CEN/CENELEC/WS EFPFInterOp

Date: 2022-06-22

CWA: XXXX

Secretariat: ASI

European Connected Factory Platform for Agile Manufacturing Interoperability (EFPFInterOp)

Einführendes Element — Haupt-Element — Ergänzendes Element

Élément introductif — Élément central — Élément complémentaire

ICS:

CCMC will prepare and attach the official title page.

European foreword

This CEN and CENELEC Workshop Agreement has been developed in accordance with the CEN-CENELEC Guide 29 "CEN/CENELEC Workshop Agreements – A rapid prototyping to standardization" and with the relevant provisions of CEN/CENELEC Internal Regulations - Part 2. It was approved by a Workshop of representatives of interested parties on YYYY-MM-DD, the constitution of which was supported by CEN and CENELEC following the public call for participation made on 2020-12-16. However, this CEN and CENELEC Workshop Agreement does not necessarily include all relevant stakeholders.

The final text of this CEN and CENELEC Workshop Agreement was provided to CEN and CENELEC for publication on YYYY-MM-DD.

Results incorporated in this CWA received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825075.

The following organizations and individuals developed and approved this CEN and CENELEC Workshop Agreement:

- 3D ICOM GmbH & Co. KG
- AM Allied Maintenance GmbH
- Ascora GmbH
- Austrian Standards International
- Brimatech Services GmbH
- Caixa Mágica Software SA
- The Centre for Research & Technology, Hellas - CERTH
- DIN Deutsches Institut für Normung e. V.
- Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e. V.
- Hanse Aerospace Wirtschaftsdienst GmbH
- Information Catalyst for Enterprise Ltd.
- AIDIMME, Instituto Técnológico Metalmecánico, Mueble, Madera, Embalaje y Afines
- Jotne IT
- Salzburg Research Forschungsgesellschaft m.b.H.
- SRDC Software Research & Development and Consultancy Corp.
- Walter Otto Müller GmbH & Co. KG
- ZDMP Project (H2020 – No 825631)

Attention is drawn to the possibility that some elements of this document may be subject to patent rights. CEN/CENELEC policy on patent rights is described in CEN-CENELEC Guide 8 “Guidelines for Implementation of the Common IPR Policy on Patent”. CEN shall not be held responsible for identifying any or all such patent rights.

Although the Workshop parties have made every effort to ensure the reliability and accuracy of technical and nontechnical descriptions, the Workshop is not able to guarantee, explicitly or implicitly, the correctness of this document. Anyone who applies this CEN Workshop Agreement shall be aware that neither the Workshop, nor CEN, can be held liable for damages or losses of any kind whatsoever. The use of this CEN Workshop Agreement does not relieve users of their responsibility for their own actions, and they apply this document at their own risk. The CEN Workshop Agreement should not be construed as legal advice authoritatively endorsed by CEN/CENELEC.

Introduction

With the advent of Mass Production the goal in manufacturing was to reduce the cost per piece down as much as possible within given quality and time constraints. A lot of methods were introduced like Lean Manufacturing or Six Sigma that fostered constant improvements in all relevant manufacturing areas.

Production capacities were distributed into a network of suppliers across several countries and continents with an optimized production flow (like "Just in time production") which results in a rather static supply chain. The supply pyramid from Figure 1 shows the hierarchy between the different tiers of suppliers.

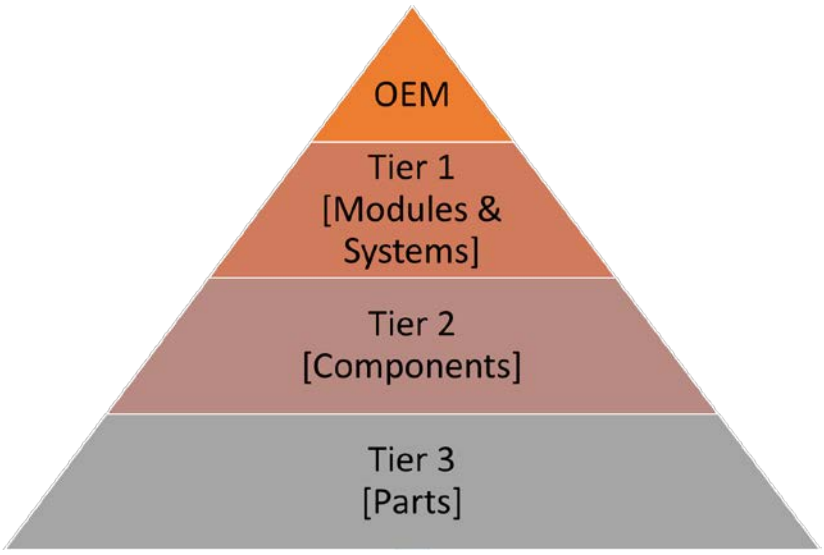


Figure 1 — Supply pyramid

In the manufacturing domain, as a result of increasing digitalization and the rapid growth and pervasiveness of smart factory and (Industrial) Internet of Things ((I)IoT) solutions, the industry trends are changing, and new possibilities are arising.

Due to changes in customer behaviour Agile Manufacturing becomes increasingly important – with the term "Lot Size 1 Manufacturing" as the keyword. This means as a company to be able to manufacture products that are tailored to the customers wants as much as possible. In Automobile manufacturing this is almost a reality as there are so many options a customer can choose from

The traditional way of producing in large quantities helps reduce the product cost per unit significantly. However,

a) new demands of highly customised products are not possible or difficult to fulfil / too time consuming with the current supply chains (difficult to find specialised suppliers quickly and establishing new ad-hoc, short-lived supply chains);
b) entries of SMEs barred as joining multiple platforms is too expensive, the SMEs don't always have the necessary digital infrastructures (e.g., ERP systems) needed to directly become part of the supply chain network of large OEMs or to prove adherence to the rigorous quality constraints set while manufacturing certain products for the OEMs – this is a hindrance to innovation.

Today's smart factory and IoT platforms are largely heterogeneous, vendor-specific, vertically oriented, functionality-wise fragmented, and behind their own closed identity and access management mechanisms. Because of such interoperability gaps between services of different platforms at the levels of interfaces, communication protocols, data formats, data models, identity providers, etc., it is not

possible to easily form composite applications with services from multiple platforms and support companies in their aforementioned endeavours.

The goal of this CWA is to provide a blueprint of a federation platform and describe components and functionalities that reduces the barrier to innovation by providing seamless access to services and solutions through an open platform. The creation of a federated ecosystem of IoT platforms of companies in the manufacturing domain can help companies form agile, ad-hoc collaborative networks, establish dynamic supply chains, and optimize production processes to meet such market demands as lot-size-one manufacturing. In addition to the definition of a reference architecture (see Figure 2 below) a specific implementation of a federated platform from the EFPF project will be described as well.

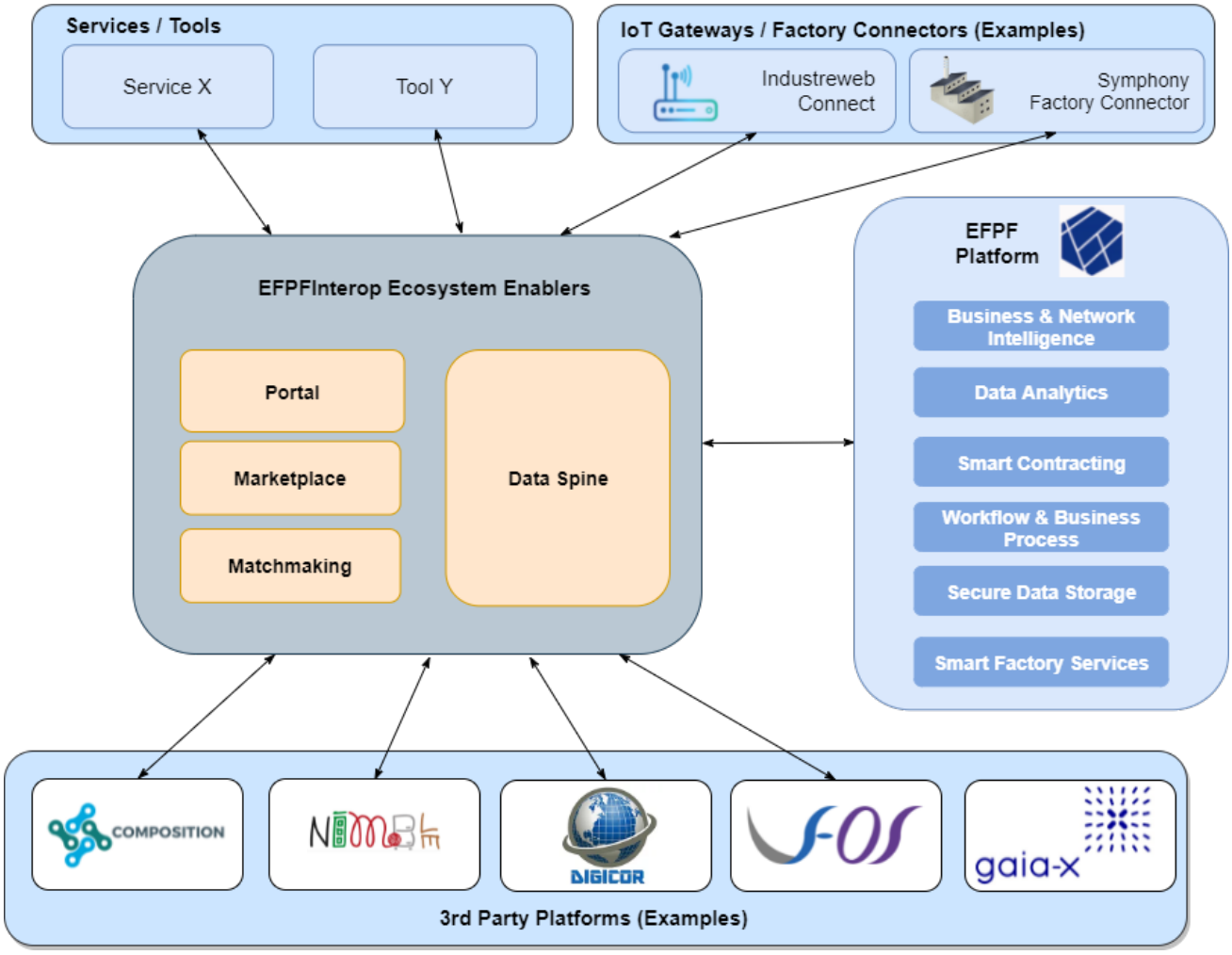


Figure 2 — EFPFInterop ecosystem overview

1 Scope

This CEN-CENELEC Workshop Agreement (CWA) defines a reference architecture for federating manufacturing platforms focusing on the interoperability on Service-Oriented Architecture (SOA), Protocol, Security and Data Model level. Additionally, a reference implementation in the form of the EFPF Data Spine and associated components will be described including Best Practices identified.

This CWA will not define requirements related to safety aspects.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at http://www.electropedia.org/
- ISO Online browsing platform: available at https://www.iso.org/obp/ui

3.1

Data Spine

federated interoperability enabler that bridges the interoperability gaps between the services of heterogeneous platforms and enables an easy and intuitive creation of cross-platform applications

3.2

Integration Flow

dataflow or workflow created using the Integration Flow Engine component of the Data Spine for realising composite applications, performing data model transformation, data enrichment, or protocol translation, etc.

3.3

Digital Platform

platform that provides offerings such as digital tools, services and data and secures access to them using its own Identity and Access Management service

3.4

Ecosystem of Digital Platforms

federation of digital platforms that enables an easy creation of cross-platform applications

4 EFPFInterop Platform Ecosystem Requirements

Building on the motivation from the Introduction chapter we will list in this chapter the requirements relevant to create federated platform ecosystems.

4.1 Interoperability Approach

The framework for enterprise interoperability described in EN ISO 11354-1 defines three different approaches to enterprise interoperability. These can also be applied to digital platform interoperability if the digital platforms are considered as heterogeneous enterprises.

- Integrated Interoperability Approach: This approach defines a common data model or a common Application Programming Interface (API) and all the digital platforms in the ecosystem must adhere to it.

- Unified Interoperability Approach: This approach defines a common, canonical data model at the ecosystem-level similar to the Integrated Interoperability Approach. However, in this approach, the common data model is defined at the meta-level and therefore it is a non-executable entity.

- Federated Interoperability Approach: This approach does not prescribe a common data model at the ecosystem and the all the digital platforms in the ecosystem are free to define and use their own data models. Interoperability needs to be enabled on-demand, when required by a use case, through data model transformation. The platforms need to share an ontology to map between their data models to establish interoperability.

The EFPFInterop ecosystem consists of heterogeneous digital platforms that are owned and managed by independent entities. Therefore, enforcing a common data model or API at the ecosystem-level is not feasible. Defining a common meta-model at the ecosystem-level is very difficult. With new tools, services and/or platforms joining the ecosystem, the administrator is burdened with updating the meta-model frequently, while also ensuring backwards compatibility with the already connected platforms. Therefore, Integrated and Unified Interoperability Approaches don't scale well with the increasing number of connected tools, services and platforms in the ecosystem. In contrast, the Federated Interoperability Approach distributes the burden of establishing interoperability among the service consumers, when they want to consume a particular service, making it scalable with the rapid growth of the ecosystem. Therefore, EFPFInterop approach aligns closely with the Federated Interoperability Approach.

4.2 High-level Functional Requirements

4.2.1 General

In this section, the challenges in creating an ecosystem/federation of digital platforms are identified. From each challenge, a requirement is formulated. To fulfil each requirement, a component is defined. These components, called 'Ecosystem Enablers', are the building blocks that enable the creation of the ecosystem.

4.2.2 Ecosystem Enablers: Data Spine

The lack of cross-platform interoperability is a major roadblock in the creation of an ecosystem of digital platforms to enable an easy creation of applications using services from multiple platforms. Interoperability gaps exist between the services of heterogeneous platforms mainly at the levels of security mechanisms, communication protocols, data models, data formats, data values, interaction approaches and communication patterns, etc., as illustrated in Figure 3. This section describes how the components of the Data Spine bridge these interoperability gaps and enable cross-platform communication.

