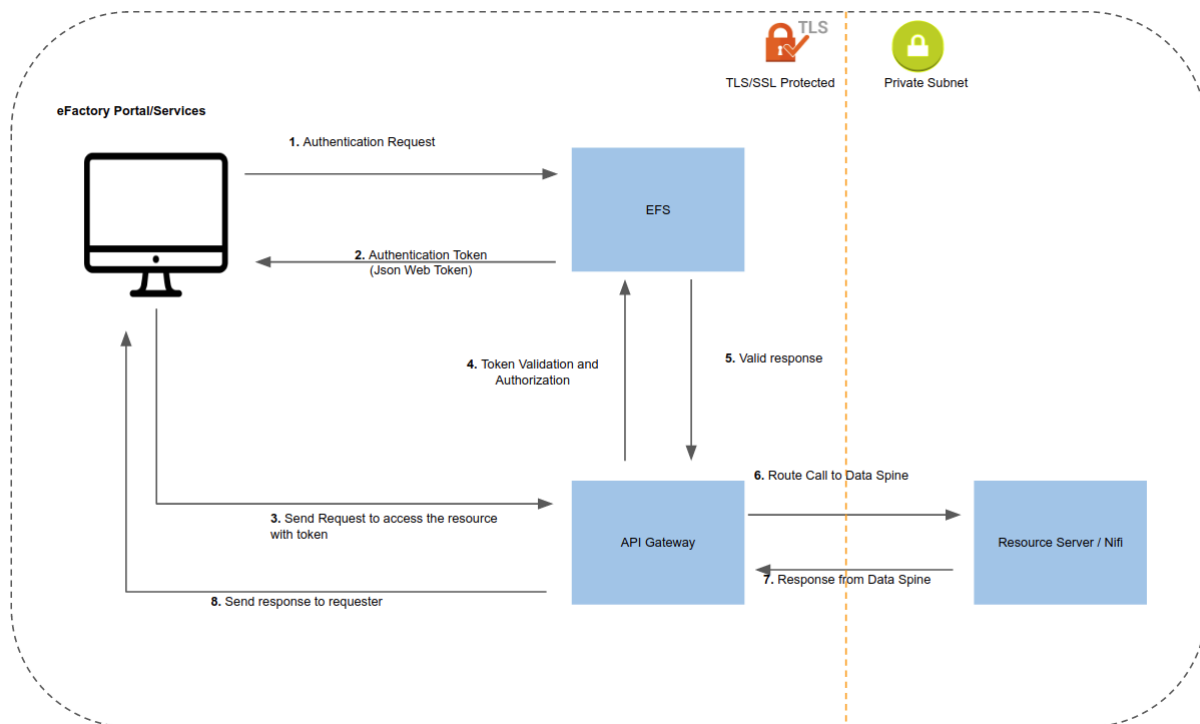


Figure 8: ASG Communication Flow

#### 1. Open ID Connect plugin

- a. The Open ID connects plugin provides token introspection. The token introspection can be done via communicating with the identity server or importing the public key of the token to the Open ID connect plugin. The introspection plugin will verify the token is generated from the EFPP identity server and does basic authorization via JSON web token scopes.

#### 2. Policy enforcement plugin

- a. Policy enforcement plugin provides additional security for the routes defined by the ASG. The identity server allows defining policies based on the user's role or attribute. The policy enforcement plugin will communicate with the policy engine to allow or reject the call based on users privileges.

### Pub/Sub Security

The admin API of the message broker is also protected by the ASG. During the purchase of a Factory connector (FC) or an IoT device, the EFPP Security Portal (EFS) assigns a root topic for the FC. From there onwards, the user is only allowed to create sub-topics under the designated root topic. Any attempt to create a subtopic/publisher/producer will result in a 401 unauthorized access code via the ASG. The following sequence diagram shows how the ASG protects the admin API of the message broker in the Factory Connector management scenario.

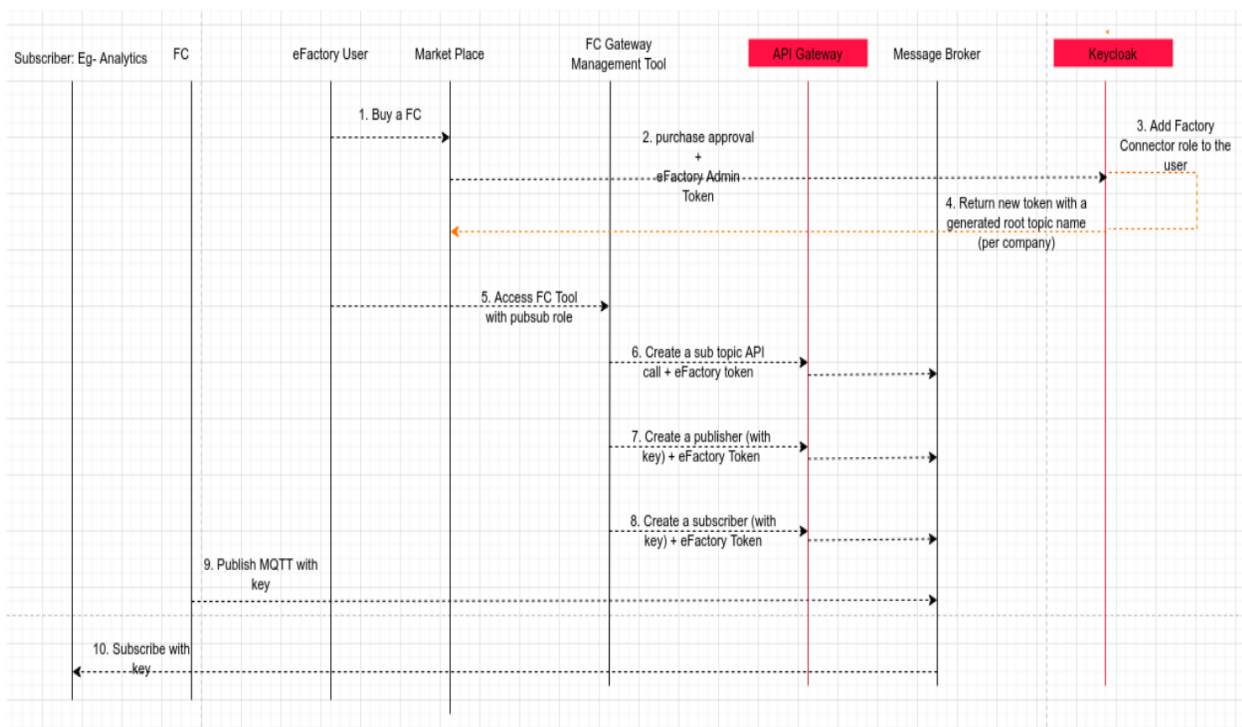


Figure 9: Pub/Sub Security Workflow

### 3 Deployment and Hosting

#### 3.1 Deployment Process

The EFPF platform is heterogeneous, and the development environment and platforms for components differ. The core components (framed in Red Figure 10) will have specified hosting, deployment and operational requirements.

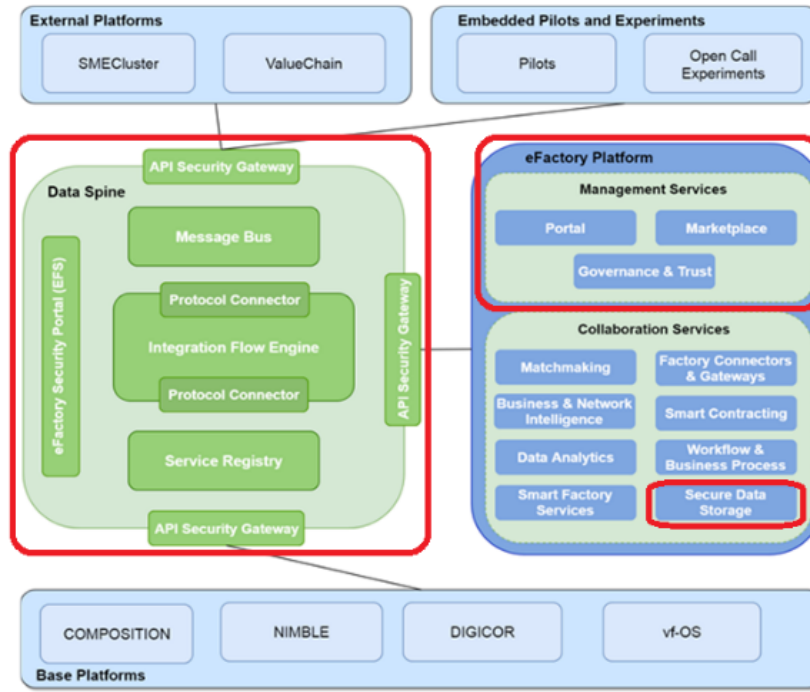


Figure 10: EFPF Platform and core components.

A GitLab repository is currently used for project planning, source code management as well as continuous integration (CI), continuous delivery (CD) and monitoring. The core components are migrating towards this setup. The continuous integration and testing environment are available to all EFPF base platform and service providers but mainly used for the core EFPF infrastructure. The repository will be successively populated and adapted for integration of additional external base platforms and/or tools.

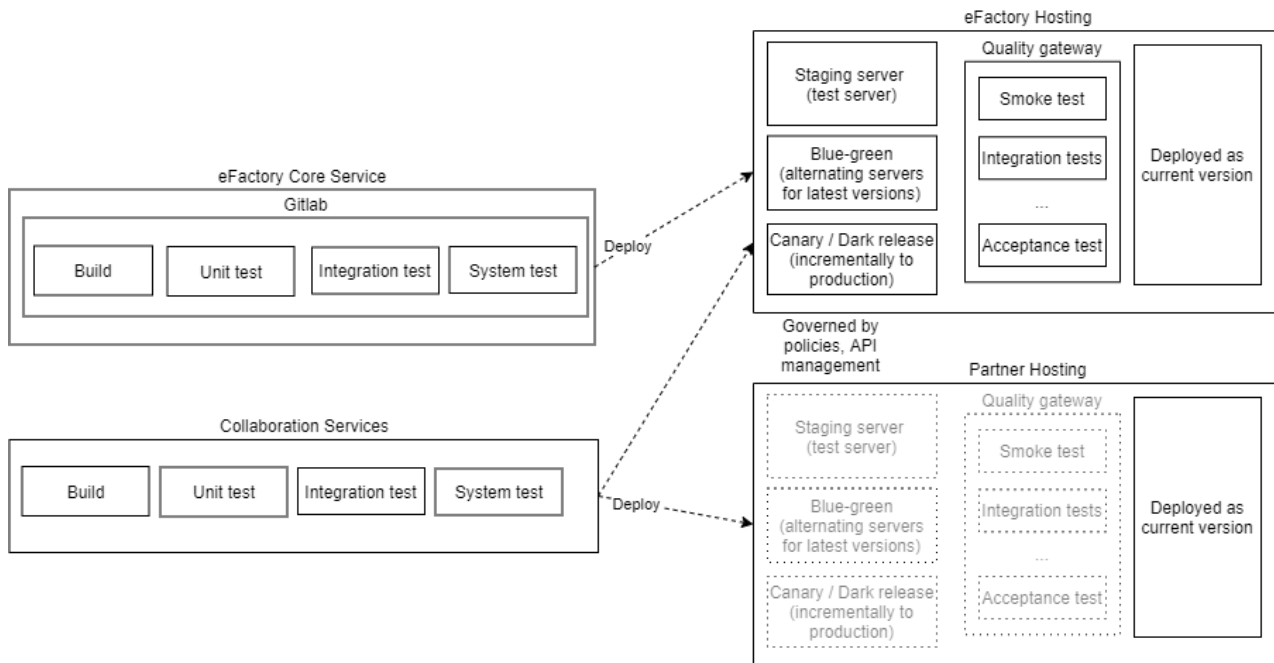


Figure 11: Deployment configuration for EFPF services

The deployment process (shown in Figure 11) 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.

The core components will have a defined deployment pipeline and a common hosting solution. Although Docker images are the recommended unit of deployment, other components may be deployed in various ways and on different platforms. Some are only provided as services in the EFPF ecosystem and will be deployed and managed by the developing partner. The table in Annex C provides an overview of the runtime environments and unit of deployment of the components.

### 3.2 Deployment Environments

Starting with the Data Spine, a 3-tier deployment model is being 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 **development** environment is a less stable environment where, e.g. component connectors, security mechanisms and network topology may change. There may be in-place changes by developers in the development environment and no quality gateways - apart from unit tests - are needed to deploy anything there. Malfunctions and conflicting versions are solved by developer-to-developer communication on instant messaging channels.

The **testing** environment serves as a more stable environment for integration testing. Although this is not the intended use, some performance testing can also be done there. Components are deployed to the test environment by script, using Docker images as a unit of deployment for EFPF hosted components. Configuration should be complete, and no in-place changes needed. If a new version does not pass the quality gateway, it is rolled back, corrected, and re-deployed, not edited in place.

Once the component passes the quality gateway, this version may be deployed to the **production** environment, which is hosted on a dedicated server. The policies for API management and a quality gateway apply in the production environment.

The three environments are completely isolated. No connections are set up between development, test and production. An exception could be factory connectors feeding data to the test environment.

An open issue is how to manage data in the test environment. Both streaming and static data is often needed for integration tests. Streaming data could be mirrored from the production environment (not recommended for security reasons), provided in a “playback loop” that provides a fixed sequence of observations and events that each component can use for test purposes, or each component can inject its own set of test data into relevant streams. Static data, e.g. in the Secure Data Store, should not be considered stable. Integration tests that change static data should set up and restore the data sets needed. Some global data may have to be versioned in sync with component versions, or policy of restoring all data on a set interval may be used. This will be selected depending on emerging needs and practices.

## 4 Smart Factory Pilot Solution Requirements

The EFPF ecosystem aims to offer tools and services in one place and to streamline the process of implementing solutions to production related issues. Within EFPF, the three pilots from the Furniture, Aerospace and Circular Economy domains have yielded a broad range of user requirements which should be satisfied using the available Tools and Services within the ecosystem in a federated manner. The process of mapping those requirements to the Tools and Services described later in this document are covered in this section, with a high-level analysis of the motivation behind the requirements.

### 4.1 Methodology

The User Requirements gathered from the three pilot partner domains were collated in “D2.3 – Requirements of Embedded Pilot Scenarios” and grouped according to Epics which reflect high level user goals. Within each Epic, User Stories and Tasks then reflected the granular functionality required including the definition of the stakeholders and Acceptance Criteria which were captured in Jira<sup>9</sup>. These were analysed collectively by WP4 partners in detail in order to map each User Story to Solutions comprised of one or more of the Tools or Services from the base platforms. These were integrated together using the EFPF Data Spine.

Once the Solutions were defined, development effort was allocated within WP4 to develop, enhance and integrate the building blocks (systems, tools, services, methods) into the EFPF platform. Where certain requirements were not satisfied by the existing offerings additional development was made to adapt or develop new services and tools to fulfil the requirement. The following sections recap the Pilot objectives followed by a view of the mapping between the Epics / User Stories and the Solutions / Tools intended to satisfy those requirements which utilise the Data Spine in order to orchestrate between them.

### 4.2 Aerospace

The Aerospace Pilot in the EFPF project addresses the ad-hoc setup of a production network involving Airbus Hamburg and local SME suppliers represented by the Hanse-Aerospace (HAW) association. This pilot focuses on realising two high-level scenarios:

- OEMs like Airbus is interested in rapidly integrating SME innovations into existing aircraft programs and building agile supply chains. Existing Airbus tools and platforms need complementary solutions for integrating SMEs directly in the supply chain. These solutions, such as tender decomposition, match-making, team building and smart contracting, can be provided by the EFPF platform
- SME clusters like Hanse-Aerospace and its member companies expect mature digital manufacturing tools for supporting agile collaborations between SMEs, shop-floor connectivity and data analysis. EFPF can provide a unified interface to distributed tools to address the diverse digitalisation needs of SMEs in the aerospace sector

The following table shows the Epics and User Stories for the Aerospace domain and the pilot solutions used to fulfil those requirements.

<sup>9</sup> <https://jira.fit.fraunhofer.de/jira/secure/Dashboard.jspa>



	<b>Epic / User Stories</b>	<b>Solutions / Tools</b>
1	<b>Joint Purchase</b> <ul style="list-style-type: none"> <li>Joint purchase of prepregs (raw material)</li> <li>Joint purchase of Energy</li> <li>Joint Purchase of personal protection equipment (PPE)</li> </ul>	<b>Product Catalogue with Federated Search</b> Defined Product / Service catalogues and share with other users on the EFPF platform <ul style="list-style-type: none"> <li>Catalogue Service (7.4.2) – Enable partner discovery allowing companies to list the products they supply and the services they provide. Used with the Federated Search Data Spine component.</li> </ul>
2	<b>Tender &amp; Bid Management</b> <ul style="list-style-type: none"> <li>Tender of prepregs which have reached the expiry date</li> <li>Tender of One-Way-Pallets</li> <li>Tender of Maintenance Services</li> </ul>	<b>Tendering Service</b> Collaborative environment to search for suppliers and publicise tenders <ul style="list-style-type: none"> <li>Tendering and Team Formation Service – Manage supplier capability profiles, business opportunity publication and associated bids with tools to allow secure document sharing and messaging (solution currently in development)</li> </ul>
3	<b>Parameter Monitoring in Production and Maintenance</b> <ul style="list-style-type: none"> <li>Temperature Monitoring Machine Environment</li> <li>Temperature Monitoring Freezer</li> <li>Vacuum Monitoring</li> <li>Tracking of Trolleys</li> <li>Visual Detection of PPE</li> </ul>	<b>Environment Monitoring</b> Capture environmental data from production processes where variations in environment have a significant effect on quality and repeatability <ul style="list-style-type: none"> <li>TSMATCH Gateway (5.1.3) – Provides real-time connectivity to machine sensors to publish to EFPF</li> <li>Symphony Data Storage (6.2.2) – Provides user visualisation of sensors data</li> <li>Symphony Visualisation App (6.2.3) – Provides user visualisation of sensors data</li> <li>FCGMT (5.2) – Provided Management of Gateway component within EFPF platform</li> <li>Event Reactor (6.2.1) – Manages Alerts to notify Users of machines exceeding recommended environmental working limits</li> </ul> <b>Resource Management</b> Detect staff compliance with the wearing of PPE in the workplace

		<ul style="list-style-type: none"> <li>• Industreweb Collect (5.1.1) – Provides real-time data collation and connectivity to EFPF</li> <li>• Industreweb Visual Resource Monitoring (7.2.2) – AI framework to trained to detect staff wearing PPE</li> <li>• FCGMT (5.2) – Provided Management of Gateway component within EFPF platform</li> <li>• SDSS (6.2.2) – Securely store sensor process usage and PPE compliance to enable real-time and historical analysis</li> </ul>
--	--	--

### 4.3 Furniture

The furniture manufacturing pilot envisions the creation of a Lot Size 1 Consolidation Center to control small scale stocking and consolidation points for large manufacturers. This pilot focuses on LAGRAMA as a large manufacturer/supplier and responsible for receiving and realising customer orders and for coordination of various suppliers. The pilot scenario is about LAGRAMA receiving an order for the furnishing and decoration of five rooms of a small thematic hotel, each of them inspired by different decorative concepts. LAGRAMA is expected to manufacture all furniture for each room (beds, bedside tables, wardrobe, bathroom furniture, etc.) and supply and coordinate the installation of the other elements required in each room (lighting, bedding, curtains, carpets and decoration objects). EFPF is expected to improve the efficiency of supply chain creation and operations based on the provisioning of digital tools and secure information exchange channels.

The following table shows the Epics and User Stories for the Furniture domain and the pilot solutions used to fulfil those requirements.

	<b>Epic / User Story</b>	<b>Solution</b>
4	<b>Search for products, services or supply partners to meet production objectives</b> <ul style="list-style-type: none"> <li>• Send and receive orders</li> <li>• Search of products/services</li> <li>• Request for quotation of customised product</li> <li>• Exchange product catalogue</li> <li>• Customer analytics</li> </ul>	<b>Product Catalogue with Federated Search</b> Defined Product / Service catalogues and share with other users on the EFPF platform <ul style="list-style-type: none"> <li>• Catalogue Service (7.4.2) – Enable partner discovery allowing companies to list the products they supply and the services they provide. Used with the Federated Search Data Spine component.</li> </ul> <b>Tendering Service</b> Publish requests for bids for customised products <ul style="list-style-type: none"> <li>• Tendering and Team Formation Service – Publish customised product business opportunity and manage associated bids with tools to allow</li> </ul>

		<p>secure document sharing and messaging (solution currently in development)</p> <p><b>Customer Analytics Service</b></p> <p>Service to support understanding of customer behaviour</p> <ul style="list-style-type: none"> <li>Catalogue Trend Analysis (7.3.4) - Models to allow companies to analyse their customer data and to predict customer behaviour</li> </ul>
5	<p><b>Search for products, services or supply partners to meet production objectives</b></p> <ul style="list-style-type: none"> <li>Invitation to join collaboration platform</li> <li>Tender publication &amp; bid assessment</li> <li>Product request</li> </ul>	<p><b>Tendering Service</b></p> <p>Collaborative environment to search for suppliers and publicise tenders</p> <ul style="list-style-type: none"> <li>Tendering and Team Formation Service – Manage supplier capability profiles, business opportunity publication and associated bids with tools to allow secure document sharing and messaging (solution currently in development)</li> </ul>
6	<p><b>Monitor production processes to get a schedule for planning activities</b></p> <ul style="list-style-type: none"> <li>Production optimization</li> <li>Production process monitoring in a supply chain</li> <li>High-level production overview</li> </ul>	<p><b>Predictive Maintenance</b></p> <p>Analyse machine condition data to develop model to predict machine breakdown</p> <ul style="list-style-type: none"> <li>Industreweb Collect (5.1.1) – Provides real-time data collation and connectivity to EFPF</li> <li>FCGMT (5.2) – Provided Management of Gateway component within EFPF platform</li> <li>SDSS (6.2.2) – Securely store sensor process usage and PPE compliance to enable real-time and historical analysis</li> <li>Analytics Tools (to be clarified)</li> <li>Risk Tool (7.3.6) - To detect and Alert risk scenarios based on results from Analytics tools</li> </ul> <p><b>Error Proofing</b></p> <p>Guide the operator with clear product specific instructions</p> <ul style="list-style-type: none"> <li>Industreweb Collect (5.1.1) – Connect with barcode camera and ERP system to retrieve process specific data for production</li> </ul>

		<ul style="list-style-type: none"> <li>• Industweb Display (7.2.1) – Visualise clear instructions based on data from the ERP system based on product barcode</li> <li>• FCGMT (5.2) – Provided Management of Gateway component within EFPF platform</li> <li>• SDSS (6.2.2) – Securely store process events</li> <li>• Risk Tool (7.3.6) - To detect and Alert errors based on process events</li> </ul>
7	Monitor delivery processes both incoming and outgoing in order to get a schedule of activities <ul style="list-style-type: none"> <li>• Delivery process monitoring</li> <li>• Local reception of goods</li> <li>• Delivery Schedule</li> <li>• Replace logistics provider to meet just-in-time delivery</li> <li>• Set up logistics supplier in new country</li> </ul>	<b>Delivery App</b> Visualise the status of delivery process and track with blockchain based audit trail <ul style="list-style-type: none"> <li>• Blockchain (6.3.1) – Connect with barcode camera and ERP system to retrieve process specific data for production</li> </ul>

## 4.4 Circular Economy

In this pilot, KLEEMANN (KLE), a global manufacturer of Lift Systems, Escalators, and Moving Walks, requires agile relationships with different partners and suppliers in the waste management domain in order to be compliant with LCA regulations.

In this circular economy model, the value of products and materials is maintained for as long as possible, which can bring major economic benefits contributing to innovation, growth and job creation. This pilot in the EFPF project specifically focuses on realising closed-loop supply chains (CLSC) in the KLE ecosystem, where the returns processes and the manufacturers in the network have the intent of capturing additional value and further integrating all supply-chain activities.

The pilot scenario addresses the agile supply network through circular economy activities involving KLE, ELD and other European companies from the business ecosystem of KLE. At present, the supply chain and business relationships in the KLE ecosystem lack the visibility and tracking of waste. The companies also face lag in material transition phases and the ecosystem pose entry barriers for new innovative European companies to join the market. Therefore, the pilot realisation focuses on establishing Closed-Loop Supply Chains (CLSC) at the European level, where the KLE has the intent of capturing additional value and further integrating supply-chain activities through return processes. Moreover, the pilot also focuses on providing new business opportunities to European companies through their inclusion in the different levels of the waste management supply chain. The tools and serviced provided by EFPF will support the overall process to ensure risk management, regulatory and environmental compliance and for optimising the production and waste

management processes. The EFPF tools and services will play a crucial role in the design, execution, monitoring and optimisation of ad-hoc collaborative processes to deliver time, cost and service improvement benefits.

The following table shows the Epics and User Stories for the Circular Economy domain and the pilot solutions used to fulfil those requirements.

<b>Epic</b>	<b>User Story</b>	<b>Solution</b>
8	<b>Enable the integration of IT systems in the supply chain</b> <ul style="list-style-type: none"> <li>Bins' fill level monitoring</li> <li>Blockchain verification to improve transportation security and reliability</li> <li>Blockchain traceability to improve closed loop supply chain management</li> </ul>	<b>Bin fill level monitoring</b> <ul style="list-style-type: none"> <li>Fill Level Sensors (ultrasonic) enable bins remote monitoring</li> <li>Symphony components (5.2.3) enables shop-floor connectivity and events management</li> </ul> <b>Blockchain for fully visibility in waste management procedures in a closed loop supply chain</b> <ul style="list-style-type: none"> <li>Web Based Blockchain Application for Wastes Tracking, see D5.12 section 3.2.2</li> <li>Mobile Blockchain App Application to support the delivery process (4.3.1)</li> </ul> <b>Security framework</b> <ul style="list-style-type: none"> <li>Login and Role Management capabilities for secure access to resources for various users provided through EFPF Security functionalities available on D5.12. EFPF Portal also used in order to list the solution there and make it available to the end-users</li> </ul>
9	<b>Reduce the burden of finding relevant partners for lot-size-one tasks</b> <ul style="list-style-type: none"> <li>Online bidding for processed wastes</li> <li>Request and receive possible suppliers' details</li> </ul>	<b>Matchmaking</b> <ul style="list-style-type: none"> <li>Matchmaking Services to support an automated online bidding process for specific services and goods. Please see D5.11, section 1.4.</li> </ul> <b>Marketplace</b> <ul style="list-style-type: none"> <li>Agents used in order to create an Agent Marketplace to support the realization of the bidding process, plus the corresponding UIs. Information available on D5.13 and section 2.2.</li> </ul> <b>Portal, Security Framework and Data Spine</b> <ul style="list-style-type: none"> <li>These general available components were used in order to provide the tool to the end-users as an installed tool on EFPF portal.</li> </ul>
10	<b>Search for specialised products/services</b> <ul style="list-style-type: none"> <li>Search for new customers</li> <li>Search for suppliers</li> </ul>	<b>Matchmaker and Federated Search</b> <ul style="list-style-type: none"> <li>Federated search is used in order to enable end-users to search for</li> </ul>

	<ul style="list-style-type: none"> <li>Equipment Selection and Purchasing</li> <li>Market research for specialised product customers</li> </ul>	<p>customers etc. The tool is described on D5.11 and 1.3 section.</p> <ul style="list-style-type: none"> <li>Furthermore, the Matchmaker component that supports the Online Bidding Process provides also the information for companies coming from COMPOSITION Agent Marketplace in order to be indexed by federated search and be available to search results</li> </ul> <p><b>Data Spine</b></p> <p>Data Spine and the corresponding integration flow engine (NiFi) were used in order to index the data from Agent Marketplace to Federated Search repository. (see D3.11 and D5.11)</p> <p><b>Marketplace</b></p> <ul style="list-style-type: none"> <li>As part of marketplace activities and in connection with federated search, landing pages for companies coming from Agent Marketplace have been created. The COMPOSITION marketplace has no interface for available companies/services, so an approach from NIMBLE project was adopted here. (see D5.11 and D5.13)</li> </ul> <p><b>Portal and Security Framework</b> also used in order to provide access to search mechanisms and results</p>
11	<p><b>Data Analytics for the Optimization of Procedures Related to Circular Economy Activities</b></p> <ul style="list-style-type: none"> <li>Optimization of Planning Activities for Effective Waste Management through EFPF Platform</li> <li>Optimization of Planning Activities for Purchasing New Materials</li> <li>New Predictive Maintenance solution for polishing machine</li> </ul>	<p><b>Factory Connectivity</b></p> <ul style="list-style-type: none"> <li>Functionalities for factory connectivity for various sensors connection coming from Symphony available solutions (5.2.3)</li> <li>Fill level and vibration sensors used for fill level monitoring and predictive maintenance as well</li> </ul> <p><b>Data Analytics</b></p> <ul style="list-style-type: none"> <li>Visual Analytics Tool for Predictive Maintenance and Optimization of Supply Chain Planning Activities as it is described in 5.3.2 section of this deliverable. The tool provides predictive maintenance support through vibration sensors analysis for KLEEMANN, fill level sensors analysis and monitoring for ELDIA and tonnage forecasting solutions for both ELDIA and MILOIL</li> <li>Deep Learning Tool for Data Analysis as it is described on section 5.3.3 of this report. The tool is used by ELDIA and MILOIL for price forecasting and provides its output through visual analytics coming from the above mention tool (5.3.2)</li> </ul>

		<b>Portal and Security Framework/Role Management</b> <ul style="list-style-type: none"><li>• The analytic tools are available to the pilots through the portal. Every user sees different custom instances of the Visual and Data analytics tool based on its data and its requirements (i.e. different dashboards for predictive maintenance and supply chain planning optimization)</li></ul>
--	--	---



## 5 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.

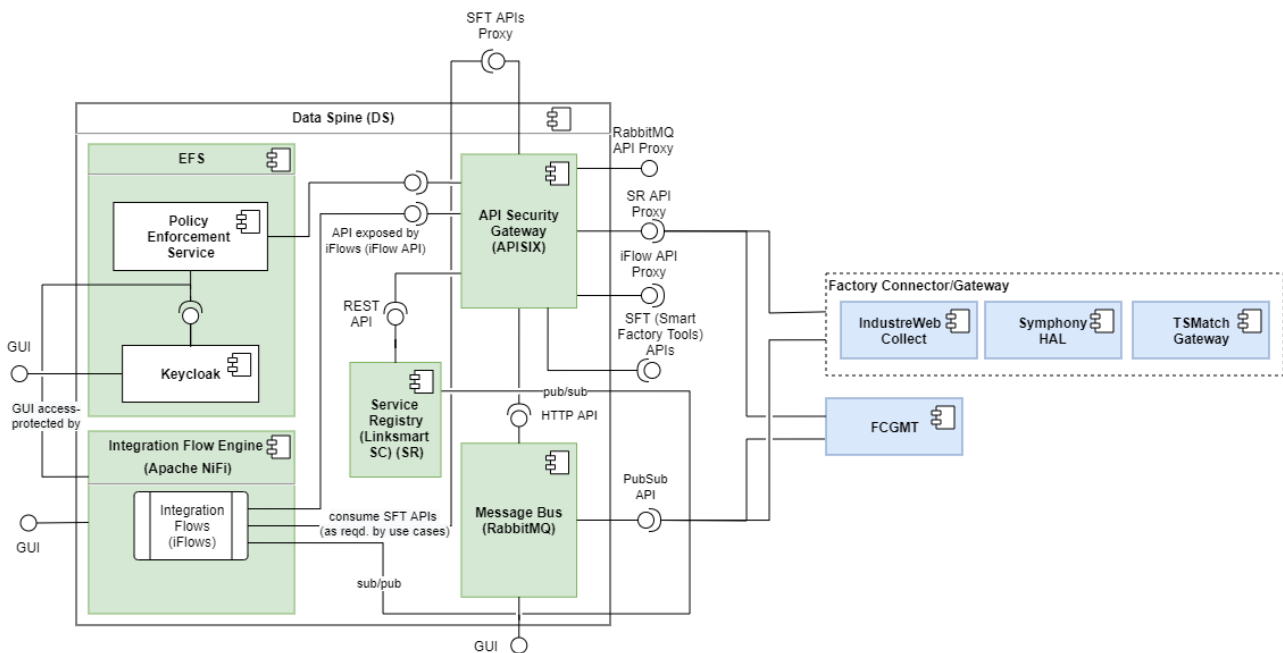


Figure 12: Interaction between the Factory Connectors and the EFPF Data Spine

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

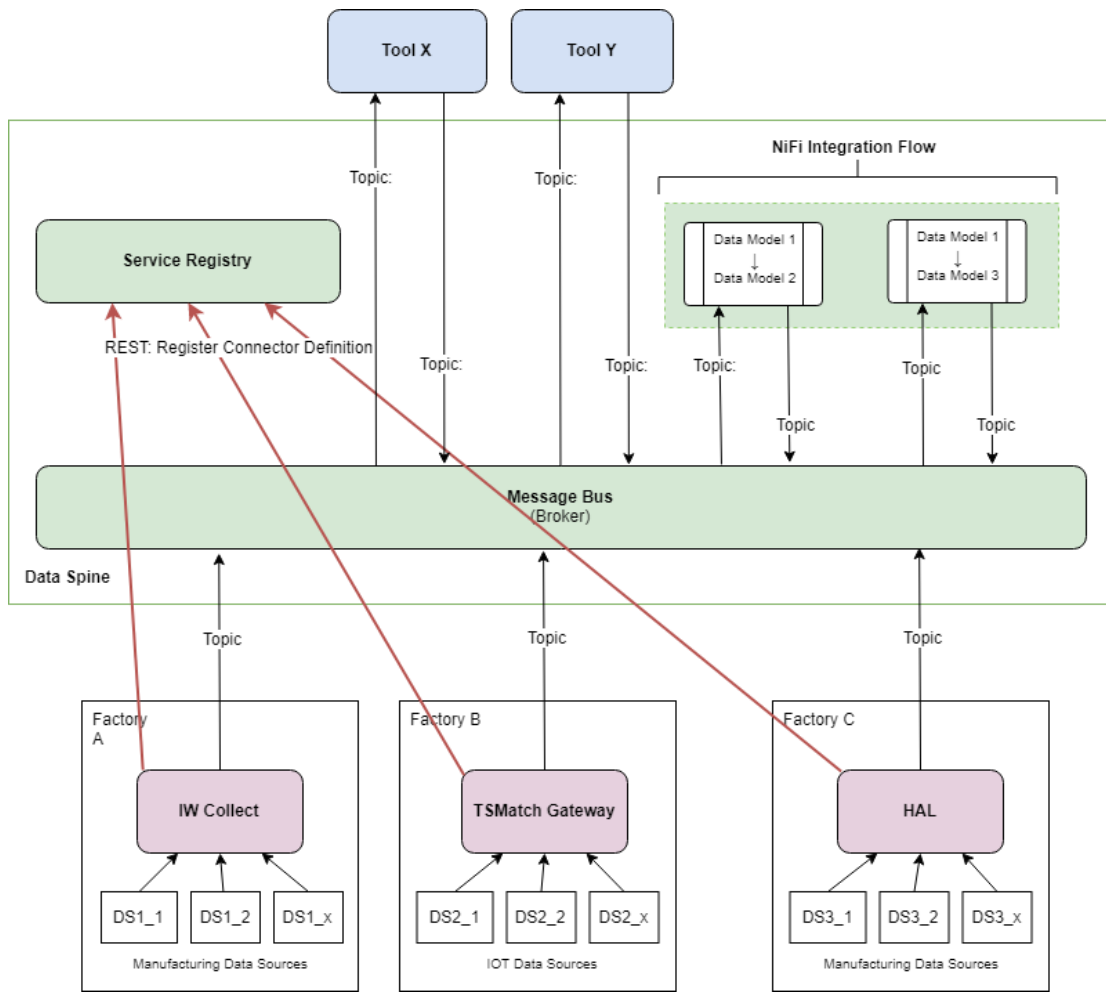


Figure 13: Factory Connectivity within EFPF

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 0, 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.

## 5.1 Connectors and Gateways

### 5.1.1 Industweb Collect

Industweb (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.