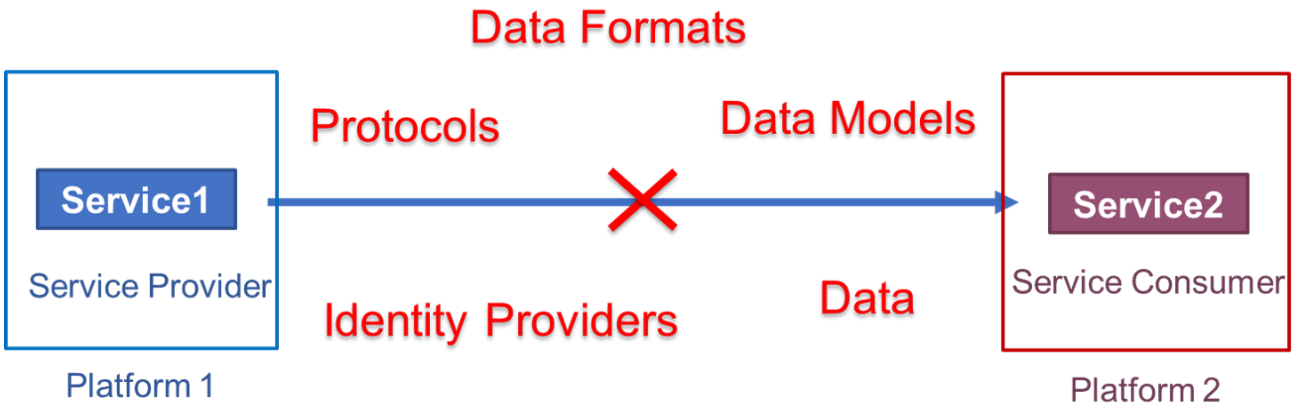


Figure 3 — Illustration of interoperability gaps between the services of digital platforms

Challenge 1) Security interoperability: The services of each digital platform are behind their own, closed Identity Providers and a user needs to get user accounts for multiple platforms to access their services.

Requirement 1: The users should be able to seamlessly access tools and services from different platforms using a single set of credentials.

Solution: This challenge is addressed by using a component called '**Data Spine Security Portal**' that federates the Identity Providers of the platforms in the ecosystem.

Challenge 2) Protocol and Data Model Interoperability and tooling support for an easy service composition: The interoperability gaps between **services** of heterogeneous platforms in the ecosystem prevent cross-platform service-level communication.

Requirement 2: It should be possible for users to create cross-platform applications using the existing services that may use different communication protocols and data models, without making any changes to them.
Requirement 3: The users should not be required to deploy and maintain additional components on their self-managed servers.
Requirement 4: The creation of cross-platform applications should be possible easily and intuitively, with minimal coding effort.
Requirement 5: It should be possible for users/developers from the same or different platforms/companies to collaboratively create composite applications.

Solution: This challenge is addressed by using a component called '**Data Spine Integration Flow Engine**' that provides a low-code development environment for the creation of composite applications.

Challenge 3) Support for asynchronous Pub/Sub communication pattern and messaging protocols

Requirement 6: The ecosystem should not only support message-based communication, but also enable the composition of tools/services that use the messaging pattern.

Solution: This challenge is addressed by using components called **'Data Spine Message Broker'** and **'Data Spine Integration Flow Engine'** that support Pub/Sub messaging pattern and protocols.

Challenge 4) Service/API discovery and metadata

Requirement 7: The users who create cross-platform applications using the existing services need a mechanism for discovering the existing services and retrieving their technical metadata such as the API specifications.

Solution: This challenge is addressed by using a component called **'Data Spine Service Registry'** that provides a mechanism for lifecycle management and discovery of service/API metadata and endpoints.

Challenge 5) Decoupling of application logic and policy enforcement

Requirement 8: The users should be able to easily secure the APIs of the cross-platform applications created by them.

Solution: This challenge can be addressed by using policy enforcement points embedded into the Ecosystem Enabler components or by using a component called **'API Security Gateway'** that intercepts incoming requests to component APIs and enforces access policies with the help of the Data Spine Security Portal.

4.2.3 Ecosystem Enablers: Portal

Challenge 6) Single point of entry to the ecosystem / "GUI interoperability": The ecosystem should have a single entry point that provides access to all the tools and services. The search and access to those tools should be simplified without the need to create multiple accounts in multiple platforms and provide the necessary documentation.

Requirement 9: The potential users of the ecosystem should be able to register in order to get a new user account.
Requirement 10: The users should be able to browse and view the information about the connected tools, services, and platforms through a unified graphical user interface.
Requirement 11: The users/stakeholders should be able to access the standard procedures for interacting or integrating their offerings with the ecosystem, based on their intention (e.g., offering provision, offering consumption, etc.), which needs prior development and/or configuration and cannot be done intuitively.
Requirement 12: The users who are offering providers should be able to publish the documentation of their offerings and the potential offering consumers should be able to access it.

Solution: The **'Portal'** component will be the unification point of distributed tools and platforms in the ecosystem. It allows the user to access to connected tools, base platforms, marketplaces, experiments and pilots through a unified interface. The portal will enable the registration process to the ecosystem. By using the portal, existing users can login to the platform, new users and visitors start the registration process and create an account to become a member of the platform.

4.2.4 Ecosystem Enablers: Marketplace

Challenge 7) Each digital platform's offerings are listed in its own marketplace, presented in a variety of formats and interfaces. The marketplaces use their own payment system, so the purchases made through the ecosystem cannot be tracked.

Requirement 13: The users should be able to see a coherent view of all the offerings of the marketplaces of connected platforms.
Requirement 14: The ecosystem should not only support redirecting users to the marketplace with the offering, but also provide tracking and tracing of purchases.

Solution: To address this challenge, the **'Integrated Marketplace'** provides access to the offerings of the different external marketplaces offered by the connected platforms. The Integrated marketplace gives users the option to filter and sort the offerings, modify the view, and access the selected offering on the external marketplace to continue with the purchase process. The purchase made by the user on the external marketplace will be traced and tracked by the '**Accountancy Service'**. The collected data logs are used to carry out a cashback mechanism which enables to charge the external marketplace by a commission charge or a referral fee where a user carries out a business transaction

4.2.5 Ecosystem Enablers: Matchmaking

Challenge 8) Single point of discovery: In addition to the challenge of finding the right offering from the best suited supplier in the right domain, interacting effectively with the partner, once found, is a long and difficult process.

Requirement 15: The users should be able to search for the participants (suppliers/service providers) and their offerings (products/services) across the connected platforms, by using custom, user-defined search filters.
Requirement 16: The user should be supported to perform negotiations and transactions with selected suppliers and service providers from different connected platforms

Solution: This challenge will be addressed by **Federated Search and Matchmaking** integrated into the Portal. The Federated Search and Matchmaking will allow the user to find the right partner and enable them to transact with efficiently and effectively by providing a federated search and recommendations of the participants (suppliers/ service providers) and their value-units(products/services, supporting the user to perform negotiations and transactions with the selected partner from the different connected platforms, enabling the user to find best supplier to fulfil a request for a service or product in a fully automated way.

5 EFPFInterop Reference Architecture

5.1 Introduction

The EFPFInterop ecosystem follows the micro-service architecture approach in which different functional modules implement individual functionalities that can be composed based on specific user needs. In order to implement this approach, all components in the EFPFInterop ecosystem are prescribed to implement and publish open interfaces, preferably REST interfaces, allowing the exchange of data and avoiding the lag-time introduced by interconnection buses.

The EFPFInterop Ecosystem Enablers are central to the federation of the various platforms and services and represent the most important features and functionalities for such a federation. Figure 4 presents an overview of the high-level architecture of the EFPFInterop Ecosystem Enablers and the ecosystem of federated platforms.

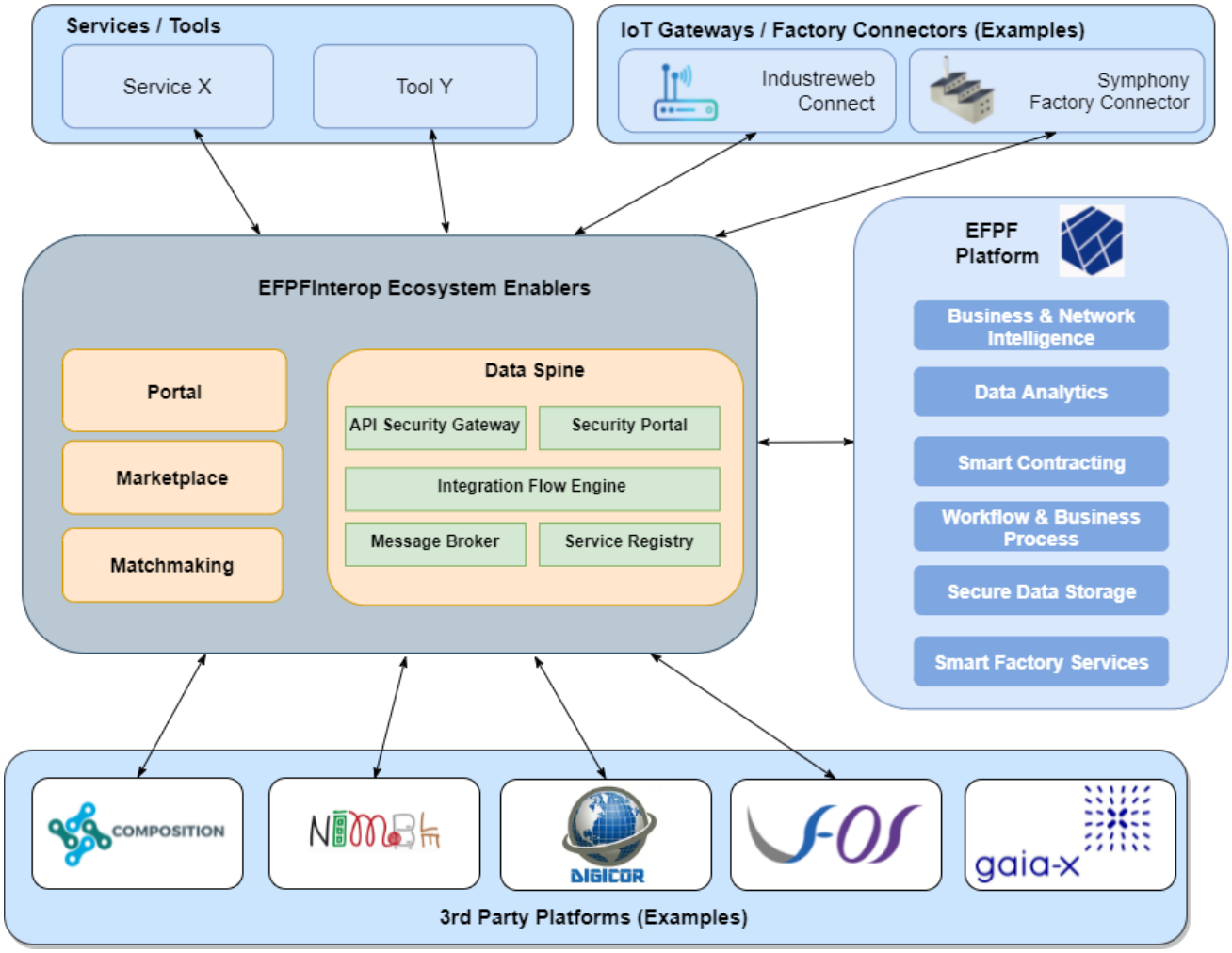


Figure 4 — EFPFInterop Ecosystem Enablers with components

The EFPFInterop ecosystem is based on a federation model, which consists of distributed platforms, tools and components provided by several partners. These individual components communicate through a central entity called 'Data Spine'. Thus, the EFPFInterop ecosystem as a whole follows Service-oriented architecture (SOA) style. The main elements in the EFPFInterop Ecosystem Enablers are:

- **Data Spine:** The Data Spine is the central entity or gluing mechanism in the EFPFInterop federation. The Data Spine provides the interoperability infrastructure that initially interlinks and establishes interoperability between 3rd Party Platforms like COMPOSITION, DIGICOR, NIMBLE and vf-OS (see [5] for more details). It adheres to common industry standards and follows a modular approach to enable the creation of a modular, flexible and extensible platform. Therefore, it can be easily extended to 'plug' new external platforms in and interlink them with the already connected platforms. Figure 4 also highlights the platform agnostic nature of the Data Spine i.e., it is evident from the high-level architecture that as far as interactions with the Data Spine are concerned, there is no distinction between any other platforms (any 3rd Party or EFPF Platform).

- **Portal**: The Portal is the unification point of EFPFInterop ecosystem and the entry point for the ecosystem users to the federated platform.

- **Marketplace:** The Marketplace connects with external marketplaces provided by the connected platforms through API interfaces. This allows the integrated marketplace framework to list the

products (from external marketplaces) and provide typical filter and sorting mechanisms, with the ecommerce features (e.g., product listing and checkout) provided by the external marketplace.

- **Matchmaking:** The Matchmaking component can be used by the ecosystem users to find the best suited suppliers and enable them to contact and transact with them efficiently and effectively.

Figure 4 depicts the architecture of the Data Spine showing a high-level conceptual view of the following core components that provide the expected functionality of the Data Spine:

- **The Integration Flow Engine** component of the Data Spine provides a platform to the system integrators, allowing them to create integration flows for interconnecting the different APIs and services.

- **The Service Registry** component allows the service providers to register their services in the Data Spine. The Service Registry provides a facility for the service consumers or system integrators to discover these services and retrieve their metadata information required to create the integration flows.

- **The Message Broker** component can be used for mediating the transfer of messages or data between asynchronous services communicating through the Data Spine.

- **The Security Portal** component of the Data Spine is responsible for providing a SSO facility across the EFPF ecosystem (see API Security Gateway). In addition, the EFS component enables data integrity, security analytics, trust and reputation mechanisms, definition of policies and governance enforcement

- **The API Security Gateway** component of the Data Spine (and a sub-functionality of EFS) acts as the policy enforcement point (PEP) for the Data Spine and the platforms communicating through it. It intercepts all the traffic to the Data Spine and invokes the security service for authentication and authorization decisions

Besides the listed 3rd party platforms the EFPF Platform can be seen as an example of functionalities that are not required as a basis for the federation of platforms but for added values services and functionalities that could be provided by several other platforms and service providers – which fosters competition and provides choices depending of the needs for the project to be realised. The EFPF Platform 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 market ready or reference implementations of the Smart Factory and Industry 4.0 tools from project partners. The collection of enhanced versions of such tools and services from the base or external platforms deployed together as micro-services would constitute the EFPF platform. These micro-services are made accessible through the EFPF Portal using the Single Sign-On (SSO) functionality.

The intended federation of a variety of platforms are interesting for the following potential stakeholders and we list potential offerings that could be provided by the stakeholders.

Software Offerings:

- Software as a Product (SaaP): Software tools that are sold/given to users as products and therefore, they generally intend to follow a one-time pricing/licensing model. E.g., Factory connectors or IoT Gateways such as TSMatch, Symphony HAL, Industreweb Collect or other products such as Google Chrome browser; Microsoft Office 2010, etc.

- Software as a Service (SaaS): Software tools/services centrally hosted on cloud, e.g., GitHub, Skype, Docker Hub, Microsoft Office 365, etc., which generally intend to follow the subscription-based or access-based (pay-as-you-go) pricing/licensing model. This includes the tools/services that make processed or value-added data available to other users over an API.

- Data API: Data provided over an API. E.g., sensor measurements available over an MQTT topic, weather data available over an HTTP API, etc.

- Dataset: Data provided in the form of a downloadable blob. E.g., historical stock data downloadable as a CSV file.

Stakeholders / User roles:

- Ecosystem Administrator (Admin): A user who has administrator-level access to the deployments and the APIs of the Ecosystem Enablers.

- Platform Provider: A user who is the provider of a digital platform

- SaaP Provider: A user who is the provider of an SaaP software.

- Tool/Service Provider: A user who is the provider of a SaaS software.

- Data API Provider: A user who is the provider of a data API.

- Dataset Provider: A user who is the provider of a dataset.

- System Integrator: A user who integrates a resource (i.e., a platform, a tool/service, or a data API, etc.) with the EFPFInterop ecosystem or creates composite applications using the existing resources.

5.2 Data Spine

The EFPFInterop ecosystem consists of distributed, heterogeneous digital platforms, tools, and components provided and hosted by independent entities. The technical features, including its interfaces, protocols, data formats, data models, identity and access management mechanisms, etc., differ significantly from each other, thus making a direct communication between EFPFInterop services problematic. As described in chapter 4.1, there could be many different ways to address this problem – one such way could be to design standardized APIs based on the identification of common standards and abstractions and ask Service Providers and Service Consumers to implement specific connectors/plugins, in order to align their proprietary APIs to these standard APIs and enable communication. However, such approaches are not desirable as they need significant modifications to the existing tools, services, systems, and platforms that need to communicate with each other, among other shortcomings. Hence, a novel solution enabling a communications layer, acting as a translator/adapter between these heterogeneous tools, services, systems, and platforms, and providing data handling, routing capabilities and API adaptation functionalities is needed. EFPF offers an interoperability enabler, the Data Spine, designed to address these challenges. The requirements identified in chapter 4.2 guide the design of the Data Spine.

The Data Spine is designed as the interoperability backbone of the EFPFInterop ecosystem that interlinks heterogeneous platforms, establishes interoperability between their services and enables the creation of cross-platform applications. The Data Spine is aimed at bridging the interoperability gaps between services at three different levels:

- **Protocol interoperability**: The Data Spine supports two communication patterns:

a) Synchronous Request-Response pattern

b) Asynchronous Publish/Subscribe pattern

The Data Spine employs an extensible mechanism to support new protocols. An implementation of the Data Spine should support standard application layer communication protocols that are widely used in the industry. For lower-level communication protocols and IoT networking technologies (e.g., BLE, Z-Wave, ZigBee, etc.), the Data Spine relies on IoT Gateways deployed at the edge.

- **Data model interoperability**: The Data Spine provides an interoperability mechanism to transform between the message formats, data structures and data models of different services.
- **Security interoperability**: The Data Spine enables single sign-on (SSO) mechanism in the ecosystem and makes it possible to access the resources of the connected platforms with a single set of credentials for each user.

Figure 5 illustrates the high-level architecture of the Data Spine. It consists of the following components:

- **The Security Portal** component is responsible for providing a SSO facility across the EFPFInterop ecosystem. It enables the users to access the resources, i.e., tools, services, and data, of the connected platforms with a single set of credentials.

- **The Integration Flow Engine** component of the Data Spine provides a platform to the users, allowing them to create dataflows or "integration flows" for interconnecting the APIs of different services in order to create composite applications.

- **The Message Broker** component is used for mediating the transfer of messages or data between asynchronous services communicating through the Data Spine.

- **The Service Registry** component allows the service providers to register their service endpoints and metadata such as API specifications. The Service Registry provides a facility for the service consumers or composite application developers to discover these services and retrieve their metadata information, which is required to create the integration flows.

- **The API Security Gateway** component acts as the policy enforcement point (PEP) for the APIs that are exposed by the integration flows created by users. It can also be used as a permalink reverse proxy endpoint for the services in the ecosystem whose direct endpoints can change.

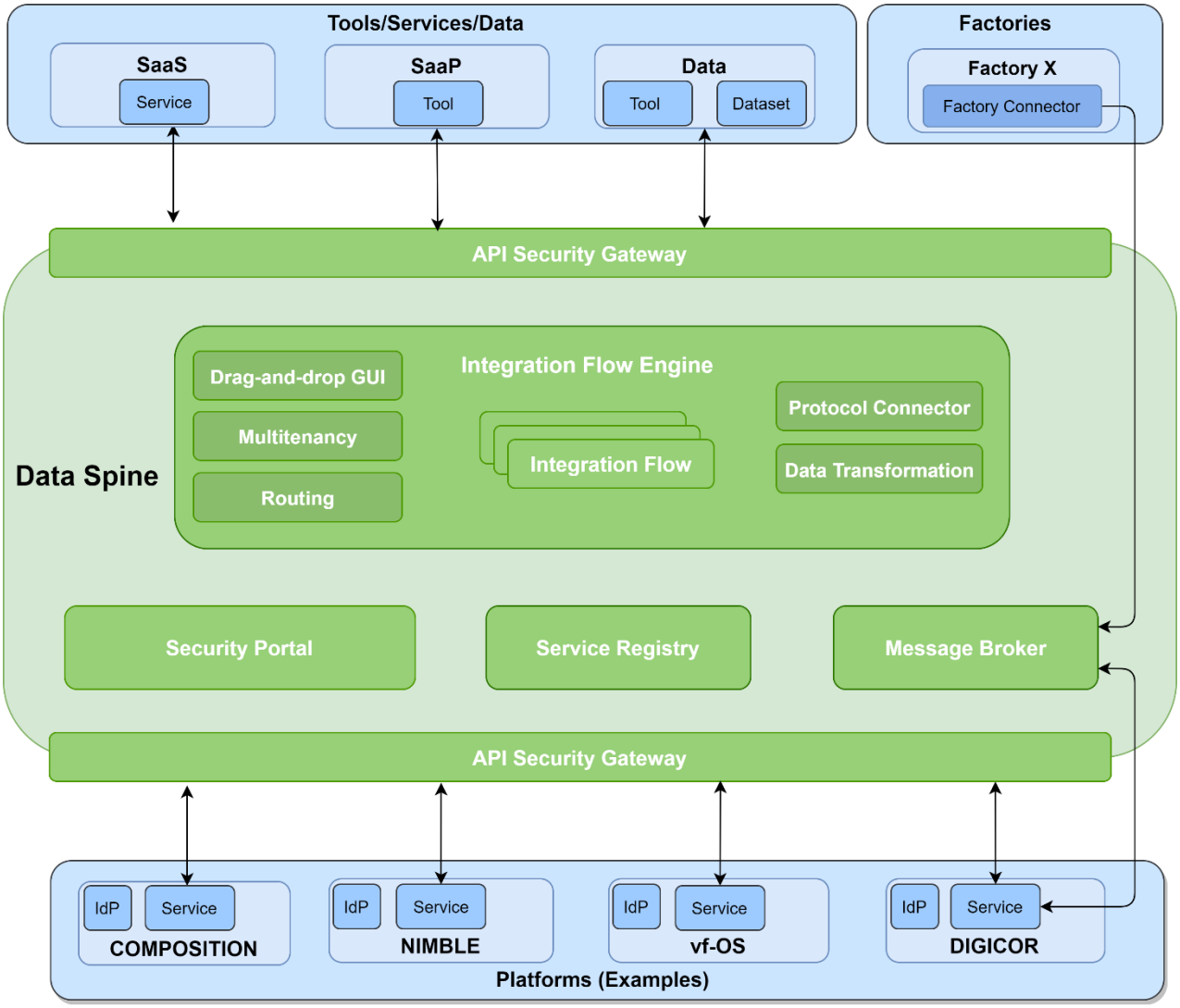


Figure 5 — High-level Architecture of the Data Spine

5.2.1 Components of the Data Spine

5.2.1.1 Security Portal

The EFPFInterop ecosystem consists of heterogeneous platforms owned by independent entities. Each of these platforms has its own Identity Provider (IdP). To enable collaboration and data exchange among the platforms, a user of one platform needs to access a service of another platform. As described in 4.2.2, an SSO functionality needs to be enabled across all platforms in the ecosystem. The objective of the Security Portal is to implement a federated identity mechanism that bridges such "security interoperability" gaps among the platforms in the EFPFInterop ecosystem.

The EFPFInterop ecosystem is envisioned to be an extensible platform ecosystem. With such a design requirement, one-to-one user federation mappings between platforms will result in a high number of login options for individual platforms (e.g., login to platform A, B, C, ..., N), which would require continuous updates to the authentication and authorization workflows for each platform in the ecosystem. Therefore, providing a completely distributed, federated solution with login options for all platforms in the Web portal of a single platform is not a scalable solution. Hence, the Security Portal is designed as a "distributed single point of trust" that federates the identity providers (IdPs) of the connected platforms in order to enable an SSO connection among them. This federation of the IdPs

enables the users to seamlessly access the resources, i.e., tools, services, data, with a single set of credentials.

Thus, the EFPFInterop Security Portal takes on the role of a central identity provider solution for the EFPF ecosystem and the Web portal of each platform provides an additional "Login with EFS" option to allow logging in with an EFPFInterop user account, as illustrated in Figure 6.

The EFPFInterop ecosystem can also be extended by adding new independent tools/services without their own identity provider. In such cases, the Security Portal acts as an identity and access management solution.

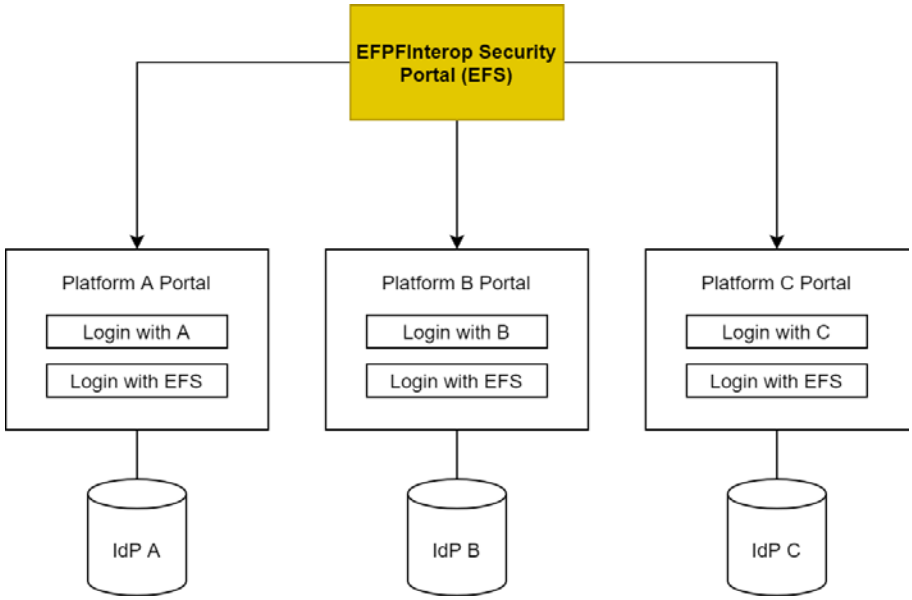


Figure 6 – An illustration of 'Login with EFS' option on the platforms' Web portals

5.2.1.2 Integration Flow Engine

Integration Flow Engine (IFE) component of the Data Spine aims to bridge the interoperability gaps at the protocol level and the data model level between the heterogeneous services communicating through the Data Spine, in order to create composite applications. To enable interoperability among the services on the fly, the IFE makes use of dataflows or "Integration Flows". The IFE is thus designed as a dataflow management system based on the concepts from flow-based programming. It provides functionalities such as connectivity, data routing, data transformation and system mediation.

The integration flows are designed and implemented as directed graphs that have 'processors' at their vertices and the edges represent the direction of the dataflow. The processors are of different types depending upon the functionality they provide: The processors of type 'Protocol Connector' address the issue of interlinking the services that use heterogeneous communication protocols, the processors of type 'Data Transformation Processor' provide the means for transforming between data models and message formats, etc. The edges that represent the flow of information support routing of data based on certain parameters. The IFE provides functionality for the lifecycle management of the integration flows.

The IFE supports the Request-Response as well as the Pub/Sub communication pattern. An instance of the Integration Flow Engine should have built-in Protocol Connectors for standard communication protocols that are widely used in the industry. The functionality of the IFE can be extended by writing and adding custom processors to the IFE instance. For example, support for a new protocol can be added by writing a new Protocol Connector and adding it to the IFE instance.

The IFE offers an intuitive, drag-and-drop style Web-based Graphical User Interface (GUI) to the service consumers or system integrators to create the integration flows which is based on the concepts from visual programming. This interface, the use of integration flows, and the built-in building blocks for connecting and aligning the APIs enable the users to create composite applications quickly, easily, and intuitively.

The IFE and its GUI support a fine-grained access control. It is possible to define access policies to allow or restrict visibility of and/or access to certain GUI elements. In addition, the IFE supports multitenancy and grouping of integration flows into "Development Spaces". The access to the development spaces at view or modify levels can be restricted per user or user-group. This enables the ecosystem administrator to assign development spaces for companies where the system integrators from those companies can create their own integration flows. This helps shifting the data transformation burden across the Service Consumers making the architecture scalable and extensible. This also enables collaboration among the users who create the integration flows. Figure 7 illustrates the process for the creation of an integration flow using the drag-and-drop style GUI of the IFE. The process steps are described below:

a) Development space: The ecosystem administrator allots a development space for the company of the Service Consumer 'consumer1'. consumer1 logs in to the GUI of the IFE and navigates to his/her company's development space.
b) Reusable processors: The built-in reusable processors are loaded by the IFE and are displayed to the users on the GUI.
c) Creation of the integration flow: consumer1 drags and drops the processors in the development space, configures them, and connects them together to create an integration flow.
d) Functionality offered by the integration flow: The integration flow in this example consumes the API endpoint EP1-a provided by the Service Provider 'provider1', performs data model transformation, and makes the transformed data available over EP1-b, that is the "interoperability proxy" endpoint for EP1-a.