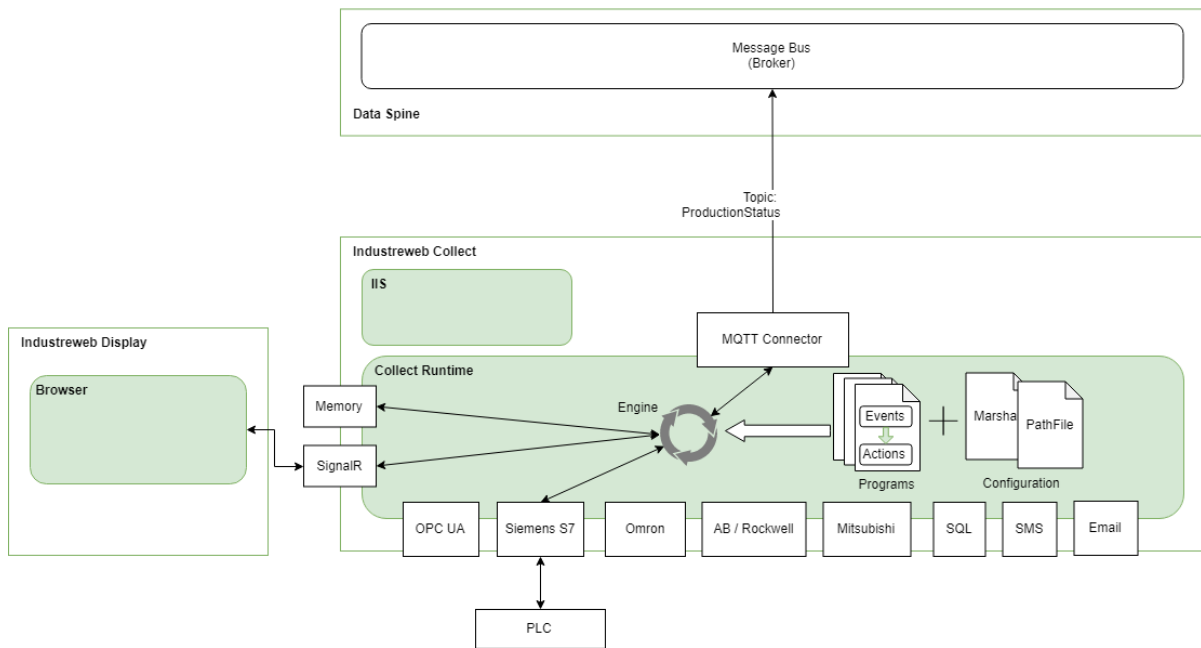


Figure 14: Industweb Collect Architecture

Data sources can include industrial control systems (e.g. PLC, CNC) both modern models with Ethernet capability and legacy equipment, wireless networks and 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 of the IW Collect is shown in

Figure 14, which 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 retro-fitted to existing machines and is based on Microsoft .Net with an option of using Microsoft SQL Server in order capture data locally.

#### 5.1.1.1 Configuration

To configure the system Industweb Global provides web based Administrative tools described in Section 7.2.1.2. The industrial PC running IW Collect must firstly be interfaced with the production data sources, which typically involves physically connecting the required networks, and the installation of any intermediate 3<sup>rd</sup> party hardware such as network switches or wireless transmitter/ receivers. Once this has been carried out the interface

settings are defined by editing the connector's configuration file, which defines each data source connector and its properties necessary to function.

Following this stage, the rules to orchestrate the collection and manipulation of data are created which are based on logic events and subsequent actions. The system is then run in the background as a Windows Service, constantly monitoring the manufacturing process.

Actions that IW Collect can trigger may include changing data in a connector, displaying an alert on a screen, sending an SMS or email, or writing a value to a database.

For interfacing with the MQTT broker in EFPPF, the data is collected from the range of production systems and sensors via the appropriate connector instance and mapped to the MQTT connector instance data model. Upon data changing within the process or at a set time interval IW Collect publishes the data to the MQTT broker with the topic configured by the EFPPF user using the Factory Connector Gateway Management Tool described in section 5.2. This is then subscribed to by the Smart Factory Tools and Services that have been granted access to that topic by the EFPPF user

#### **5.1.1.2 Operation**

Industweb Collect typically runs at start-up as a Windows Service, but can also be run manually under a command window (see

Figure 15). On start-up the application reads in the configuration data to setup the connectors and runtime logic defined in the programs, and once this is complete the Collect Engine starts cycling continually evaluating the runtime logic using the data it gets from the connectors. A cycle can set at anything from a minimum of 100 micro seconds or higher depending on the needs of the application. Whilst evaluating the runtime logic if an event evaluates as true the associated Actions are sent to the Action Queue for execution asynchronously to the main engine, with each Action fulfilling the user requirements defined at design time. This continues indefinitely or until the Collect Engine is stopped.

```

IndustrieWeb Collect Engine
2019-01-30 13:42:42,593 DEBUG ABB.ABBConnector - Auto
2019-01-30 13:42:42,608 DEBUG ABB.ABBConnector - MotorsOn
2019-01-30 13:42:42,684 DEBUG IW.DataInput.ExcelConnector - Schema loaded C:\Users\Admin\Desktop\Collect v3 17.9.18\CollectRuntime\bin\Debug\Programs\dataSch
emas\Oee.json
2019-01-30 13:42:42,701 DEBUG IW.DataInput.ExcelConnector - Schema loaded C:\Users\Admin\Desktop\Collect v3 17.9.18\CollectRuntime\bin\Debug\Programs\dataSch
emas\Oee.json
2019-01-30 13:42:42,753 DEBUG IW.DataInput.ExcelConnector - Schema loaded C:\Users\Admin\Desktop\Collect v3 17.9.18\CollectRuntime\bin\Debug\Programs\dataSch
emas\Oee.json
2019-01-30 13:42:42,781 DEBUG IW.SiemensS7.S7NETConnector - S7 Reconnected Successfully
2019-01-30 13:42:42,783 DEBUG IW.DataInput.ExcelConnector - ABB0EE started
2019-01-30 13:42:42,783 DEBUG IW.REST.RESTConnectorV2 - REST started
2019-01-30 13:42:42,785 DEBUG IW.DataInput.ExcelConnector - OMRON0EE started
2019-01-30 13:42:42,783 DEBUG ABB.ABBConnector - ABBROBOT started
2019-01-30 13:42:42,785 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ Reconnected Successfully
2019-01-30 13:42:42,849 DEBUG IW.DataInput.ExcelConnector - Schema in connector ADRIG0EE updated to row 1
2019-01-30 13:42:42,798 DEBUG IW.SiemensS7.S7NETConnector - S7 started
2019-01-30 13:42:42,816 DEBUG IW.SiemensS7.S7NETConnector - S7:DATACHANGED:[ROBOTSTATUS, DB601.DBX4]:0
2019-01-30 13:42:42,822 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK1WARNING, DB601.DBX6.0]:False
2019-01-30 13:42:42,857 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ started
2019-01-30 13:42:42,829 DEBUG IW.DataInput.ExcelConnector - ADRIG0EE started
2019-01-30 13:42:42,846 DEBUG IW.DataInput.ExcelConnector - Schema in connector OMRON0EE updated to row 1
2019-01-30 13:42:42,849 DEBUG IW.Omron.OMRONConnector - OMRON started
2019-01-30 13:42:42,848 DEBUG IW.DataInput.ExcelConnector - Schema in connector ABB0EE updated to row 1
2019-01-30 13:42:42,876 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK2WARNING, DB601.DBX6.1]:False
2019-01-30 13:42:42,895 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK1ALARM, DB601.DBX6.3]:False
2019-01-30 13:42:42,905 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK2ALARM, DB601.DBX6.4]:False
2019-01-30 13:42:42,913 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK2FILL, DB36.DBX0.2]:False
2019-01-30 13:42:42,922 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK2EMPTY, DB36.DBX0.3]:False
2019-01-30 13:42:42,932 DEBUG IW.SiemensS7.S7NETConnector - S7ADRIQ:DATACHANGED:[TANK1EMPTY, DB36.DBX0.1]:False

```

Figure 15 Industreweb Collect running in command prompt

### 5.1.2 Symphony Hardware Abstraction Layer (HAL)

Symphony Hardware Abstraction Layer (HAL) is a software module that is part of Nextworks' Symphony platform<sup>10</sup> (see section 7.2.3). It primarily abstracts the low-level details of various heterogeneous fieldbus technologies and provides a common interface to its users. It adapts the device protocols and provides the necessary logic to manage them accordingly to their respective constraints (e.g. timing constraints). It also implements optimizations, e.g. avoid spamming the KNX bus<sup>11</sup> with too many messages, pack contiguous Modbus reads into a single multi-register read.

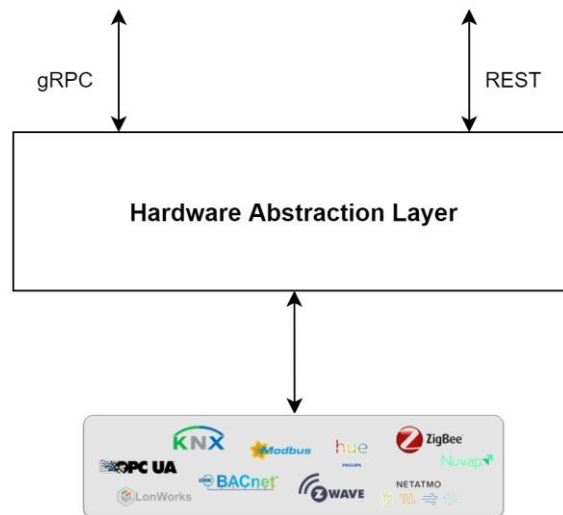


Figure 16 Symphony Hardware Abstraction Layer (HAL)

The HAL supports KNX, BACnet, Modbus-TCP and Modbus-RTU as well as, several other proprietary control protocols. It can be extended by developing modules that can be dynamically plugged into its core.

<sup>10</sup> <http://www.nextworks.it/en/products/brands/symphony>

<sup>11</sup> <https://www2.knx.org/no/knx/association/what-is-knx/index.php>

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, ...) or are presented according to the OGC SensorThings<sup>12</sup> data format standard. The HAL supports access via REST and gRPC and furthermore enables publish/subscribe features via MQTT.

### 5.1.2.1 Configuration

The configuration is possible through its GUI and REST API. In order to configure the HAL, it should be interconnected with its supporting 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. After interfacing, user can perform the configuration through the GUI based on the fieldbus device specific parameters. For example, if the fieldbus protocol is Modbus, the user can import the project as single csv file or configure the buses (IP address, Port, Poll, Slaves) and related datapoints (Address, Register Type, Mask, Function Code, Binding Slave). Also, the configuration is possible via REST API with fieldbus specific schemas. For configuring the MQTT message broker to publish the data, user should use the specific configuration schema for API.

### 5.1.2.2 Operation

Symphony HAL is part of Symphony Platform and start-up when the platform is starting up. It works in LINUX environment and after starting up, the GUI and APIs are up and ready to use.

### 5.1.3 TSMatch Gateway

TSMatch (Thing to Service Matching) gateway is a software module that enables the automatic and dynamic matching between Things and IoT services or end-user requirements (the end-user requirements are provided through the TSMatch App).

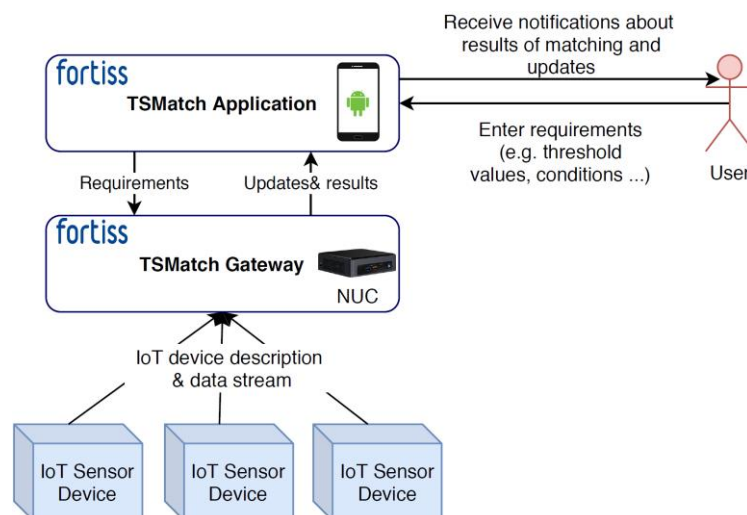


Figure 17: TSMatch solution

<sup>12</sup> <https://www.ogc.org/standards/sensorthings>



Due to an increasing amount of heterogeneous data generated by IoT devices, data exchange between IoT devices and IoT services is becoming more and more inefficient and does not support easy integration. What ends up being done today is either select closely integrated solutions usually from the same provider or invest in additional integration effort making the processing time consuming, prone to errors, and/or costly. This challenge is especially relevant in the context of EFPF where services from the various platforms should be integrated with IoT data. Therefore, the objective of the TSMATCH solution is to ease integration between Things and IoT services via data matching based on service requirements.

TSMATCH solution is used in Industrial IoT scenarios in the context of smart facilities, which includes but is not limited to, shop floor, commercial buildings, and building offices. Implementing smart facilities use cases implies in combination with TSMATCH solution a physical infrastructure, services, IoT devices “Things”, and IoT technology supporting real-time data exchange to monitor various aspect such as:

- Monitoring safety requirements (e.g. CO2 levels) of the facility.
- Monitoring comfort parameters (e.g. temperature, humidity) for an improved experience for the workforce.
- Monitoring space utilization in the facility for optimization.
- Reducing facilities operating costs (e.g. by avoiding unnecessary heating, open windows, and turned on lights in empty spaces).

Figure 17 illustrates the interaction of the TSMATCH solution with the user and the IoT devices, and shows the main components of the TSMATCH solution as follow:

- TSMATCH Gateway: is a software component that can assess if the request provided by the IoT service or the end-user is possible to deliver based on the available Things, select the optimum set of Things to respond to the request, and select the needed micro-services (data processing function) to transform the raw IoT data to match the information needed by the service or the user.
- TSMATCH Application: is an Android application that provides an interface for users (end-users and IoT service) to specify their requirements.

#### 5.1.3.1 Configuration

Configuration of TSMATCH gateway includes:

- Integration of IoT devices: IoT devices installed in the facility to monitor desired aspects are expected to be integrated with the TSMATCH gateway by using semantic standards for the IoT device description such standards are OGC Sensor Things API<sup>13</sup> and Web of Things Thing Description<sup>14</sup>.
- Integration with the message Bus: The TSMATCH is expected to use GUI to configure the TSMATCH gateway relating to aspects such as the message bus to be used for communication between the various components

<sup>13</sup> <https://www.ogc.org/standards/sensorthings>

<sup>14</sup> <https://www.w3.org/TR/wot-thing-description/>



- User requirement description: TSMATCH App is used to collect users' requirements regarding aspects to be monitored as well as view matching results and updates/notifications. To model these users' requirements an existing service description standards is yet to be selected.

#### 5.1.3.2 Operation

Upon start-up, the TSMATCH gateway first discovers Things available in the environments and stores their descriptions. Upon receiving an IoT service or end-user request containing a description of the requirements, TSMATCH gateway selects the optimum set of Things capable of responding to the request. If the raw data of the selected set of Things does not directly match the requirements specified in the request, a micro-service that provide base functionalities such as clustering (i.e. clustering based on an area of interest), event triggering (i.e. triggering notification based on a specified threshold) is selected so that the raw data matches the specified requirements. The result of the matching is sent to the TSMATCH app. Results of matching could be positive meaning that the requirements can be fulfilled or negative meaning that the currently available Things are unable to answer the user requirements. Finally, the user receives updates and notification about the monitored aspects through the TSMATCH App.

## 5.2 Factory Connector Gateway Management Tool (FCGMT)

The Factory Connector Gateway Management Tool (FCGMT) enables the administration of IoT devices such as Factory Connectors and Gateways in the EFPF platform. The tool may interact with the Service Registry and external repositories through APIs to get information about these items.

Relationships between devices and topics can be defined through the tool, in order to represent what devices publish data or are subscribed to particular topics. The FCGMT consists of two main components:

- **FCG UI or frontend:** This is mainly the Angular web client application which provides the interface components and navigation to interact with the repositories. It also provides the visualization of the location of connectors and gateways through the Google Maps API. The tool is designed to consume information about factory connectors based on the service description schema which has been defined for the persistency of information about services in the Service Registry, as well as consuming extended information about factory connectors and gateways from an additional repository. This additional document-oriented repository of devices contains extra information about publishers and subscriber devices, such as bindings between these devices and users or topics available, and it can be accessed through the FCG API, which manages the persistency of these resources.

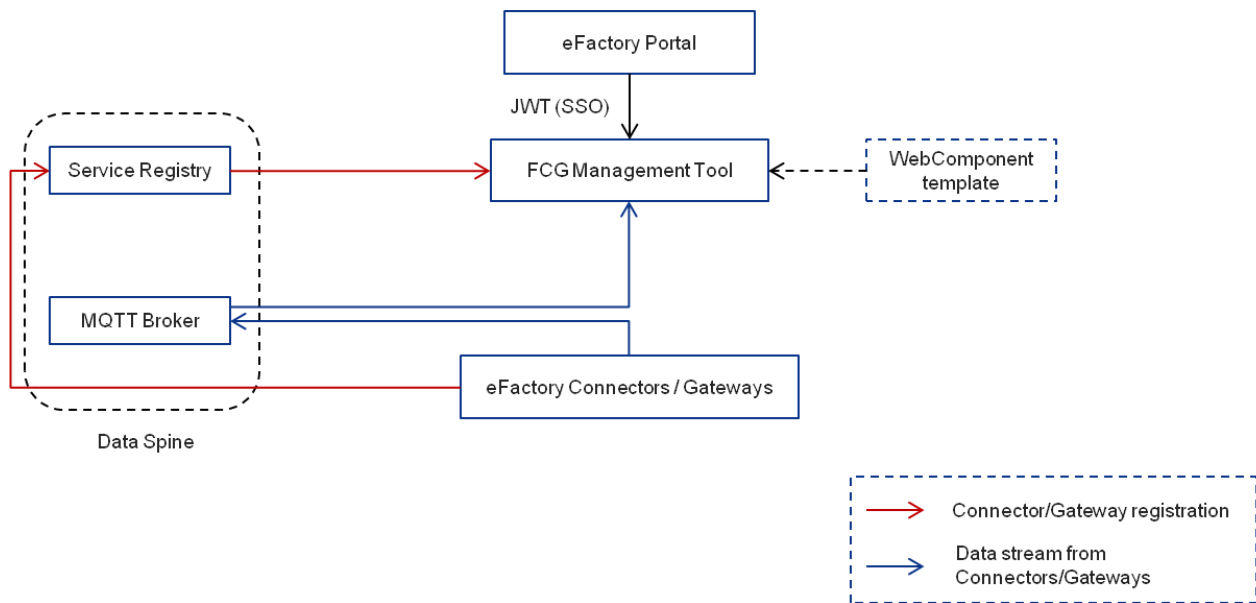


Figure 18: Preliminary FCGMT Design Architecture

In order to implement a first approach to the appearance and navigation through the tool, a collection of mock-ups was designed and discussed. Then, an initial design of the UI was implemented just to represent the main functionalities.

- **FCG API:** It provides access to an external repository that contains information about the IoT devices and the topics they produce or consume. The API offers main operations on these resources, such as creating, reading, updating and deleting. The devices may play the role of producers (data publishers) or consumers (data subscribers). The producer devices are able to create and publish data on particular topics, while consumers are able to subscribe to these data. In order to consider the authorization of the devices to publish or subscribed to specific topics, the Security Gateway API is consumed.

Besides this, it should be noted that also a preliminary version of a Message Broker (RabbitMQ) API has been developed. This API provides administration operations to interact directly with the Message Broker, such as the creation of exchanges, queues, bindings, send messages or receive messages one by one.

The UI design, the FCG API and the RabbitMQ API functionalities are open to improvements and new additions based on particular needs during the experimentation in the Factory Connectivity related scenarios.

### 5.2.1 Configuration

The FCG UI is an Angular client application that can be built and deployed as a standalone application or built through a particular script in order to be later integrated in the EFPF platform.

Some configuration parameters should be considered before building the application. In the current version, they are in the *Globals.ts* file, which is a class that manages the values of properties which are consumed in different components and services of the application. They involve aspects such as:

- The default latitude and longitude to be placed in the location maps component that consumes the Google Maps API
- The level of zoom in the map
- The endpoint to the Service Registry API to retrieve Factory Connector information
- The endpoint to the FCG API which interacts with the
- The Message Broker API that performs administrative operations in the MQTT Broker (RabbitMQ)
- The default type of the Factory Connectors services in order to filter the list
- The default Time To Live (TTL) of new created devices through the FCG UI

Furthermore, the Message Broker API includes a configuration file *application.yml* in resources folder which manages some relevant parameters mainly focused on the RabbitMQ instance properties such as:

- RabbitMQ host and port
- RabbitMQ administration credentials

## 5.2.2 Operation

Once the tool is deployed, either as a standalone application or as an integrated EFPF platform tool, the available functionalities are prompted through menus and action buttons.

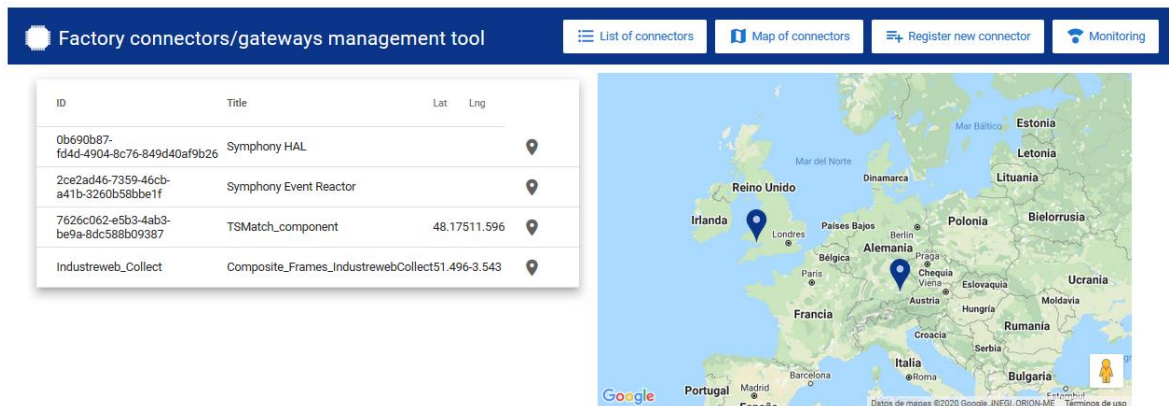


Figure 19 Preliminary design of the FCG frontend

At the time of this document, just some core basic functionalities are implemented:

- Get the list of available Factory Connectors from the Service Registry
- Obtain more detailed information about FCGs, such as the preliminary data exposed by the devices, the APIs available, etc.
- Manually registration and edition of FCGs through user forms

However, once the requirements demanded by IoT devices are refined, additional functionalities will be covered, such as:

- Management of subtopics from root topics: publication and subscription
- Management of user access to topics through the API Gateway
- Creation of any required resource in the Message Broker at administration level through the RabbitMQ API

The FCG API will be accessed in order to interact with the persisted detailed information about devices and topics which is expected to be saved in an external document-oriented repository.

Furthermore, additional functionalities are being considered. They are for example the visualization of log information produced by the IoT devices, or a Message Broker testing component to check the communication with the Broker.

The FCGMT is available as an open source tool designed to be deployed as a standalone application in any web server or integrated in the EFPF portal, so this is generated from an Angular template to accelerate the integration process.

## 6 Foundation Tools and Services

### 6.1 Introduction

This section will describe the Tools and Services available through EFPF that provide a diverse range of functionality to support implementing Smart Factory solutions. This includes functions such as data transformation and storage, data co-ordination and tracking, which can be done prior to or in addition to processing carried out by the productivity tools. Each Tool and Service will outline what functionality is provided, how it is configured and what the output is to the user.

### 6.2 Digital Tools for Smart Factory scenario

This set of tools provide back-end services that are necessary to support data handling, storage, event management, visualisation and security features in other user-centric solutions.

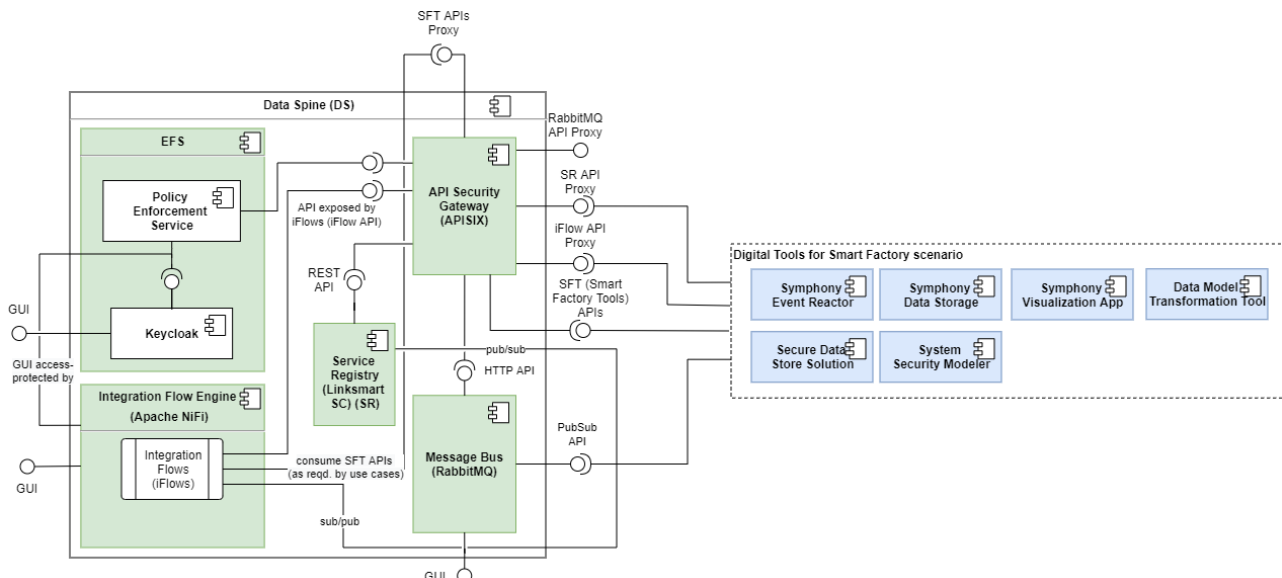


Figure 20: Interaction between the Digital Tools and the EFPF Data Spine

#### 6.2.1 Symphony Event Reactor

The Symphony Event Reactor gives the ability to trigger actions and alarms through its Event Manager/ Alarm Manager in response to different kinds of event types. Moreover, it offers a logging and lifecycle system for alarms. The event reactor is written in Python and has a GUI written in Blockly (google), a free and open source client-side JavaScript library for creating block-based visual programming languages (VPLs) and editors.

The Symphony Event Reactor leverages on a highly customisable logging system that allows to handle events locally and synchronise them remotely, together with user activity and alarm history.

The Symphony Event Reactor is composed of two separate software modules:

- **Event Manager (EM):** The EM executes custom rules that combine information coming from different sources (local sensors and device monitors, user actions, video-cameras, intrusion detection systems, etc.) and data brokers (e.g. AMQP, MQTT) to determine actions to be taken (see Figure 26). Actions include actuations on field devices, activation of scenarios, generation of events, notifications and alarms, and so on.
- **Alarm Manager (AM):** Alerts can be raised in order to present the situation to specific users or user-groups. The system provides a configurable priority based alert routing system that allows to target a single, a group or mixed sets of users with SMS, emails, pop-ups, etc. (see Figure 22) The AM has an internal state machine to track each alarm's status (Open, Close, Acknowledged, Resolved, Delivered). Also, it logs and keeps alarms history in a log database which is accessible through a REST interface. Different notification channels for the AM are being developed.

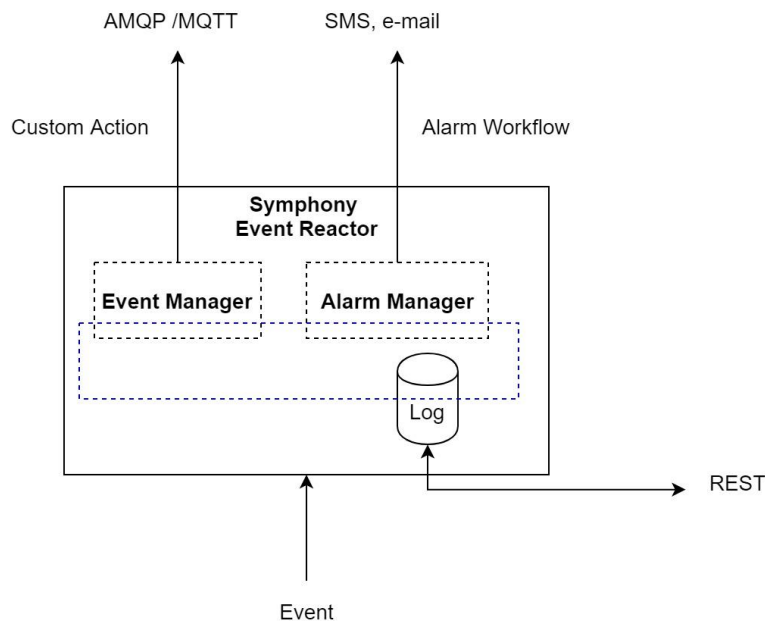


Figure 21 Symphony Event Manager

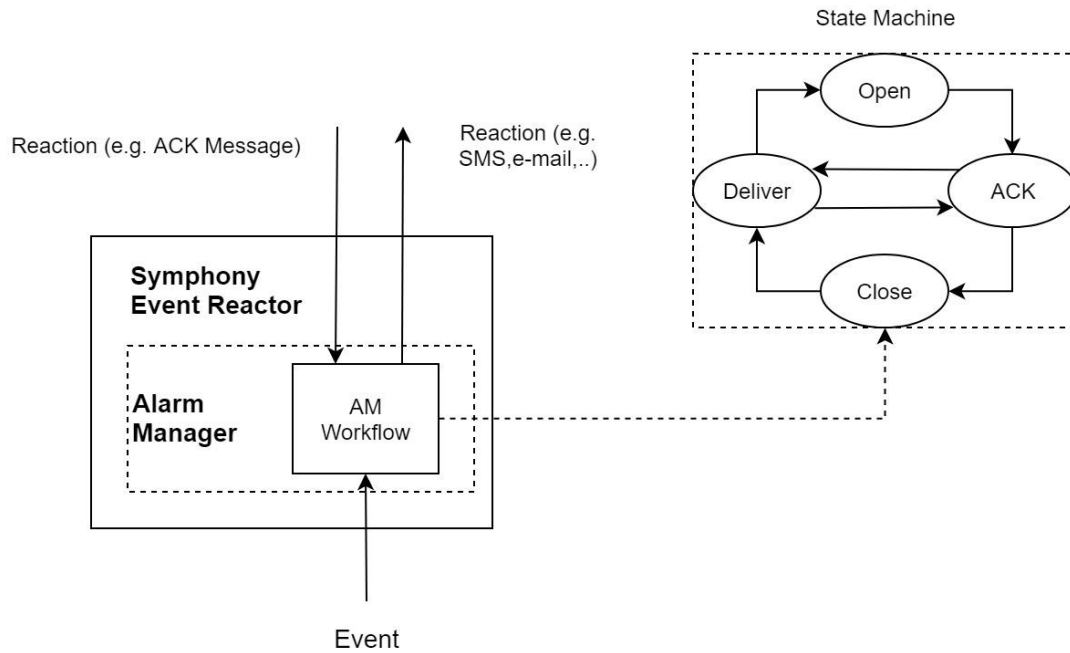


Figure 22 Symphony Event Reactor's Alarm Manager

#### 6.2.1.1 Configuration

To configure the system, after deploying the Symphony Event Reactor, the user can configure Event Manager and Alarm Manager through the GUI and set required message bus subscription information:

1. Event Manager (see Figure 23): user can define, modify and monitor block-based rules via Visual Programming Language for triggering the actions and build scenarios.



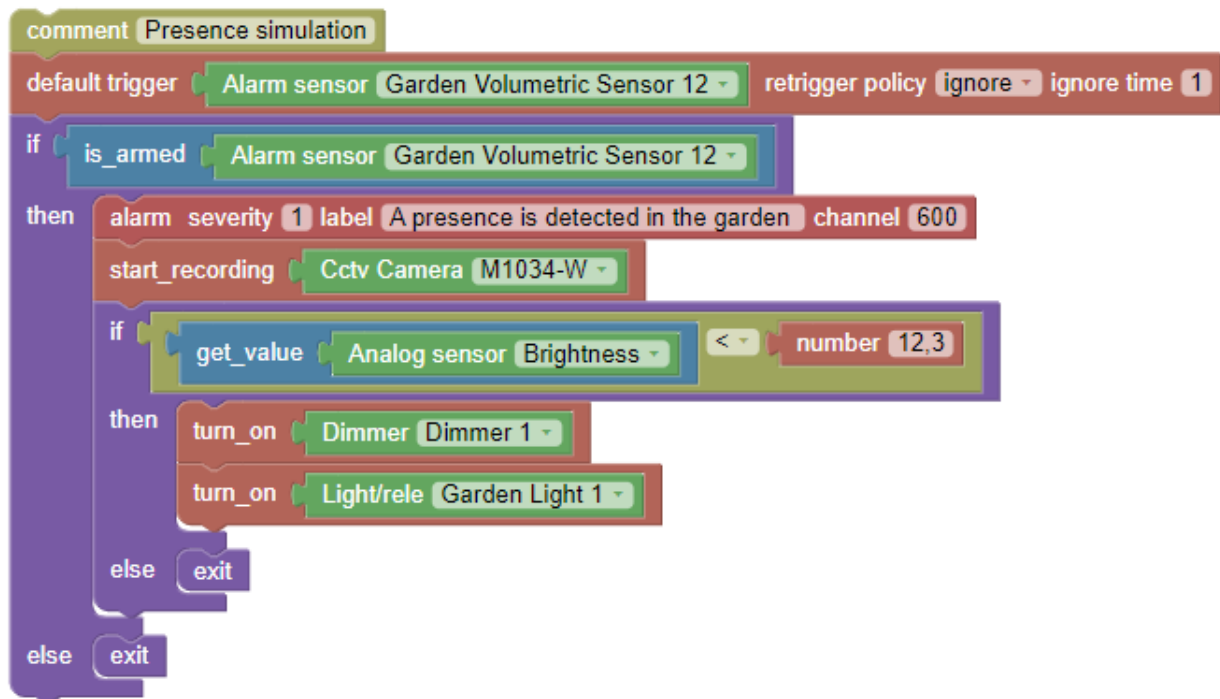


Figure 23 Symphony Event Reactor's Event Manager

2. Alarm Manager (see Figure 24): user can create and modify,
  - Gateways for email and SMS
  - Users whom receive the alarm with name, last name, email and phone number
  - Specific template for email and SMS
  - Chain of deliveries for users and gateways
  - See history of the alarms

Figure 24 Symphony Event Reactor's Alarm Manager

### 6.2.1.2 Operation

Symphony Event Reactor is a part of Symphony Platform, after deploying the platform it is up and ready to be used.

### 6.2.2 Symphony Data Storage

Symphony Data Storage is a highly scalable and high-performance data storage which is designed to handle large amounts 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. Symphony Data Storage is designed to handle large amounts of data and providing high availability with no single point of failure. Also, it supports PostgreSQL and Elastic Search as backend.

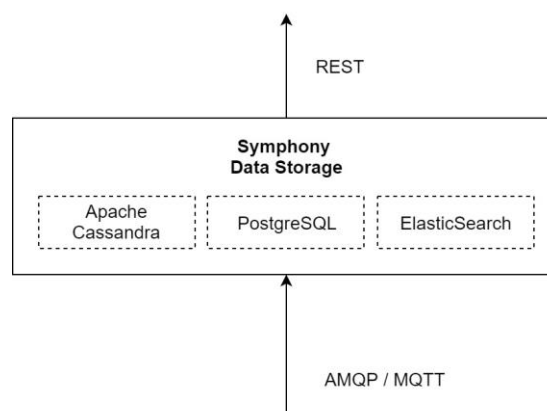


Figure 25 Symphony Data Storage

### 6.2.2.1 Configuration

To configure the system, after running the docker container user can configure the Symphony Data Storage through its configuration shell. User can define backends (Apache Cassandra, PostgreSQL and ElasticSearch) and AMQP/MQTT data sources for subscribing. After configuration, data immediately begin to flow to the Data Storage and user(s) can query the data through REST interface. Also, user should define sufficient resources based on the backend(s) and the rate of incoming data.

### 6.2.2.2 Operation

Symphony data storage is running in Docker Container environment so user can simply run the docker container and start configuring/using the Data Storage. The Data Storage will run indefinitely until in stopped by user or by lack of resources like HDD.

## 6.2.3 Symphony Visualization App

The Symphony Visualization App is the interface that stands between the operators and the process. This is the primary tool by which operators and line supervisors coordinate and control the industrial and manufacturing processes in the plant. Symphony Visualization App offers customizable, flexible and user-friendly user interface, getting information from the shop floor level.

In an ordinary Symphony installation, it is supposed to get inputs from Symphony components, but in general it can be configured for receiving data from any factory connector, under certain conditions (i.e. the data format). The Visualization App gets data through a southbound RESTful interface, and relies on ontology and semantics used in the Symphony platform such as SAREF and SAREF4BLDG, plus some extensions to provide part of its functions. Therefore, the app will be able to provide views for any factory connector that exposes a compatible REST interface.

At the time of writing of this document, this component has some dependencies with some of the Symphony components, thus it needs some development in order to be decoupled from the platform and become ready to be integrated in the EFPF ecosystem.



Figure 26 Symphony Visualization App

### 6.2.3.1 Configuration

Since the component is part of the Symphony Platform and it is not decoupled yet, the configuration is possible through Symphony Platform. User can configure the required

information for getting the data from southbound REST interface. Furthermore, user can create graphical panels and HMI interfaces with ready to use objects library and bind them to the data coming from ICS.