Thus, the IFE enables interoperability and creation of applications in an easy and intuitive manner.

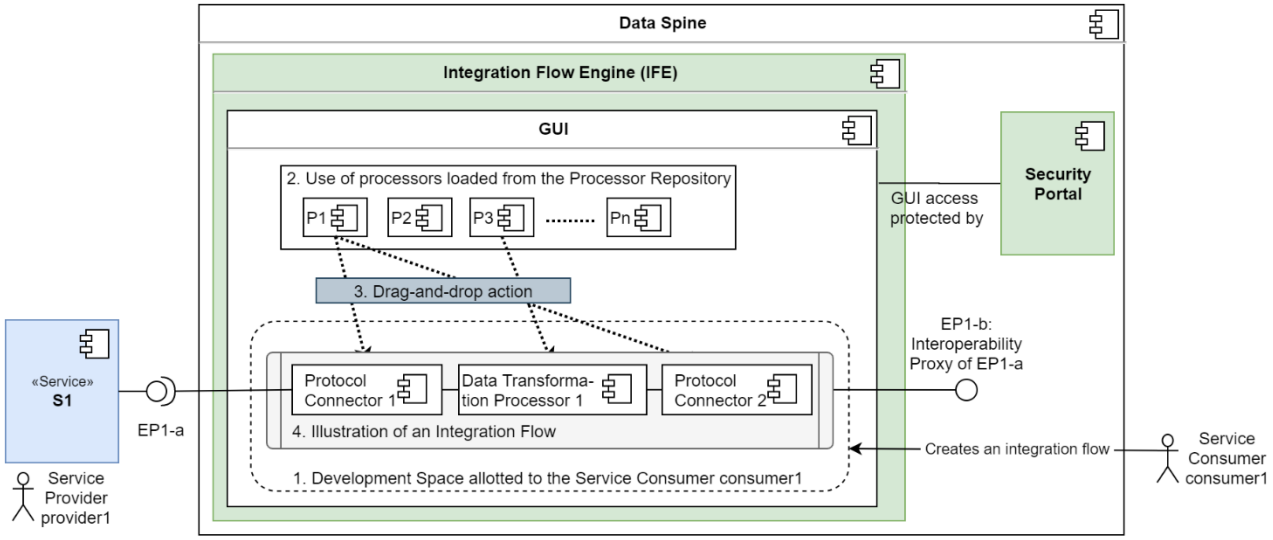


Figure 7 — An illustration of the creation of an integration flow

Furthermore, the IFE supports standard authentication and authorization protocols such as OpenID Connect (OIDC) and OAuth2.0 to secure access to its GUI using a pluggable identity provider. This enables connecting the IFE with the Security Portal in order to enable user authentication. Finally, to ensure high availability, throughput, and low latency, an instance of the IFE should be scalable and capable of operating in a clustered fashion.

5.2.1.3 Message Broker

In the manufacturing domain, the Pub/Sub communication pattern is widely used. The shop-floor data from sensors is typically collected by a Factory Connector or an IoT Gateway and is made available to other services through a message-oriented middleware. The Data Spine Message Broker supports the Pub/Sub communication pattern. In addition, the Message Broker supports multitenancy and fine-grained access control. It provides interfaces for topic, user, and policy management.

An implementation of the Message Broker should support messaging protocols such as MQTT, AMQP, etc. that are widely used in the industry. The Message Broker can be extended to add support for new protocols via a plugin mechanism.

5.2.1.4 Service Registry

In an interconnected platform ecosystem such as EFPFInterop, the services of different platforms need to be composed together to achieve common objectives. For this purpose, the service consumers should be able to discover the available services, retrieve their API metadata, and consume them without the active involvement of the service providers. The Data Spine Service Registry provides the following mechanisms to fulfil these requirements:

- Registration and lifecycle management of service/API metadata for synchronous (Request-Response) as well as asynchronous (Pub/Sub) services in a uniform manner
- Discovery, lookup, and filtering of services
- Use of standard API specifications (specs) to capture service metadata to ensure the completeness of and uniformity across the API descriptions.

Figure 8 shows the abstract class diagram for the Service Registry that illustrates composition relationship between its classes. The notation '0..*' in the diagram denotes 'zero or more instances' of the concerned entity. As illustrated in the diagram, the Catalog of the Service Registry can have zero or more services. Each Service has zero or more APIs, and each API has exactly one API specification (Spec).

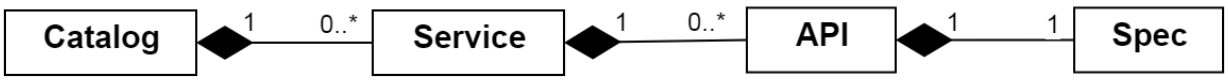


Figure 8 — Abstract Class Diagram for the Service Registry

Figure 9 further shows the abstract schema for the Service object. The API Spec is obtained from an API Spec document. This API Spec document needs to conform to one of the following standards in order to ensure uniformity across and completeness of API specifications:

- For synchronous (Request-Response) services: OpenAPI/Swagger Spec[1]
- For asynchronous (Pub/Sub) services: AsyncAPI Spec[2]

[1] OpenAPI/Swagger Spec: https://swagger.io/docs/specification/about/

[2] AsyncAPI Spec: https://www.asyncapi.com/

Thus, this design makes the schema capable of managing metadata for synchronous (Request-Response) as well as asynchronous (Pub/Sub) type of services. All the technical metadata for the APIs of services that is needed for consuming the API can be obtained from these API Spec documents.

The 'type' could be used to categorise the services by giving a type to them based on the functionality they offer. In addition, any additional functional metadata related to the services, or the individual APIs can be stored in the respective 'meta' objects as key-value pairs. Thus, the basic schema can be extended to include additional metadata for the entire service or for a specific API.

```
{
    "id": "<unique id - custom or uuid>",
    "type": "<string>",
    "meta": {},
    "apis": [{
            "id": "<string>",
            "url": "<base url of the API>",
            "spec": {
                "mediaType": "<mediaType type of the API Spec document>",
                "url": "<url to API Spec document>"
            },
            "meta": {}
        }
    ],
    "doc": "<URL to service documentation>",
    "createdAt": "2022-06-06T15:46:36.793Z",
    "updatedAt": "2022-06-07T15:46:36.793Z"
}
```

Figure 9 — Abstract Service Description Schema of the Service Registry

5.2.1.5 API Security Gateway

The API Security Gateway (ASG) acts as the Policy Enforcement Point (PEP) for the synchronous HTTP-based APIs exposed by the integration flows created by users in the IFE. It connects to the Security Portal for making authentication and authorization decisions. The ASG can also provide permanent reverse proxy endpoints for the services that do not have fixed/permanent API endpoints.

Moreover, the ASG can be used to secure communication to the HTTP-based APIs of other components deployed in the same internal network, such as the Service Registry. Finally, the ASG automates the process of creation of secure proxy endpoints for the services registered in the Service Registry.

5.2.2 Data Spine Components' Interaction

Figure 10 illustrates the architecture of the Data Spine and interaction among its components. The access to the GUI of the IFE and its elements is protected by the Security Portal. The ASG acts as the Policy Enforcement Point and relies on the Security Portal to make the access control related decisions. The ASG secures the REST API of the Service Registry and offers a secure proxy endpoint to access it. The API Security Gateway is configured to check the Service Registry for new service registrations periodically in order to automatically create secure proxy endpoints for them. The Service Registry publishes service status announcement related messages to the Message Broker.

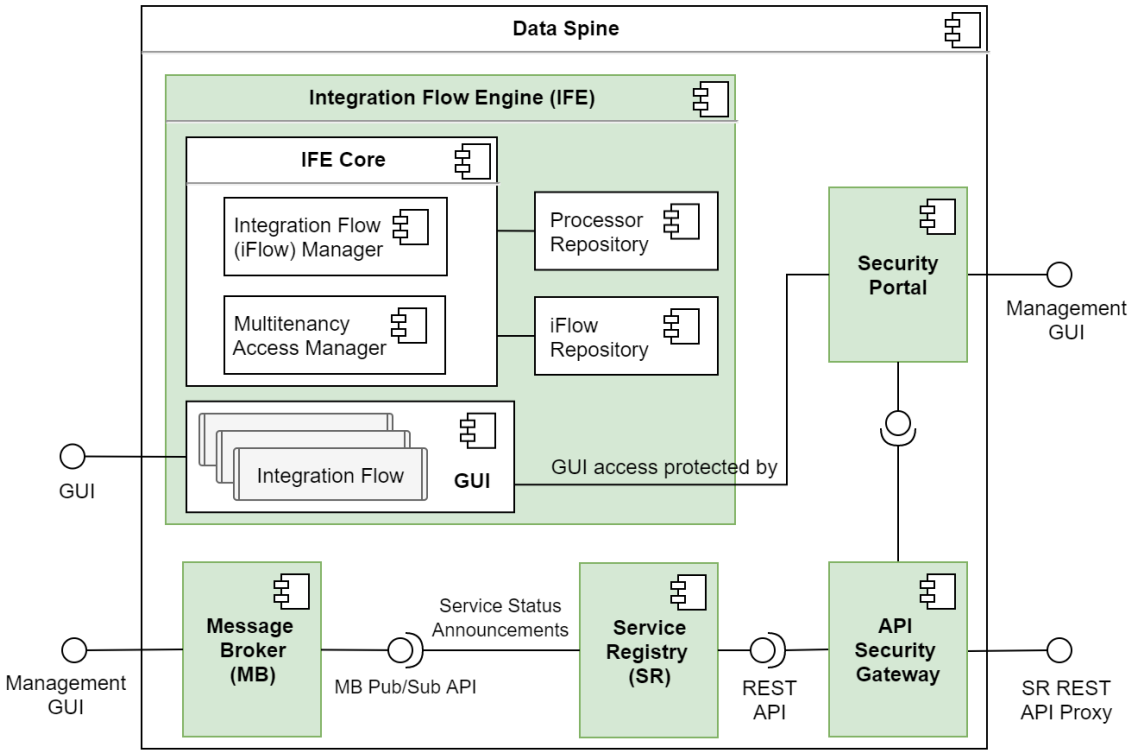


Figure 10 — Architecture of the Data Spine

Figure 11 illustrates how the interoperability proxy endpoint EP1-b exposed by the integration flow is secured by the ASG. When the Service Consumer, consumer1 registers EP1-b to the Service Registry, the ASG automatically creates a security proxy endpoint EP1-c for EP1-b. consumer1 can then invoke EP1-c in through S2 with an access token obtained from the Security Portal.

In this way, the components of the Data Spine work together to enable integration of and communication between the services of different platforms.

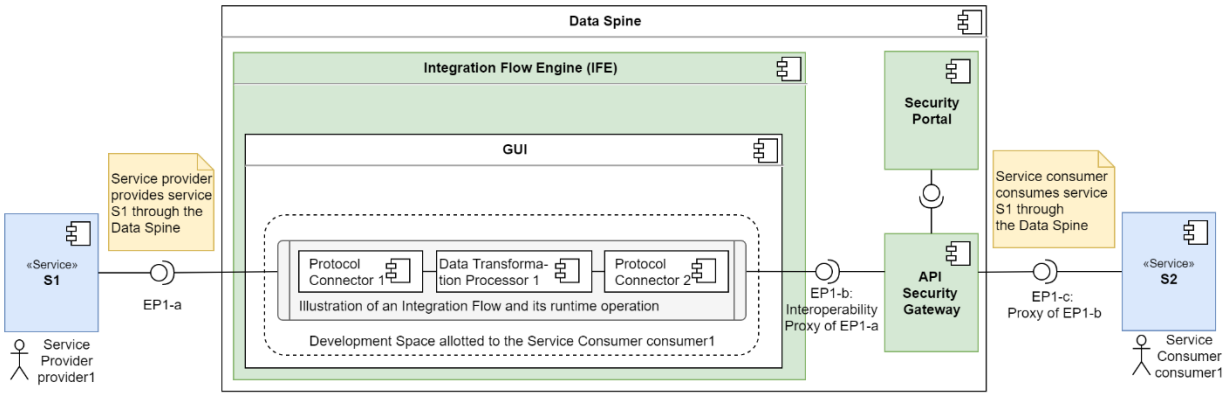


Figure 11 — Illustration of an integration flow and its runtime operation

In summary, the Data Spine provides the following functionalities:

- Authentication, authorization and SSO;
- Service/API metadata lifecycle management and discovery;
- Infrastructure and tooling for protocol connection, data transformation, routing, and system mediation;
- Multitenant, Web-based, drag-and-drop style GUI for an easy and intuitive creation of applications with minimal coding effort;
- Message brokering.

5.2.3 Creation of Composite Applications

5.2.3.1 General

Together with enabling cross-platform interoperability, the Data Spine also enables the creation of cross-platforms in an easy and intuitive manner. In order to create composite applications using the services of different platforms, those platforms need to be integrated with the EFPFInterop ecosystem beforehand. The prerequisite for the creation of composite applications is the federation of the platforms' identity providers with the Security Portal to enable SSO. It is also possible to integrate the services that do not belong to any platform, or in other words, do not have an associated identity provider, in which case the Security Portal becomes their default identity provider during their integration. Once this integration is complete, the services can be composed together using the Data Spine to create applications.

The design-time steps for the creation of composite applications and dataflow through the Data Spine at runtime for both synchronous (Request-Response) and asynchronous (Pub/Sub) communication patterns are described below.

5.2.3.2 Synchronous (Request-Response) Communication

Figure 12 and Figure 13 show how provider1's service 'PS1' and consumer1's service 'CS1' interact with the components of the Data Spine, in order to provide and consume services respectively. The actions to be performed for service provision and consumption through the Data Spine are described below.