### 6.2.3.2 Operation

Symphony Visualization App is a part of Symphony Platform and as mentioned before it still has dependencies to other Symphony components. After deploying the platform, the Visualization App is up and ready to use.

## 6.2.1 Data Model Transformation Tool Suite

In order to realise an agile collaborative manufacturing ecosystem, a service integration platform that integrates, enables communication between and bridges interoperability gaps between services of different heterogeneous platforms is needed. To bridge the protocol interoperability gaps, the state-of-the-art service integration platforms in the industry provide protocol connection and translation tools that work out-of-the-box as protocol definitions and specifications are standard and fixed. The interoperability gaps arising due to the use of different data formats are also bridged in a similar fashion. On the other hand, to address the data model interoperability gaps, these platforms offer data transformation tools which facilitate the system integrator user in writing the data model/schema mapping rules or source code for data transformation that is specific to the interaction between a particular pair of services. The examples of such data transformation tools include XSLT, Jolt, etc. for specifying the data transformation rules and use of (scripting) languages such as JavaScript, Python, Java, R, etc. to perform the data transformation directly. Thus, as the data transformation process is manual, it is tedious, costly and time consuming. These data transformation tools aren't intuitive - for some tools, there is a steep learning curve (e.g. XSLT); for others, they aren't 'Turing complete' (e.g. Jolt).

In addition, in order to write rules or source code for data transformation, the developers need to know the semantics for the data. The semantic specifications are not always available and even if they are, they are often based on different 'non-standard' vocabularies, which are service, platform or organization specific. Therefore, the system integrator cannot write the transformation rules alone and depends on information from service provider and consumer sides.

This involves even more humans in the loop for giving information and feedback on the semantics of the data, for checking the correctness of the transformations, etc. making the data transformation process even more 'tightly coupled'. Even if the description of these service/platform/organization specific semantics for service interfaces are available, they are often present in documentation form. These documentations need to be complete and descriptive enough for the service integrator to write mapping rules without any feedback from the providers and consumers of services. Even then, the system integrator has to study this service specific documentation to create mapping rules for a particular pair of services.

There is no system available in the surveyed open source service integration platforms for Data Spine that could facilitate the system integrator by providing data model attributes and their meaning in one place, predict possible mappings or at least guide him/her for writing the mapping rules in a semi-automated way. The details of this survey were included in the deliverable D3.1: EFPF Architecture-I. Thus, the data transformation process is slow,

cumbersome and error prone. Therefore, a new data transformation tool is needed which solves these problems.

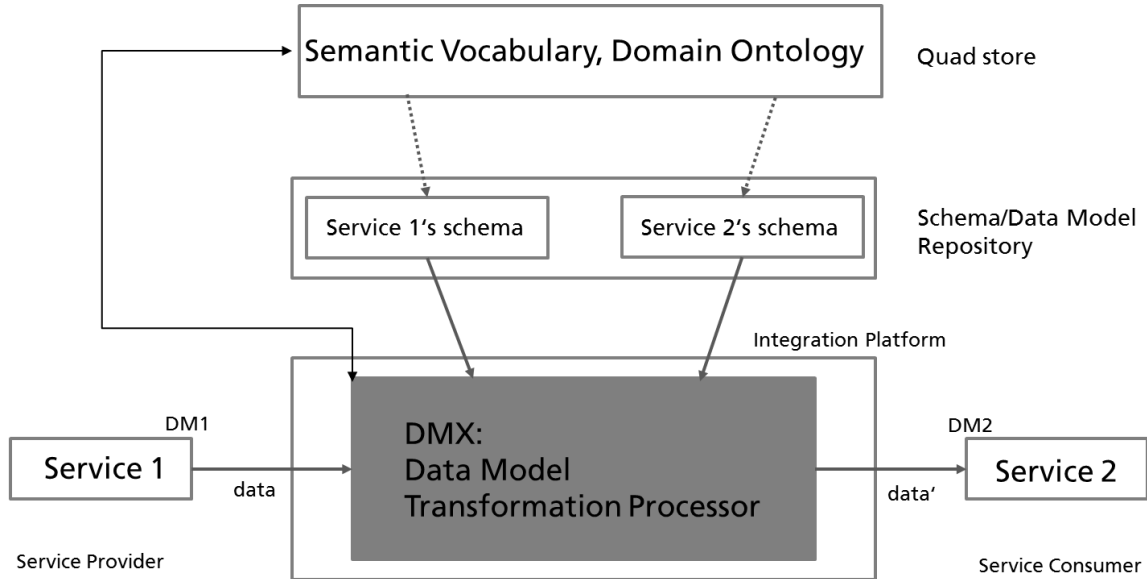


Figure 27: Conceptual Components of Data Model Transformation Tool Suite

### 6.2.1.1 Overview and Expected Operation

The objective of the Data Model Transformation Tool Suite is to address the problems listed in the previous Section. As shown in Figure 27, it contains semantic repositories to facilitate the lifecycle management of functional as well as technical metadata of services based on standard vocabularies and the lifecycle management of these standard vocabularies. In addition, it provides a Data Model Transformation (DMX) Tool that guides the system integrator for writing the mapping rules in a semi-automated way, thereby reducing the human involvement to make it faster, easier and more efficient. The Tool Suite is currently in conceptualization phase and a detailed description of it would be included in the next version of this deliverable D4.14.

### 6.2.2 Secure Data Store Solution

Secure Data Store Solution provides a customizable solution to the problem of maintaining balance between data privacy and sharing, giving data owners the opportunity to manage data locally, on site, giving them complete control over their data, while allowing for specific configurations of how to share data.

Secure Data Store Solution handles this with a focus on providing the data owner with the utmost control over data sharing, where the data owner retains physical control over the computing systems storing the data, eliminating any uncertainty of how their data is handled in cloud computing solutions. With the caveat that the data owner must now properly administer their computing resources with respect to cybersecurity, they can also choose the level of effort they wish to secure their data, and if they wish to use extraordinary measures.

Secure Data Store Solution is intended to facilitate connecting sensor data with analysis tools, by not only providing the capabilities for storing data a later time, but to allow for the explicit definition and access scope to the data to be shared. To this end, timeseries data collected from sensors on the shop floor is stored to build a historical record of activity, that can later be selectively used as input for analysis tools.

Secure Data Storage Solution has two major security goals, enforcing data access along the lines of a meaningful authorization scheme, and ensuring that the stored data is sufficiently protected while in storage.

The strongest guarantee that Secure Data Store Solution offers to protect stored data is that individual data owners can deploy an instance on site, allowing the data owner to retain physical control, and allowing the data owner to directly verify the security. By nature, utilizing cloud-based solutions require trusting external entities to ensure this. However, while retaining the control, the data owner is now responsible for the proper administration for the system to ensure the proper handling of the data with regards to industry best practices. To this end, Secure Data Store Solution is built around the concepts of High Availability and Redundancy. Notably, backups are possible, but are dependent upon the data owner.

To further assist in data privacy vs. usability, pseudonymization routines and analysis techniques are provided to afford additional solutions for data owners.

### **Data Integrity and Quality**

Secure Data Storage Solution provides options for maintaining data integrity, it is ultimately up to the individual deploying organizations' operators to ensure that resources sufficient in quality and number are given to ensure the success of Data Integrity.

The Secure Data Store Solution must be able to operate so that the loss of individual computer does not break the system. To this end, all the subcomponents of Secure Data Store Solution can be deployed in clusters. Furthermore, the subcomponent clusters should be able to still operate given a sudden and unexpected machine failure, in order to ensure High Availability and avoid gaps in service.

Another consideration is the potential for a complete loss of the computing resources, for example, severe site damage from a fire or natural disaster. Secure Data Store Solution provides capabilities for backing up the system, but it is left to the system operators to ensure procedures are in place regarding frequency and offsite backup locations.

### **Data Sharing Controls – UMA**

User Managed Access (UMA) is a protocol standard for authorization that integrates with OAuth and OpenID connect for authentication and identity management. Within UMA, Secure Data Store Solution acts in the role of a Resource Server.

Users can request authorization to access specific resources, which can then be granted by the data owner. This allows greater flexibility in usage.

### **Limiting Data Access Scope**

Secure Data Store Solution is intended to record data for latter access, especially with respect to investigating incidents after the fact to ascertain incidents, it is possible that much data will be accumulated.

During such an investigation, Secure Data Storage allows data to be addressed in partitions via time ranges. Thus, access to data can be requested, controlled, and granted in a flexible manner limiting data scope

### Pseudonymization

Pseudonymization replaces personally unique identifiers with an alternate unique identifier, seeking a balance where privacy is maintained, but analysis techniques can still relate associations to individuals for stronger results. To facility additional privacy management solutions, Secure Data Store Solution provides utilities for managing pseudonymization in the stored data.

Given that individuality is still maintained within the dataset, there is some risk that additional information contained within the dataset is still revealing enough that the anonymity granted by pseudonymization can be broken. As such, Secure Data Store Solution also provides an analysis tool to estimate this risk.

#### 6.2.2.1 Configuration

Notably, as a deployment on-site foregoes cloud resources, it is important that sufficient computing resources of adequate quality be provided by the administrative team. The onsite administrative team needs to have the skills and knowledge of such systems for successful implementation fulfilment, with the capacity to understand on-site demands and provision corresponding computing resources.

To support on site deployments, Secure Data Store Solution is provided as a collection of Docker containers. Docker is a "container" system, whereby services are packaged in small complete systems isolated from the hosting environment, which can be easily scaled as virtual resources.

The following services need to be established, as per the following table. To maintain redundancy, this should include at least two computers capable of running the software alone, and ideally at least a small third system that can ensure a voting does not result in a split cluster.

Service	Docker Image
Redis	redis:6
MongoDB	mongo:4
InfluxDB	influx:1.8

The main Secure Data Storage docker image needs to be configured with the parameters as environmental variables as specified in the following table.

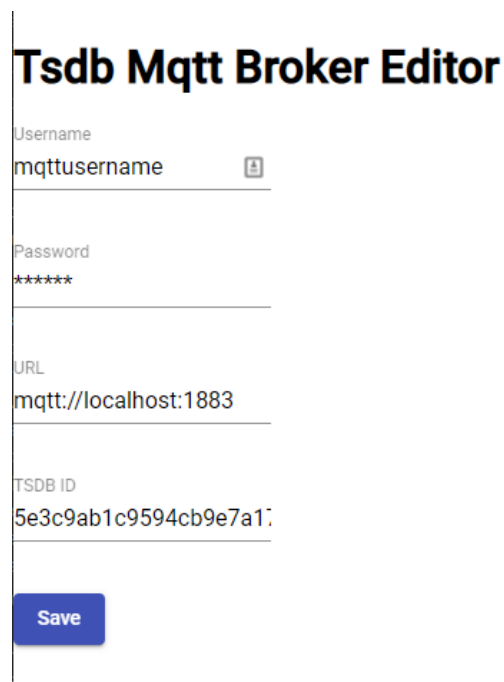
Environment Variable	Description
MONGO_URL	Connection URL for the MongoDB database.
INFLUX_HOST	Connection URL for the InfluxDB database
KC_CLIENT_ID	Client ID within the target KeyCloak realm
KC_CLIENT_SECRET	KeyCloak client secret.



KC_WELLKNOWN	URL of the target KeyCloak realm's OpenID configuration
SECRET	Unique secret value per deployment.

### 6.2.2.2 Operation

After the service is launched, users can login using the configured KeyCloak instance. Users can then configure how to select data to record and store. Figure 28 shows a prototype screen where the source Data Broker connection can be configured. The URL and login credentials can be supplied here.



**TsdB Mqtt Broker Editor**

Username  
mqttusername

Password  
\*\*\*\*\*

URL  
mqtt://localhost:1883

TSDB ID  
5e3c9ab1c9594cb9e7a1

Save

Figure 28: Configure Secure Data Storage Solution source

Inside of a data resource, users can specify timeseries data for specific sensors. To this, users can specify a name for the timeseries, the source data broker, and selection criteria within the broker. Figure 29 shows a prototype screen where the timeseries data to record can be configured.

## Timeseries Editor

Name  
TestTimeSeries

TSDB ID  
5e3c9ab1c9594cb9e7a17

MQTT Broker ID  
5eb1140977069ec5a3db

MQTT Topic  
efactory/ascoratest/sens

Save

Figure 29: Configure Secure Data Storage Solution connection

After data has been recorded, it is possible to execute queries to retrieve data. A demonstrator in the prototype UI is shown in Figure 30, where stored timeseries data is sorted by time to select the 3 most recent data points.

```
select * from "5eb1140977069ec5a3db4391"
  order by time desc
  limit 3
```

Query

Response:

```
[
  {
    "time": "2020-06-09T06:45:31.019Z",
    "efactory/ascoratest/sensor1": 14970.688922072028
  },
  {
    "time": "2020-06-09T06:45:31.019Z",
    "efactory/ascoratest/sensor1": 14970.688922072028
  },
  {
    "time": "2020-06-09T06:45:30.321Z",
    "efactory/ascoratest/sensor1": 15394.841721805902
  }
]
```

Figure 30: Secure Data Store Solution query evaluation

### 6.2.3 The System Security Modeler

Risk assessment is fundamental in common scenarios present in the Industry 4.0 domain and in particular in EFPF. In fact, any set of systems sharing any type of proprietary, sensitive or process data in digital form with any of your suppliers or other third parties is subject to ICT related risks. It is an established good practice to perform due diligence of conducting a cybersecurity and GDPR compliance risk analysis. Any time the ICT services used in a supply chain change, there is a need for performing a risk assessment. The insurers Hiscox spoke to 5,400 small, medium and large businesses in the UK, Germany, the US, Belgium, France, the Netherlands and Spain<sup>15</sup>. They found that the number of cyberattacks rose significantly in 2019, with 60% of businesses reporting one or more attack in 2019 (up from 45% in 2018). Official UK government statistics from the National Cyber Security Council (2019)<sup>16</sup> indicate that 31% of UK small businesses identified a cyber breach or attack last year (mainly phishing emails, impersonations and malware). The impact of these attacks included lost files, lost network access, websites taken down and software systems corrupted or damaged – at an average financial cost of £3,650 p.a. to fix<sup>7</sup>. Therefore, as suggested in a Deloitte's Insights article “organizations should perform risk assessments across their environment, including enterprise, Digital Supply Networks,

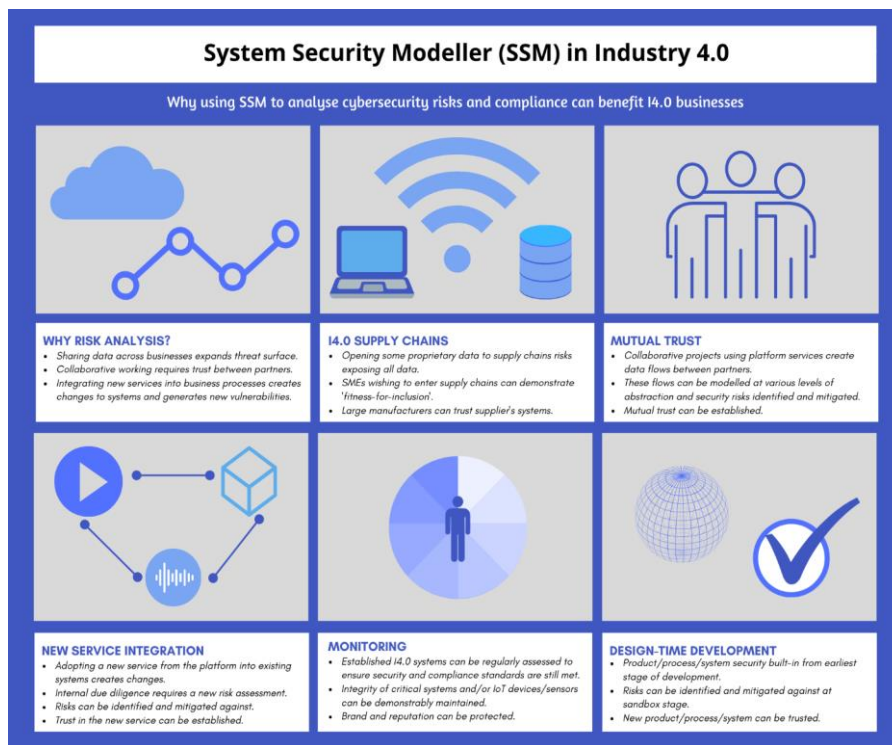


Figure 31:SSM overview

<sup>15</sup> BBC News, 23rd April 2019. 'More than half of British firms 'report cyber-attacks in 2019' [Available on: <https://www.bbc.co.uk/news/business-48017943>]

<sup>16</sup> UK Government Official Statistics, 2019, *Cyber Security Breaches Survey 2019: Micro/Small Business findings*. National Cyber Security Council. [available on: [https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment\\_data/file/791943/CSBS\\_2019\\_Infographics\\_-\\_Micro\\_and\\_Small\\_Businesses.pdf](https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/791943/CSBS_2019_Infographics_-_Micro_and_Small_Businesses.pdf)]

industrial control systems, and connected products, and use those assessments to determine or update their cyber risk strategies”<sup>17</sup>.

In an Industry 4.0 manufacturing environment, typified by large quantities of proprietary data moving rapidly between businesses and their supply chains, effective cybersecurity has become central – because whenever data flows between different systems it introduces security risks. Large manufacturers want the confidence to enter into ad-hoc collaborations with a range of SME suppliers, and SMEs want to prove they are cybersecure to large manufacturers in order to enter into the supply chain.

Yet, becoming cybersecure is often seen as a complex and expensive process for SMEs, many of whom may lack the expertise in-house to conduct extensive risk assessments. For example, 68% of SMEs have no cybersecurity policies or risk management processes and 70% have not conducted any risk assessment in the past year<sup>7</sup>. This can put them at a competitive disadvantage when compared with large suppliers. The System Security Modeller (SSM) tool, provided as one of the component services available to SMEs, and others, on the EFPF platform, can help overcome these barriers (see Figure 22).

In general terms, a range of benefits can be seen when conducting effective cybersecurity risk and compliance assessments either on existing supply chains by individual companies, or as a vital part of introducing new services or suppliers into a system. Equally, risk assessment can be undertaken mutually, in an end-to-end process, by manufacturer and supplier together. This can develop strong mutual trust in each other’s systems and promote positive collaborations.

#### **6.2.3.1 Configuration**

The tool is provided as a service and does not need installation or configuration. The tool required the design of the system model to be analysed and it provides a graphical interface to support this action. The tool is integrated with the EFPF platform Single Sign On and all the models are associated with the specific user who created them.

#### **6.2.3.2 Operation**

The tool follows the workflow defined in the ISO27005<sup>18</sup> 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 10). The asset models represent the topology of the physical (the infrastructure), logical (the processes) and dataflow (the data) layers.

---

<sup>17</sup> Waslo et. al., 2017. *Industry 4.0 and cybersecurity: Managing risk in an age of connected production*. Deloitte Insights Article. [Accessed on <https://www2.deloitte.com/us/en/insights/focus/industry-4-0/cybersecurity-managing-risk-in-age-of-connected-production.html>]

<sup>18</sup> ISO/IEC 27005: 2018 – Information Security Risk Management <https://www.iso.org/standard/75281.html>

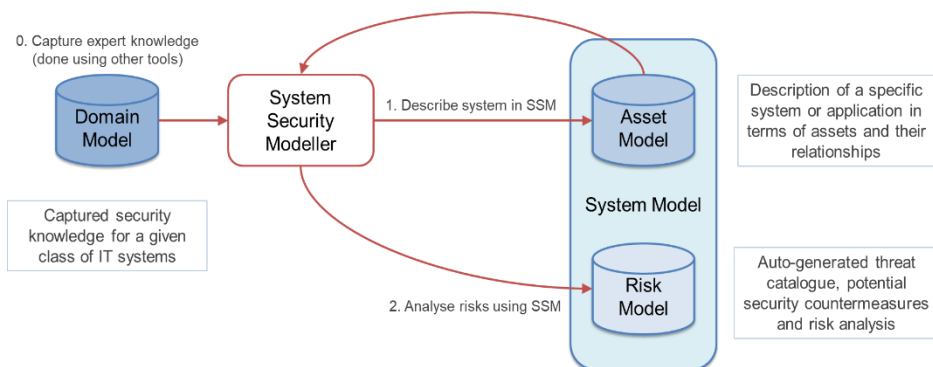


Figure 32: Overview of SSM Data Models

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

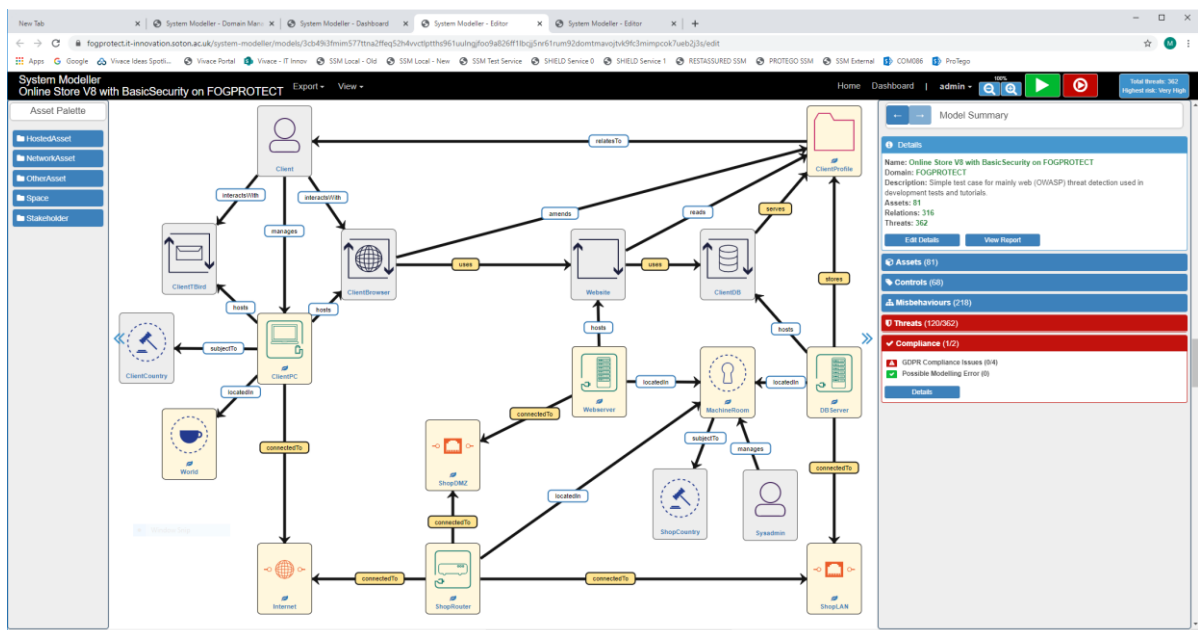


Figure 33: SSM screenshot

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.

## 6.3 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.

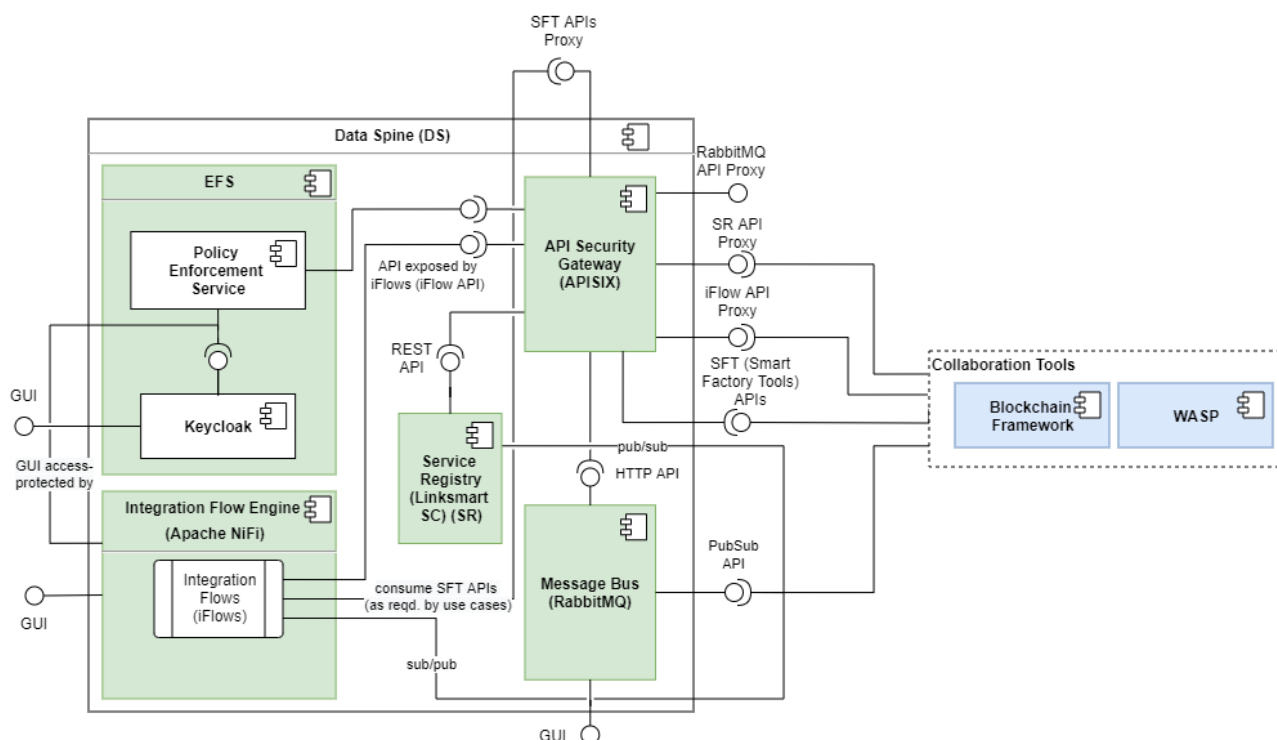


Figure 34: Interaction between the Collaboration Tools and the EFPF Data Spine

### 6.3.1 Blockchain Framework

The blockchain application for shipping from the base platform COMPOSITION has been brought into EFPP and is being enhanced to be a specific application of the e EFPP Blockchain and Smart Contracting Platform. The DApp, also further elaborated in EFPP, uses mobile devices and physical data from the device and external sensors to provide evidence for the information that is entered into the blockchain.

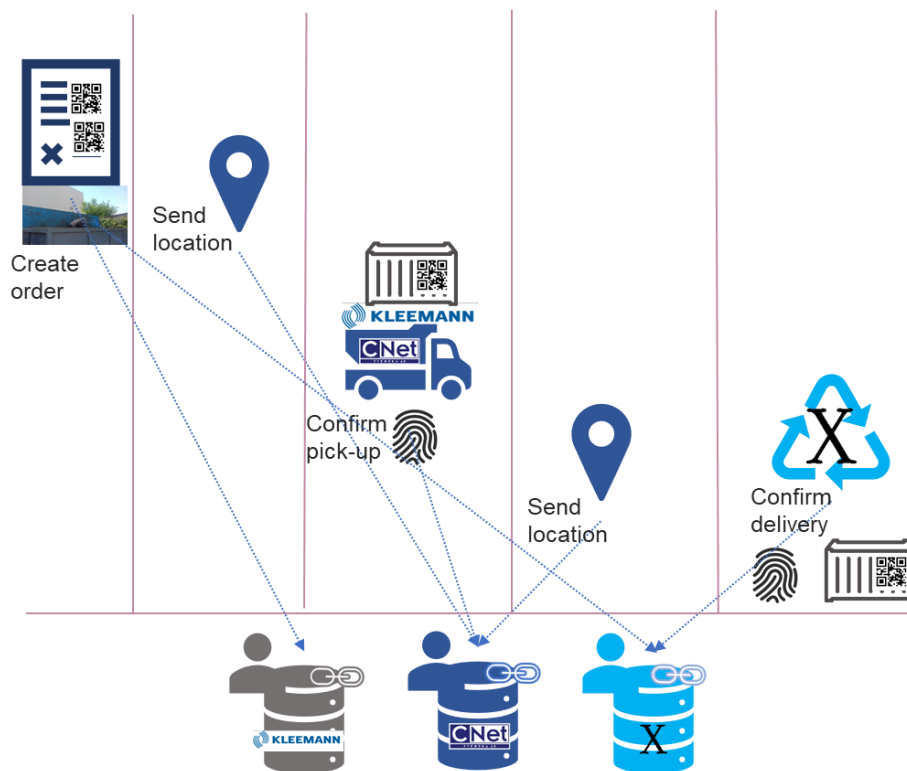
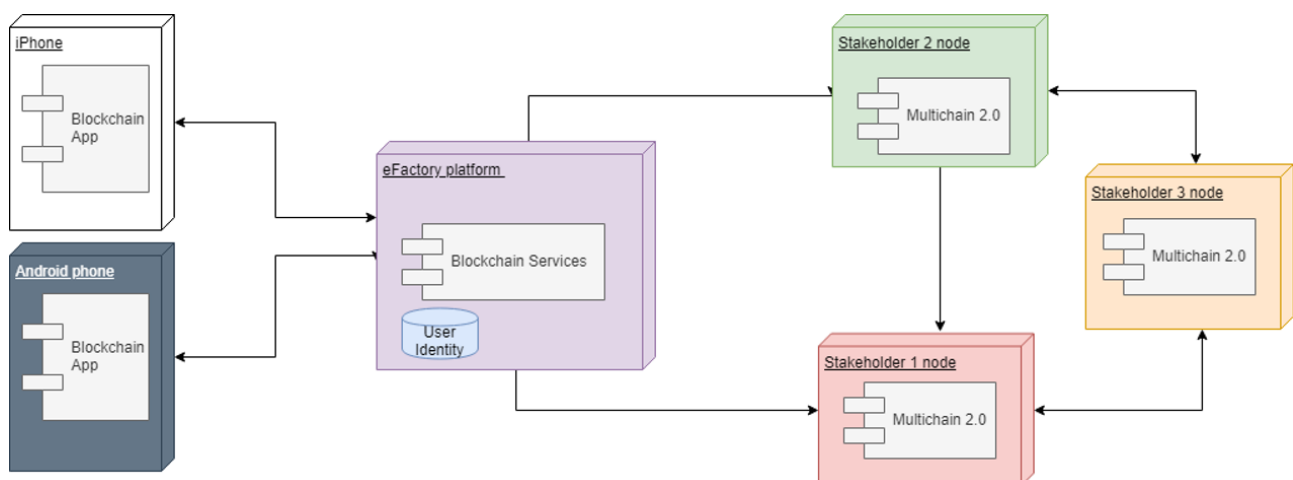


Figure 35: Delivery process in the first version.

This blockchain framework from base platform COMPOSITION has been upgraded and the blockchain implementation is being modified to use the blockchain implementation chosen for EFPP, Hyperledger Sawtooth. The mobile DApp has been made multi-platform and is being adapted to EFPP pilot needs. To further strengthen the integration with EFPP the shipping process is being reengineered to work with a BPMN process engine, which will allow the user to configure the process in the WASP tool.

### 6.3.1.1 Configuration

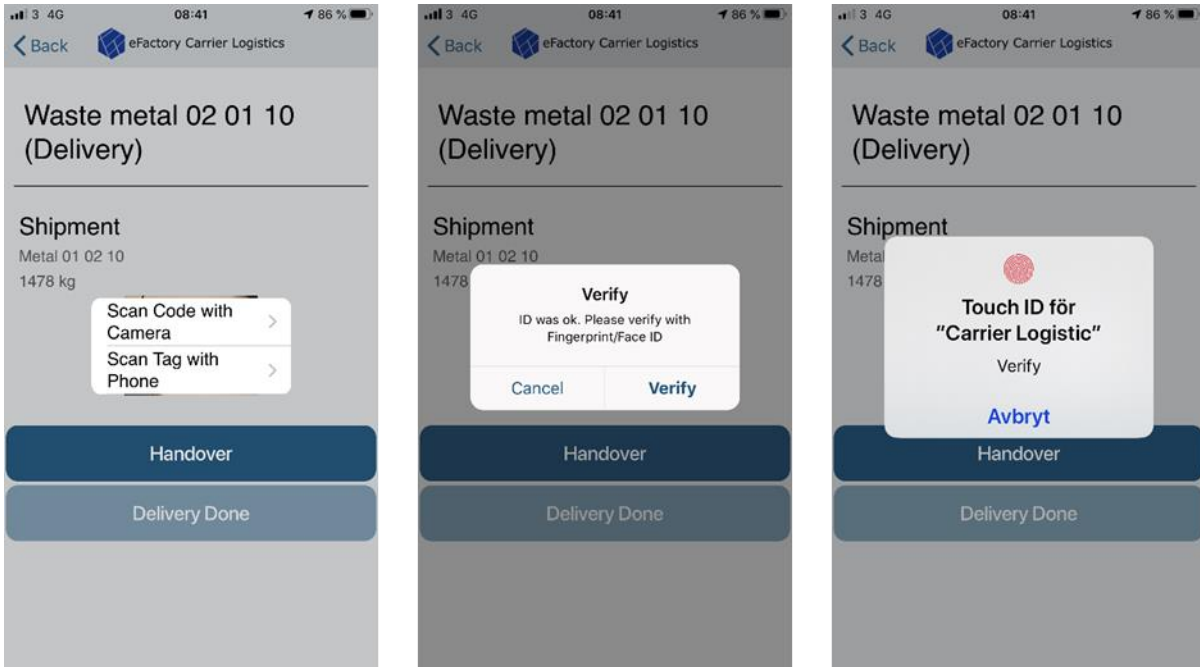


To deploy the shipping DApp, a peer-to-peer network of blockchain nodes needs to be set up. There are already nodes set up that is currently used by the DApp, new nodes can attach to this network or build a new private consortium blockchain. The first version used Multichain 2.0 nodes, while the next update one will use Hyperledger Sawtooth as the other



EFPF blockchain applications. The DApp needs to be deployed to the mobile unit (iOS or Android) through the development IDE. At the time of writing, it is not published to App Store or Play.

### 6.3.1.2 Operation



The key feature of the mobile app is to use physical interaction to address weak points or loopholes where to manipulate the supply chain. We use added sensor data and biometric identification to corroborate the transaction data (e.g. a receipt of received goods). As the data is a representation of a transaction taking place in the physical world and entered into the blockchain, it cannot be validated by the blockchain transaction rules or smart contracts directly. To circumvent this reliability problem of physical interfaces we add other metadata such as NFC tags, weight, images and sign in with face or fingerprint id to provide other evidence that the event took place as described.

### 6.3.2 Distributed Workflow and Business Process Design and Execution

WASP is a platform for fast automation of processes regardless of the type of activity, whether it is an internal task like providing direct input from people or making a decision; an external task done by other software-based services e.g. a manufacturing machine that can be controlled remotely; or a combination of these. WASP follows a SaaS operation model; thus, users subscribe to one of our membership types in order to have access to a range of functionalities that will automate and overall help you improve your services and processes.

The different components of the WASP platform are described below. For graphical representation, see Figure 36.

**Camunda engine.** This component is the core for controlling all processes. It tracks all process instances, process templates, activities and their execution. This is effectively the brain of the whole Workflow and Service Automation Platform. Communication occurs via REST API that sits on a Tomcat distribution; it has its own database.



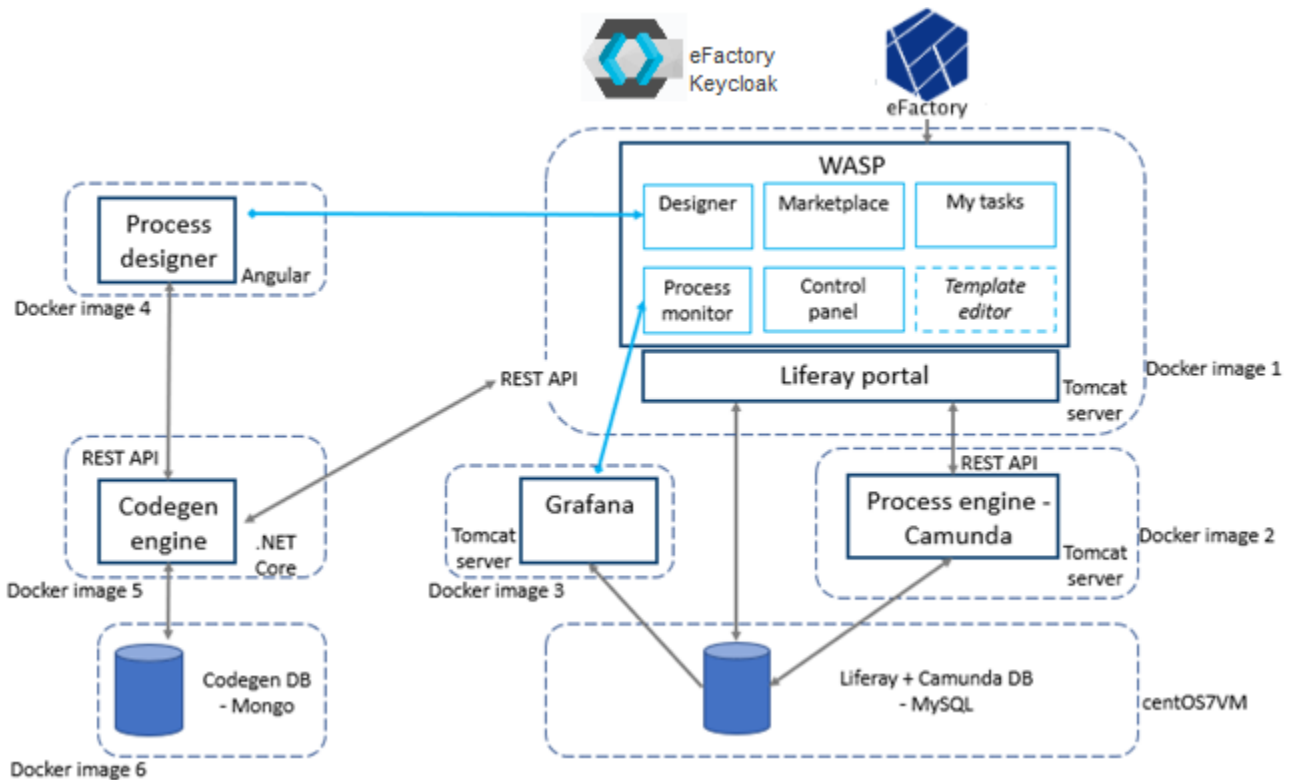


Figure 36: System architecture

**Liferay portal.** This component is a subsystem in its own right; it manages websites including user administration. It hosts the website with internal applications called portlets. Each portlet represents a distinct functionality/tool within our Workflow and Service Automation Platform.