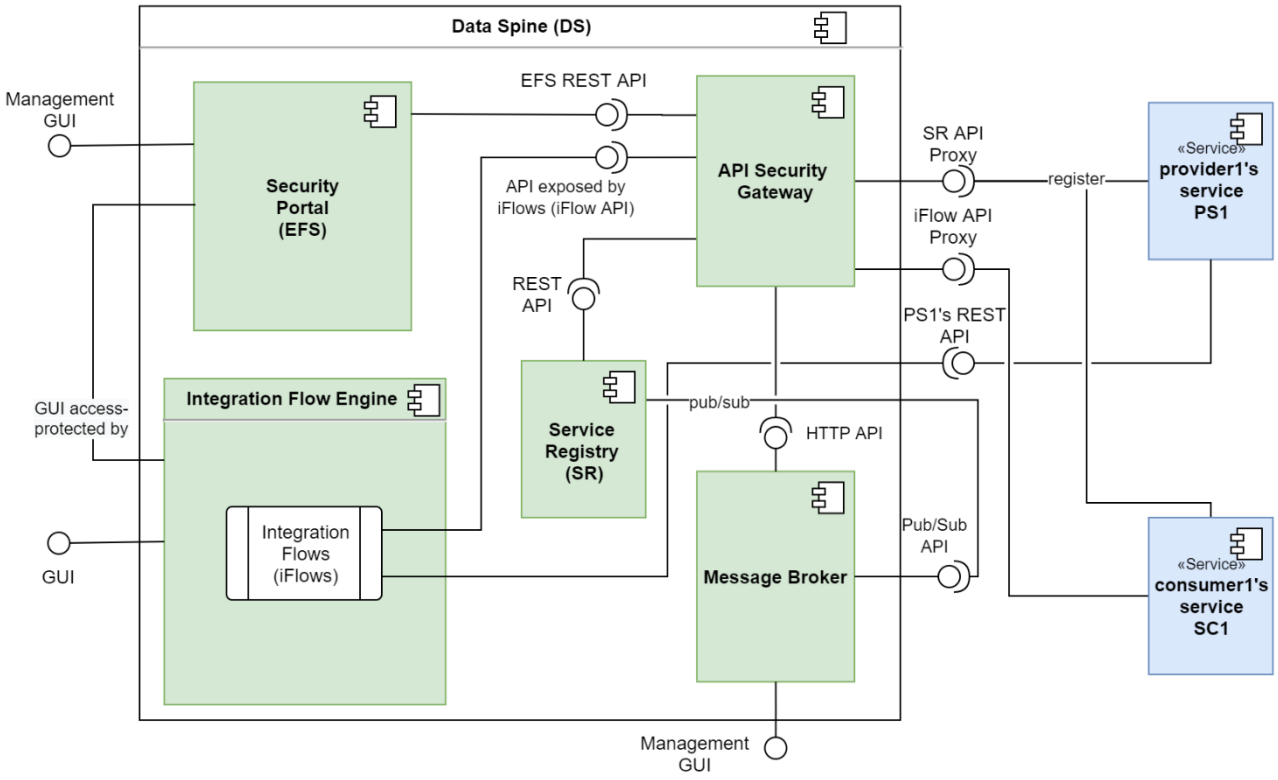


Figure 12 — Synchronous services' integration through the Data Spine.

Prerequisites:

- The service provider 'provider1' and service consumer 'consumer1' both have user accounts for the EFPFInterop Security Portal.
- provider1 and consumer1 have the necessary permissions required to access the Service Registry.

Design-time service integration activities:

a) Service Registration: provider1 registers his/her service 'PS1' to the Service Registry with an appropriate service 'type' (e.g., 'marketplace-service'). Here, PS1's REST API endpoint, EP1 is already secured by its platform's identity and access management service (not shown in the figure for simplicity).
b) Service Lookup and Metadata Retrieval: consumer1 uses Service Registry's filtering API to find PS1, decides to consume it, and retrieves its technical metadata including its API spec from the Service Registry.
c) Access Configuration: consumer1 requests for and acquires the necessary access permissions to invoke EP1.
d) Access Configuration: consumer1 requests for and acquires a development space in the IFE to create integration flows.
e) Integration Flow Creation: consumer1 creates an integration flow using the GUI of the IFE that invokes EP1, performs data transformation to align request/response payload to its own data model/format, and finally creates and exposes an "interoperability-proxy" endpoint EP1-C for EP1.
f) Service/API Registration: consumer1 registers this new EP1-C API endpoint to the Service Registry.
g) Creation of a Secure Proxy API: ASG automatically creates a secure proxy API endpoint EP1-C_P for EP1-C.

h) Access Configuration: consumer1 requests for and acquires the necessary access permissions to invoke EP1-C_P.
i) Integration Complete: provider1's service PS1 and consumer1's service CS1 are now integrated through the Data Spine and CS1 can start invoking PS1 and obtain a response in the format required by it as illustrated in Figure 13.

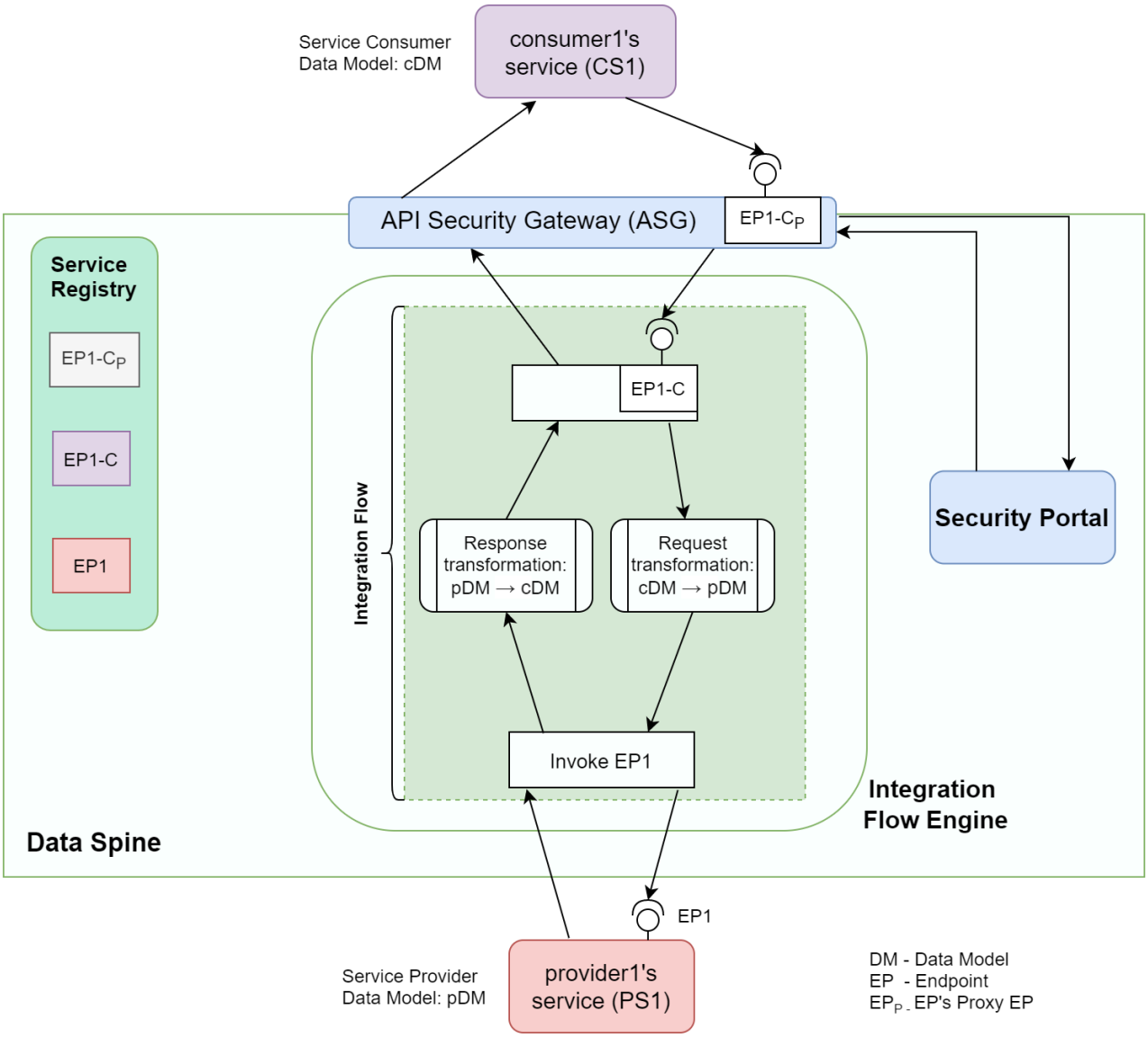


Figure 13 — Example of synchronous communication dataflow through the Data Spine.

5.2.3.3 Asynchronous (Pub/Sub) Communication

Figure 14 and Figure 15 show how publisher1's service 'pub1' and subscriber1's service 'sub1' interact with the components of the Data Spine, to provide and consume services, respectively. The actions to be performed for service provision and consumption through the Data Spine are described below.

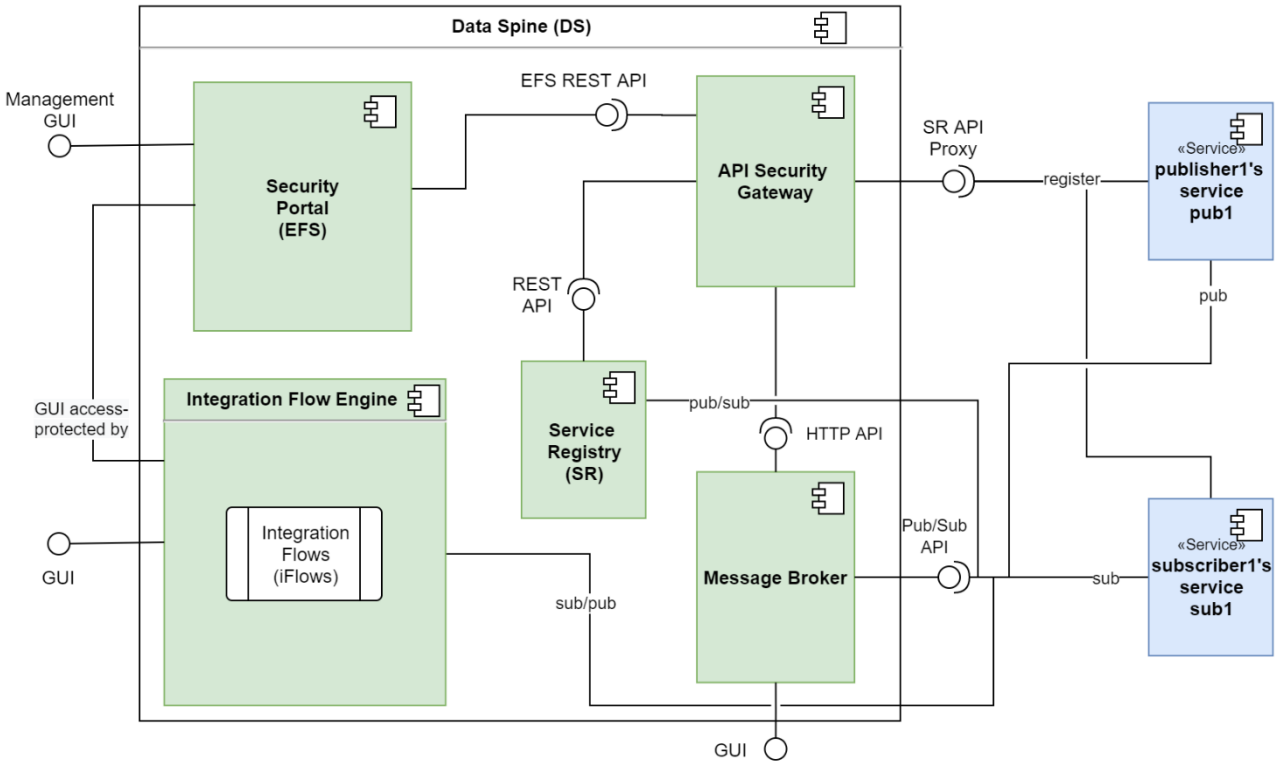


Figure 14 — Asynchronous services' integration through Data Spine.

Prerequisites:

- The service provider 'provider1' and service consumer 'consumer1' both have user accounts for the EFPFInterop Security Portal.

- provider1 and consumer1 have the necessary permissions required to access the service registry.

Design-time service integration activities:

a) Access Configuration: publisher1 requests for and acquires the necessary access permissions to publish to the Message Broker over topic 'p1/topic1'.
b) Publisher Configuration: publisher1 configures his/her service 'pub1' to publish to the Message Broker over the topic 'p1/topic1'.
c) Service Registration: publisher1 registers pub1 that consists of this Pub/Sub API containing its publication information to the Service Registry.
d) Service Lookup and Metadata Retrieval: subscriber1 uses Service Registry's filtering API to find pub1, decides to subscribe to pub1's topic 'p1/topic1' and retrieves the technical metadata for pub1 including its API spec from the Service Registry.
e) Access Configuration: subscriber1 requests for and acquires the necessary access permissions to subscribe to p1/topic1 and to publish back to the Message Broker over a new topic 's1/topic1'.
f) Access Configuration: subscriber1 requests for and acquires a development space in the IFE to create integration flows.
g) Integration Flow Creation: subscriber1 creates an integration flow using the GUI of the IFE to subscribe to p1/topic1, perform data model transformation to align the message payload to its own data model/format, and finally to publish the resulting data to the Message Broker over the topic s1/topic1.

h) Service Registration: subscriber1 registers his/her service with the APIs containing its subscription and publication information to the Service Registry.
i) Integration Complete: publisher1's service pub1 and consumer1's service sub1 are now integrated through the Data Spine and, sub1 can subscribe to the topic s1/topic1 and obtain data in the format required by it as illustrated in Figure 15.