**Process designer portlet.** This portlet sits inside the Workflow and Service Automation platform as an independent application. It allows the user to manage processes in a graphical way, where processes are based on BPMN 2.0 standard.

**Marketplace portlet.** This portlet also sits inside the Workflow and Service Automation platform as an independent application. It allows the user to manage services for later use in processes.

**Control panel Portlet.** This portlet also sits inside the Workflow and Service Automation platform as an independent application. It allows the users to manage their processes. Users can run a new instance of a process, view the current running instances and see which stage the process is at; also suspend and delete a process.

**Process Monitor Portlet.** This portlet also sits inside the Workflow and Service Automation platform as an independent application. It connects to Grafana which is a data analytics suite. This module is still under development and not currently visible.

**Task Portlet.** This portlet also sits inside the Workflow and Service Automation platform as an independent application. It allows the users to complete the user tasks which are assigned to them. Completing the task could show data to the user from previous tasks as input parameters and also have output boxes for the user to fill in which could be used in the next stages of the process.

**Template Editor.** This portlet also sits inside the Workflow and Service Automation platform as an independent application. New portlet we haven't started yet.... editor of forms, so that

the Designer can add them to user tasks and Task portlet can show it for user inputs / outputs in a more customisable way.

**Liferay DB.** Database where website information and configuration are stored. Also, user data is stored there.

**Camunda DB.** Database where process information and any engine configuration are stored. No track of user exists in this one.

**EFPF Portal.** This represents the only access point from the Internet, which is via a web browser, in this case an iframe is used to display WASP on the EFPF portal.

**EFPF KeyCloak.** KeyCloak has been added to WASP as an option to sign in via OpenID Connect/ Oauth

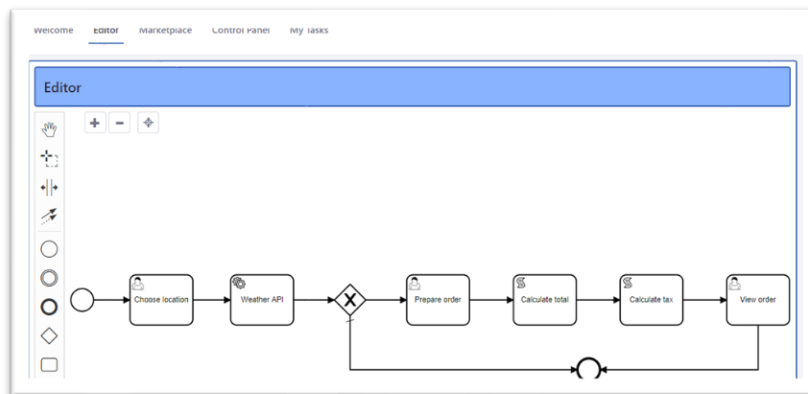


Figure 37: Snapshot of WASP GUI

### 6.3.2.1 Configuration

WASP is offered as Platform-as-a-Service in the EFPF ecosystem. The WASP GUI is integrated in the EFPF Portal and access is enabled by through the EFPF Single-Sign-On and user management service. This means, any EFPF user can make use of the WASP functionality e.g. design and execute processes, add services in the WASP marketplace and share processes or specific activities in a process with other EFPF users.

### 6.3.2.2 Operation

WASP offers an intuitive GUI that allows users to design processes or workflows using standard BPMN notations. The operations of the WASP platform can be wide ranging in nature depending on the types of activities or processes being modelled. So far, WASP is being used by the EFPF users to manage distributed activities in a supply chain scenario and also to design delivery processes that provide track and trace functionalities to the involved stakeholders. A detailed user guide, providing step by step instructions, is made available on the EFPF Portal.

## 7 Productivity Tools

### 7.1 Introduction

This section describes the Tools and Services available through the EFPF ecosystem that help manufacturing companies manage and optimise production related activities. These solutions range from the visualisation of shop-floor and production data in order to get a better understanding of production status, through to complex machine learning and analytics tools that help understand complex issues in the production processes. Each Tool and Service described in this section will outline what functionality is provided, how it is configured and what the output is to the user.

### 7.2 Industry 4.0 Tools

Industry4.0 is a now a well-recognised phenomenon. It is about digitalisation of traditional activities based on the advances in the smart factory, Cloud computing, big data and IoT domains. The Industry4.0 tools in EFPF interact with the core EFPF infrastructure, such as the Data Spine, to deliver a range of solutions to the manufacturing users.

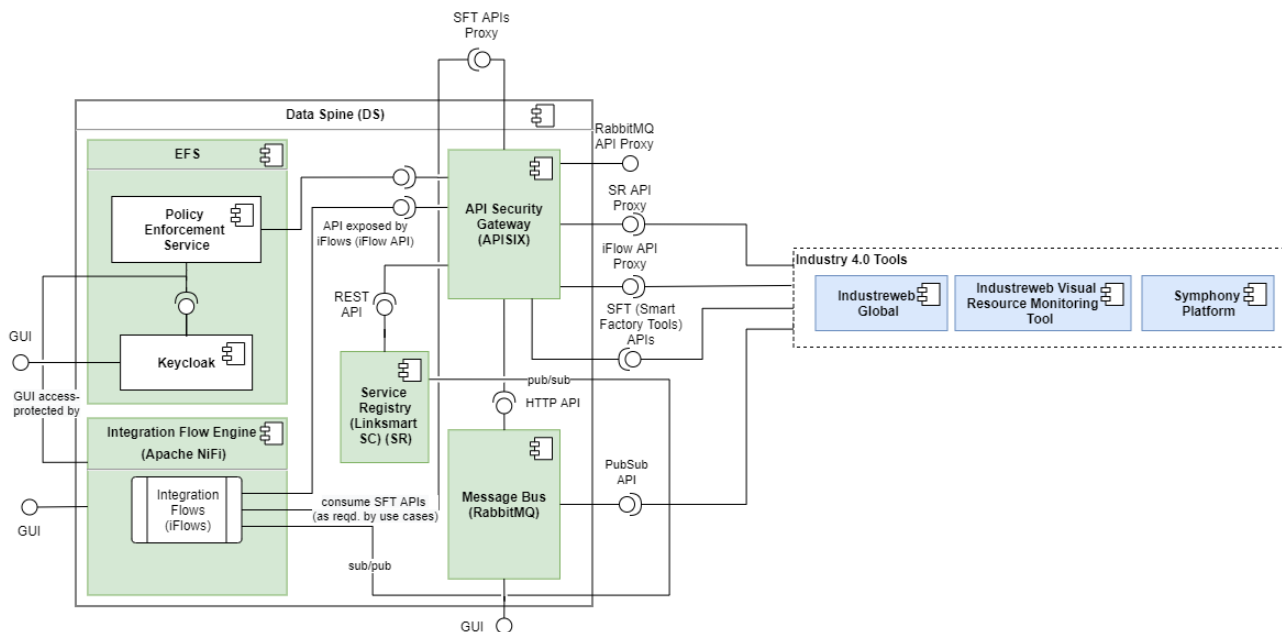


Figure 38: Interaction between the Industry 4.0 Tools and the EFPF Data Spine

#### 7.2.1 Industreweb Global

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 39 the main elements and how it interacts with Industreweb Display screens and Industreweb Collect Factory Connectors. This section will explain what Industreweb is able to offer from a data visualisation and an application perspective

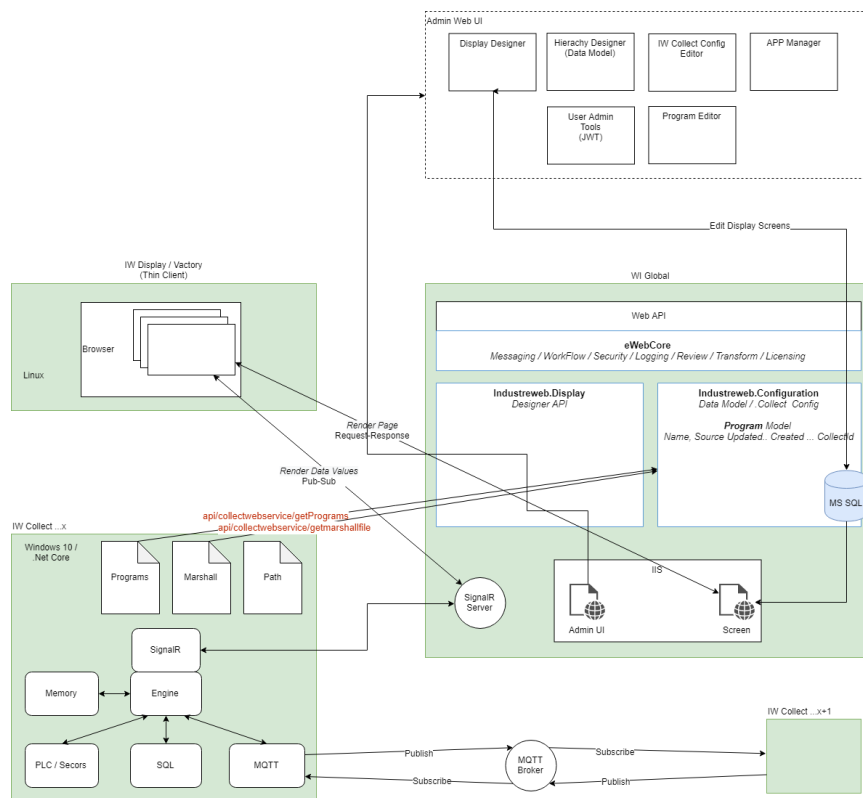


Figure 39: Industreweb Global Architecture and interfacing

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

The architecture for IW Display is shown in Figure 40.

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.

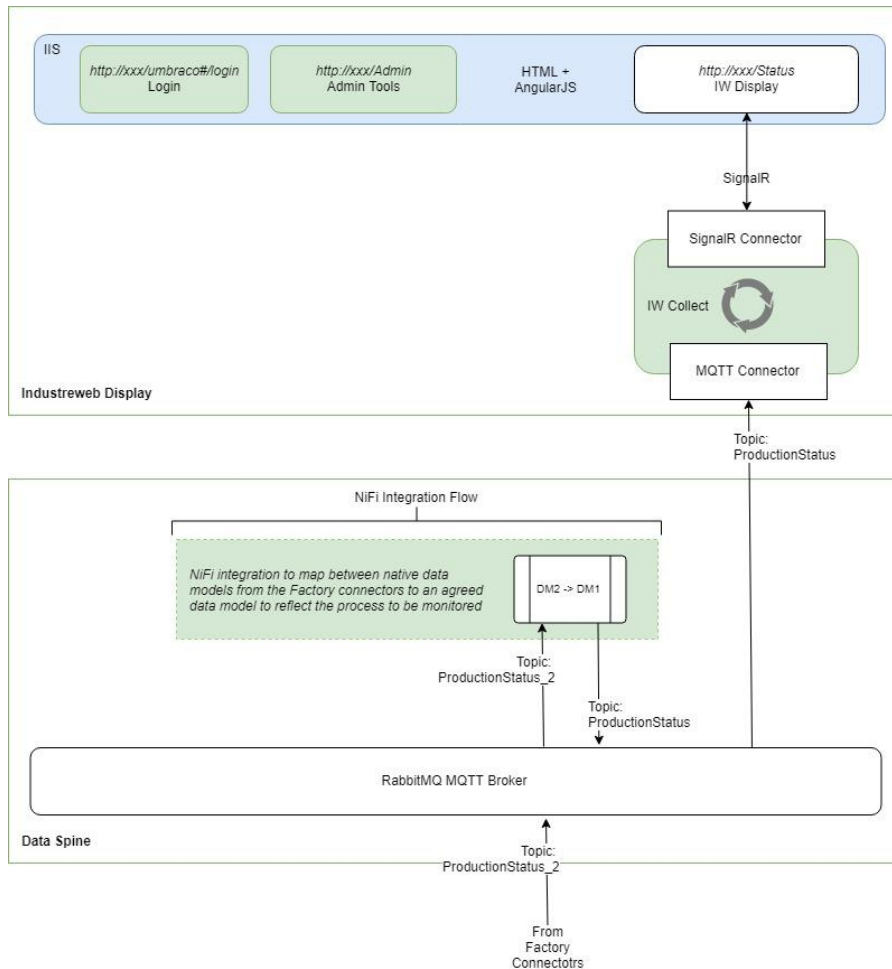


Figure 40: Industreweb Display Architecture

**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.

#### 7.2.1.1.1 IW Global Applications

IW Global allows solution specific Industreweb Applications to be utilised. These consist of readymade screens and additional application specific services along with connectors within IW Collect.

Applications that are available include:

**Work Instructions:** Allow screens or documentations to be display to users based on the operation that they need to perform. This would often be a machine or process operator who would need to know how to perform their role safely and without error, where work instruction could include machine setup, safe operating procedure or the best practice for any manually processes.

**Error Proofing:** Error proofing allows for operator error scenarios to be modelled so that when they occur either the process or machine can be inhibited or a corrective message can be displayed. The goal is to detect and to minimise or prevent user error and thereby maximise quality.

**Fault Escalation:** When machine issues occur it is important to alert this to maintenance staff to call for corrective action to be undertaken. Associated metrics are captured including duration of faults and the nature or fault or error code. Should a fault remain unresolved within a user defined period the fault can be escalated for attention.

**Knowledgebase:** The Knowledgebase Application detects machine fault scenarios and presents existing solutions to maintenance staff. In addition new solutions to faults can also be captured, including descriptions photos, videos and machine documentation with bookmarks. The goal is to capture valuable knowledge on how to fix machines and share best practice.

#### 7.2.1.2 Configuration

Using Industreweb Global, users can perform the following administrative functions:

- Edit Industreweb Collect node configurations
- Manage, create and edit Industreweb Display programs
- Manage, create and edit Industreweb Display screens
- Manage user logins for Administrators and Users to view screens
- Configure settings on back end services to affect systems functionality

For the authoring of Collect Programs IW Global uses a drag and drop interface based upon Google Blockly, shown in Figure 41. Users can define Events they wish to monitor and the Actions they wish to occur when these events are detected as true. Events could include a simple thresholds for specific values or more complex compound conditions coming from multiple data sources. Actions can include saving data to a database, sending messages via SMS or email, or publishing data to a IW Display screen or a Pub/Sub broker.

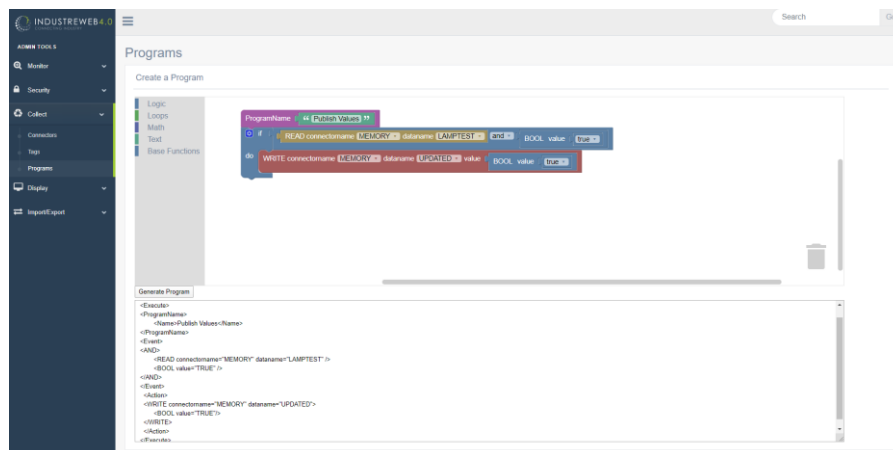


Figure 41: Industreweb Global Logic program Administration

IW Display screens are edited using a Responsive Bootstrap drag and drop UI which includes a palette of screen components. Components can be dragged on to the page within predefined html structures and their attributes configured to adjust presentation and to bind it to data tags from IW Collect. Figure 42 shows the editing tool UI which is being used to design a KPI dashboard.

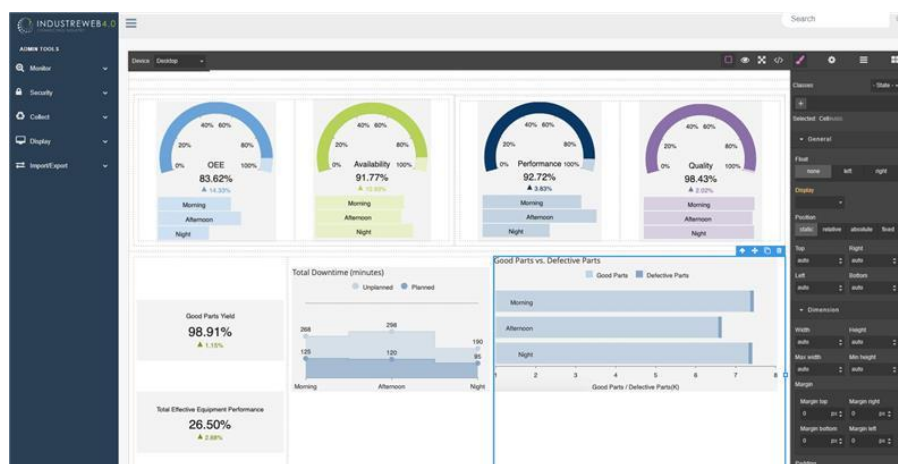


Figure 42: Industreweb global Display Screen editing Tools

### 7.2.1.3 Operation

Since Industreweb Global runs under Microsoft IIS webs server it starts as soon as the server starts. Display dashboards and Admin tool UI's are server based on web requests for each page so will render based on users opening dashboards or Admin tools. The WebApi Service framework that includes the Workflow Service are passive in nature so can be executed when they are required to perform certain utility and orchestration functions. There are two types of Workflow that can be executed in the Workflow Service, persistent and non-persistent, where persistence is the durable capture of a workflow instance so that a workflow can occur over a period of time, maintaining its current position.

When IW Global is running IW Display screen hosting is automatically initiated. Users can access the screens, but depending on the User and Role access defined may have to login to their User account. When the screen renders the page values automatically subscribe to the Industreweb Collect Engine to get real-time updates. This can include numerical or text



labels, graphs or graphics to reflect the machine status. Figure 43 shows a simple line graph and tabular data reflecting the number of components parts in a process. Figure 44 shows the Vactory 3D UI where the colour of the 3D asset relates to a specific KPI such as energy consumption, efficiency or machine breakdown state.



Figure 43: Industreweb Display data visualisation



Figure 44: IW Global Digital Twin UI

### 7.2.2 Industreweb Visual Resource Monitoring Tool

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.



Figure 45: Monitoring unit and camera

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 EFPP cloud or display a message on a screen or dashboard.

#### 7.2.2.1 Configuration

The learning stage can take 1-5 days depending on the complexity and size of the visual library, and must be undertaken as a setup step using an high powered PC provided by the commissioning company Control 2K. Preconfigured AI models supplied with the solution can be used to make implementations easier.

Subsequently using the IW Global Administration tools the Events and Actions that should happen when an object is detected need to be created and deployed to the IW Collect node.

#### 7.2.2.2 Operation

Three scenarios are proposed for this technology in EFPP:

- 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 its continual scanning. When an object is detected a probability of match is made, which if greater than a redefined user threshold the associated Actions can be carried out as shown in Figure 46.



Figure 46: 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.

### 7.2.3 Symphony Platform

Symphony is a complete BMS platform whose basic building blocks can be used to support the automation of different production sites.

The main functions provided by the platform include:

- Support for major automation standards (e.g. KNX, BACnet, LonWorks, OPC-UA, Modbus), allowing seamless interconnection of heterogeneous systems for cross-technology operation
- 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 (e.g. for lighting, curtains, lifters, HVAC, automation control) 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
- Security, Access Control and Anti-intrusion
- 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)

Symphony BMS control logics and system behaviors can be defined both at a local level (single system) and at a global level (multiple systems controlled by cloud-based services).

The generic development framework can define reactions to events which are fed to more or less complex processing rules (ranging from simple algorithms to more complex decision systems), which eventually result into 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.

A simplified graphical interface exists, which can be used as an alternative to low-level programming languages (LUA, Python) that are also available.

An action scheduler allows programming functions with fine-grained timings or simply in a daily, weekly, monthly or seasonal fashion.

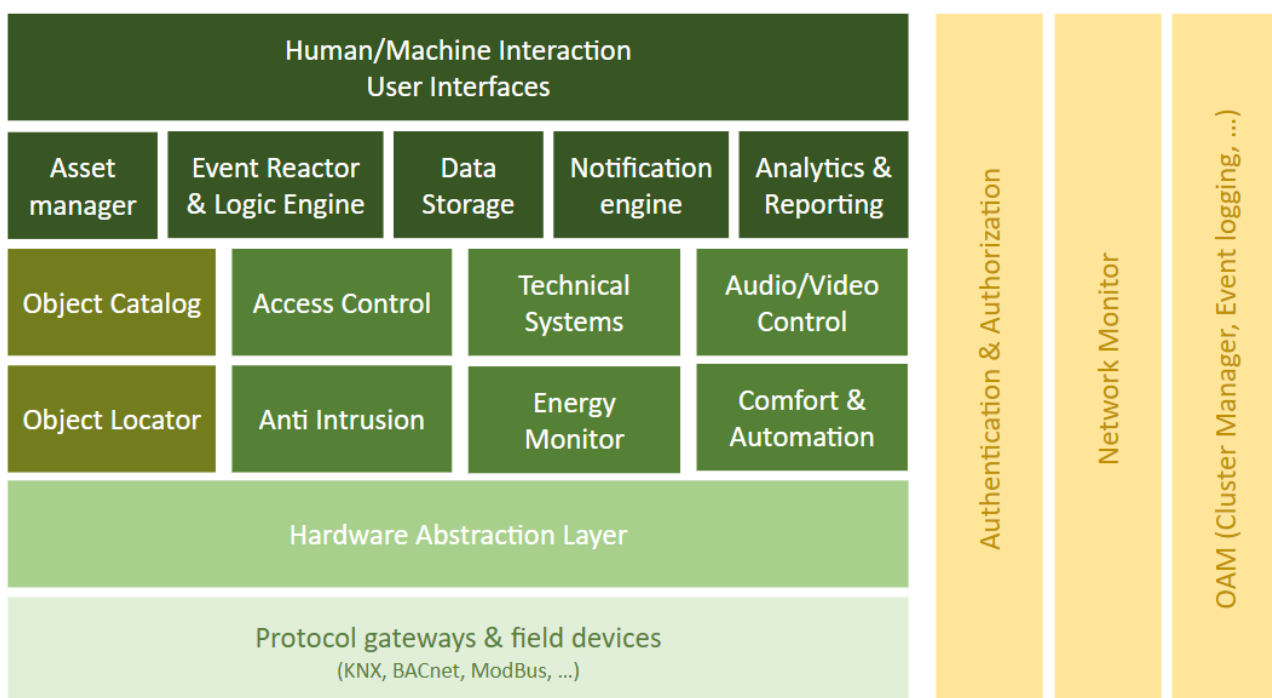


Figure 47: Symphony Building Blocks

The basic building blocks of the Symphony BMS are:

- Low level fieldbus controller (Hardware Abstraction Layer (HAL))
- Specialized functional modules:
  - Comfort automation system (CAS)
  - Technical monitoring system (TMS)
  - Energy manager (SEM)
  - Access control / anti-intrusion system (AIS)
  - Video surveillance system (CCTV)
- Communications
- Network controller (NET)

- Event Reactor (ER)
- Multi-tier storage system (Data Storage)
- Analytics & reporting engine
- Authentication and authorization manager (AAM)
- Event reactor & logic engine (ERLE) to develop processes and workflows
- Cluster manager (high availability)
- Visualization App (UI)
- Cloud gateway
- Cloud reflector

The building blocks provide a complete functional stack covering an automation chain from the field bus level up to the user interface level.

### 7.3 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.

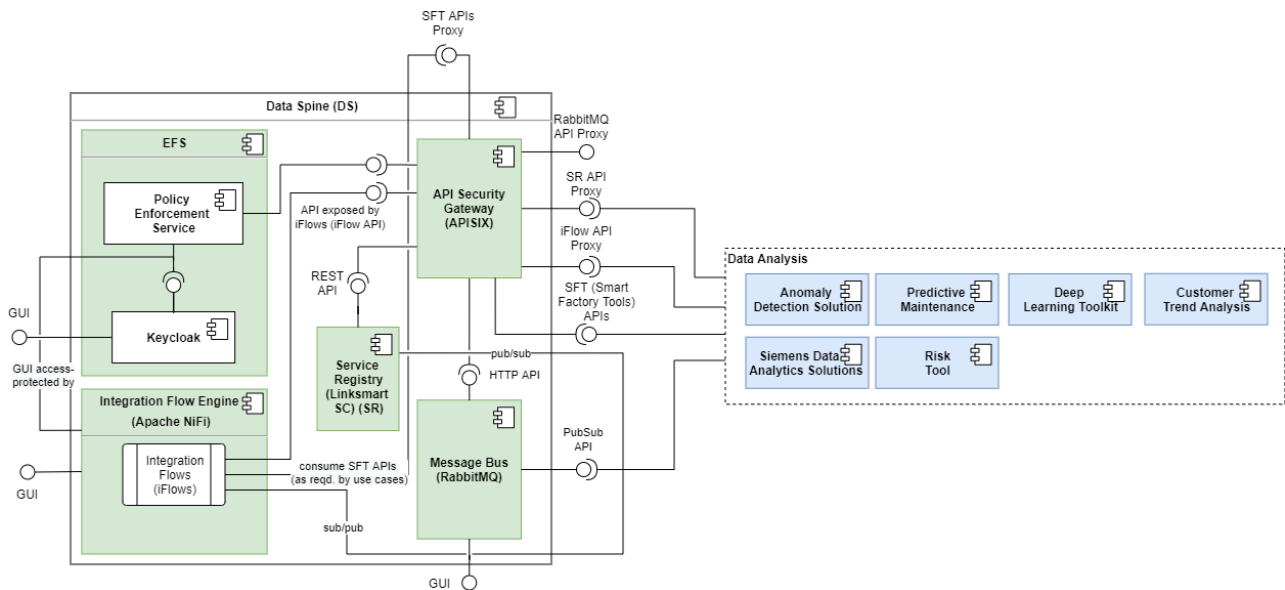


Figure 48: Interaction between the Data Analysis Tools and the EFPF Data Spine

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:

### 7.3.1 Anomaly Detection Solution

The EFPF Anomaly Data Solution (ADS) enables the creation of the building blocks and execution of machine learning-based analytic algorithms on the sensor data. This includes machine learning algorithms supporting supervised and unsupervised scenarios. A broad and diverse range of sensors data serves as inputs for the ADS. The ADS is designed to operate on real-time data as well as historic data.

The design of the ADS takes into account the real-world needs for developing anomaly detection models that can capture the operating behaviour of manufacturing assets. ADS allows using those models to not only detect anomalies (in machine behaviour) in real-time but also to predict the occurrence of anomalies in future. The operating of the ADS requires the capturing of streaming data to be stored in databases as historical data. This historic data is used by the Machine Learning (ML) algorithms embedded in the ADS to build models, which represent the normal and problematic behaviour of manufacturing assets. ADS also enables the deployment of machine learning models and publishing of sensor data as data streams that can be connected with the previously developed models. The processing of data streams through the machine learning models delivers the real-time analytics.

The overall process of model creation, publishing of data streams, processing of real-time data and delivering decision support through visualisations is supported by intuitive step-by-step GUI. This makes the ADS an easy to use tool by the target audience, who are not expected to know too much technicalities of data analytics or the underlying algorithms.

In essence, the ADS is integrated by the following internal components:

- Frontend
  - Workflows. Building of Machine learning Models
  - Deployer-Manager. Board to manage the actions for an ML models: Deploy/Stop, Delete, visualization
  - Broker. Web Interface of the Broker to explore queues
  - Publisher. Machine simulator to publish data on the broker
- Backend
  - Controller
  - Machine engine
  - Model Manager
  - File storage
  - Deployer: A REST API to deploy in real-time the models built through the Web-UI

Within the EFPF platform, the ADS interacts with the following external components:

- EFPF Portal, the portal provides the secure entry point to the ADS interface
- EFPF Data Spine, the Data Spine provides the necessary data brokerage and data transformation support to the ADS

#### 7.3.1.1 Configuration

The ADS is implemented as a web-based solution, which is accessible from the EFPF Portal. The ADS is packaged and deployed as a Docker container. The web-based GUI of the ADS is designed as a dashboard with several tabs, as shown in Figure 49: Web-based GUI of the ADS



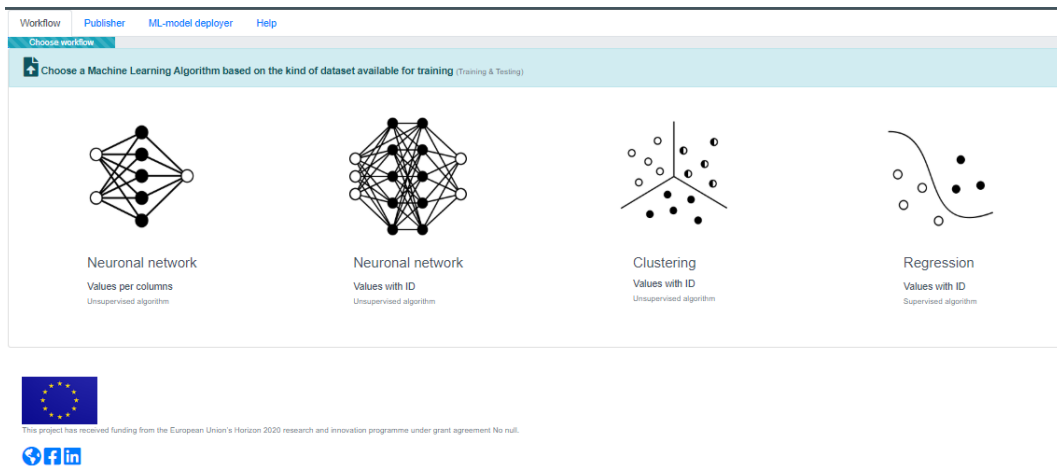


Figure 49: Web-based GUI of the ADS

Each tab on the ADS dashboard is designed to perform a specific function. A brief overview of the different tabs is provided below:

**Workflow:** This tab contains a set of the analytic workflows based on the programming of different machine learning algorithms in the ADS. The workflows provided on this tab allows users to build machine learning models. Users can upload their data files through the online form. The uploaded data files are needed to create the models

**Publisher:** The Publisher component is used to simulate a sensor/machine data. It is useful to test the ADS functionality before it is used in real-world environment. From ADS dashboard, the Publisher tab allows the users to read a dataset (CSV file) and create the messages to be sent to a specific queue in the EFPF Broker, from where the predefined Machine Learning models can acquire the data and use it for analytic functions

**ML-Model Deployer:** The ML-Model Deployer tab provides users the information about existing models and the operations that can be performed on the available models. In this way the ADS GUI not only provides some details of the already developed models, it also provides the operations (run/stop, delete, visualise) that can be performed on the already developed models.

**Broker:** From the ADS dashboard, the Broker tab provides an interface to the EFPF message/data broker (<https://rabbitmq.smecluster.com/#/queues>) that is used to orchestrate data between different ADS components

**Algorithms:** In order to deliver advanced analytic functions in a form or shape that makes them useable directly by the manufacturing companies, the ADS hides the details of the underlying analytic algorithms and allows users to communicate with them using only the GUI. The AI-based Machine Learning algorithms are pre-coded in the ADS and their unsupervised nature means they require no configuration at the time of their use

### 7.3.1.2 Operation

Using the above configurations, the ADS can be operated through the intuitive GUI – as described below:

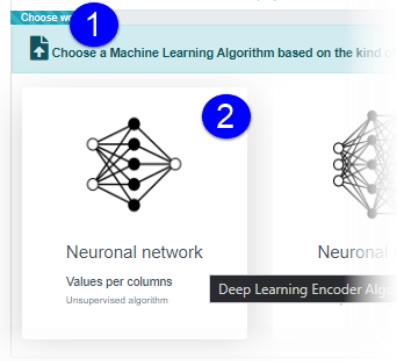
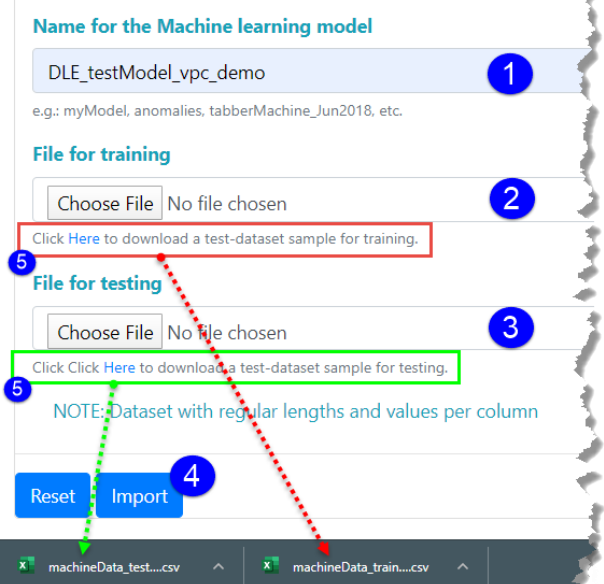


Each workflow requires two datasets in CSV format (samples can be downloaded from the link given on the interface). The first dataset is used to create and train a machine learning model and it should contain the values that represent the expected behaviour of the machine. The second dataset is used for testing of the machine learning model by looking for anomalies in the data. The recommended proportion of the datasets could be any of the following for training and testing correspondently: 60/40, 70/30, 75/25 (percentages).