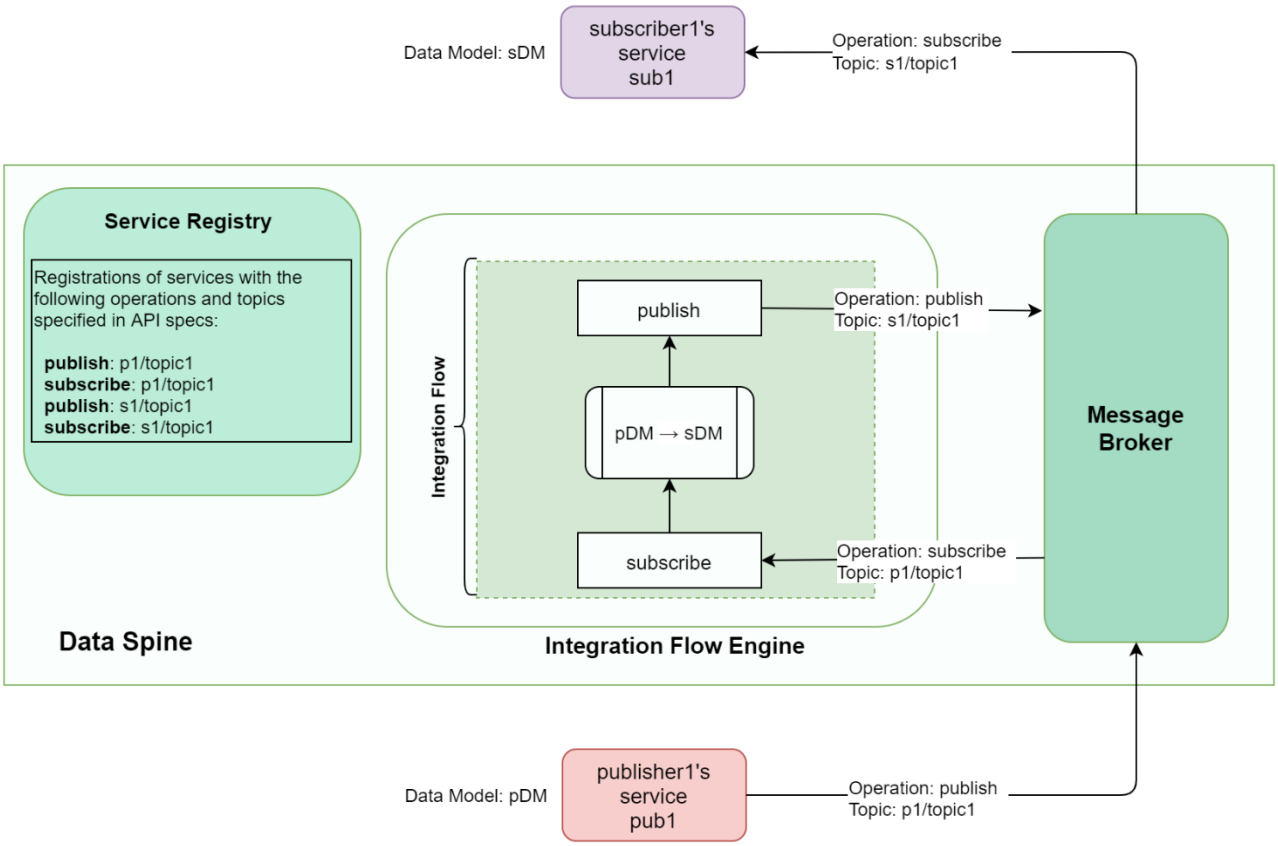


Figure 15 — Example of asynchronous communication dataflow through the Data Spine.

5.3 Marketplace

5.3.1 Integrated Marketplace Framework

One of the core requirements of the EFPF platform as a federated ecosystem is to bring together different stockholders into a level playing field where they can perform collaborative activities and develop relationships such as those between providers and suppliers of products and services. In this respect, the EFPF integrated marketplace framework is able to bring different types of offerings to the users in the EFPF ecosystem and provides a unified interface that allows users to search for different types of offerings through an intuitive and unified interface – available through the Portal component. The marketplace provides filtering and sorting and easy access to the product pages on the external marketplaces.

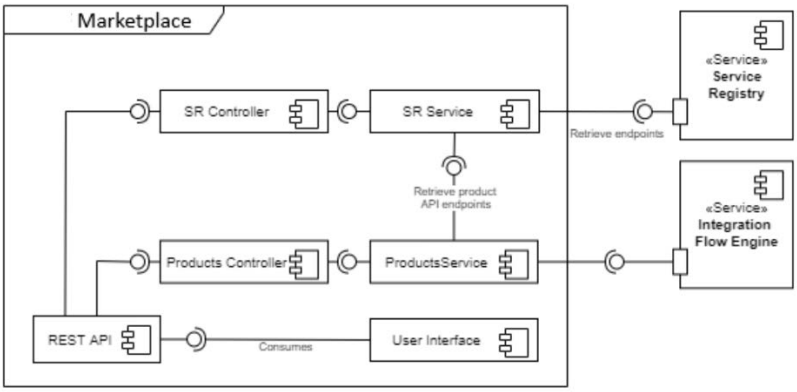


Figure 16 — Architecture of the Integrated Marketplace

By default, the Marketplace Framework lists all available items sorted by the name attribute. Additionally, the user can sort the products based on the item category, which is provided by the source marketplace. The user has also the option to negate the sorting.

The user can also use the provided filters, which allows to filter the list based on the source marketplaces and the categories. Both filters can be reset to their default values.

Each item provides basic information, which are the name, categories, item image and an external link, which leads the user to the detail page of the source marketplace.

5.3.2 Automated Agent-based Marketplace and Online Bidding

The automated Agent-Based Marketplace aims to provide mechanisms that enable the automated negotiations within the participants. In this marketplace, each company sets up an agent which represents the company in an Online Bidding Process in which the agents negotiate for specific goods. This provides automation of existing manual procedures and reduce time costs for traditional negotiations carried out by phone calls and email. The Online Bidding Process component is composed of three main components:

- Agent ecosystem
- Matchmaker
- User Interface

The Agent ecosystem is composed of a set of agents built to communicate with each other and with other platform's components to perform an automated negotiation. To achieve this an Agent includes modules to support different communication protocols using their own communication language called CXL (Composition eXchange Language) derived from the FIPA ACL communication standard. Agents are differentiated by the following roles:

- Requester: Represents a company who is requesting for a service or/and a related good
- Supplier: Represents a company who provides a service

The Matchmaker component provides a full semantic framework for the agents with CRUD operations for agents/companies and two type of matchmaking functionalities. It matches requesters with suppliers for a service/good and in a second level matches the request with the best available offer (coming from

supplier agents' bids) based on different evaluation criteria such as price, payment and delivery methods, reliability etc.

The UI of the bidding process is web-based and enables a user to carry out the following functions:

- register/set up and agent;
- initialize a bidding process in order to request for a service/good (requester);
- add information and priorities for the request (requester);
- online monitoring of the process for both suppliers and requester;
- opportunity for supplier to bid for a request;
- suggestion for best offer and details of all submitted offers in the case the user want;
- to ignore system suggestion and evaluate by himself (requester);
- visual notification for win/lose in bidding process (supplier);
- bids history available to user (requester).

The Online Bidding Process is a stand-alone component that is also made of agents representing companies to enable them to perform automated negotiation following predefined protocols and logics

5.3.3 Accountancy Service

The Accountancy Service is an integral part of the marketplace framework and provides insights into user's interaction with federated platform as well as its connected marketplaces. The Accountancy Service also monitors and provides a cashback mechanism to process all the transactions that a user makes with each external marketplace and calculates a corresponding cashback amount.

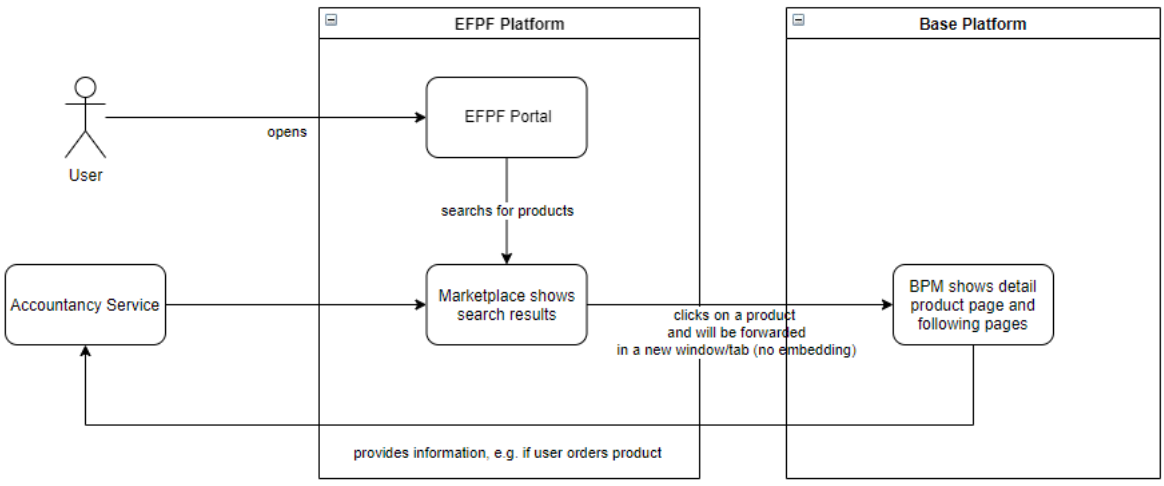


Figure 17 — Accountancy Service Cashback Mechanism

This component keeps track of all the trackable user actions in which action items are listed in 'subject, verb, object' manner and these actions include users' basic interactions with various parts of the federated Platform such as login, register, inviting other users as well as payments realized on external marketplaces if the user has initiated his/her journey from Integrated Marketplace. In this way, when a

user performs a certain action on either the Portal or a connected marketplace, corresponding information is sent to Accountancy Service to be persisted so that it can later be visualized to extract valuable information.

Based on the collected information, customizable dashboards are also needed for better and easier tracking of user interactions and currently, there 4 different dashboards, each focusing on different types of user interactions:

- User Activities: Visualizes simple user actions such as login and register
- Platform Engagement: Displays base platform visits and tool/service usages and tracks the frequency of these usages
- Marketplace Usage: Visualizes marketplaces usages in terms of most frequently used search keywords, queried platforms, and their distribution
- Payments: Displays all payments realized on integrated marketplaces as well as the corresponding cashback (commission) amounts calculated for each transaction

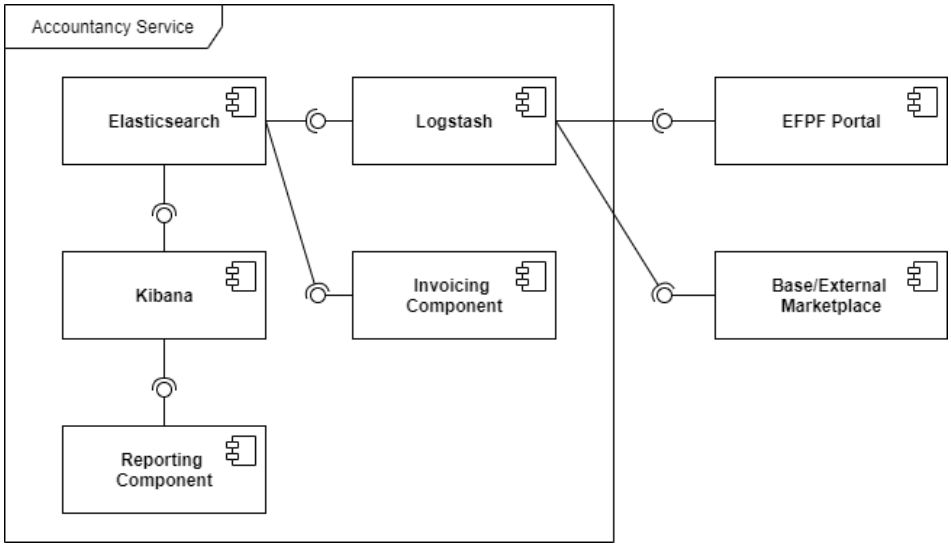


Figure 18 — Architecture of the Accountancy Service

5.4 Portal

The Portal is the unification point of the interconnected platform and the entry point of the user to the ecosystem.

The portal will enable the registration and management of users on the platform, the integration of new tools in the EFPFInterop platform and the utilisation/instantiation of integrated tools and services by ecosystem users. In this respect, the portal component will have three planes, a user facing pane, a resource facing plane and an administrative plane. The administrative pane will provide real-time monitoring functionality by actively and passively providing the infrastructure health and resource utilisation data for platform management and QoS auditing. In addition, the portal will integrate the holistic security, privacy and user management mechanisms (provided by the Security Portal – Data Spine) to realise single sign-on, user rights management and privacy of information exchanges between the platform, tools and the users.

Through the portal, users have access to all tools in one place, such as the Integrated Marketplace, the Matchmaking component, the pub sub security component, etc.

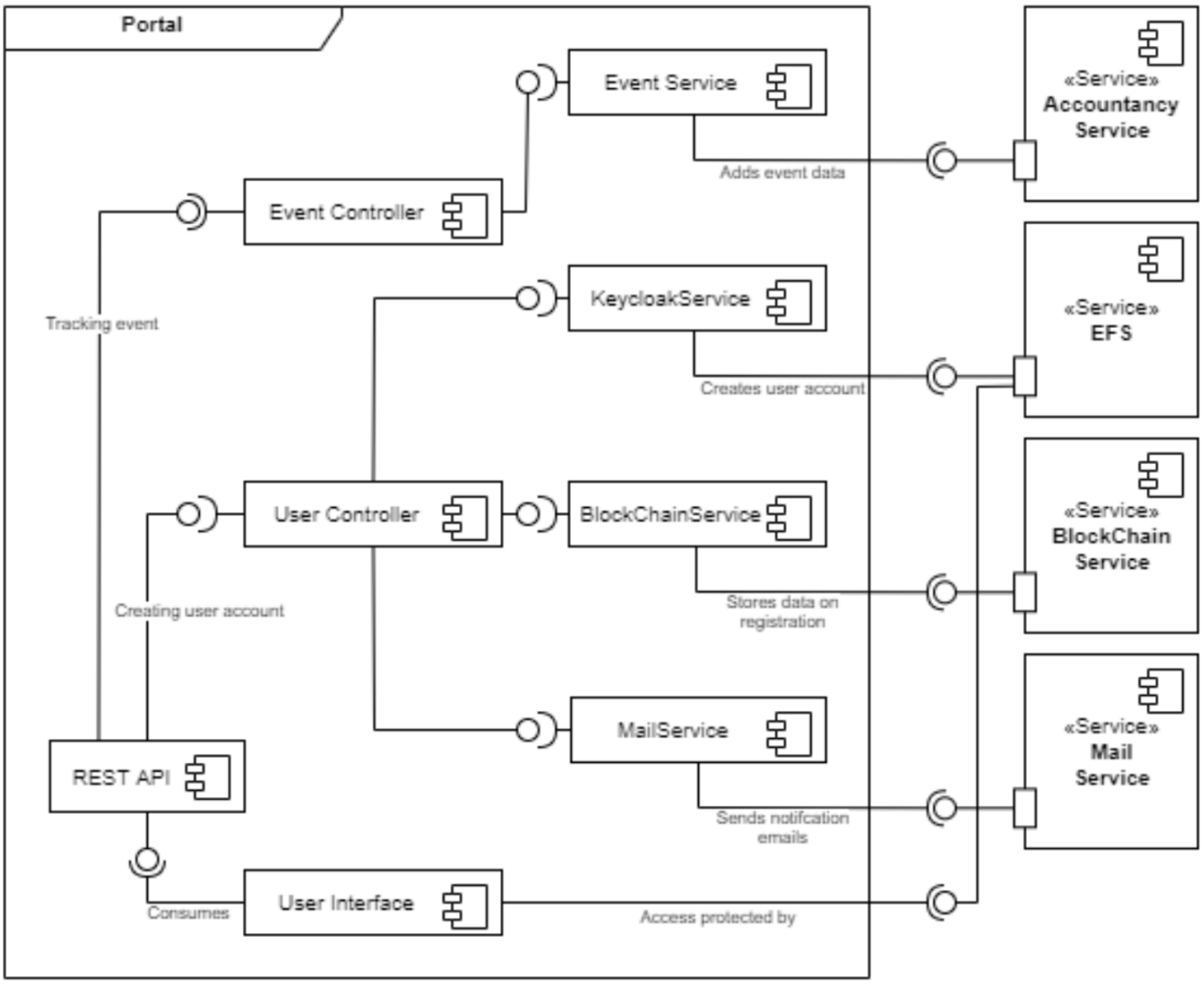


Figure 19 — Architecture of the Portal

5.5 Matchmaking

The matchmaking service supports stakeholders to effectively perform transactions over the EFPF platform and find the best suppliers/service providers. The matchmaking service is enabled through federated search and recommendation services that are designed for the EFPF ecosystem.