This workflow takes the users through the following:

- Select a workflow from the dashboard
- Provide the two datasets in the web-based form
- Analyse correlation matrix that is developed based on the analysis of the datasets
- Perform Training & Testing of the machine learning model
- Download the generated model for deployment or reuse

The following table describes the different steps mentioned above:

	<p>Choose a workflow</p> <ol style="list-style-type: none"> <li>1. Select the Workflow tab</li> <li>2. Choose a workflow e.g. the Neuronal Network Values per columns figure</li> </ol>
	<p>Workflow datasets form</p> <ol style="list-style-type: none"> <li>1. Type a name for the ML model to be developed</li> <li>2. Choose the Dataset file (CSV) for training</li> <li>3. Choose the dataset file (CSV) for testing</li> <li>4. Click on import, to import the datasets</li> </ol> <p><b>Note:</b> Users can download a sample of each dataset from the link “here” shown under each dataset field</p>

**Imported Datasets (Training & Testing)**

Dataset for training [upload-dir/machineData\_trainData.csv] **1**

Columns, choose several **3**

	TR37	TR7	TR2	TR100	TR101	TR200
26.4139	33.9821	36.4777	172.528	31.5543	150.854	
26.4411						
26.4953						
26.5903						
26.6174						
26.6038	25.9528	33.2226	36.2336	160.538	30.1845	153.079
26.631	25.9664	33.1412	36.1251	160.552	30.198	153.052
26.7259	26.1223	33.1276	36.0301	160.579	30.1506	152.957
26.7937	26.3054	33.1005	35.8945	160.579	30.1166	152.916
26.8208	26.3733	33.1412	35.8267	160.566	30.1031	152.808
	26.4818	33.1412	35.9352	160.511	30.198	152.74
	26.0884	33.1141	35.9759	160.484	30.1845	
	25.8307	33.209	36.0437	160.566	30.1709	
	25.2611	33.1005	36.0708	160.593	30.198	
	24.6643	32.9513	36.0708	160.525	30.1302	

Dataset for testing [upload-dir/machineData\_testData.csv] **2**

Correlation analysis **4**

### Importation

1. Verify the sample of the training dataset
2. Verify the sample of the test-dataset
3. Check the box for each data parameter that should be to be included in building of the ML model
4. Click on the Correlation button to display the Correlation charts

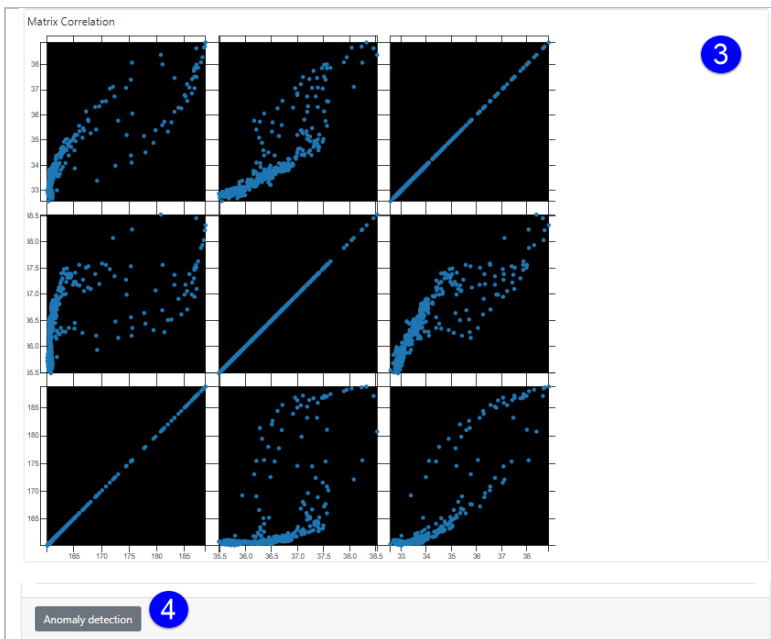
**Correlation Analysis**

	TR7	TR2	TR100
TR7	1.000000	0.859902	0.889061
TR2	0.859902	1.000000	0.602276
TR100	0.889061	0.602276	1.000000

Heatmap Correlation **2**

### Correlation matrix

1. Analyse the correlation matrix table
2. Correlation matrix chart is shown (in dark colour the strong correlations)
3. Matrix correlation is shown to display how spread are the values between a pair of sensors
4. Click on the button to perform the anomaly detection training



Anomaly detection

H2OAutoEncoderEstimator : Deep Learning

Model Key: DLE\_testModel\_vpc

ModelMetricsAutoEncoder: deeplearning

\*\* Reported on train data. \*\*

MSE: 0.0006154693338636158

RMSE: 0.024808654414611362

Anomalies

detected:

Empty

Columns:

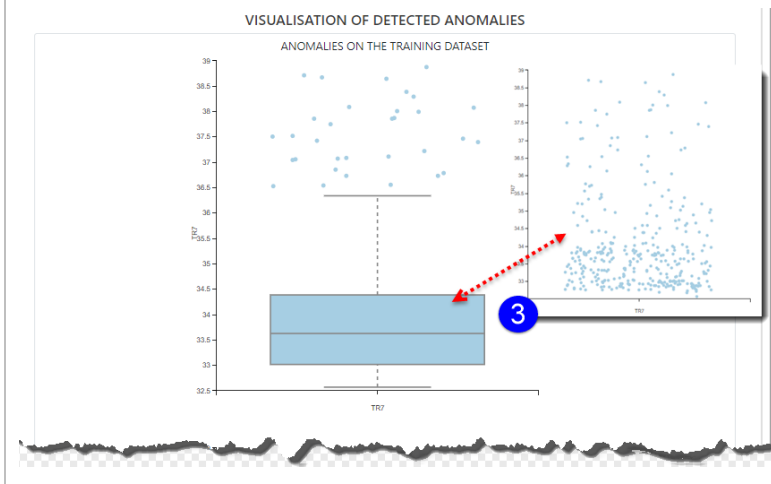
Index:

DataFrame

[Reconstruction.MSE

Rank]

2

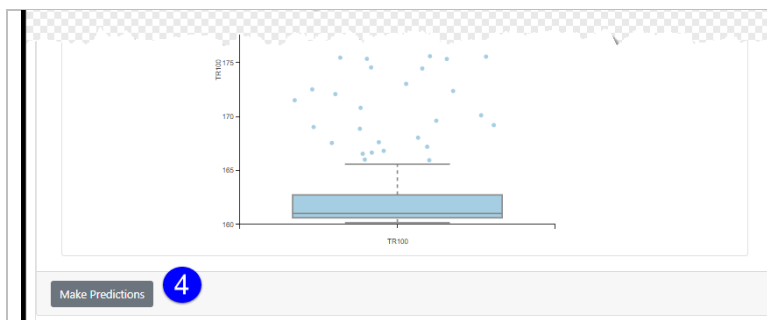


## Training & Testing

This stage uses the training dataset to train the model, looking for the hidden patterns that make possible the anomaly detection.

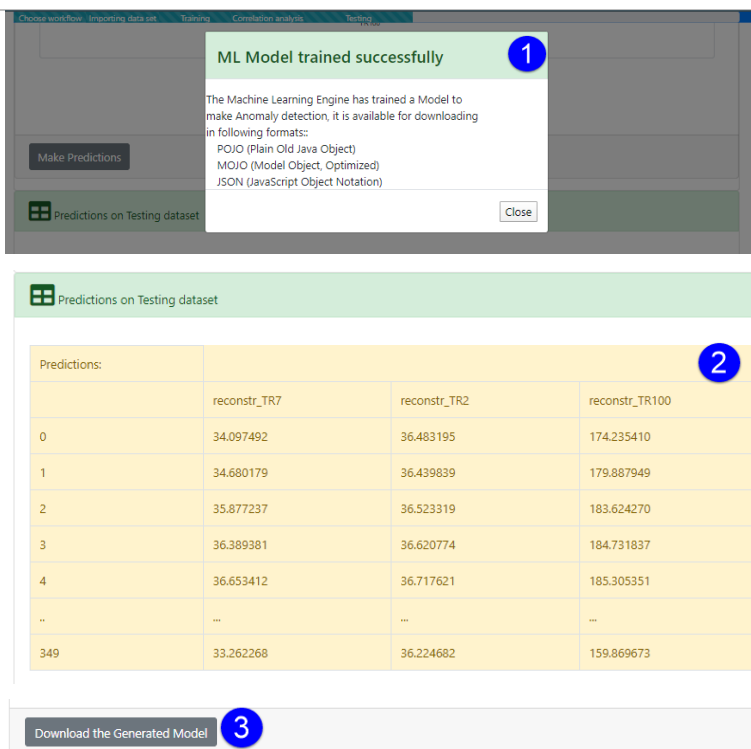
A table with the values of the selected parameters is shown with the relevant threshold. This is to enable users to identify the values that are out the normal behaviour. The following steps are made available to the user:

1. Analyse results of the training
2. Analyse the table of anomalies in the test-dataset
3. Analyse the visualization by a Whisker Plot that shows the values in/out the RMSE threshold, the chart are spited in two groups to show: Train and Test dataset
4. Click on the button to Make a prediction test on the test-dataset



Downloading the generated model

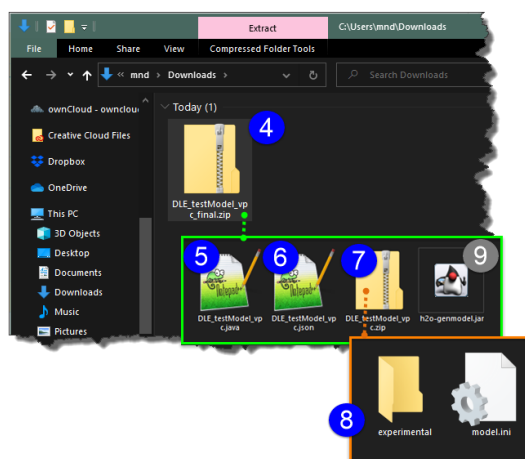
1. Once the user clicked the button to make predictions, a message is displayed to notify the user that the developed ML model built is available in POJO, JSON and MOJO format
2. The table with a sample of the prediction is displayed
3. Click on the button to download already trained model in a ZIP file. This Zip file (4) contains the models in POJO (5), JSON (6) and MOJO (7) format.



What is a MOJO?

A MOJO (Model Object, Optimized) is an alternative to POJO; MOJOs do not have a size restriction. A MOJO is a ZIP file as well (8).

Note: The file shown in the figure by the number 9 is a file with the Machine learning libraries used to implement a model in the production environment



Once the models are developed, the ML-Model Deployer tab provides users the information about existing models (1) and the operations that can be performed on the available models (2). In this way the GUI not only provides some details of the already developed models (3), it also provides the operations (run/stop, delete, visualise) that can be performed on the already developed models (4).

The screenshot shows the 'ML-model deployer' tab with a table of models. The table has columns: #, Model name, Training fields Number, Category, Family, Status, and Actions. There are two rows of models. The first row is 'DLE\_testModel\_vpc\_demo.zip' with status 'Not implemented'. The second row is 'DLE\_testModel\_vpc\_factory.zip' with status 'Implemented'. The Actions column contains icons for play, stop, delete, and visualise. Numbered callouts 1 through 4 highlight specific elements: 1 points to the 'ML-model deployer' tab, 2 points to the table header, 3 points to the table body, and 4 points to the Actions column.

#	Model name	Training fields Number	Category	Family	Status	Actions
1	DLE_testModel_vpc_demo.zip	3	AutoEncoder	Unsupervised	Not implemented	[Play] [Stop] [Delete] [Visualise]
2	DLE_testModel_vpc_factory.zip	2	AutoEncoder	Unsupervised	Implemented	[Play] [Stop] [Delete] [Visualise]



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



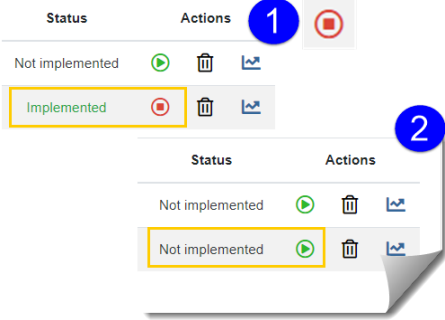
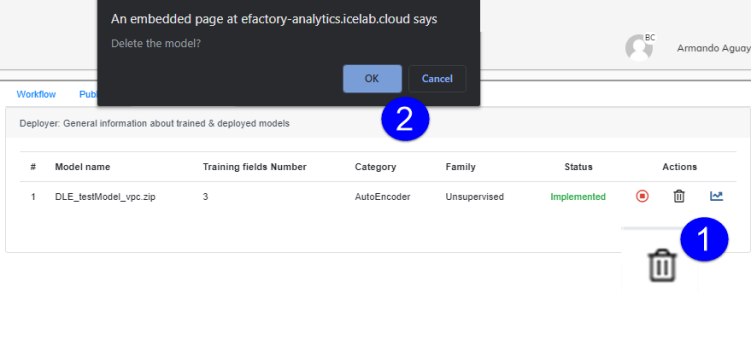
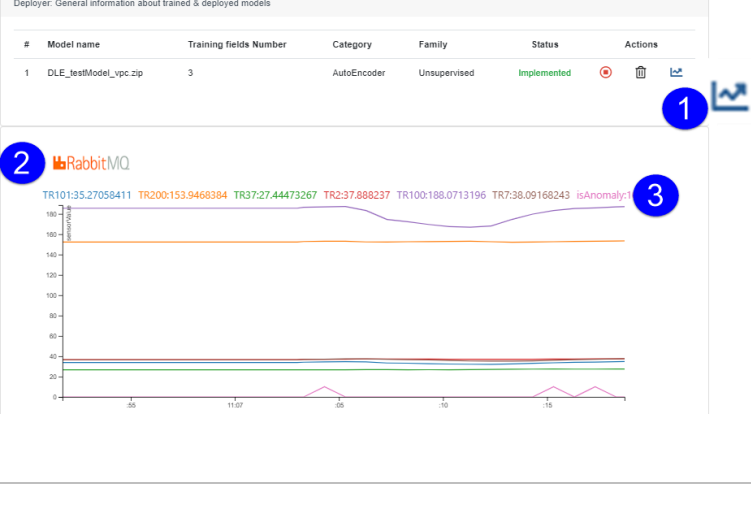
The following table describes the steps that users can perform on the Deployer tab:

The screenshot shows the 'ML-Model Deployer' tab with a form for deploying a model. The form has a dropdown menu for 'ML Model Name' and a 'Submit' button. Numbered callouts 1 through 4 highlight specific elements: 1 points to the 'Actions' column in the table above, 2 points to the 'ML Model Name' dropdown, 3 points to the 'Submit' button, and 4 points to the 'Status' column in the table below. The table below shows the same two models as the first screenshot, but the first model's status is now 'Implemented'.

#	Model name	Training fields Number	Category	Family	Status	Actions
1	DLE_testModel_vpc_demo.zip	3	AutoEncoder	Unsupervised	Implemented	[Play] [Stop] [Delete] [Visualise]
2	DLE_testModel_vpc_factory.zip	2	AutoEncoder	Unsupervised	Not implemented	[Play] [Stop] [Delete] [Visualise]

#### Implementing a model (Deploying)

1. Click on the button play
2. From the form confirm the model by choosing it from the list
3. Click on the button Submit
4. The model will display the implemented status and the stop button

	<p>Stopping a model</p> <ol style="list-style-type: none"> <li>1. Press the stop button</li> <li>2. The status change from Implemented to Not Implemented and the button from Stop to Run</li> </ol>
	<p>Deleting a model</p> <ol style="list-style-type: none"> <li>1. Click on the bin button</li> <li>2. Confirm (Ok) or Cancel de action</li> </ol>
	<p>Stream visualization</p> <ol style="list-style-type: none"> <li>1. Click on the Visualization button</li> <li>2. The Stream chart will be displayed</li> <li>3. Sensor values (updated at the time a data is streamed)</li> </ol>

### What is an implemented or running model?

Once a model is created and stored, it can be implemented or run to connect it with a SOURCE (data stream) queue in the data broker (1). The running model will receive messages (2) from the specific que on the broker and analyse them in real-time through the anomaly detection model implemented. Upon detecting an anomaly the particular message (or sensor reading) is tagged with an extra parameter named isAnomaly (3). This additional parameter is a Boolean value that means whether a row data is an anomaly (1/true) or not (o/false). This process is represented in the figure below.



Once the Models are validated, the Broker tab provides an interface to the EFPF message/data broker (<https://rabbitmq.smecluster.com/#/queues>) from where relevant data can be sourced to connect with the models. Using the intuitive interface of the broker the user is able to perform the following actions (as shown in the following figure):

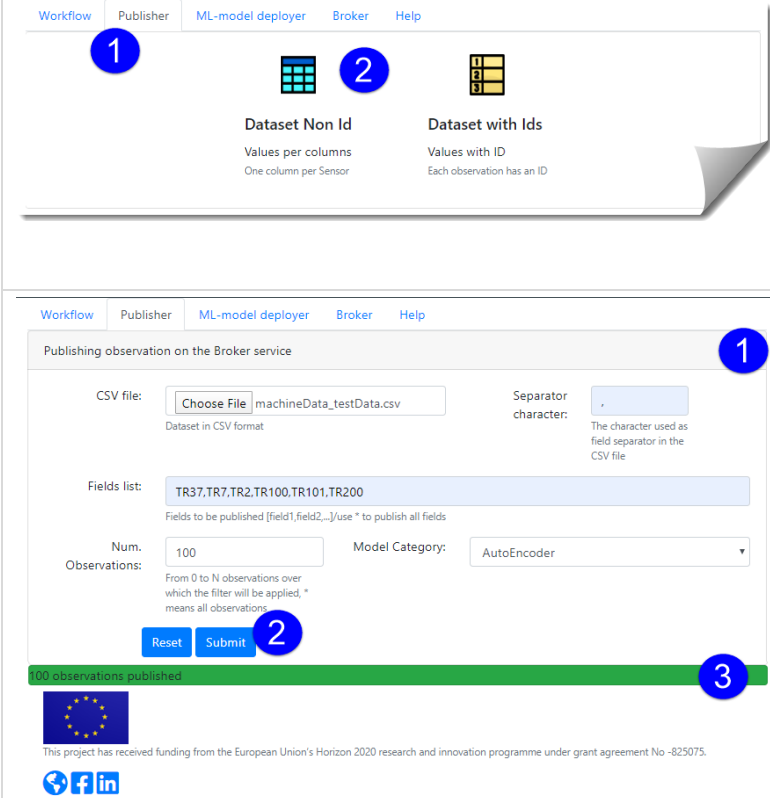
1. Choose the Broker board
2. Choose a virtual host
3. Explore the SOURCE queue
4. Explore the ANOMALIES queue





Once the Broker is setup, the Publisher tab allows the users to read a dataset (CSV file) and create the messages to be sent to a specific queue in the EFPF Broker. In this respect, the publisher can be used to simulate a sensor/machine.

To publish a Dataset without ID (where the raw data has not associated identifier e.g. time stamp), the user can follow the following steps:



The screenshot shows the 'Publisher' tab in the EFPF interface. It highlights the 'Dataset Non-Id' option, which is described as 'Values per columns' and 'One column per Sensor'. Below this, the 'Publishing observation on the Broker service' form is shown. The form includes a 'CSV file' field with a 'Choose File' button and the filename 'machineData\_testData.csv'. A 'Separator character' field is set to a comma. The 'Fields list' field contains 'TR37,TR7,TR2,TR100,TR101,TR200'. The 'Num. Observations' field is set to '100'. The 'Model Category' dropdown is set to 'AutoEncoder'. A 'Submit' button is highlighted. At the bottom, a green progress bar indicates '100 observations published'.

### Dataset Non-ID

1. Choose the Publisher tab
2. Choose the Dataset Non-Id icon

### Publisher form

1. Fill up the form
2. Click on submit button to publish the dataset by messages
3. The number of messages published will be shown in this bar

Params	Description
(1) CSV file	CSV_file (Source-Dataset)
(2) Separator character	The character used as field separator in the CSV file
(3) Fields list	Columns headers
(4) Num. Observations	First rows from the dataset to be published
(5) Model Category	Name of the Model category referred by the SOURCE queue

The current implementation of the ADS component allows users to use all advance analytic features. The upcoming work will focus on the tuning the underlying analytic functions and also GUI aspects to better match the needs of the EFPF user partners.

### **7.3.2 Visual Analytics Tool for Predictive Maintenance and Optimization of Supply Chain Planning Activities**

A web based data and visual analytics tool has been developed in COMPOSITION, one of the four base platforms of EFPF project. This tool has been updated in the EFPF project and is made available through EFPF portal. The tool is integrated aiming to address two major challenges for EFPF. The first one was to keep providing all the analytics solutions to pilots that were already available and the second one was to provide analytic services to new partners.

The tool aims to address real world needs and to provide analytics methodologies for both production and supply chain domains. In particular, the tool provide the following main functionalities:

- An analytics and monitoring dashboard for machine's vibration profile behaviour. All the abnormal vibrations during the production process are detected and visualized by the tool
- 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 that 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.

Main Interactions within the EFPF platform, the Visual Analytics tool interacts with the following external components:

- EFPF Portal, the portal provides the secure entry point to the tool's interface
- EFPF Security Framework in order to use role management capabilities for providing different views to different users

- Factory connectors provided by NXW in order to access sensors data from KLEEMANN and ELDIA 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 solution to the end-users that are related to planning activities

### 7.3.2.1 Configuration

The visual analytic tool has been implemented as a complete web-based tool. It is developed by using AngularJS framework, which takes care of content management features of the application. MongoDB is used for the tool's storing requirements. The graphical representations are enabled by the use of the Chart.js and D3.js visualization libraries. These libraries are ideal for producing dynamic and interactive data visualizations in web browsers. To this direction, the libraries allow control over the final visual results by using the widely implemented SVG, HTML5 and CSS standard. In order to be able to connect databases and IoT devices to the analytics platform both MQTT and HTTP communication protocols are supported. Furthermore, authentication and the authorization services are supported by the platform for the secure data exchange. All the analytic methodologies has been implemented using Python language and corresponding libraries.

There are no specific configuration that should be done by users. As soon as, an authorized user is logged in based on his role coming from the corresponding KeyCloak account, he is able to view a custom dashboard that has been configured for him. Then the user should just select a data source in order to deploy data and to select a corresponding analytic method.

### 7.3.2.2 Operation

The tool is delivered to the users as type of dashboard that provides three main options: (1) visit the real-time monitoring *dashboard*, (2) visit the *analytics* tab and (3) visit the *administration* tab in order to add new data to database.



The monitoring dashboard provides real-time visualizations for machine' condition or for sensors' data. An example for fill level sensors monitoring view is available in next figure:

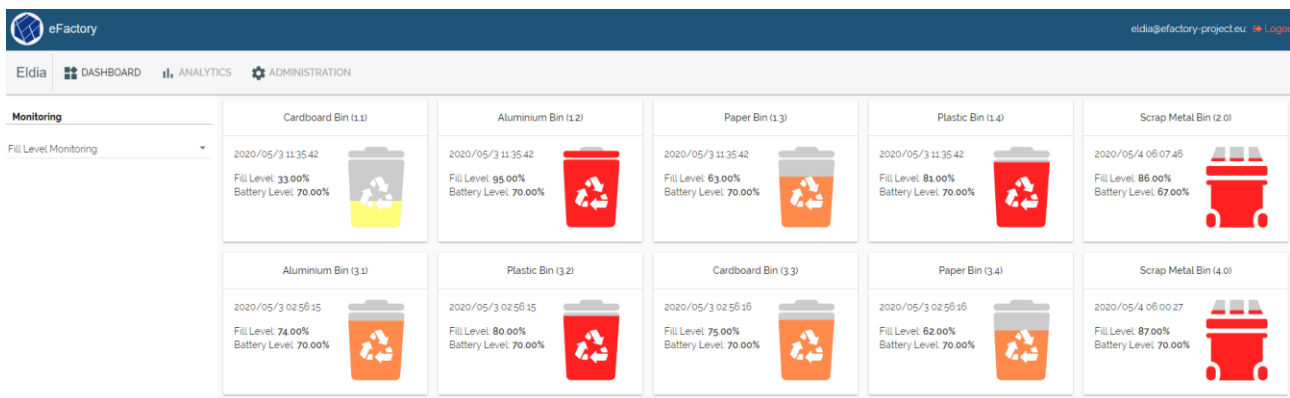
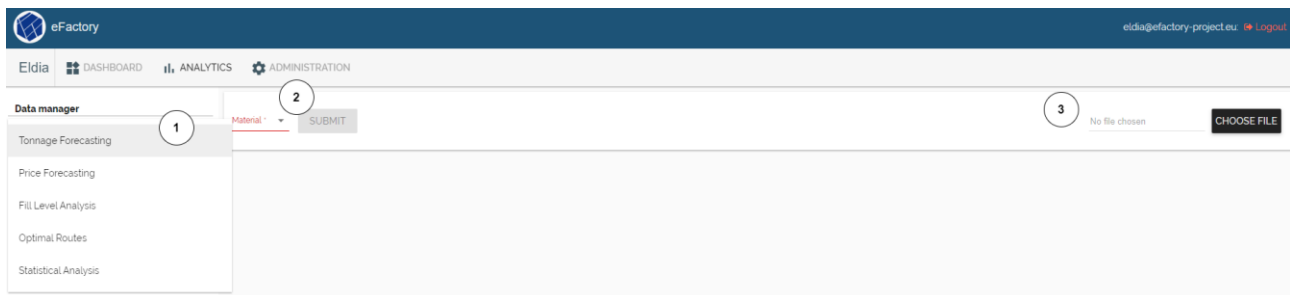


Figure 50: Main Dashboard of the Predictive Maintenance Tool

The analytics tab that is the main view of the Visual Analytics tool provide to the user the following options:



The first option enable the user to choose the type of analysis that he/she wants to perform, the second option is to select a raw material that want to load relevant data from a database in order to analyse them or skip this option and choose the third option that enables to load data from csv file.

As soon as the user defines, the type of analysis he/she wants to perform and deploy the corresponding data then the following options/interfaces are available:

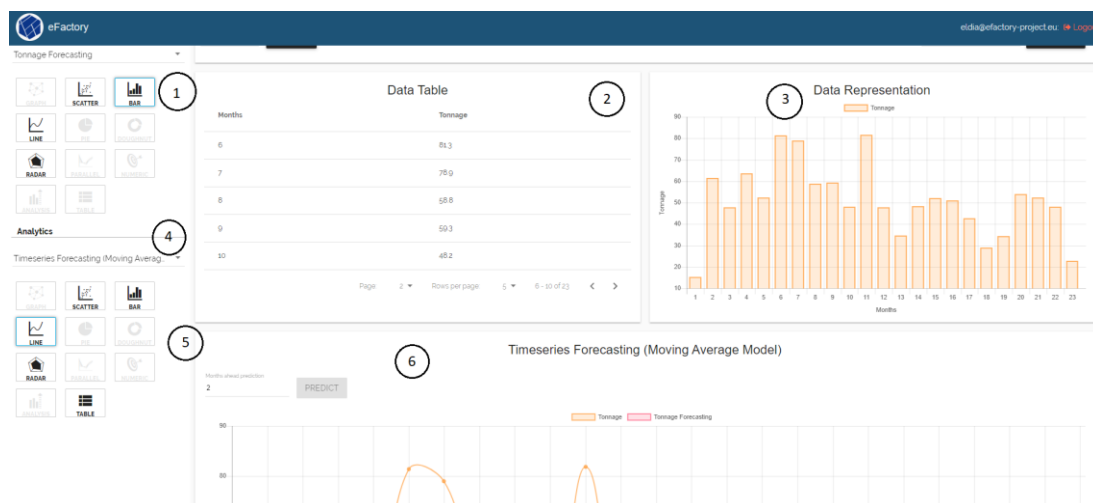


Figure 51: Analytic output of the Predictive Maintenance Tool

In 1<sup>st</sup> step the user is able to select the type of graph that he prefers to show the data that just load. In 2<sup>nd</sup> step the user is able to see the loaded data in a table and in 3<sup>rd</sup> step is able to get the data representation that it was selected on step number 1. In 4<sup>th</sup> step the user is able to select the data analytics methodology and on 5<sup>th</sup> step the diagram type that he wants to get the analytics output. The output is available on 6<sup>th</sup> step.

The following analytics methodologies are available to the user:

### Machine Vibration Diagnosis Profile

- Dynamic solution based on real-time data coming from deployed vibration sensors. The method creates a profile of machines normal operation
- Real-time detection of 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
- Maintenance manager visually informed via the Visual Analytics when the machine's activity surpasses the abnormal vibration threshold

The method has been deployed on KLEEMANN polishing machine during COMPOSITION project. For EFPF purposes, the data broker has been replaced with one dedicated to this project and the method alongside with the visualization dashboard are now available through EFPF portal only to KLEEMANN users by exploiting project's security framework capabilities.

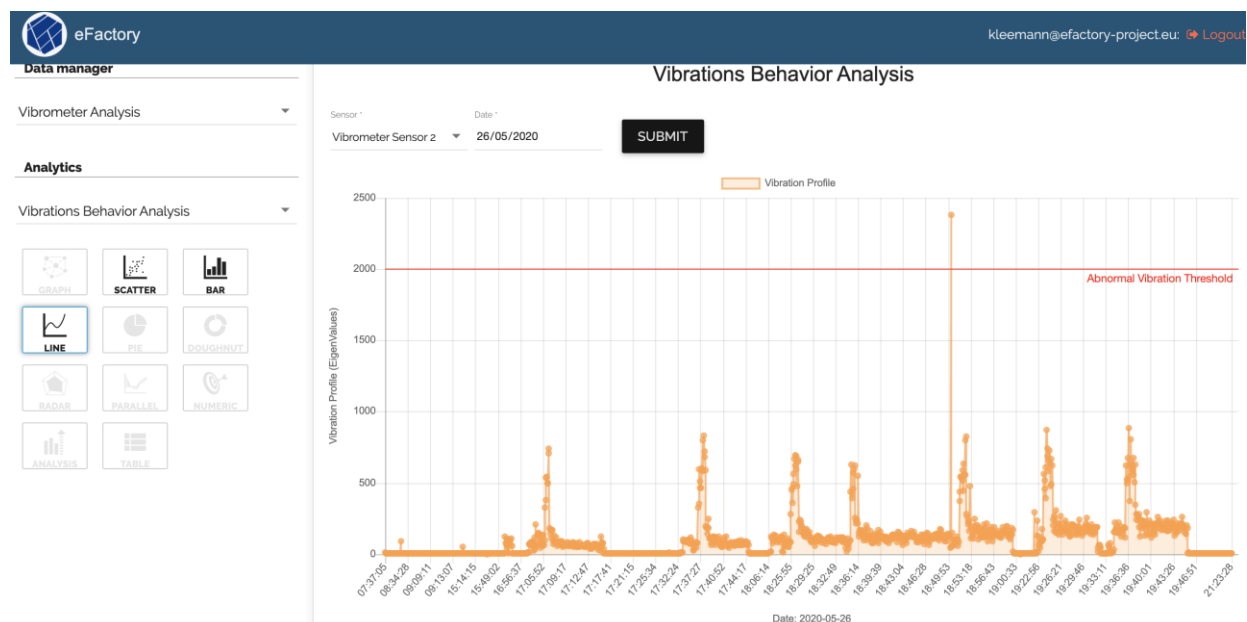


Figure 52: Visualisations on the Predictive Analytic Tool

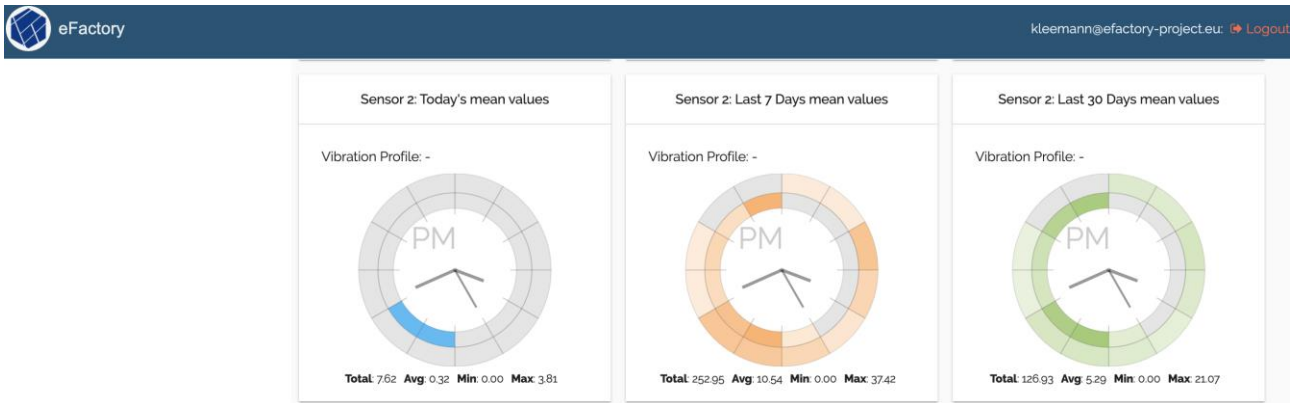


Figure 53: Visualisations showing the Mean Values

In COMPOSITION project, the threshold that indicates abnormal vibrations was computed by some custom tests on the available data as the method was originally designed for one pilot. For EFPF, the solution will be available as a kind of service to other platform users in order to use it in the case they have similar type of data. This is enabled by a modified threshold selection process. The process that is implemented during this project is described in next paragraph.

The user initially provides vibration data of the machine during normal operating conditions. That data is split into continuous time series using the provided time labels. The vibration data may be in the form of x, y, z coordinates or the distance from the vibrator's initial position. For each second of vibration data, the Euclidean Distance (ED) between each of the points and the 3-dimensional coordinate system origin is calculated and the pairwise distance matrix is created. Using principal component analysis, we extract the sum of the 3 biggest eigenvalues. For each time series of vibration data the maximum values for eigenvalues sum, eigenvalues sum variance and their product are calculated. These values, increased by a 10% tolerance percentage, are the threshold against which all further vibration data is judged. Afterwards the user provides vibration data during unknown operating conditions. After undergoing the same preprocessing as above, the maximum values for eigenvalues sum, eigenvalues variance and their product are calculated and compared to the safety thresholds previously calculated. If those thresholds are at any point crossed, an alert for abnormal vibrations is logged.

Besides the previous described methodology for vibration data analysis, a second one based on Standardised Mahalanobis Distance is in progress in this project. The method is testing in lab environment and as the results seems positive it is going to be deployed and tested on KLEEMANN's polishing machine. In particular, MD is measuring the distance between a point P of k given samples with n dimensionality, where  $X = (P1, P2, P3, \dots, Pk)^n$  is the dataset, to the distribution T of the samples. MD has the ability to calculate distances in multi-dimensional spaces and bearing into consideration the correlation between variables or features might exist. To detect if a point of a sample belongs to normal or anomalous condition the following steps were utilized. First, all data were raw vibration signals and there was no signal processing or feature extraction. Second, all data were standardized by:

$$P_{stand,i} = \frac{Pi - \mu(P)}{\sigma(P)}$$

where  $\mu$  is the mean of the data and  $\sigma$  is the standard deviation of the data. Afterwards, we calculated the covariance matrix and the inverse covariance matrix by:

$$S = E[(X - \mu_X)(X - \mu_X)^n]$$

where  $E$  is the mean value of the covariance matrix,  $X$  is the dataset,  $n$  is the dimensionality of the dataset and  $\mu_X$  is mean value of the dataset. The SMD of all the points belonging to dataset was calculated by<sup>19</sup>:

$$SMD = \sqrt{(Pi - \mu)^n S^{-1} (Pi - \mu)}$$

where  $\mu$  is the mean of dataset  $X$  and  $S^{-1}$  is the inverse of the covariance matrix of the data. For an instance belonging to a dataset, in order to label it as outlier, it will be outside of the norm of the population. When a machine works without any faults then identically all the datapoints will be in the same norm and will be normally distributed. In order to raise a threshold for flagging normal vs anomalous, we considered that the probability distribution of SMD of every datapoint is the normal distribution. So we set a threshold as three standard deviations of the normal distribution of our data. The algorithm is first trained on data belonging only in normal condition and the threshold is generated. The testing phase of the algorithm includes data belonging both in normal and anomalous condition. Furthermore, the SMD methodology is able to process in real time raw vibration data in order to detect whether they belong to normal condition or abnormal condition and categorize them.

The SMD algorithm is compared with two known machine learning algorithms. The Local Outlier Factor (LOF) algorithm and the Isolation Forest (IF) algorithm. The performance of these approaches were tested utilizing two datasets, one collected with a vibration sensor on a motor inside our premises, as local, and a public one. The local dataset contains vibration data corresponding to normal and abnormal condition. The abnormal condition refers to malfunctions occurred by voltage drop on rotating machines. The public dataset is NASA's bearing dataset No 3, where vibration data were collected over the lifetime of the bearing until a failure occurred with crack in the outer race of the bearing number 3 after 100 million cycles<sup>20</sup>.

For the local dataset, the results indicated that Standardized Mahalanobis Distance detects anomalies more efficient than Local Outlier Factor and Isolation Forest models when the voltage value of the abnormal condition is not close to the nominal voltage value. For the public dataset, Standardized Mahalanobis Distance achieved better evaluation results than Local Outlier Factor and Isolation Forest and has the ability to identify in a more efficient way outliers before a fault on a bearing occurs.

### Fill Level Trend Analysis

The fill level sensors data that are available in Circular Economy pilot analyzed in order to provide estimations that can optimized planning procedures. A Trend Analysis methodology is available. The key aspects of the deploying solution that are available to the user through EFPF portal are the following:

- Real-time analysis of fill level sensors data
- Trend Analysis applied in order to create a profile for fill level trend

<sup>19</sup> P. Mahalanobis, On the generalised distance in statistics (vol. 2, pp. 49–55), Proceedings National Institute of Science, India. Retrieved from <http://ir.isical.ac.in/dspace/handle/1/1268>.

<sup>20</sup> C. C. L. Abhinav Saxena, Kai Goebel, F.-K. Chang, Cfrp composites dataset. URL <http://ti.arc.nasa.gov/project/prognostic-data-repository>



- Slope Statistic Profile method is applied on the time series of recordings (percentages) of a fill level sensor
- Waste management company is able to define which bin has the most aggressive trend in order to arrange a pick-up

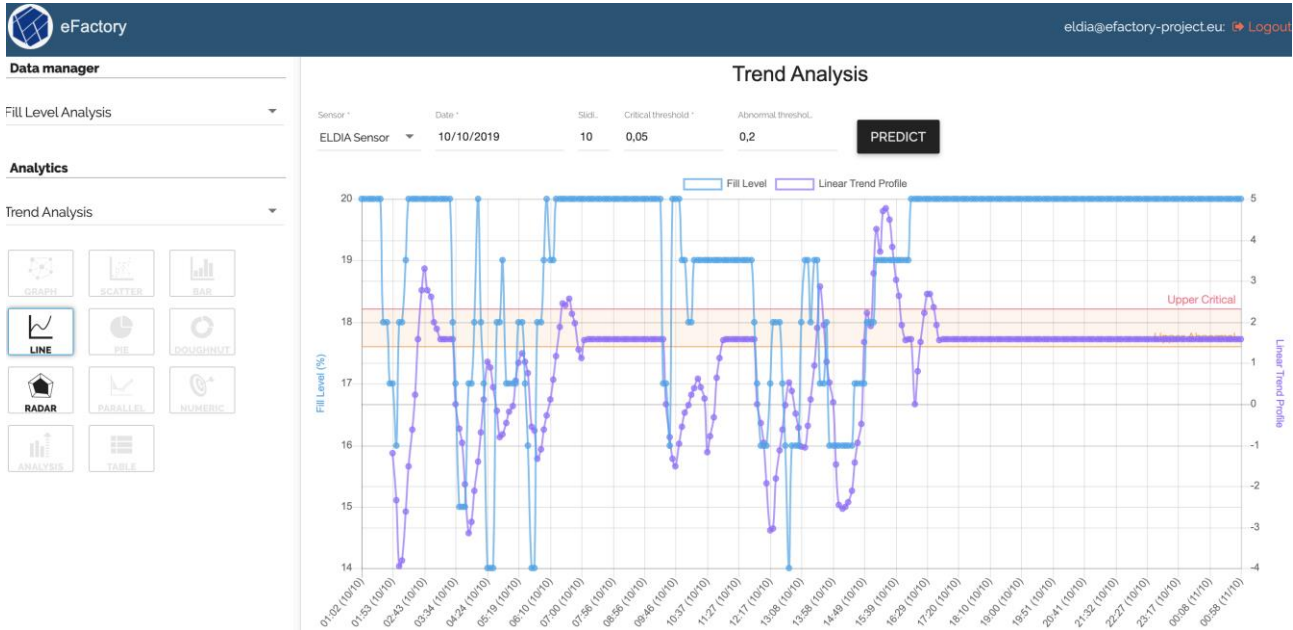


Figure 54: Visualisations showing the Trend Analysis

### Methodologies for Tonnage Forecasting:

The methodologies are related to the estimation of the future tonnage of raw material that a company is going to buy based on historical data. Two of the three presented methodologies in this section are derived from COMPOSITION project. They are originally designed for ELDIA in order to enhance planning activities by giving estimations of the future wastes/materials they have to pick up in upcoming months. For EFPF purposes, the methodologies are available to ELDIA through EFPF portal (as did for KLEEMAN case as well). Besides this, an analytics dashboard with similar solutions has been setup during EFPF for MILOIL (not COMPOSITION partner) as it was defined the type of available data and the need fits with ELDIA ones. In addition to two available methodologies (Moving Average and Markov Chain) for tonnage forecasting a third one is developed in EFPF methodology, and is based on auto-regression models. In brief details, the three available algorithms for tonnage forecasting are:

#### 1. Moving average

The moving average algorithm aims to predict future values for the time series by finding an  $n$ -length series of values (where the value of  $n$  is chosen by the user from between 2 to 5) that, when repeated for the whole length of the time series, best fit the time series. To compute these  $n$  values the time series is split into  $n$ -length segments and the average value of each of the  $n$  points is calculated. The prediction is made by repeating this series of average values for the required number of values. In order to deal with time series that display a trend, the moving average value of the time series is computed and from this the

average trend is computed. This trend is then removed from the original time series, the moving average prediction is calculated, and then the trend applied to the prediction.

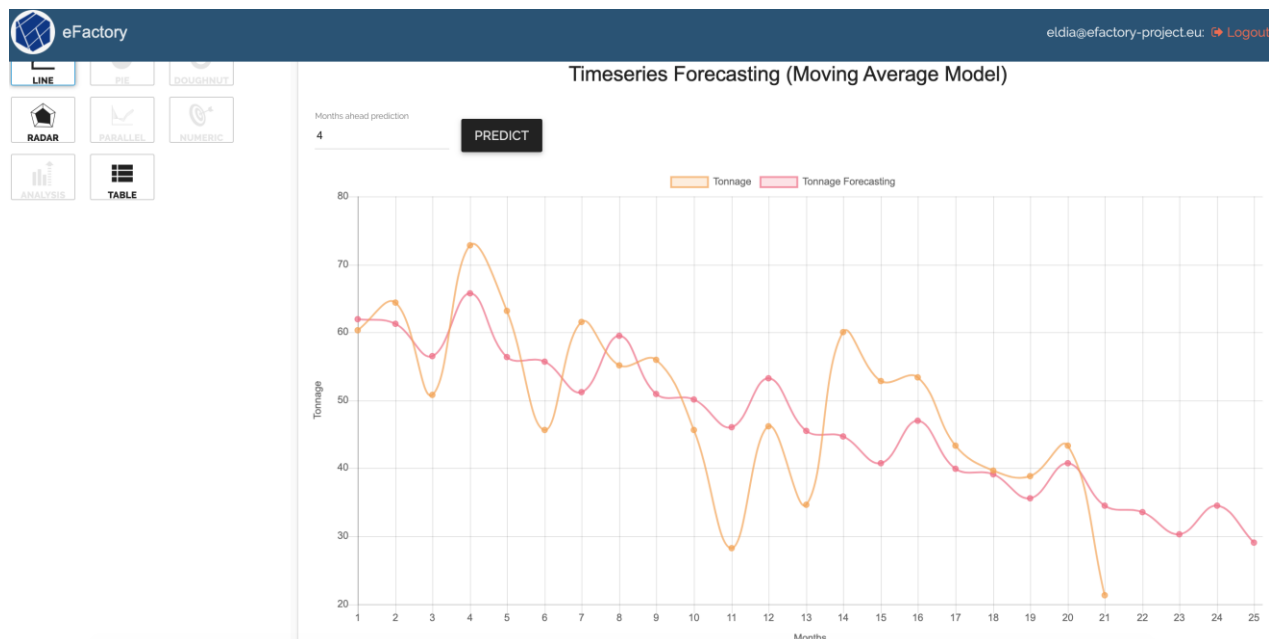


Figure 55: Visualisations Showing the Timeseries Forecasting

By applying the Moving Average methodology to new pilot (MILOIL) data the following were observed:

- Although moving average is a very simple method, it can offer surprisingly accurate predictions in time series with a strong seasonality component.

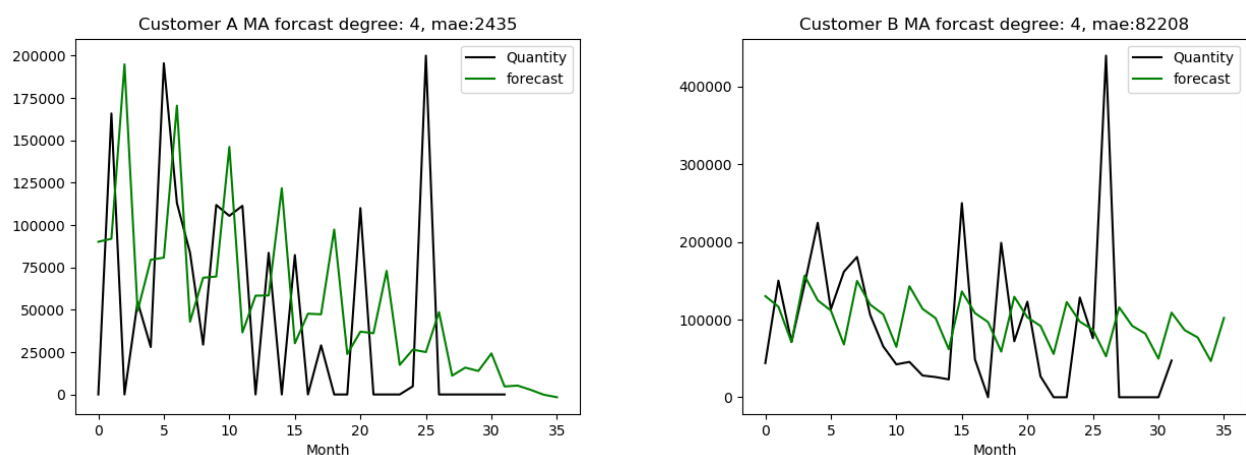


Figure 56: Moving average prediction comparison between time series with and without strong seasonality components.

- The selection of the moving average degree is also very important. Although one would expect higher degrees to offer better predictions, choosing a degree that better matches

the time series' seasonality yields better results. For this reason, the algorithm constructs moving average predictions for a number of different degrees and presents the user with the best prediction.

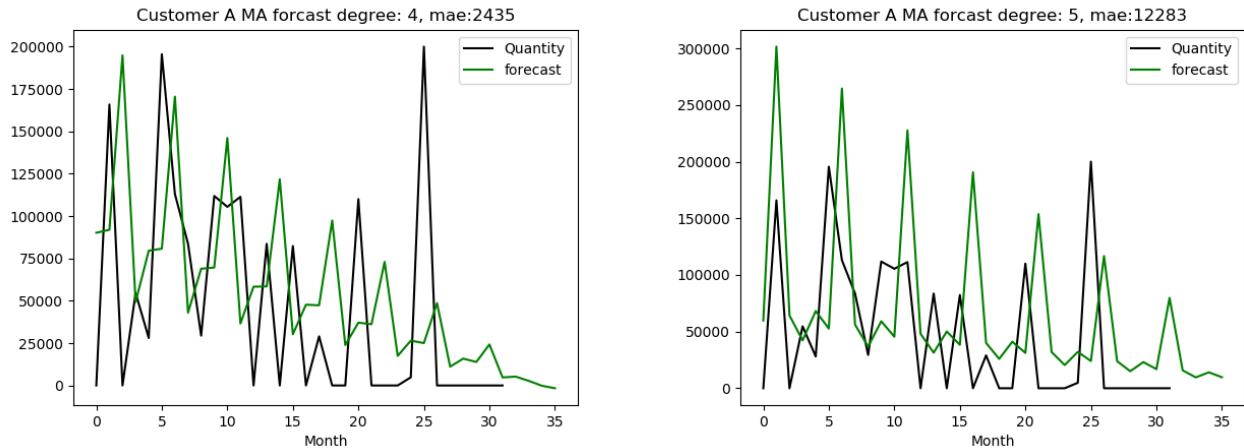


Figure 57: Moving average prediction comparison between using different degrees

## 2. Auto-regression models

A n-order auto-regression model provides a prediction for a time series by predicting future values as a linear sum of the n previous values. For example, a second-order auto-regression model predicts each future value  $y_t$  as:

$$y_t = b_0 + b_1 \cdot y_{t-1} + b_2 \cdot y_{t-2}$$

We use the function `ar_model.AR` from the python library `statsmodels.tsa` to create the auto-regressive model. The function uses the conditional maximum likelihood estimation method to automatically determine the best order for the model and compute the model's parameters. In order to deal with seasonalities in the time series, the user can apply a de-seasonalization method by choosing the length of a seasonality period (e.g. 12 for yearly seasonality for monthly data). In that case, the algorithm computes the average value for each point in the period, subtracts those values from the time series, computes the auto-regression model, uses it to predict the future values of the time series and finally adds the seasonal averages into the predictions.

In general, the auto-regression prediction method is one of the most commonly used methods for time series prediction, as it can offer very good results when each value in the series is heavily dependent on a few of the previous values. However, this method can struggle to detect long-term dependencies, which is why the user is allowed to deseasonalize the series first by providing a seasonality period length. However, this option must be used carefully. As we tried to deseasonalize a time series that does display seasonal dependencies (or even displays seasonal dependencies of a different period) in MILOIL data, we defined that the approach it may result in a worse prediction, as it is depicted in next figures below:

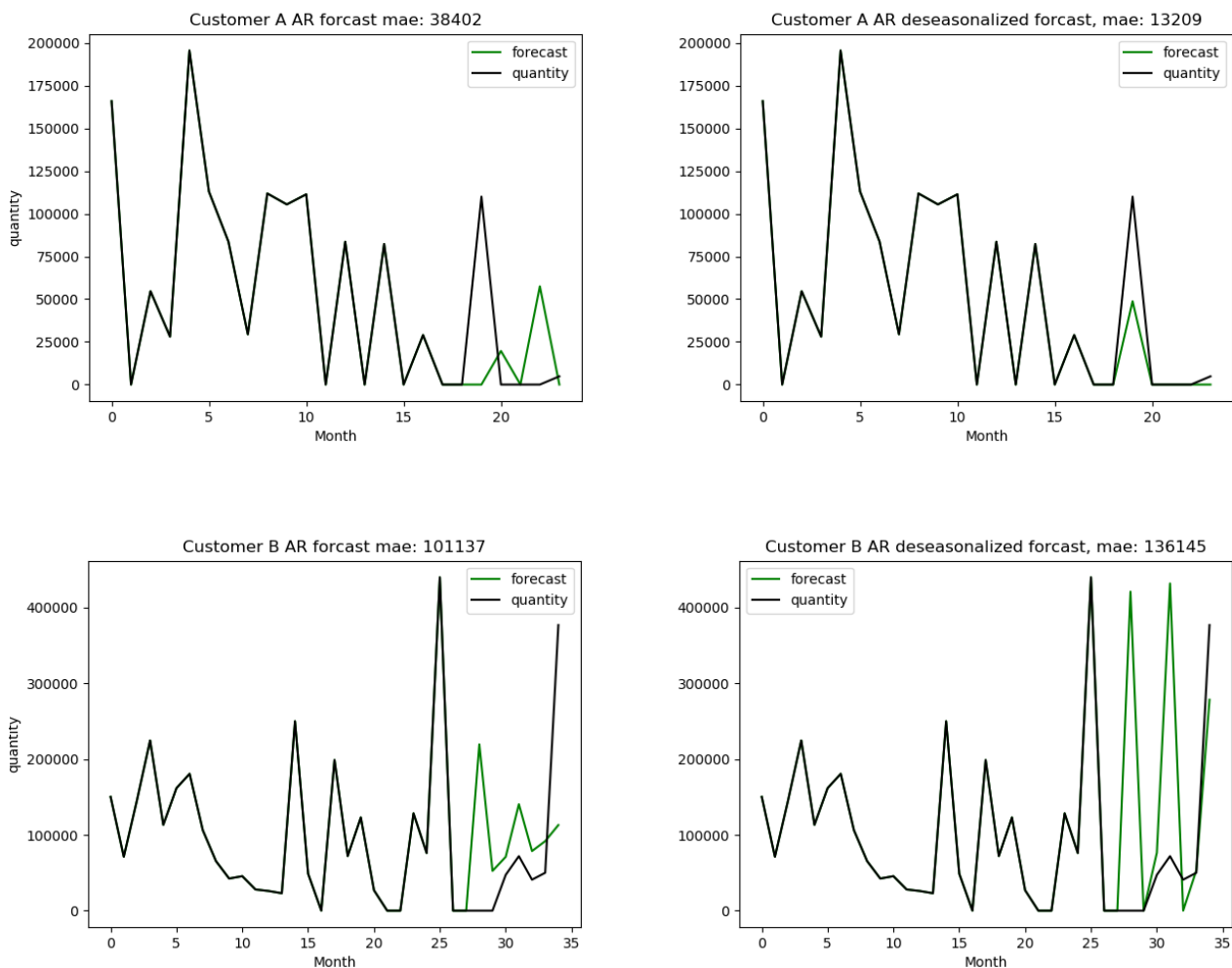


Figure 58: Example of applying deseasonalization on time series that do (Customer A) and do not (Customer B) display seasonal dependencies.

### 3. Markov chain

The Markov chain prediction algorithm uses the time series values to compute a transition matrix between three different time series states: value increases, value decreases, and value remains constant. Using the transition matrix the algorithm can not only compute the probability of the time series moving from one state to another (e.g. the probability of the value increasing after decreasing in the previous time step), but also the probability of the time series moving from one state to another after any number of time steps. The method does not provide the user with specific prediction for future steps of the time series, but it can provide a prediction for general movement of the time series in an easy to understand format while also providing specific statistical probabilities. Moreover, the Markov chain method allows the user to produce effortlessly the probabilities for the time series' movement over a large number of time steps, again providing precise and easy to understand statistical probabilities.

### 7.3.3 Deep Learning Toolkit for Data Analytics

The Deep Learning Toolkit (**DLT**) is a tool which leverages the Machine Learning and Deep Learning techniques for time series shaped data, like Long Short Term Memory (**LSTM**) and Encoder-Decoder for providing solutions of predictive maintenance and/or anomaly detection in the Industry scenarios. The DLT provides as well, using the same techniques, time series forecasting that can be used for predicting the prices of goods.

The Deep Learning Toolkit is made from two different modules, the price prediction and the predictive maintenance one. Both have been developed during the COMPOSITION project and have then been adapted to the EFPF ecosystem. The modules can be independently deployed and are agnostic one from the other while being part of the same suite and sharing part of the codebase.

Both modules do not include a graphical user interface or a web portal by default. This is because of the variety of the processes they can be used in, so that an ad-hoc interface can be developed according to the needs of each user.

**Price prediction:** This module tries to predict the next values of a dataset containing information about purchases of a given good. To obtain the best accuracy possible the dataset shall contain entries close in time between one another and each entry should provide information other than the timestamp about the price of the good at that time and the quantity purchased.

A single instance of this module can be used to predict the prices outlook of different goods since there is the possibility of choosing different models when invoking the APIs. This feature can also be used to have different models for the same good in case the price behaviour changes in different scenarios.

Different accuracy metrics are made available to the user in the deployment phase to provide the possibility to pick the one which reflects better the data provided.

Unfortunately, it is almost impossible to guarantee the long-term module behaviour, because it is strictly dependent on the input data that will be provided to the network. But it is safe to assume that, if the boundary conditions do not change from the time of obtaining the data to the time of using new data, the network will learn the price's pattern with the help of online learning. That means that if the boundary conditions when the network have been trained are the same of when they will be used to predict or to apply continuous learning the network will converge.

The module can be trained using datasets already collected, then, when deployed, it keeps learning using the single entries provided when updates to prices are issued. This process is called online learning. If the initial dataset is small, initial results can be extremely inaccurate because the network is predicting the results using random weights. Its accuracy improves as additional data becomes available.

**Predictive maintenance:** This module differs from the previous since it uses datasets more rich in features. This is done in order to provide more accurate predictions about the possible failures in a process which can be monitored through the data supplied to the Deep Learning Toolkit.

The data this tool is designed to use is readings from different sensors applied to a machine. These readings should represent features that are proved to have a strong correlation with

the correct operation of the machine they are referring to, and should be equally spaced in time to allow the prediction process to work.

Under ideal conditions the Deep Learning Toolkit would predict, based on the previous data, when a fail will occur. In order to achieve this result after studying the state-of-the-art and most innovative models of time series classification/forecasting and comparing with the well-known ones, Recurrent Neural Networks (RNN) have been chosen. This happened because each read of the sensors and each event can be seen as discrete variables that changes through time and RNNs are the most fit kind of deep neural networks for this task.

When training the network, the data is fed to it in consecutive time series of length 32 and the network will predict the following (starting from the last point of the time series) 5 future events. In standard conditions each entry of the dataset is spaced 5 minutes from the other so this means using the last 160 minutes of sensors readings to predict failures up to 25 minutes in the future.

After the deployment, the tool keeps learning each time it is provided with new data and it will provide a warning in case an anomaly is detected in the sensors' data streams.

### 7.3.3.1 Configuration

Both the tools composing the Deep Learning Toolkit have been put into separate Docker containers. Each container can be deployed independently and can live both on premise or remotely. Configuration files are placed inside the volumes assigned to each container to perform small tuning of the tools; major changes may need a software update process. Since the tools are independent each other there are some differences in how the neural network – the main component of each tool – and the respective API are designed.

### Price prediction

The Recurrent Neural Network used in this tool is made of two main building blocks:

- Encoder: the encoder layer is mainly composed of a LSTM node.
- Decoder: the decoder consists of a LSTM layer, followed by a fully connected one, which produces the output (through a linear activation).

Since the goal is to predict the future prices, the metrics used in order to evaluate the results are the MSE and the MAPE. In order to prevent the RNN to over-fit too much (despite collecting more data is the only way to make the model able to generalize properly), the following techniques have been applied:

- Adding dropout;
- Adding L2 regularization on the weights and activation parameters, L1 demonstrated to work poorly in this use case;
- Early stopping, in order to prevent over-fitting on the training dataset when this brings no value on the validation set;
- Adding more layers and making the network deeper had no effects on improving the results. It means that the capacity of the abovementioned model is enough to describe the problem.



The hyper-parameters of the model have been chosen with a combination of manual tests with some provided values and runs leveraging the hyperopt library (Hyperopt, n.d.) in order to determine the best ones for the specific use case.

The considered hyper-parameters are the learning rate, the dropout percentages, the batch size, the kernel regularization factors, the activation regularization factors. Any user edit to the considered hyper-parameters performed through the configuration files should be validated.

### Predictive maintenance

The neural network used in the tool consists – as in the tool for price prediction – in an encoder and a decoder stacked on each other. The encoder consists in the first LSTM with 256 units while the decoder is the last LSTM with 64 units as well. The first layer takes in input 32 time steps for the N features (the sensors which are providing the input data) and returns only 64 vectors representing the “encoding” version of the input data. These encoding data is repeated 5 times (the number of time step in the future we want to predict) by the Repeat Vector layer creating a new time series of [5x256]. This newly created times series is given in input to the decoder, the second LSTM. The decoder returns the new 5 values plus their “sequence”: this can be seen as a time series (of [5x64]) too that passes through a Dense Layer in a “time distributed fashion”. It means, citing Keras documentation, that “This wrapper applies a layer to every temporal slice of an input”. The last layer uses the SoftMax activation: this because the target variable is considered as the one-hot-encoded variable indicating the correct functioning of the machine the tool is monitoring.

Two techniques are used for preventing the overfitting:

- First, the dropout, which is a special layer in a neural network that has the function of randomly “turn off” one random neuron from the previous layers and readjusting all the weights on each training step. The percentage of the “random” can be set and usually has value  $\geq 0.5$ .
- The second technique used for preventing the over-fitting is the so-called “Early Stopping”. This technique allows training of the network until the validation loss stops improving. Of course, there are some epochs of “waiting” before stopping the training in order to allow the network to keep learning even after some not-learning epochs.

Fine tuning on these parameter can be performed by editing the configuration files inside the docker volume.

#### 7.3.3.2 Operation

##### Price prediction

The users can interact with the DLT for price prediction through a set of HTTP based API shown in the table below.

Path	HTTP verb	Description
/good	POST	Creates a new good if it does not exist. It creates the correspondent network, untrained with the structure defined in the configuration files



/good	GET	Returns the list of all the goods created in the DLT
/goods/{id}	GET	Returns the details about the good {id}
/goods/{id}	DELETE	Deletes the good and the relative network and model
/goods/{id}/values	POST	Adds a new value to the good. Each time a new value is added, the network predicts the new value, then the network is trained with the input value
/goods/{id}/values	GET	Returns all the input values added through time.
/goods/{id}/predictions	GET	Returns all the predictions for a good in an array shaped form
/goods/{id}/predictions/last	GET	Returns the last prediction of the DLT (without adding a new value)

### Predictive maintenance

As of today the DLT for predictive maintenance exports four different functionalities that can be remotely invoked from the users through the Pyro framework:

- Predict: it generates output predictions for the input sample. It returns a Python list of three values: o prediction: 1 if the prediction is a fault, 0 otherwise; o accuracy: it is a decimal number between 0 (worst) and 1(best) that represent the accuracy of the prediction; o validity: the time of validity in minutes of the prediction. When a new prediction is generated, it overwrites all the oldest.
- Predict on batch: the returning values would be [[prediction0, accuracy0, validity0], [prediction1, accuracy1, validity1], [prediction2, accuracy2, validity2]]
- Learn: this method can be used for continuous learning. It runs a single gradient update (unless specified differently) on a single sample of data.
- Learn on batch: this method can be used for continuous learning. It runs a stochastic gradient update on a batch of data.

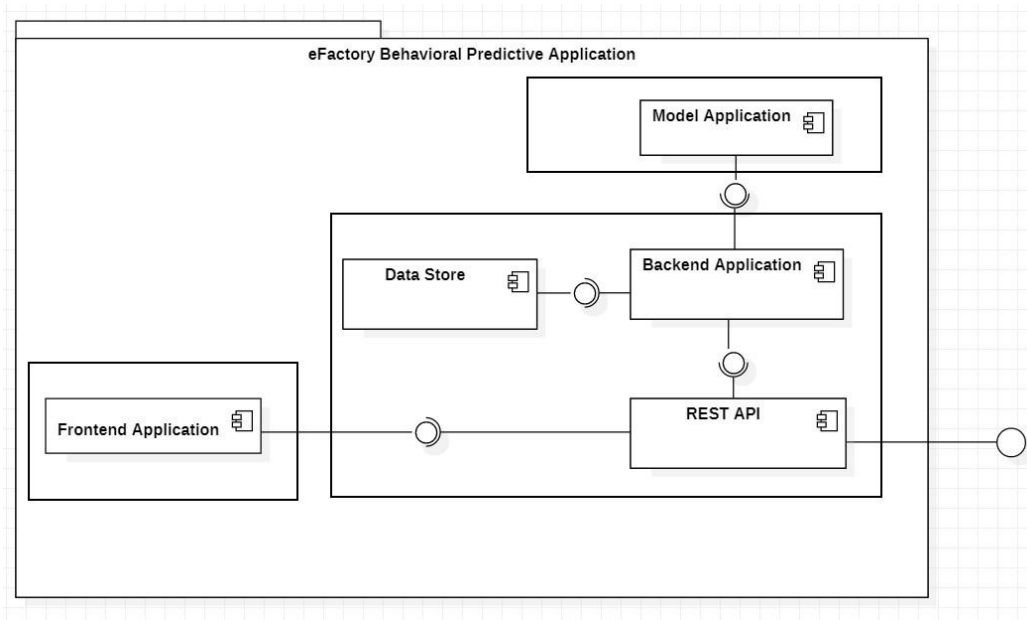
This set of API is currently deprecated and will be removed in future versions of the tool. A set of equivalent HTTP based API is currently under development and will be released prior to the removal of the Pyro API.

### 7.3.4 Customer Trend Analysis

Customer trend analysis will comprise different models that will allow companies to analyse their customer data and to predict customer behaviour. Based on the user requirements gathered in the EFPF project, the first customer trend analysis model solution 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.

The customer trend analysis solution is developed using the base functionality offered by the Integration of the eElanyo data analytic platform. Therefore the integration of Elanyo platform into the EFPF ecosystem is one of the key priorities. Once this is done, APIs will be created to ease the use of certain functionalities its use (e.g. for uploading data). Then, further more specialised analytic models will be added if suitable models can be identified based on the consultations with EFPF's user partners.

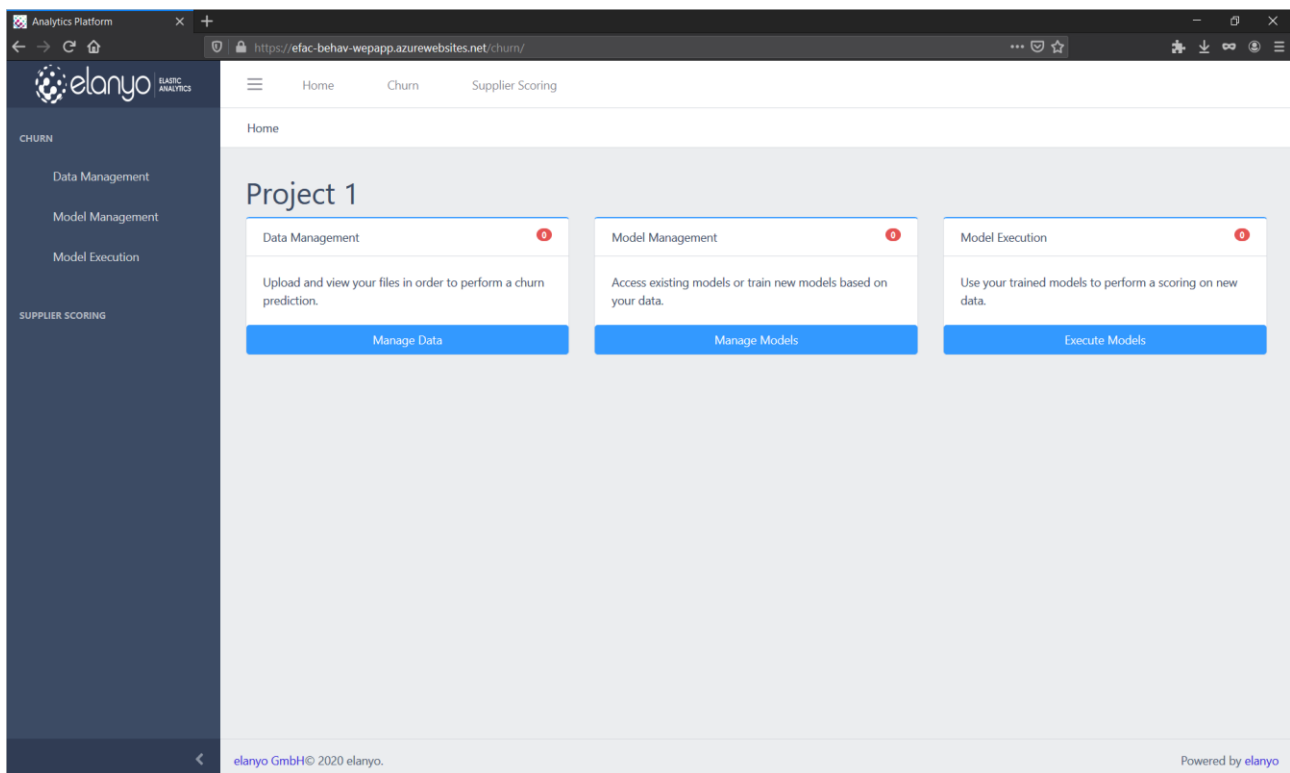
Diagram of Elanyo's platform is shown below:



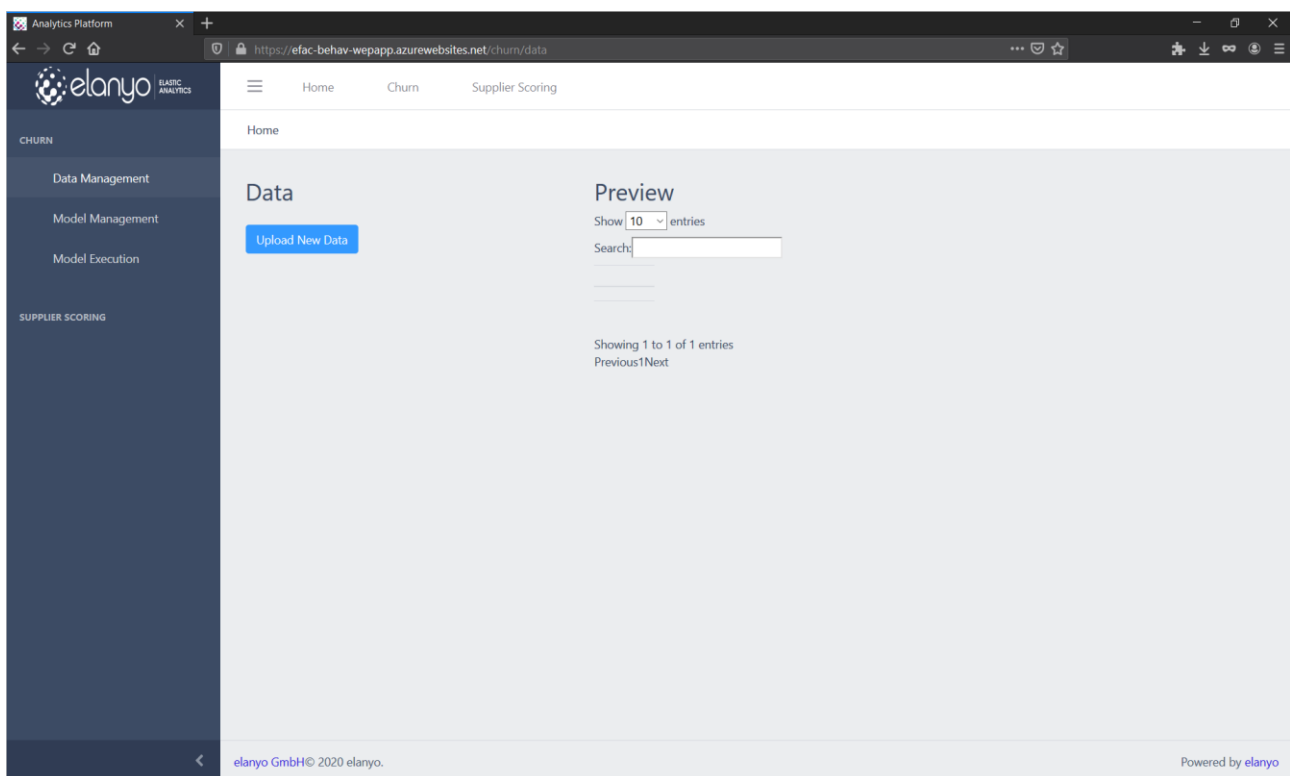
#### 7.3.4.1 Configuration

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.

Right now, this implementation only includes the churn model. Other models will be added later once the models have been discussed and specified with users.

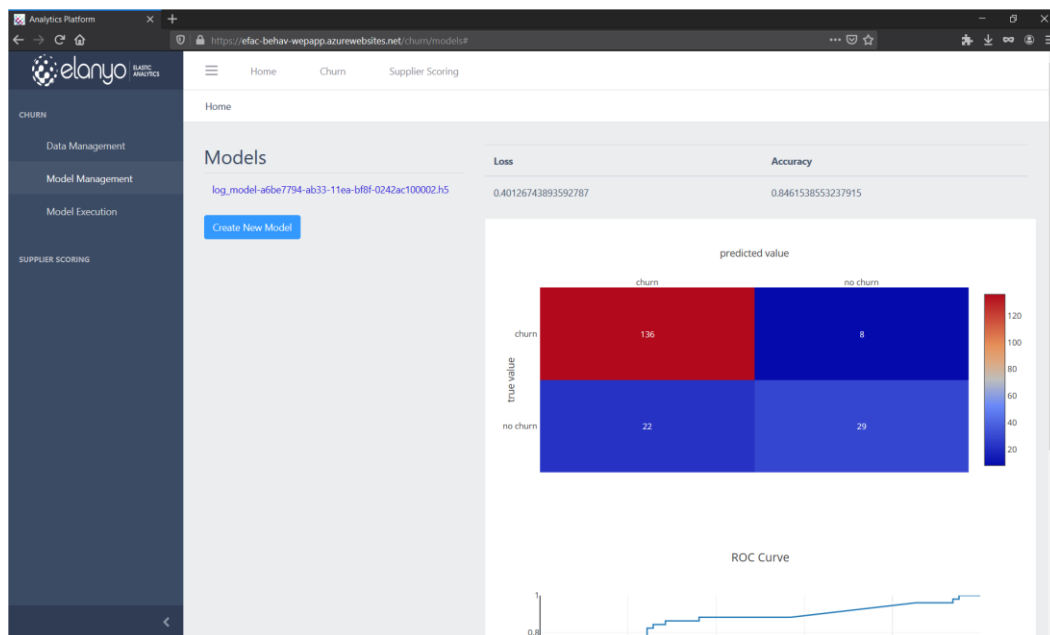


On the data management tab, users can upload data on their users. Right now, it only works for data of a given structure, but functionality to add generic data will be added.

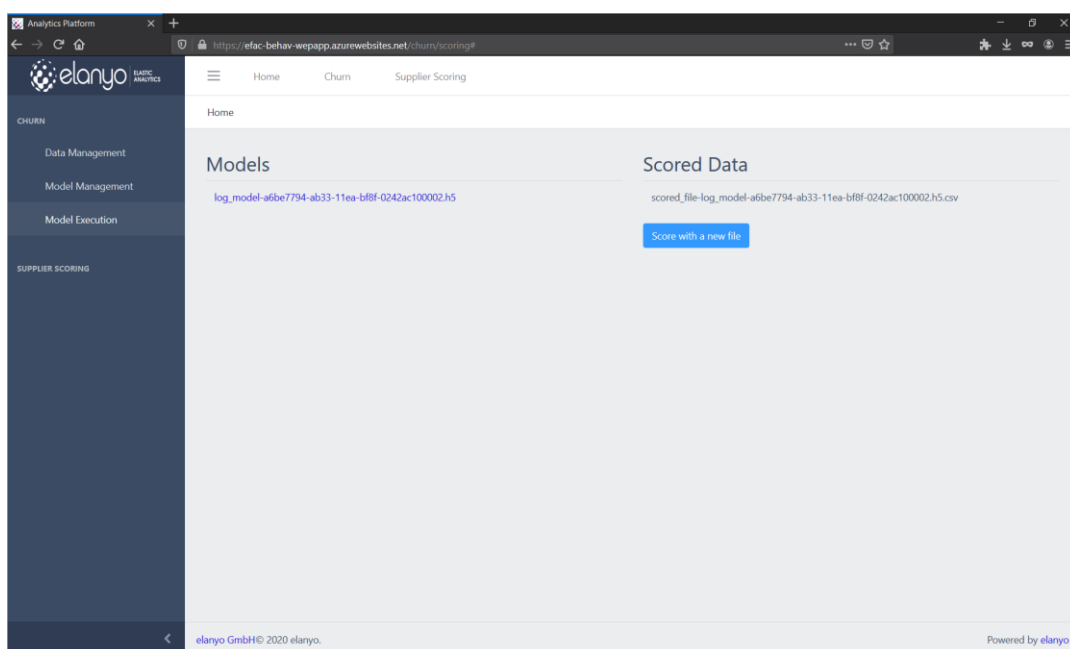


On the model management tab, users can train models (with the step they have previously uploaded). Right now, the only model that can be trained is the churn prediction model and this one is based on a logistic regression in Keras.

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.