5.5.1 Matchmaking components overview

The Matchmaking in EFPF is achieved using the following steps;

- Federated search of participants (suppliers/service providers) and their features (products/services)
- Recommendations of suppliers/service providers and products/services
- Control of negotiations and transactions with selected suppliers and service providers from different base platforms

- Enabling users to find the best supplier to fulfil a request for a service or product in an automated way (using automated agents).

5.5.1.1 Federated Search Service

Federated Search enables simultaneous search of multiple data sources using a single query and search interface. The service enables searching for partners and/ or products and services, across different connected platforms and using different criteria, e.g. capabilities of partners, their geographic locations and acquired feedback and online rankings. The user is also able to search for products and services based on product/service-related criteria.

The Federated Search approach is implemented using the index-time merge architecture, which reduces the risk on relying on availability of search indices in the connected platforms, in comparison to search-time merge and hybrid architectures. The index-time merge search requires content from connected platforms to be acquired into a central index at the ecosystem, in order to enable platform level search for products/services and partners/companies across the base platforms in the EFPF ecosystem. It is also used to implement traditional enterprise search systems, in which information can be retrieved across heterogeneous data sources in an enterprise. Figure 20 depicts the index-time merge architecture that supports the EFPF federated search.

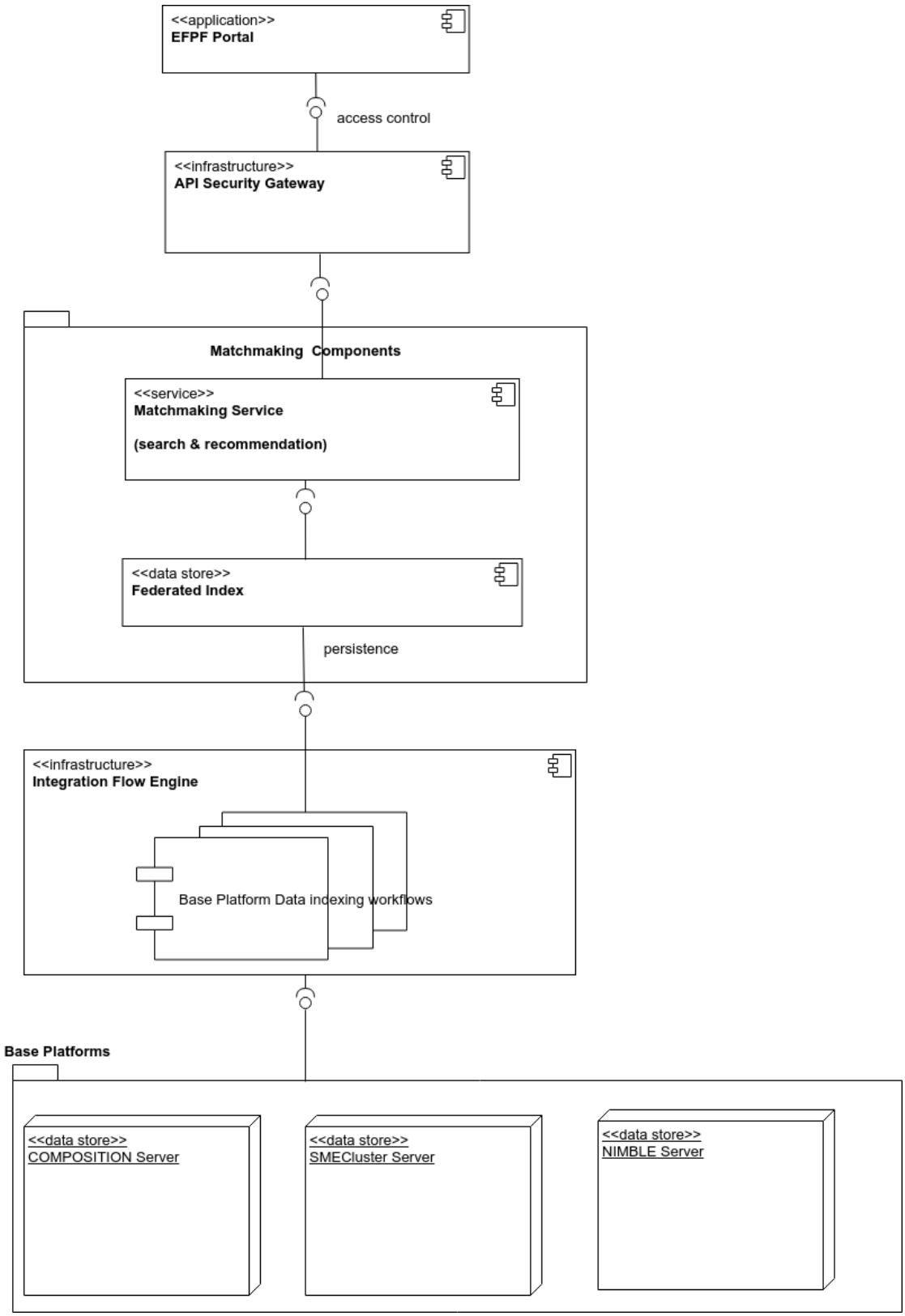


Figure 20 — Architecture of the Index-Time Merge Federated Search

The major advantages of the index-time merge architecture, as shown in Figure 20, are as follows:

- Through acquiring all data into a central index, sophisticated query enhancement and relevancy algorithms can be applied, providing the user with excellent search results.

- The selected search architectural approach allows for flexibility in the implementation of the recommendation and matchmaking engine.

- The indexed data and ML algorithms can be used to provide product/services/partners recommendation.

5.5.1.2 Recommendation service in EFPF

The results of the Federated Search service are collected to further support federated Recommendation service. Both Federated Search and Recommendation services support advanced matchmaking and agile network creation mechanisms for business transactions, across the digital platform ecosystem. The Federated Search supports queries for products, services and business partners. The search criteria and the results are collected to support the recommendation processes, based on different techniques of information pattern matching, e.g., information retrieval and similarity matching techniques, which are both based on Machine Learning (ML) and data analysis. After the recommendation algorithm has identified the most suitable products, services and/or partners, the user evaluates the results based on selected indicators, e.g., cost, reliability, quality, etc. In the final step, the user makes decisions and initiates a business transaction.

The main data flow in the relevant matchmaking components is depicted in Figure 21.

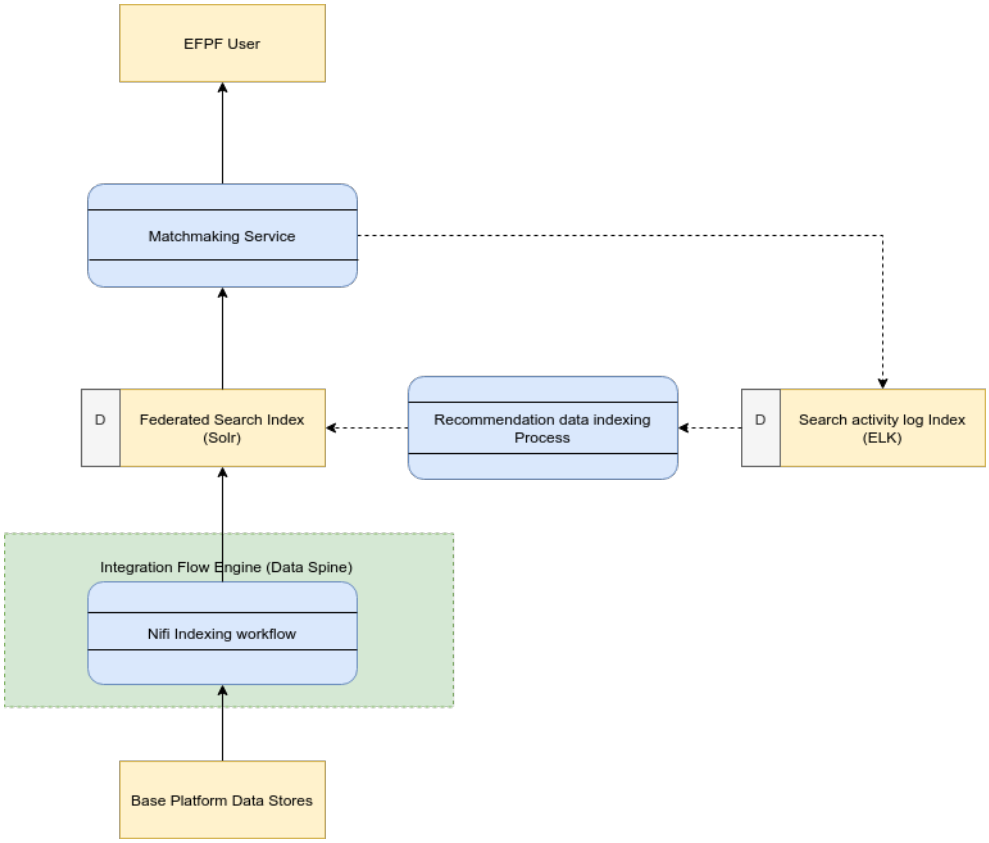


Figure 21 — Data Flow in Matchmaking Components

The main data flow is the one coming from base platform data stores (most base platforms expose their data stores via an API) to the federated index in matchmaking, during the indexing processes that happens on the integration flow engine (Data Spine). The data ingestion process is implemented as a set

of Apache NiFi workflows. This data flow is triggered periodically to retrieve the latest data from the base platform. The data retrieval schedule is configured as part of the Apache NiFi configurations.

5.5.2 Matchmaking Implementation

The Federated Search is developed as a Spring Boot microservice. It consists of the Matchmaking Ontology and an API for accessing the indexed data. It hides the complexity using a full featured search engine (Apache SOLR[3]) and provides simplified access to the indexed data, supporting full text search and faceted search over the indexed data collections. Apache Solr is a scalable and fault tolerant search platform that provides distributed indexing, replication and load-balanced querying, automated failover and recovery, centralized configuration and more.

The Matchmaking Ontology in EFPF is the foundation of the SOLR data model and is inspired by the SKOS[4] (Simple Knowledge Organization System) Ontology. SKOS is a common data model for sharing and linking knowledge systems using the Semantic Web technologies. In short, SKOS specifies the notion of concepts which might be connected by means of broader and narrower relationships. Concepts might be grouped together in concept schemes and allow for multilingual naming with preferred, (multiple) alternate and hidden names.

Apache SOLR allows for storing arbitrary documents with flexible data models. However, to support the matchmaking and federated search processes, the EFPF Matchmaking Ontology has been defined (see Figure 22 below).

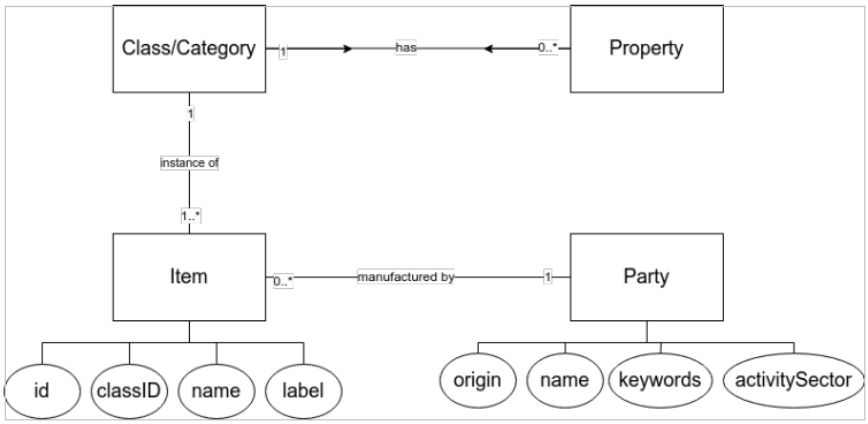


Figure 22 — Matchmaking Ontology (simplified view)

Every data item of interest is categorized using arbitrary concepts or treated as an instance of a certain class. The Concept class specifies dedicated attributes which are commonly present. This is reflected in the Matchmaking Ontology with the `Class/Category` and the `Property` collections. As outlined in Figure 22, each indexed product (`Item`) might be annotated as an instance of a predefined `Class/Category`. The `Class/Category` however will then provide the relevant attributes which are expected to be present with the indexed product.

These relationships (`Item` is instance of `Class/Category`; `Class/Category` has `Properties`) is important for providing search interfaces since the data items need to be presented in a human readable way.

The Matchmaking Service is based on the Matchmaking Ontology that defines the internal data structure and format of the indexed data. This data structure must be adhered when using the services for storing and retrieving the data.

[3] https://solr.apache.org/

[4] https://www.w3.org/TR/swbp-skos-core-spec/

In the indexing workflow (see Figure 21), the base platform's data stores provide their data to the Matchmaking Service. To align the distinct platform's data with the Matchmaking Ontology, a data transformation occurs from the incoming data format to the federated data model. The data transformation is implemented using NiFi Jolt (Json Language for Transform) transformation processor, which is an open-source JSON to JSON transformation library. It allows the developer to define rules of transformations as a JSON specification file. The Jolt transformation processor in Apache NiFi processes the incoming data flow file and executes the transformation rules and converts the data flow file to the target schema in the NiFi workflow. In a final step of the Nifi workflow, the transformed data is stored in the EFPF federated index.

6 EFPFInterop Platform Reference Implementation

6.1 Data Spine

The components of the Data Spine described in chapter 5.2 can be realised using the some of the tools and technologies available today or can be implemented from scratch. Table illustrates one such reference implementation of the Data Spine components using open-source technologies. These technologies are further described in the subsequent sections.