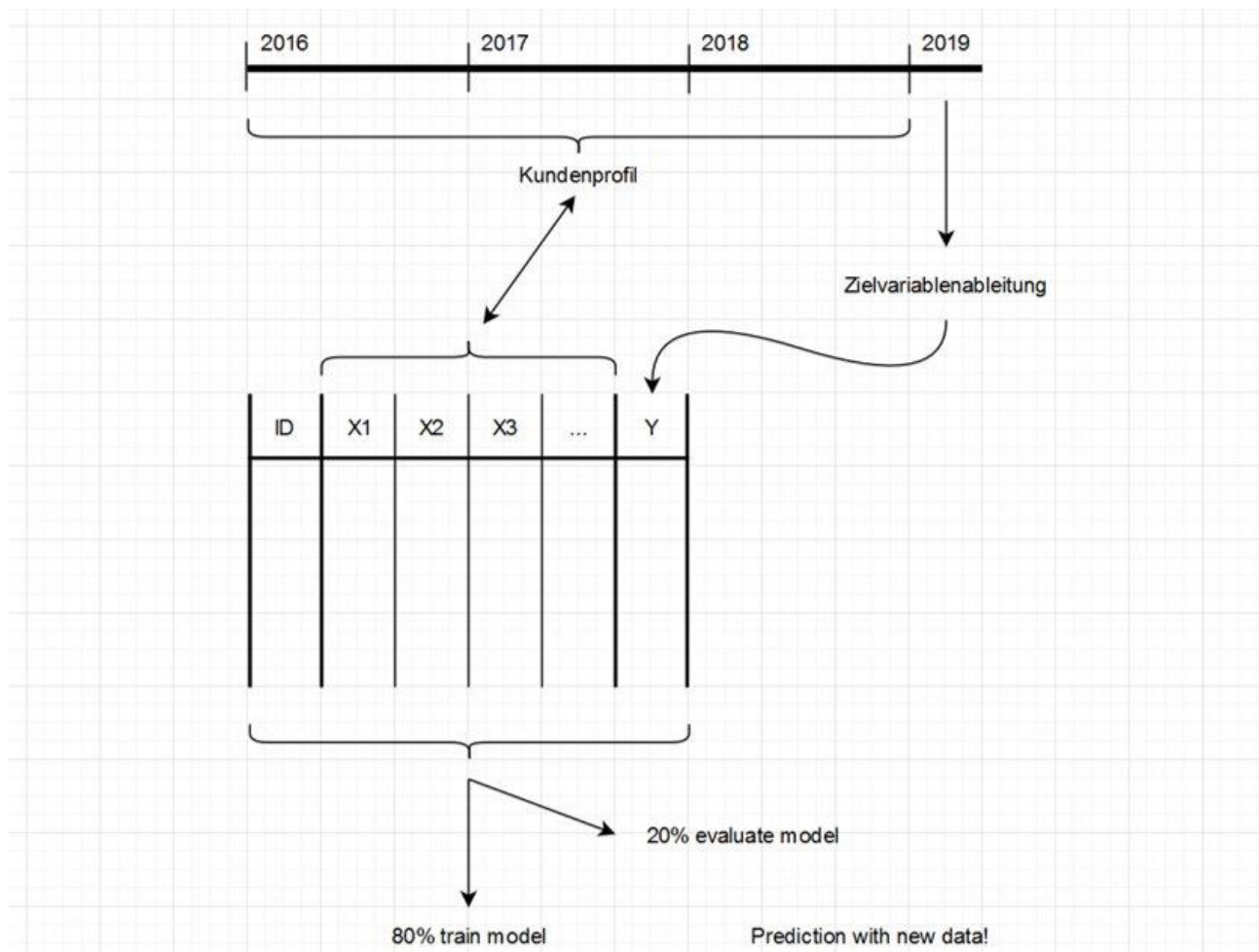


On the final tab (model execution), users can upload scoring data to run it through the trained models. Right now, users can only upload customer data and score it with the churn prediction model. The result is a list of customers that may churn and of customers that may not churn.



### 7.3.4.2 Operation

The user does not have to do anything but provide data and choose the suitable model. So far, only one model can be chosen. It automatically configures itself, as explained in the following. A sample churn prediction model is shown below:



The 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, as shown in Figure 59.

personId	churn
11453	1
15579	0
16588	0
16832	0
19343	0
19635	1
29686	0
32077	0
35917	1
38125	1
45737	0
50445	1
53880	1

Figure 59: Churn Prediction Model evaluation output

### 7.3.5 Siemens Data Analytics Solutions

In recent years, we are witnessing the digital transformation of production lines as part of manufacturers' transition to the fourth industrial revolution (Industry4.0) and the emergence

of a new paradigm where by adding value is extracted from data all the way from to the shop floor data from machines and exposing them to analytics to business services. Based on Cyber Physical Systems (CPS) and digital technologies like cloud computing, the Industrial Internet of Things (IoT) and Artificial Intelligence (AI), Industry4.0 is enabling flexible production lines and supporting innovative functionalities like mass customization, predictive maintenance, zero defect manufacturing and digital twins. AI is currently the most disruptive digital enabler of the Industry4.0 era and enables novel use cases like predictive quality management, effective human robot collaboration, agile production and generative software design. AI's disruptive potential is currently propelled by advances in parallel hardware and scalable software systems (e.g. which have enabled the development of advanced machine learning frameworks and novel algorithms that are suitable for large scale problems in realistic settings). In the manufacturing sector, advanced AI technologies enable solutions to large scale optimization and control problems.

Business know-how and strategies becomes data centric, based on increased amount of data from multiple sources and a multitude of vendor-specific technologies, making resources sharing between legacy applications a challenge. Aligning the digital platforms at the level of EFPF brings new challenges such as data interoperability, implementing data models and semantics to ensure compatibility, automated accurate information exchange, producing meaningful results etc. The highly dynamic environment acting cross-company and cross-domain, requires high-level interoperability and scalability of the federated digital platform system's services. In 2019, it was noted that "Industry 4.0 is now being deployed on shop-floors across Europe as part of smart factory solutions comprising of Internet of Things, Cyber-physical systems and cloud-based services" [WABH18].

Interoperability enables artificial intelligence implementations of the Federated EFPF Platform to correctly interpret data, learn from such data and adapt to reach the goals of the numerous offerings. The essential condition is to extract, transform, load (ETL) data of specific quality from multiple sources. Machine learning workflow or data science workflow may be applied to determine actionable insights for EFPF domain centric offerings. Despite the these advances in data sciences, the use of data analytic techniques in the manufacturing domains , state of the art AI deployments in manufacturing are far from leveraging the most advanced capabilities of machine learning and robotic systems. Rather, they are still less sophisticated and mostly focused on the consolidation of datasets from heterogeneous sources towards enabling advanced analytics (such as deep learning) for use cases like predictive maintenance and industrial simulations. This is largely due to that real-life manufacturing environments can be complex, dynamic and unpredictable, which raises safety, reliability and trustworthiness concerns for data analytics AI deployments.

Within the EFPF project, the data analytic solutions offered by Siemens are based on proven methodologies and state of the art technologies. The Siemens data analytic solutions are primarily built upon the analytic-as-a-service concept supported by the MindSphere platform. These solutions (currently under development) are designed in consultation with user partners to address the needs for optimisation of manufacturing activities.

#### **7.3.5.1 Implementation Details**

Siemens rely for its data analytics implementation strategies for data analytics are based on few different types of experiences and technology offerings. First one is related to MindSphere (<https://siemens.mindsphere.io/en>), an industry grade environment, where Predictive Learning functionalities are considered flexible and usable to bring in the



supported Data Spine the “bring your own analytics” concept at the level of EFPF Data Spine. This set of MindSphere components allow full lifecycle of analytics from data collection & cleansing, ML model development (using for example Amazon SageMaker), Model Management and deployment to cloud of edge enabled infrastructures. In addition, it can be considered the visualisation of analysis both on demand and also as monitoring of various type of alert.

It should be mentioned that MindSphere use its own asset management model in order to normalize the IOT Model in place. Having this controlled data space, analytics experts can define along the process which processing libraries imports or generate, plan and schedule jobs over collected data and decide visual platform to be used. From the EFPF point of view, MindSphere can be configured to be collected connected to data sources and offer the processing in a normalized way.

Another type of implementation is relevant for a usual asset monitoring scenario where specific devices or data sources are monitored via specialized agents. For example, a critical part of distributed systems is the communication infrastructure where routers should be monitored for potential attack patterns. Collected data is processed as events via a Complex Event Processing infrastructure in order to detect complex dependencies between temporal occurrences of disparate events, like network attacks.. Figure 36 illustrates an example of such a case:

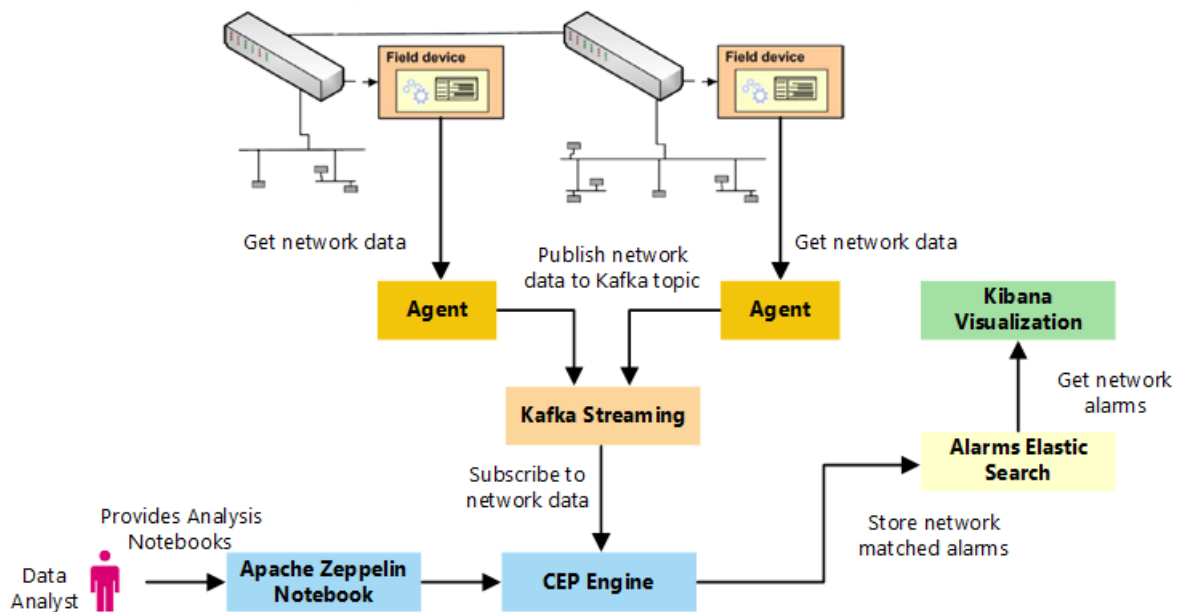


Figure 60: Example of collected data processed via CEP infrastructure

A data analyst builds a detection model that is invoked by detection pattern and correlated various field data.

For this sample case a network topology is considered and detailed about the way how process data is acquired out of communication nodes and how attacks are observed. As a key observation the process observation in this case is flexible enough to handle a large variety of observation agents, each of them fitted to specific processes, ingest them in an analytic chain and either deliver them to a situation awareness layer or perform local pattern detection in a CEP engine or do extended pattern matching in a Python notebook. Bellow

the monitoring infrastructure is described, followed by the adaptation toward the application use.

The network monitoring module acquires data about network status using some detectors and process it to detect near real time events. These events are special network state, based on which some decisions can be taken.

The detectors are SNMP (Simple Network Management Protocol) agents deployed directly on routers/switches (if the field device allows) or on devices as Raspberry Pi that are installed near the monitored network. The field devices should allow communication over SNMP protocol. The agent applications connect to the field device and interrogates it about network status. Another functionality of the agents is to send the data to a message broker, in this case Kafka.

The network monitoring data analyst defines the processing logic in the form of Analysis notebooks. Apache Zeppelin is an open source notebook which is used to enable interactive data analytics. The processing logic, defined as queries, is deployed to the CEP engine. The CEP engine subscribes to the data published on Kafka and process it.

The processing results (the network alarms) are saved on ElasticSearch file system. The Kibana data visualization is used to show generated network alerts.

Figure 61 describes the deployed services of the Network Monitor and the used hardware.

Service name	Service description	Involved hardware
kafka-conn	Publish/subscribe system, based on Apache Kafka, used to decouple the data providers and data consumers.	Machine connected running platform (network connected)
network-analyzer	Complex event processing based on Apache Flink. It runs the business logic to detect the required scenarios.	Machine connected running platform (network connected)
sie-snmp-data-collector	Application that acts a bridge between the snmp-agent and kafka-conn. It uses SNMP to get the data from a device and publishes to kafka-conn.	Machine connected running platform (network connected)
SNMP agent/daemon	SNMP agent service that enables a device to be monitored by sie-snmp-data-collector.	Raspberry PI running Raspbian (operating system based on Debian) connected to Defender platform network.

Figure 61: Deployed services of the Network Monitor

### 7.3.6 Risk Tool

The Risk Management & Analysis Tool enables users to create risk recipes and workflows to determine risks and risk scores from data provided by REST API and MQTT.

#### 7.3.6.1 Configuration

To determine risks, the tool allows for users to create customizable risk recipes and risk workflows through REST API and a frontend that makes use of the REST API.

##### Risk recipes

The tool makes use of *risk recipes*. These are data transformation modules which process the data into a new format. In some cases, data specific to the configured workflow may be required in order to compute a risk. For example, a recipe might require information about the deadline for a process, and this data may not be produced by the relevant data stream. In this case, the deadline would be configured in the tool ahead of time. The tool comes with premade risk recipes, but has a highly customizable risk creation functionality.

##### Workflows

Risk recipes can be chained into a risk workflow, which outputs the risks, and its visualisations, to the user. This allows for users to create workflows with multiple smaller recipes that can be reused in other workflows as well. Additionally, this allows users to receive multiple metrics at once, contained in the same output, instead of having to invoke outputs of multiple recipes independently. These workflows are defined by a chain of recipes, a data source, and a data sink. For testing purposes, a workflow can be run once, with test data. In practice, the source is defined as an input MQTT topic, and the sink is defined as an output topic.

##### Frontend

The Risk tool also includes a frontend web-app, which has the full functionality of the REST-API, without requiring the user to make API calls. Instead, they allow the user to fill in more visually intuitive forms. Specifically, it allows the user to see all available custom recipes, hardcoded recipes, and workflows. For each of these, their details can be seen by clicking on them, and in addition, they can be edited, there is “copy and edit” functionality, to make a similar recipe, and they can be deleted. In their details, creation, and edit windows, there is a testing panel that enables the user to verify whether they created a properly working recipe, or what kinds of input work well for their workflows. This testing functionality includes output comparison testing as well. Lastly, the tool allows the user to subscribe to workflow input topics on MQTT, to look at workflow metadata (such as last run time or last patch time), and to retrieve the last workflow output for a given workflow.



## Risk analysis service customizable risk recipe configuration.



Toggle Mockup UI

All Custom Recipes

▼

All Hardcoded Recipes

▼

Available Workflows

▼

Figure 62: Configuration frontend UI

All Custom Recipes			^		
Add a new recipe			+		
StandardStatisticsRecipe					
HistoricNormalizedDelaysRecipe					
FinalDelaysMeanRecipe					
FinalDelaysVarianceRecipe					
FinalDelaysStdRecipe					
FinalDelaysRecipe					
HistoricDelaysRecipe					
ProportionDelayedRecipe					

Figure 63: Recipe list in configuration UI, with create, edit, and delete buttons.

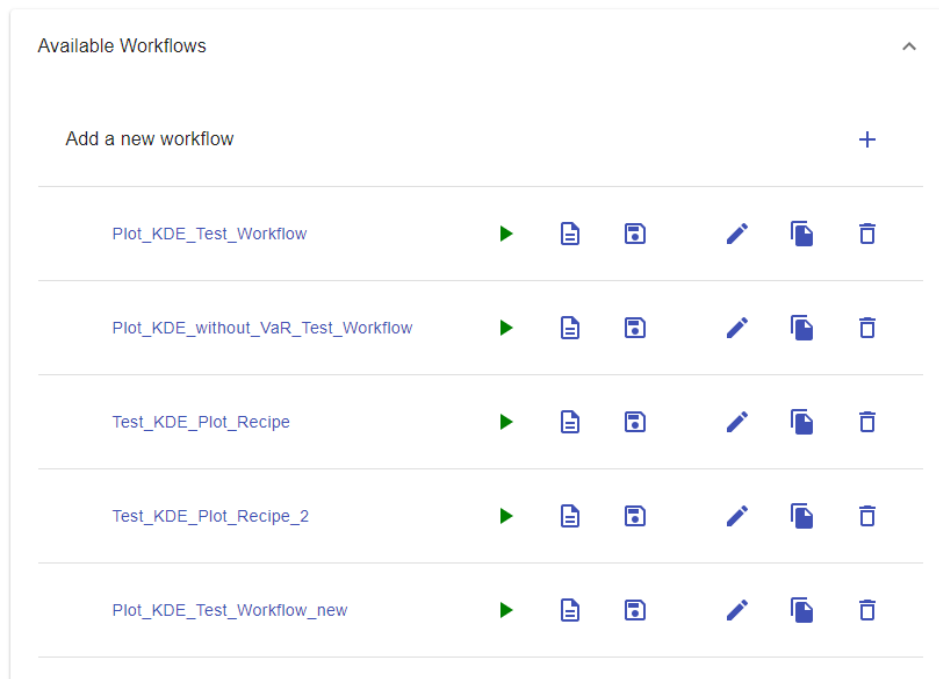


Figure 64: Workflow list in configuration UI, with subscription and workflow metadata

Create Recipe 🔒 🗑️

To create a new recipe, please fill in all the fields.

Recipe definition method

☒ Manually

☐ Upload a file

Recipe Name \*

Recipe Description \*

Recipe Type

Custom equation based ▾

Add formula Remove formula

Output field 0 *	Formula 0 *	
["output"]	a*x + b	Add operand for formula 0 Remove operand for formula 0
Operand name 0 of formula 0 *	Operand value 0 of formula 0 *	
a	["slope"]	
Operand name 1 of formula 0 *	Operand value 1 of formula 0 *	
b	["offset"]	
Operand name 2 of formula 0 *	Operand value 2 of formula 0 *	
x	["data", "specific_data_entry"]	

Add previous transform Remove previous transform

Test the recipe here:

Test Data Type

☒ Manually

Submit

Figure 65: Recipe creation and edit form.

Create Workflow

To create a new workflow, please fill in all the fields.

Workflow definition method

☒ Manually

☐ Upload a file

Workflow Name *	Workflow Description *	Workflow Input Topic *	Workflow Output Topic *
NewWorkflow	Workflow Description	in_topic	out_topic

Specify output field    Unspecify output field

Data → AutomaticallyConvertListsToNumpyArraysRecipe → ConvertAllNumpyArraysToListsRecipe → ConvertNumpyDataTypesToRegularDataTypesRecipe

Recipe Name

No custom recipe exists with this name. It could however have been manually added through an API call.

Remove current Recipe

Test the workflow here:

Test Data Type

☒ Upload a file

☐ Plain text

☐ Data already present or not needed

Drop the file(s) here, or click to load.

Selected Files:

Submit

Figure 66: Workflow creation and edit form.

### 7.3.6.2 Operation

As mentioned before, the tool includes a REST-API, which exposes endpoints which are used to (1) add, edit, and remove both recipes and workflows; and (2) to run workflows both a single time, and continuously via MQTT subscriptions.

After the workflow has been created, and the input topics have been subscribed to, the workflow will automatically run with the received input data, whenever an input is then received on a certain topic through the EFPF message bus. This data is then processed into statistics, risks, and visualisations. Except for the visualisations, these are all streamed through the message bus, to notify the users about their risks, and for the users to use directly.

In addition to the before mentioned configuration UI, the frontend will also have a monitoring UI, focussing more on a managing role, rather than an engineering role. This UI is still a work in progress, but it will contain the visualisations of the risks, along with their outputs.

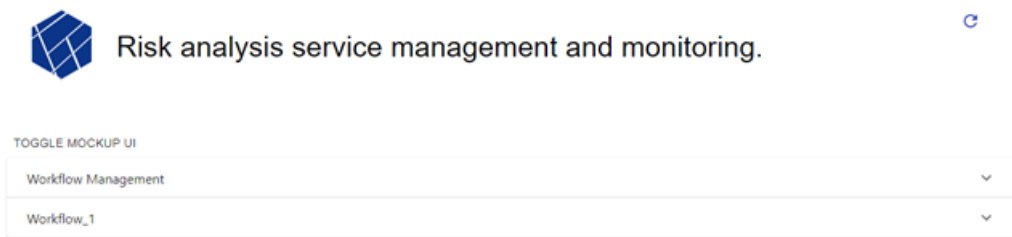


Figure 67: Management and monitoring UI prototype.

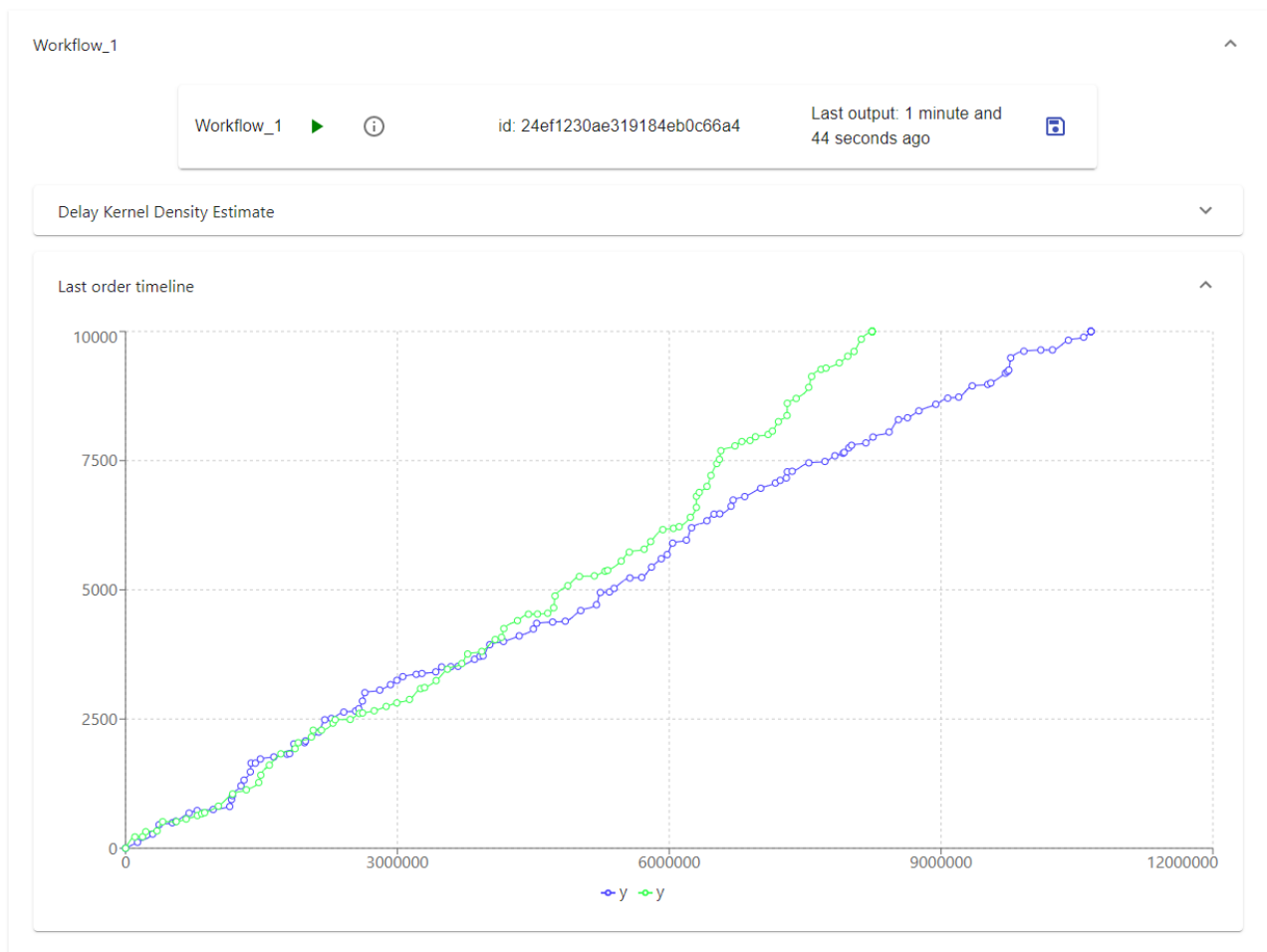


Figure 68: Visualisations in the management and monitoring UI prototype



## 7.4 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.

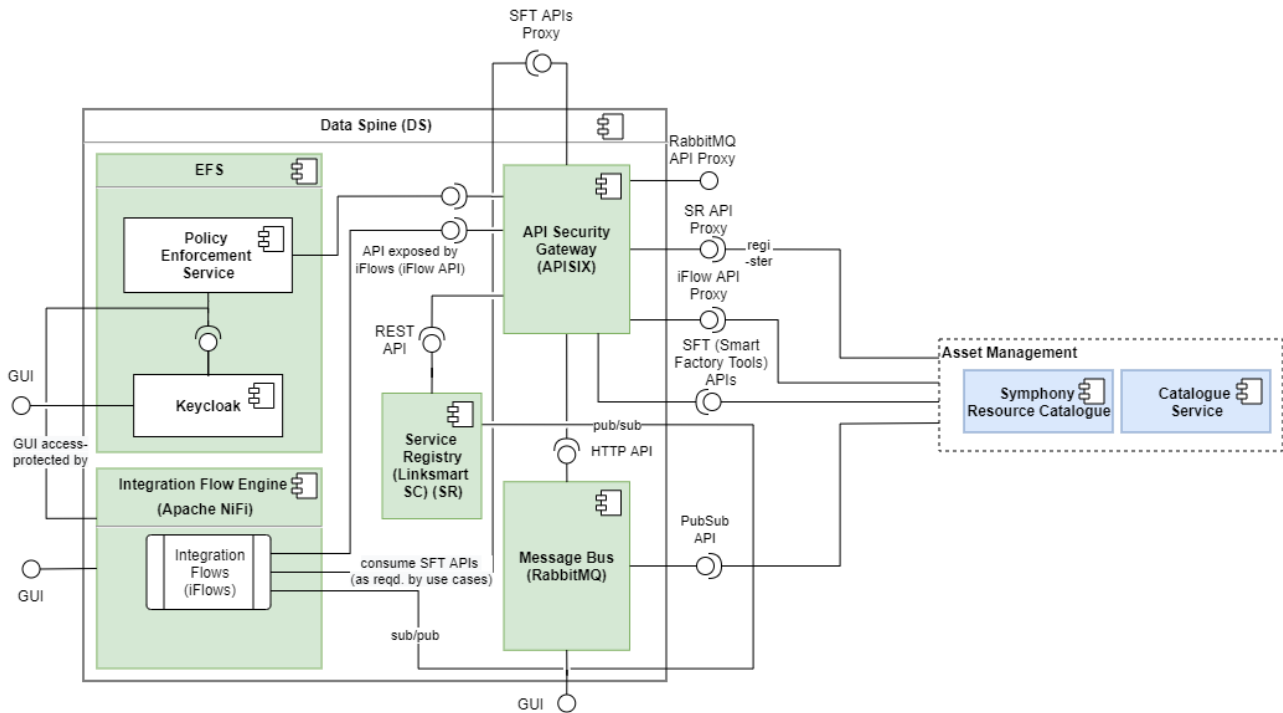


Figure 69: Interaction between the Asset Management Tools and the EFPF Data Spine

### 7.4.1 Symphony Resource Catalogue

Symphony Resource Catalogue is used for storing the description of all objects and mapping between objects to endpoints. Also, it leverages on REST API based on SPARQL for searching.

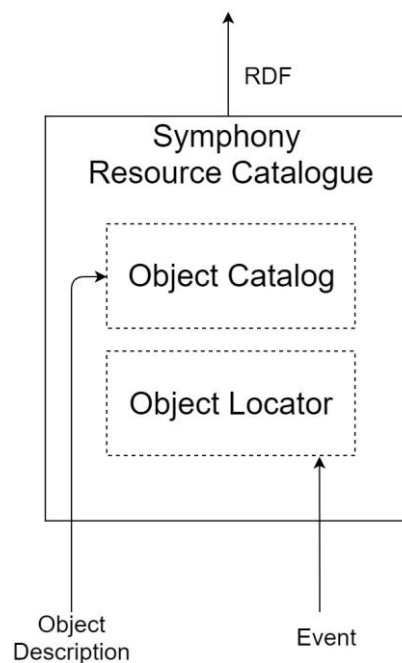


Figure 70 Symphony Resource Catalogue

Symphony Resource Catalogue is composed of two separate components:

1. **Object Catalog:** Contains the list of all the "objects" existing in a given installation of a system. Each object is univocally identified by an ID, and has further properties which include: the services it provides, the "space" it is located in, the physical properties it affects (e.g. temperature) and in which "space", the relations with other objects, and so on. This ontology has been originally derived from SAREF<sup>21</sup> and SAREF4BLDG<sup>22</sup>, with additions made to include Nextworks' previous proprietary information model, and is completely flexible. The "services" are the interfaces that each specific object provides.
2. **Object Locator:** Contains a map of the object IDs (as specified in Object Catalog) to the protocol-specific endpoints that a client application needs to use to access that particular object. Object Locator also contains endpoints for platform-level services which are not referred to by using an object ID (e.g. storage service, configuration service, orchestration service). Endpoints can be provided (and looked for) for multiple protocols, i.e. REST endpoints, gRPC endpoints, CORBA IORs, raw sockets. A specific object will be accessible by using one or more endpoints, depending on the type and version of the Symphony HAL which is controlling it.

Object Catalog contains static information (what the object is, what it can do, how it relates to other objects), whereas Object Locator contains dynamic information to reach an object at runtime (which might change, e.g. if another node takes control of a physical object in a high availability setup). Object Catalog is implemented as an ontology in Apache Jena Fuseki<sup>23</sup>, plus a wrapper to provide higher level methods to SparQL. Object Locator is

<sup>21</sup> [https://www.etsi.org/deliver/etsi\\_ts/103200\\_103299/103264/01.01.01\\_60/ts\\_103264v010101p.pdf](https://www.etsi.org/deliver/etsi_ts/103200_103299/103264/01.01.01_60/ts_103264v010101p.pdf)

<sup>22</sup> [https://www.etsi.org/deliver/etsi\\_ts/103400\\_103499/10341003/01.01.01\\_60/ts\\_10341003v010101p.pdf](https://www.etsi.org/deliver/etsi_ts/103400_103499/10341003/01.01.01_60/ts_10341003v010101p.pdf)

<sup>23</sup> <https://jena.apache.org/documentation/fuseki2/>

implemented as an API wrapper to etcd<sup>24</sup> which is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines

#### 7.4.1.1 Configuration

Since the component is part of the Symphony Platform and it is not decoupled yet, the configuration is possible through Symphony Platform. It is possible to view list of objects with ID, Type, Description and other characteristics. Also user can create Objects based on supported type of sensors and actuators.

#### 7.4.1.2 Operation

Symphony Resource Catalogue is a part of Symphony Platform and as mentioned before it still has dependencies to other Symphony components. After deploying the platform, the Resource Catalogue is up and ready to use.

### 7.4.2 Catalogue Service

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.



Figure 71: eClass root categories vs category hierarchy for logistics services

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 (Figure 72). 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,

<sup>24</sup> <https://etcd.io/>

products, product properties and so on. Moreover, products and services as well as catalogues are persisted on a UBL-compliant relational database. UBL attributes of a product or service within a catalogue can be seen in Figure 72:

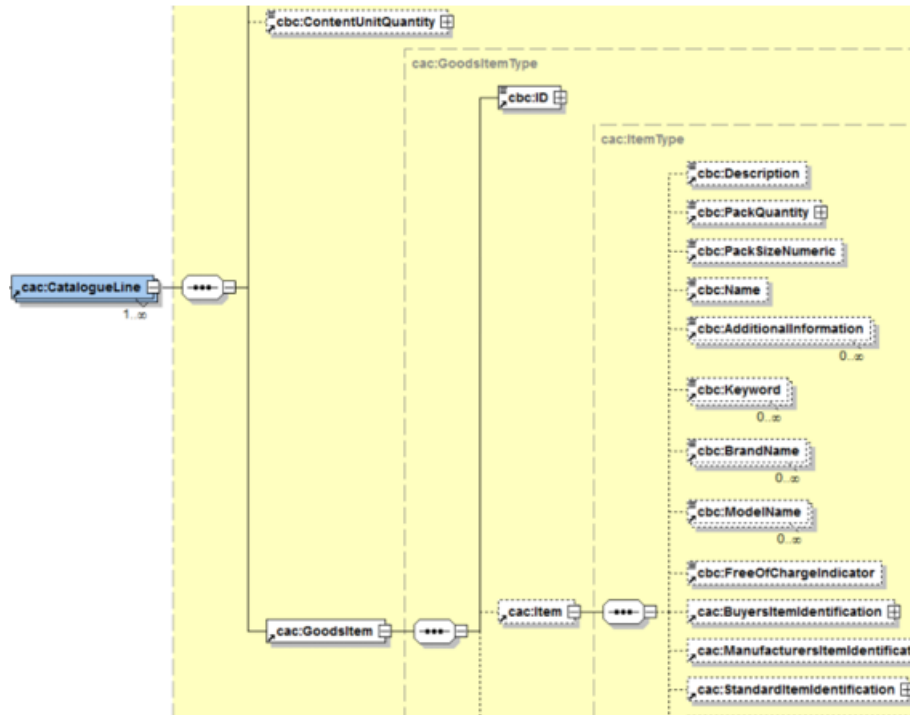


Figure 72. UBL Graphical representation for Catalogue Line schema

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. The overall, high level overview of Catalogue Service can be seen in Figure 73.

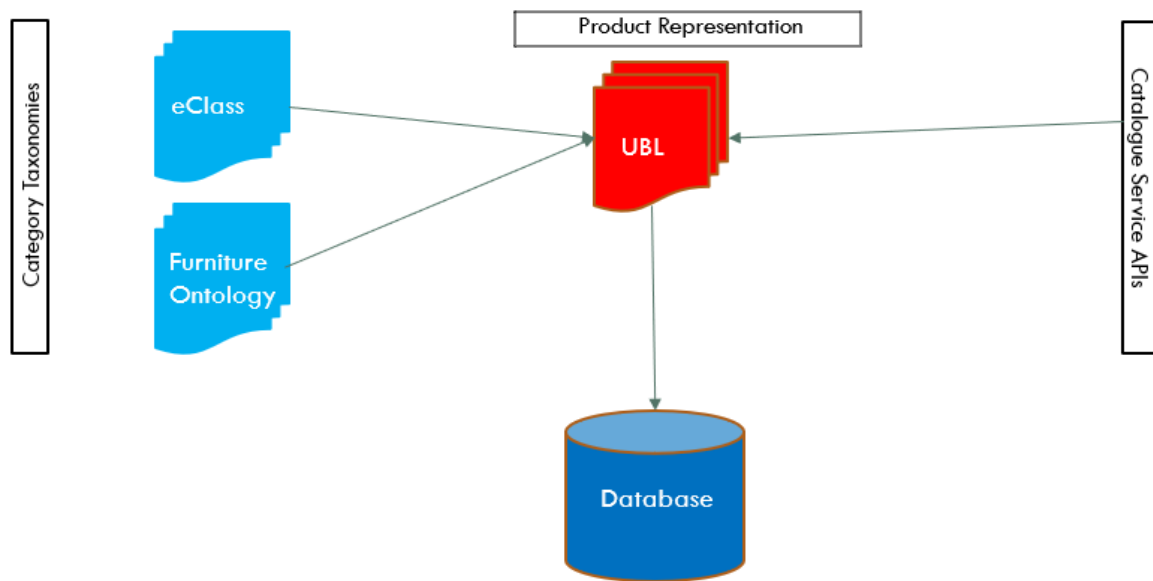


Figure 73. Overview of Catalogue Service

#### 7.4.2.1 Configuration

Catalogue Service uses YAML based default common configuration parameters defined on the NIMBLE platform itself such as the REST points of the other microservices that the Catalogue Service depends on, database connection parameters, etc. and all these default configurations can be overridden by setting appropriate environment variables or passing appropriate JAVA Virtual Machine arguments during the NIMBLE platform start up.

#### 7.4.2.2 Operation

Catalogue Service is one of the core components of the NIMBLE base platform and it is readily available when the platform is deployed on production environment.

## 8 Smart Factory Solutions

The previous sections describe the Tools, Services, Connectors and Gateways that are available within the EFPF ecosystem. Users of the ecosystem can use Tools and Services on their own, however one of the main goals is that they can be used together to solve larger or more comprehensive use cases. This is supported by the functionality offered by the EFPF Data Spine described in section 2, which includes data interoperability and transformation. The following sections describes the approaches a technical user can take to create Smart Factory Solutions and provides examples relating to the Pilot user requirements.

### 8.1 Composite Applications

In order to create Smart Factory solutions to meet current manufacturing challenges, it may be necessary to use multiple tools together to deliver broader functionality. Creating such a composite application is typically challenging since tools from different publishers will almost certainly use and offer different interfaces, protocols and data models. Traditionally the user selects the tools which are the closest “match” and human manual processes must provide the bridge between tools.

EFPF Ecosystem provides features to make interfacing Tools and Services more achievable through the Data Spine which enables communication, data transformation and protocol interoperability. The function of the Data Spine is provided in section 2 and in greater depth in D3.11 EFPF Data Spine Realisation.

Smart Factory solutions will generally involve two main patterns of communications, vertical communications between the Factory and the Data Spine and horizontal between the many Tools and Services that are needed for the specific application.

Figure 74 shows the vertical communication with the specific factory connector instance publishing data to the Data Spine Communications Broker on an application specific Topic configured by the user. Should the data or protocol need to be subsequently changed, the Integration Flow Engine can be used to develop an Integration Flow to make data transformations or protocol conversions.

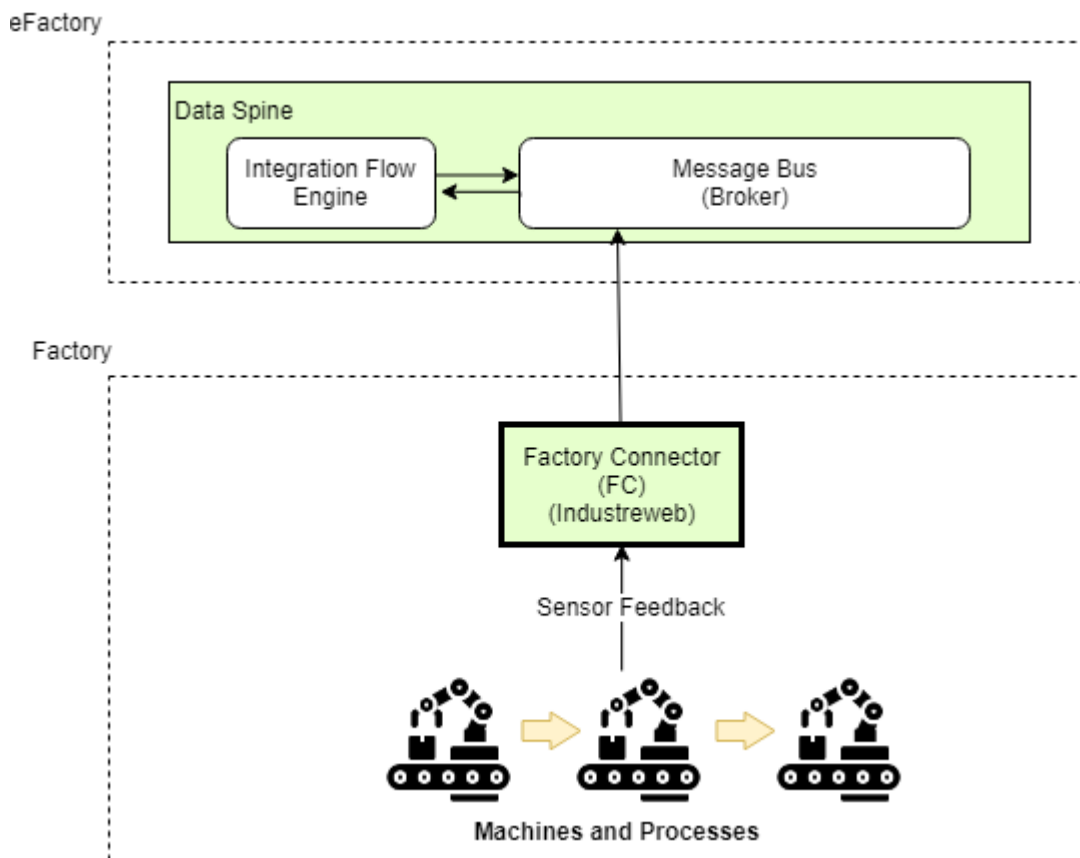


Figure 74: Vertical Communications within Smart Factory Solution

Figure 75 shows the horizontal data flows that are made between Tools and Services within a Smart Factory solution. This allows functionality of each tool to work together to create a broader overall function set. Tools and Services can utilise Pub/Sub communications via the message broker, or via using REST or alternative protocol using the Integration Engine to map between them. In addition the Integration engine allows for the data model a tool exposes to be transformed to a data model that will be required by another Tool or Service.

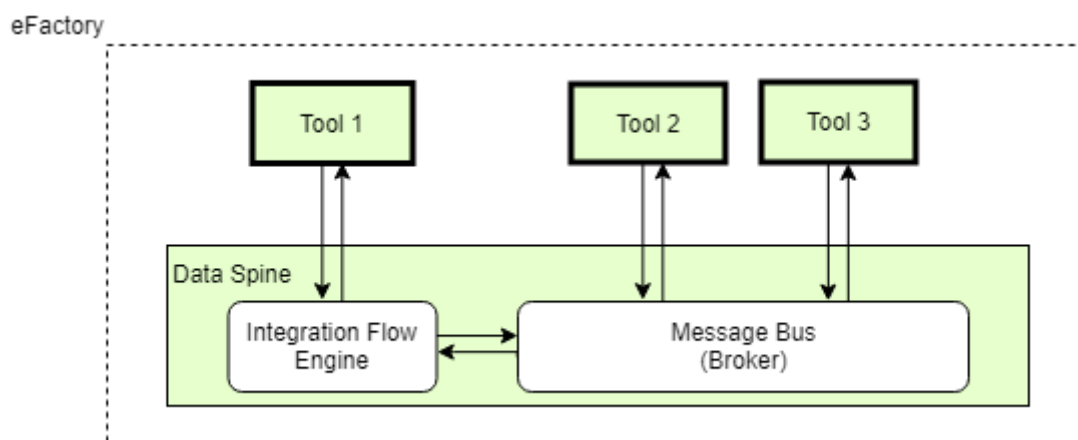


Figure 75: Horizontal Communications within Smart Factory Solution



### 8.1.1 Case Study

As an illustration of what can be implemented we will look at a scenario within the existing pilots that involves the integration of multiple EFPP Tools and Services to achieve a scope broader than what is offered by each tool individually. This composite application is designed to meet the user requirements gathered during the requirements gathering stage of the project. Figure 76 shows a User Story from Jira related to the Epic “Monitor Production Process”, whereby the requirement is to be able to optimise production.

Details

Type:

Story

Status:

**TO DO** (View Workflow)

Priority:

Nice to have

Resolution:

Unresolved

Affects Version/s:

None

Fix Version/s:

None

Labels:

Lagrama

Epic Link:

Epic 07 - Monitor production processes

Description

As a **Production manager**, I want to speed up the production so that I can reduce the time to serve the customer.

Acceptance Criteria:

- Mechanisms available to configure and optimise production processes and the supply chain in order to improve its performance

Figure 76: Production Optimisation User Story

The Smart Factory solution to fulfil this User Story is shown in Figure 77, where a Factory Connector is utilised to monitor specific condition sensors on a machine, and then publish that data to the EFPP Data Bus. This data is then subscribed to Store the data, and then using that data, multiple Analytics Tools can build models to describe the machine behaviour.

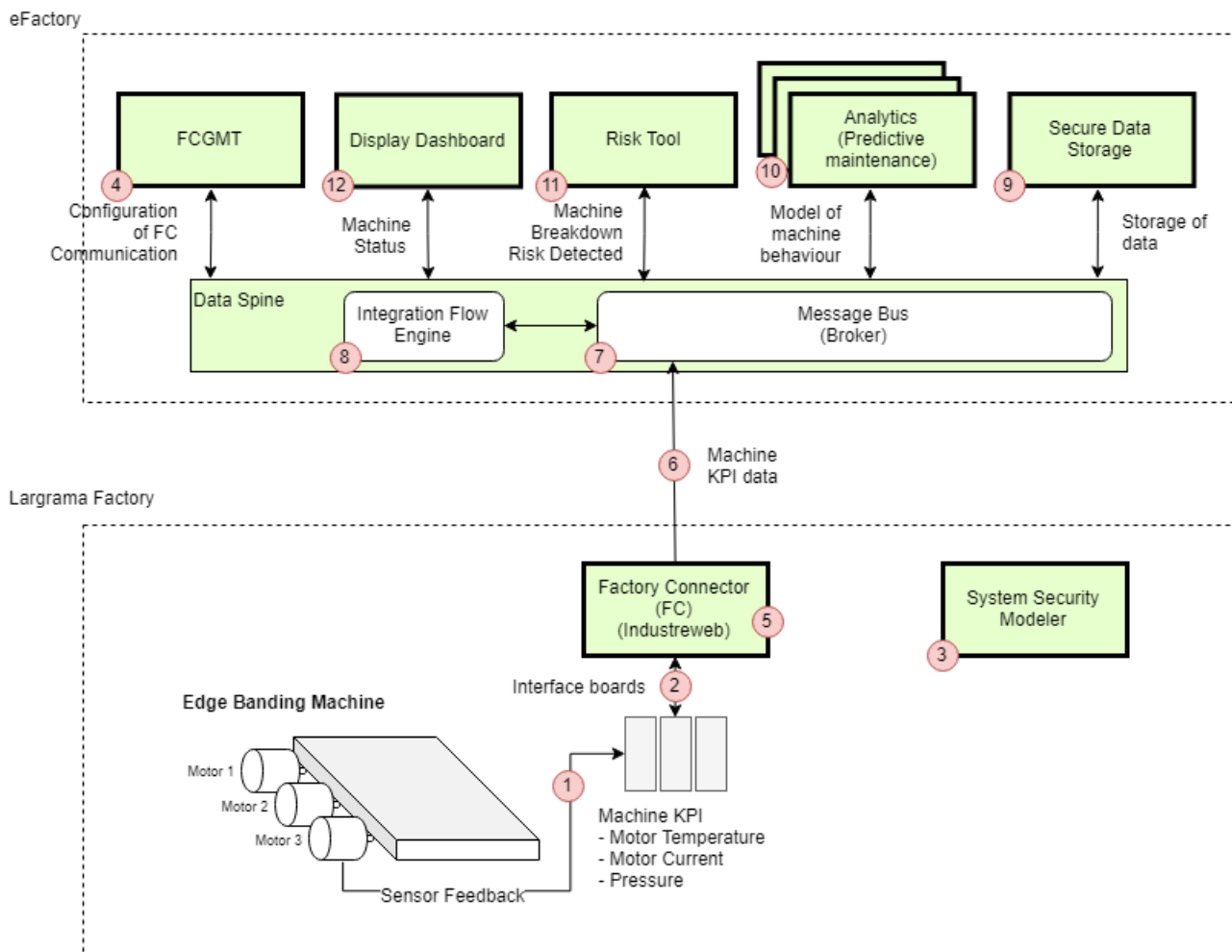


Figure 77: Production Optimisation Smart Factory Solution scenario

A description of the steps involved in this scenario are described below:

1. Data from the sensors is connected to the Factory Connector via a physical interface. In this case the sensor is wired to an EthernetIP Analogue Interface board.
2. The EthernetIP interface board is connected to the Factory Connector via the ethernet network within the factory.
3. The System Security Modeler can be used to verify the installation and to report on potential security risks from the network architecture and the components used.
4. To publish data on EFPF Data Spine as security token must be generated by the user configuring the factory connector, along with a topic on which to publish the data. The Factory Connector Gateway Management Tool is used to create these in order to commission the Factory Connector.
5. The Factory Connector is commissioned in order to interface with the data from the interface board. Also the Factory Connector is configured to publish on the user defined topic using the security token triggered either on a data change event or a user defined frequency
6. Data is published to the broker on the topic using the security token.

7. Data from the Factory connector is received by the broker and available on a user specific queue.
8. If any data model changes are required the Integration Engine can be used by the users creating an Integration Flow that will map between the fields in the original Data Model and the new one. This activity is dependent on the specific Tools used.
9. The Secure Data Store has been subscribed to the Topic by the User, so that the data gets saved for subsequent use by the other Tools.
10. The user can then use one or more Analytics Tools to model and analyse the data. In this scenario three solutions are used each providing a different result set to help the user understand when the machine exhibits behaviour which indicates it need maintenance.
11. The Risk tool is also utilised by designing a Recipe to subscribe to the data and detect when certain values published by the Factory Connector exceed warning thresholds. The Risk tool then generates an alert by publishing the alert to the broker and also sending an email alert to the user.
12. The runtime values, historical values from the Secure Data Store and Risk Tool alerts can be visualised on an Industreweb Display dashboard, to provide a high level UI for the User to monitor the machine.

In this way the Smart Factory solution provides a comprehensive solution to the original requirement to optimise and monitor production.

## 8.2 Software Development Kit

While the EFPF ecosystem provides a heterogeneous set of services for multiple purposes to support manufacturing businesses and improve their capability and virtualisation, there are numerous times where a simple service is not able to fully satisfy the business needs. Hence, there is often the need of not only composing and orchestrating two or more services provided by e-Factory, but also sometimes building something totally different. These applications are able to use services from EFPF and, using its Data Spine, have their own behaviour possibly including combining services with other third-party applications.

This capability to build applications with access to EFPF infrastructure whilst having access to full APIs, allows developers to include their own added value and expertise, whilst empowering them with functionality that would be hard to replicate by combining existing tools and services.

In addition to enabling applications to be developed, EFPF allows these applications to be deployed and published in its marketplace, the same that can also be used for retrieving elements to build new applications, therefore fostering reuse and modular development, standardisation and best practices. Moreover, having this development capability also creates an interesting opportunity for developers or software development companies to create their own markets for selling their applications to manufacturing businesses.

The development process is supported and complemented by an integrated Engagement Hub Portal which fosters the involvement of the development communities to support, debug and contribute ideas and suggestions for the improvement of the published applications. The hub will also allow pre-defined code snippets and patterns to be reused to support a natural evolution and development of best practice patterns.

The EFPF development environment itself is implemented by the services provided by the EFPF Software Development Kit (SDK). The EFPF SDK is a suite of tools (depicted in Figure 78: Software Development Kit (SDK) overview) built on top of and extending the vf-OS SDK Open Development toolset. Using the SDK developers are able to build and develop applications (manufacturing empowered apps) using the EFPF environment and data spine to offer Smart Factory solutions. This suite is a central environment for the development of applications and, generically, for the centralised access of the EFPF assets and functionalities. The SDK itself will not have a user interface per se, instead, it will be accessed as a set of APIs to access the main development resources. That way, the SDK will be able to provide to the Studio and to other application development components, the resources, and services that they require, as well as to access design orientated data stored in EFPF (models, patterns, and behaviours).

In general, data and models will be able to be retrieved from multiple sources including the following:

- EFPF Secure Data Store
- EFPF Data Spine Message Bus
- Marketplace
- EFPF Assets (enablers, drivers, internal and external services)
- Real manufacturing devices, sensors and other mechanisms via standard manufacturing API's

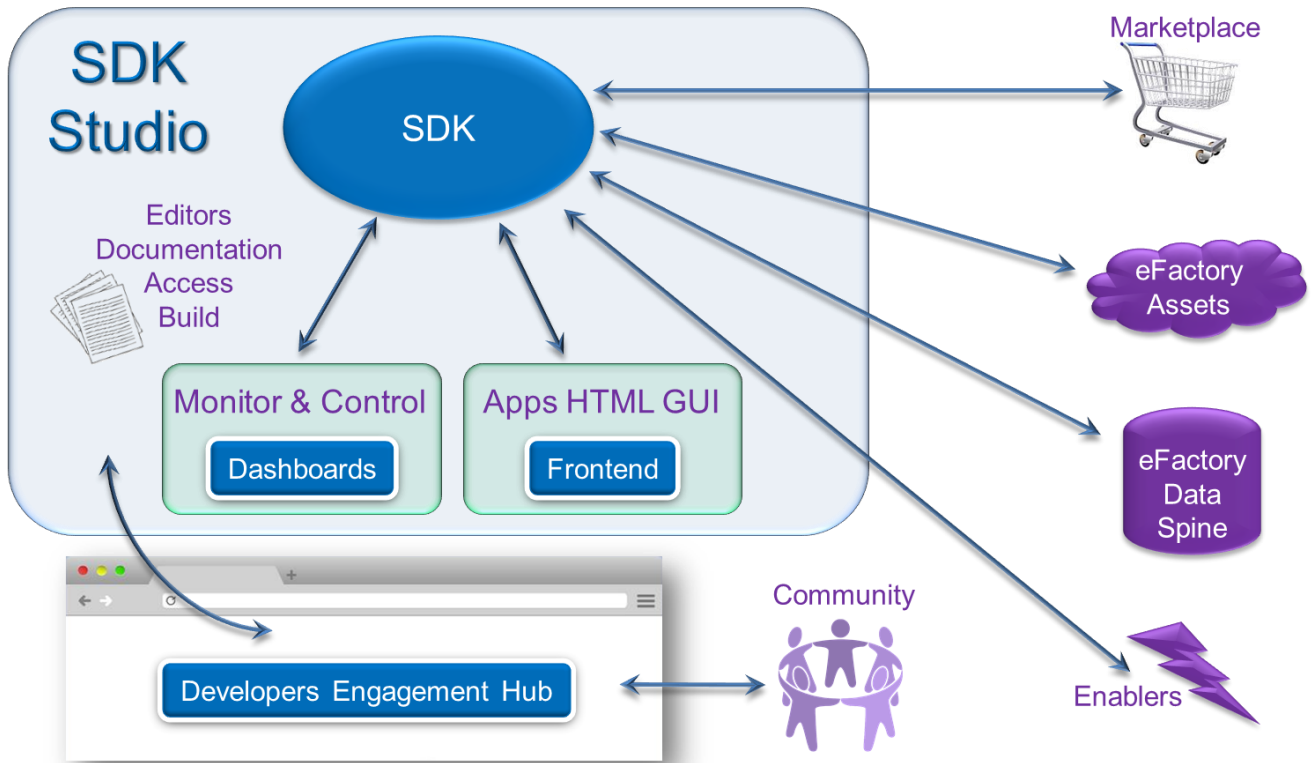


Figure 78: Software Development Kit (SDK) overview

The suite of tools provided by the SDK includes, among others:

- The SDK Studio (see Figure 79), a powerful web-based Integrated Development Environment (IDE) that includes state-of-the-art capability in editing (with features such as code editor with drag and drop of elements, build automation, code completion, compiler), debugging and building of code targeted to Docker images and applications that can then be published in the EFPF marketplace. This tool is the SDK Graphical User Interface (GUI) and will be used to integrate and embed other developing tools. Since EFPF aims to allow the development of manufacturing solutions, it is essential that the application development environment includes a user-friendly graphical interface that is simultaneously simple and clean, and fully functional for the development support

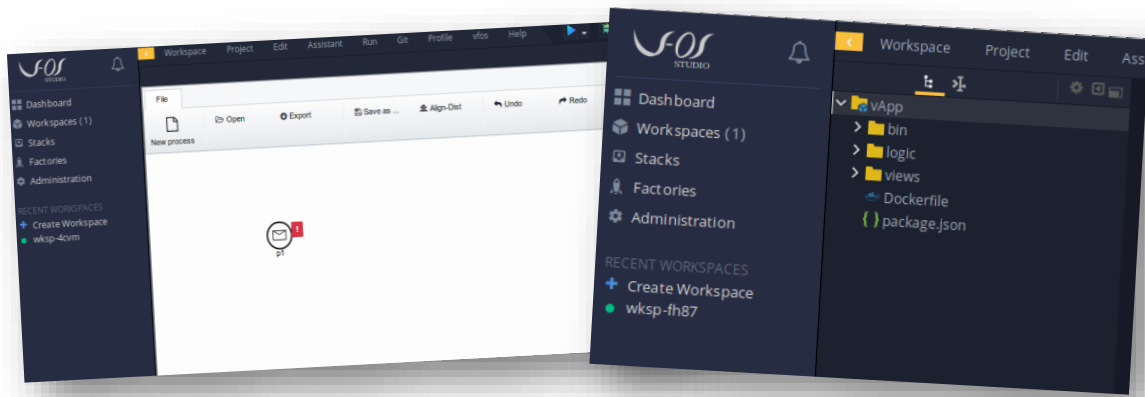


Figure 79: SDK GUI: The EFPF Studio IDE

- The SDK Front-end editor, which provides a set of classes for developers, which are integrated into the SDK Studio. These classes are made to support developers for different use cases: On one hand, UI templates, which use predefined EFPF styles which are also customisable; on the other hand, behaviour templates that process various operations successively. This tool will be used to build HTML5 + CSS + JavaScript contents for the building manufacturing application. Its purpose is to ease the development of applications by providing uniform User Interface (UI) elements to developers, so that they are able to plug and play these UI elements into programs under development. Thus, this results in a high-quality usability of the applications, which is of paramount importance for their success and adoption by the businesses, and can also establish a 'branded' environment. The Front-end editor provides a framework that facilitates a general 'look, feel, and composition' to the applications and will assist rapid development, by providing a compilation of UI elements including business logic via the OAK Studio
- The SDK itself, a set of centralised libraries and APIs to access all the EFPF potential: the data spine model and structures and the tools being developed in the other EFPF WPs. This will allow the developers of EFPF to have a unique endpoint to reach all the EFPF framework. The SDK provides more than building applications that support manufacturing. Industry requires rapid development of solutions that are able to serve the demand needs and time to market requirements. This should be established by providing best-of-breed solutions that result from the integration, composition and orchestration of pre-developed components, customised to meet the specific manufacturing needs
- The EFPF Developer Engagement Hub (see Figure 80) is an environment whose purpose is to foster and promote the productivity of developers, whether by integrating the developed Open Source code of previously developed applications, but also including a set of other tools to encourage development community building such as wikis, issue trackers, forums, and blogs. It supports the continuous integration of the developed applications by triggering automated tests upon the commit of versions in the version control environment. Additionally, it is planned to allow the definition of an organised hierarchy of development projects to cope with complex environment

structures that include multiple applications. It is web portal that has the purpose of fostering and promoting the developers integration and engagement in the development process, allowing the development community to interact with the existing modules by providing tutorials, documentation, collaboration tools (e.g., wikis, forums, issue trackers, chat) and mostly, by hosting and maintaining the source code of the applications in version control. One of the core motivation enablers to getting the application developers on board is for them to use a development environment, which they are familiar with, e.g., GitHub/Stackoverflow/JIRA. These allow developers to interact with each other at both a personal and technical level, or to store code for improvement, or track bugs.

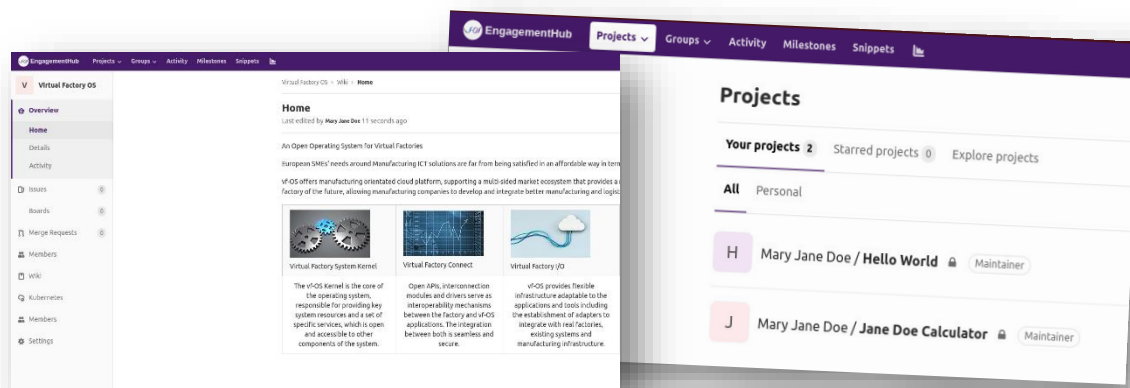


Figure 80: Developers Engagement Hub

The process to develop an application in EFPF is described in Figure 81.

It starts of course with the idea and scope of the application to be created, followed by the creation of mock-ups for the application interfaces.

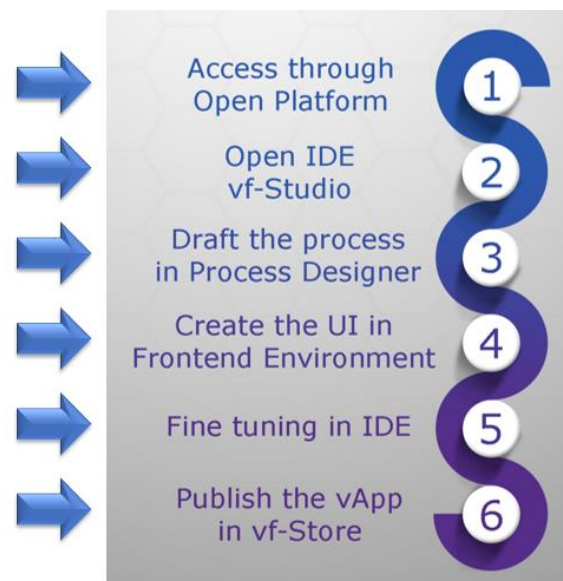


Figure 81: EFPF Application Development Process



When all the definitions are complete, the development process starts by 1) accessing the EFPF platform Figure 82, authenticating and then proceeding to 2) opening the EFPF SDK Studio IDE, shown in Figure 83. There, the developer will be able to create the application project and determine which integrations it will perform with the other EFPF assets.

The following step 3) comprises drafting the process that will undertake the algorithm of the developed application. This can be done manually or using a support application, the Process Designer, which is a tool derived from the EFPF WASP tool (see section 6.3.2) which can be invoked by the Studio (in a Studio view). If the Process Designer tool is used, the tool is able to generate code that will implement the defined processes, and that code is then automatically imported by the Studio.

After defining the application flow, the developer should 4) design the application front-end, based on the defined mock-ups and behaviours. Again, the Front-end editor will be integrated by the Studio and it will be able to be run in a Studio view. The HTML, JavaScript, and other outcomes of the Front-end definition will then be automatically imported by the studio in the shape of more code.

Subsequently, 5) the developer will be able to edit all the resulting code in the Studio, making all necessary changes, improvements and value-added actions, and customising the behaviour, algorithm and configurations of the application.



Figure 82: Login EFPF

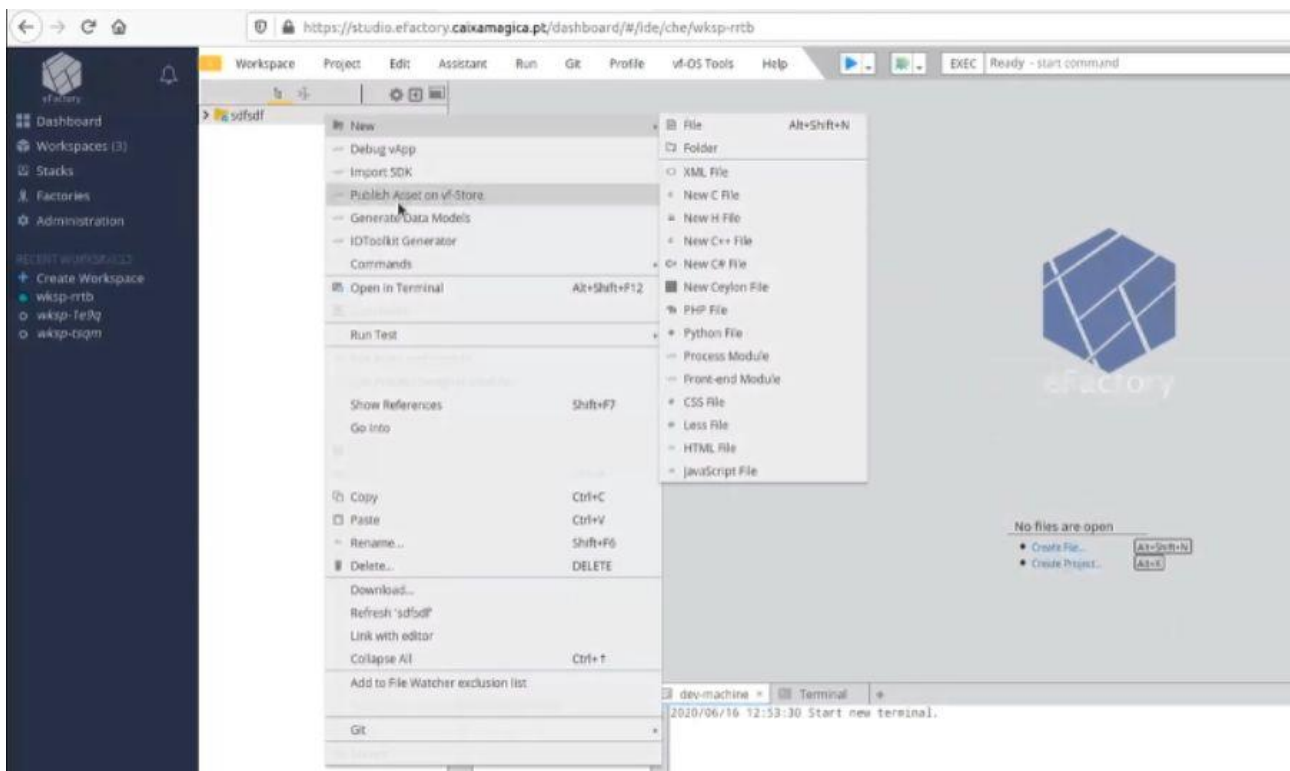


Figure 83: EFPF Studio

The final step of the development of the application will be publishing it on the EFPF marketplace, action which also will be able to be performed right from the Studio IDE (Figure 84).

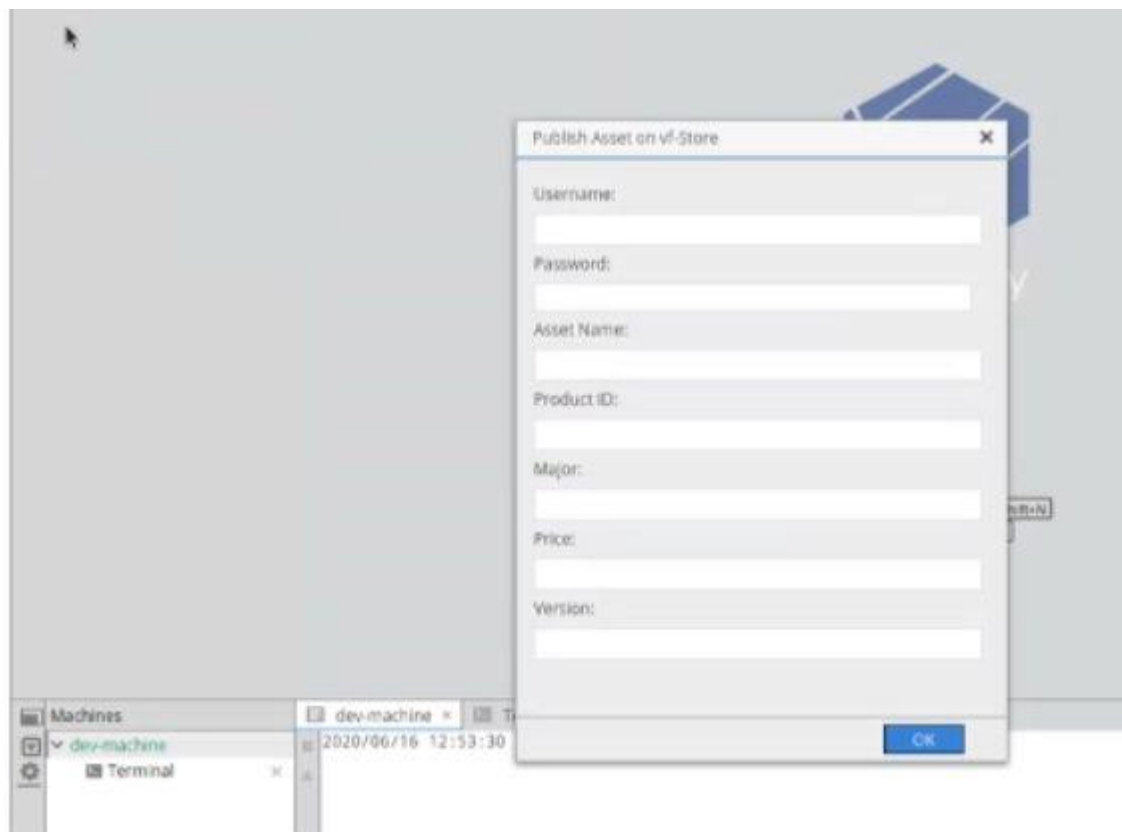


Figure 84: Studio publishing application in the EFPP Marketplace

The finalised application is then able to be sold, distributed or even reused in other applications developed in EFPP.

## 9 Conclusion and Outlook

The EFPF ecosystem already has a comprehensive and diverse catalogue of Tools and Services to offer. This is enabled by the federated approach to the base platforms which themselves support a range of mature tools and services via their existing marketplaces. Users can purchase and deploy Tools from EFPF via the base platforms and in doing so gain productivity, quality and process improvement gains with the convenience of logging in to a single ecosystem.

The greatest opportunity to users however is the ability to combine tools and services to satisfy the needs of today's manufacturing Industry 4.0 environments. This combination of Tools and Services is enabled by the EFPF Data Spine which offers both protocol integration and data transformation so that regardless the interface a Tool or Service implements it can interact with other Tool interfaces in an automated fashion. Traditionally the isolated nature of Smart Factory tools has restricted users from the implementing solutions to tackle broad scope production issues. With the EFPF Ecosystem users can now achieve these composite solutions, and also 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.

In addition to this, the ability for users to develop their own Tools through the integrated SDK framework means that should the required functionality be unavailable on the EFPF Tool catalogue, Users can develop their own Smart Factory solutions. 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 3<sup>rd</sup> party tools.

The execution of the EFPF pilots will be a valuable opportunity to evaluate the strengths of this approach, with the Open Call experimentation providing an exciting opportunity to support a different set of end-user challenges and gain feedback to enable the ecosystem to mature.

## Annexe A: History

Document History	
<b>Versions</b>	<p>V2.0</p> <ul style="list-style-type: none"> <li>Revised version of the document to accommodate Reviewer feedback</li> </ul> <p>V1.0</p> <ul style="list-style-type: none"> <li>Final version of the document</li> </ul> <p>V0.6</p> <ul style="list-style-type: none"> <li>Final content revisions</li> </ul> <p>V0.5</p> <ul style="list-style-type: none"> <li>Updated formatting and added content for section 5</li> </ul> <p>V0.4</p> <ul style="list-style-type: none"> <li>Consolidated contributions</li> </ul> <p>V0.3</p> <ul style="list-style-type: none"> <li>Revised subheading structure</li> </ul> <p>V0.2</p> <ul style="list-style-type: none"> <li>Example content added and circulated for contributions</li> </ul> <p>V0.1</p> <ul style="list-style-type: none"> <li>Final packaging – Following updates are implemented based on M18 Review report: <ul style="list-style-type: none"> <li>The objectives and architectural considerations and implications need are explained</li> <li>The data model interoperability section is strengthened to explain architectural design and maintainability aspects</li> <li>The architectural approach is elaborated for all key components and made consistent</li> <li>The user-requirements are introduced and linked with developed solutions</li> </ul> </li> </ul>
<b>Contributions</b>	<p>C2K</p> <ul style="list-style-type: none"> <li>Simon Osborne</li> </ul> <p>NXW</p> <ul style="list-style-type: none"> <li>Ali Nejabati</li> </ul> <p>FOR</p> <ul style="list-style-type: none"> <li>Nisrine Bnouhanna</li> </ul> <p>CNET</p> <ul style="list-style-type: none"> <li>Mathias Axling</li> </ul> <p>ASC</p> <ul style="list-style-type: none"> <li>Brian Clark</li> </ul> <p>FIT</p> <ul style="list-style-type: none"> <li>Rohit Deshmukh</li> </ul> <p>ICE:</p> <ul style="list-style-type: none"> <li>Usman Wajid</li> <li>Mitch Hancock</li> </ul> <p>CERTH:</p> <ul style="list-style-type: none"> <li>Alexandros Nizamis</li> <li>Ioannis Iakovidis</li> </ul> <p>ALM:</p> <ul style="list-style-type: none"> <li>Carlos Hermans</li> <li>Carolyn Langen</li> </ul> <p>SRDC:</p> <ul style="list-style-type: none"> <li>Senan Postaci</li> </ul>

	<p>CMS:</p> <ul style="list-style-type: none"><li>• Carlos Coutinho</li></ul> <p>UoS:</p> <ul style="list-style-type: none"><li>• Stefano Modafferi</li></ul> <p>LINKS:</p> <ul style="list-style-type: none"><li>• Edoardo Pristeri</li></ul> <p>SIE:</p> <ul style="list-style-type: none"><li>• Raluca Maria Repanovici</li></ul> <p>ELN:</p> <ul style="list-style-type: none"><li>• Maximilian Norz</li></ul>
--	--

## **Annexe B: References**

[WABH18] U. Wajid, G. Bhullar. Enterprise Interoperability. Towards Interoperability Across Digital Manufacturing Platforms, pp. 81-92 (2018).



## 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 consortium nodes	Docker	Docker image
Distributed Workflow and Business Process Design and Execution	WASP process design and execution platform, together with integrated service marketplace	ICE server	Docker	Docker images
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 and identification 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		
--------------------------	--	------------------	--	--



[www.efpf.org](http://www.efpf.org)