Table 1 — Data Spine Reference Implementation

Data Spine Conceptual Component	Technology	Version
Security Portal	Keycloak	3.4.0
Integration Flow Engine	Apache NiFi	1.11.4
Message Broker	RabbitMQ	3.8.5
Service Registry	LinkSmart Service Catalog	3.0.0-beta.1
API Security Gateway	Apache APISIX	2.3.0

6.1.1 Security Portal

Keycloak can be used to realise the Security Portal component. It is an open-source identity and access management solution that support the lifecycle management of users, roles, groups, and access policies, etc. Keycloak supports the OpenID Connect and OAuth 2.0 authentication and authorization protocols. In addition, it supports identity brokering to federate multiple identity providers in order to enable SSO.

Furthermore, Keycloak offers a management GUI and an HTTP/REST API to the administrators, as well as a user registration and account management GUI to the users.

6.1.2 Integration Flow Engine

Apache NiFi (NiFi) can be used to realise the IFE. It is a dataflow management platform based on the concepts of Flow-based programming. It automates the flow of information between systems through directed graphs called dataflows. The dataflows support communication, data routing, data transformation and system mediation logic with the help of 'processors' as their vertices.

The processors are responsible for handling data ingress, egress, routing, mediation, and transformation. The edges that connect these processors with each other are called 'Connections'. NiFi offers a Web-based, highly configurable, drag-and-drop style GUI for creating such dataflows. Figure 23 highlights the

elements of NiFi's GUI and shows a sample dataflow. NiFi's GUI offers a functionality to search for a particular processor, view its short description, and include it in a dataflow as shown in Figure 24. NiFi contains as many as 284 different processors as of version 1.11.3. The integration flows can be realised using the dataflows in NiFi. Apache NiFi offers connectors for protocols such as HTTP, MQTT, AMQP, etc., and data transformation processors such as Jolt, XSLT, ExecuteScript, ReplaceText, MergeContent, etc. Apache NiFi's functionality can be extended by adding new custom processors. Moreover, it supports scaling-out though the use of clustering to ensure high performance and availability.

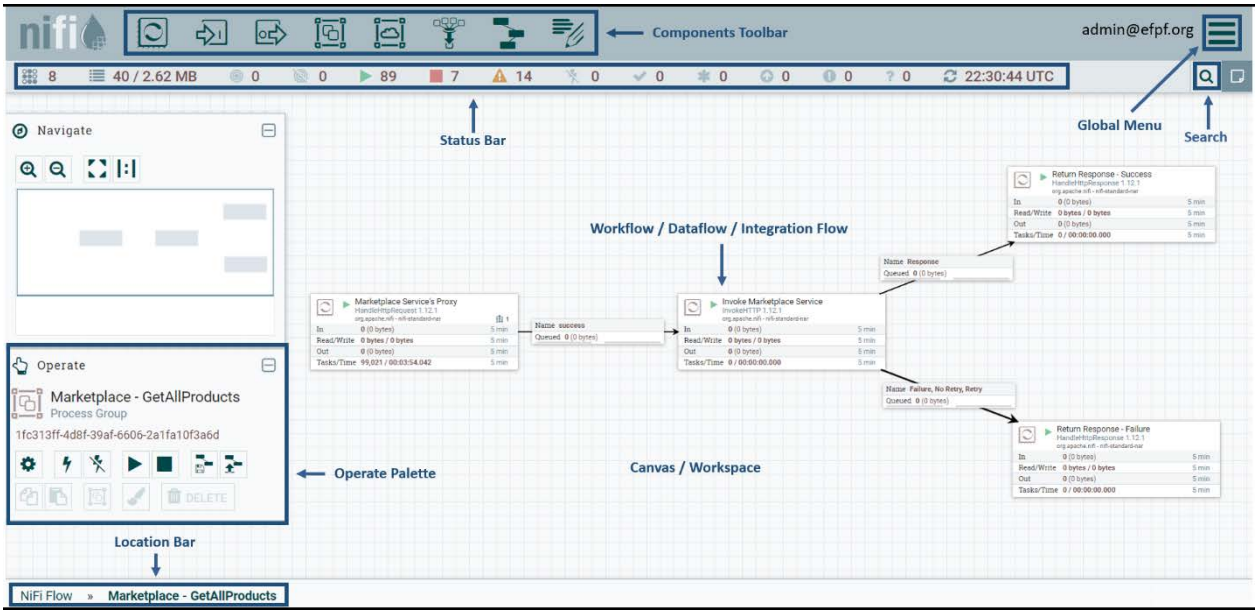


Figure 23 — Apache NiFi's GUI

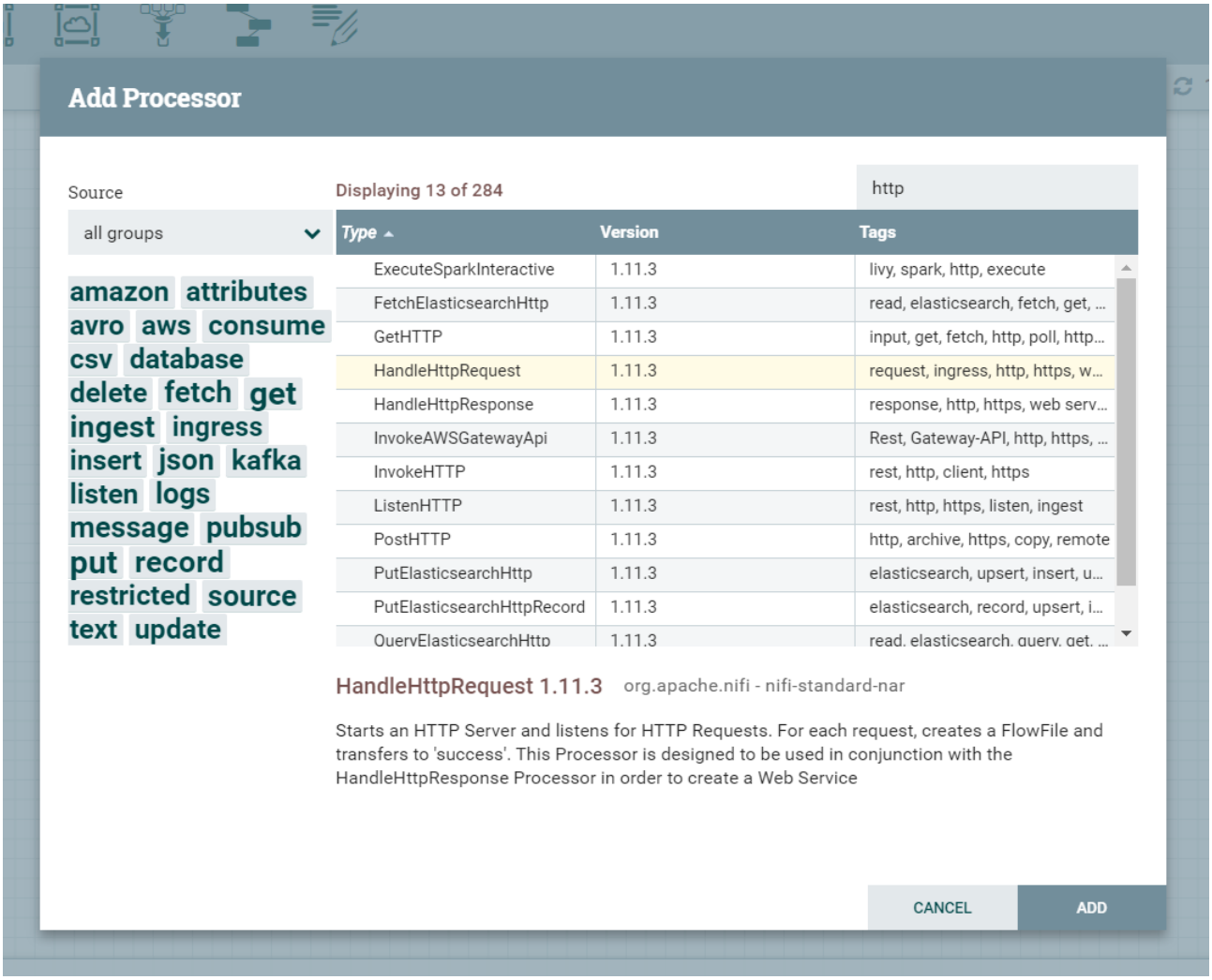


Figure 24 — Apache NiFi Processors

6.1.3 Message Broker

RabbitMQ satisfies the design requirements for the Data Spine Message Broker. It is a message-oriented middleware that implements AMQP (Advanced Message Queuing Protocol) 0-9-1 and supports the Pub/Sub communication pattern. Together with AMQP 0-9-1, it also supports other messaging protocols such as MQTT, AMQP 1.0, STOMP, and WebSockets through plugins. Its functionality can be extended by adding custom plugins.

RabbitMQ provides a management GUI and an HTTP-based API for the management of users, exchanges, queues, and permissions, etc. In addition, it supports multitenancy and fine-grained access control. Finally, RabbitMQ supports clustered deployment for high availability and throughput.

6.1.4 Service Registry

LinkSmart Service Catalog can be used to realise the Service Registry component of Data Spine. Service Catalog is the entry point for web services. Its functionality mainly covers the lifecycle management of services i.e., the registration, viewing, updating and deregistration of services' metadata. In addition, it supports browsing of the registered service entries and provides a service filtering functionality that can be used by service consumers to search services by known capabilities.

The schema of Service Catalog supports storing metadata for both synchronous (Request-Response) as well as asynchronous (Pub/Sub) type of services. Moreover, the Service Catalog provides an MQTT API for announcing the service registration/deregistration events over predefined MQTT topics.

6.1.5 API Security Gateway

The ASG can be realised using Apache APISIX. Apache APISIX is a cloud-based microservices API gateway that delivers performance, security, and an open-source and scalable platform for APIs and microservices. It can be used as a traffic entrance to process all business data, including dynamic routing, dynamic upstream, dynamic certificates, A/B testing, canary release, blue-green deployment, limit rate, defence against malicious attacks, metrics, monitoring alarms, service observability, service governance, etc. Compared with the traditional API gateways, APISIX has dynamic routing and plug-in hot loading, which is especially suitable for API management under microservice systems.

In addition, Apache APISIX supports delegating the authentication and authorization decisions to an external identity provider.

6.2 Marketplace

6.2.1 Integrated Marketplace

The integrated marketplace framework is implemented within the EFPF project so that it connects with external marketplaces in the EFPF ecosystem through API interfaces. This allows the integrated marketplace framework to list the products (from external marketplaces) and provide typical filter and sorting mechanisms, with the ecommerce features (e.g. product listing and checkout) provided by the base/external marketplace. To expand the product offering, the integrated marketplace framework allows users to publish new products and services by providing redirects to the relevant external marketplaces. The newly published products/services become available on the unified GUI provided by the integrated framework at the EFPF Platform. As connected external marketplaces may provide different feature sets and may have different focus e.g. a marketplace may focus on software apps and the other marketplace may provide features that are more suited to physical products. In this respect, the EFPF integrated marketplace framework can bring different types of offerings to the users in the EFPF ecosystem and provides a unified interface that allows users to search for different types of offerings through an intuitive and unified interface – available through the EFPF Portal.

The marketplace integration approach adopted in the EFPF project allows a fast integration of external marketplaces and their products into the EFPF ecosystem and therefore is more suited for a federated environment, where multiple marketplaces co-exist. As each marketplace may have a different data model and also a different design of the marketplace, the integrated marketplace framework performs the necessary transformation at the level of data model and interface – through custom configurations and integrations that are also supported by the Data Spine.

Additionally, the EFPF Marketplace provides federated services such as single sign-on (SSO), which allows customers to use connected marketplaces without registering on each single marketplace again. This all leads to a unified user experience and similar user journeys.

The Marketplace Framework provides access to different external marketplace.

The following external marketplaces are connected and their products are listed trough the unified user interface of the EFPF marketplace framework:

- vf-OS: A marketplace of software applications that are mostly plug and play in nature
- Nimble: A marketplace of physical products
- SMECluster: A marketplace of systems that require consultancy and configurations

- WASP: A marketplace of software/web services

Each external marketplace has been adapted to enable Single Sign-On (SSO), so EFPF platform user can access the marketplace without having to register again. Nevertheless, a marketplace may ask for providing additional information as the EFPF portal may not gathered all information at registration time.

The following list provides an overview of the relationship regarding connected components and a description of the functionality it enables:

- EFPF Portal: This component acts as a container for the Marketplace Framework UI as can be seen in Figure 4. The portal at this point also provide access control as only registered members have access to this section.

- Integration Flow Engine: This component provides REST interfaces to retrieve product data from external marketplaces.

- Service Registry: This component is an alternative to the Integration Flow Engine that also provides the REST interfaces in order to retrieve the products from the different externals marketplaces

- Identity and access management component EFS: This component secures access and communication with both components mentioned before.

6.2.2 Accountancy Service

The Accountancy Service is a standalone component that runs independently of existing EFPF tools and services and can be integrated with unlimited number of external marketplaces. It consists of three main components:

- Log Aggregator: Gathers user behaviour data from various components of the EFPF Platform, executes different transformations and filters the content, before sending the data to the Log Persistence component

- Log Persistence: Stores, indexes, provides and manages user logs to be later analysed. Since relational databases are not well-suited for managing log data, a NoSQL database like Elasticsearch is preferred due to their flexible and schema-free document structures, enabling analytics of the log data.

- Visualization: Enables interactive dashboards, filters and advanced data analysis and exploration of user logs.

In addition, the following custom modules listed below were developed to provide additional functionality:

- Reporting Component: Creates periodic (i.e. monthly) reports for each dashboard at the end of each month in PDF format and sends it as an email

- Invoicing Component: Processes all the payment data accumulated within each month, sums all the amounts from successful transactions realized on each marketplace, calculates a corresponding cashback amount and creates a detailed invoice with the information including purchased products, dates of transactions as well as the calculated commission for each product. The invoice will then be used to charge marketplaces.

In order to process the accumulated log data and provide advanced visualization mechanisms, the Accountancy Service uses Elastic Stack (Elasticsearch, Logstash, Kibana) as an advanced log persistence, monitoring, processing, and visualization framework. The Logstash component is utilized as a data

ingestion and server-side data processing pipeline. The data sent to Logstash from various EFPF components are forwarded to Elasticsearch for persistence after executing certain ingestion pipelines; and then Kibana dashboards are automatically updated based on the certain fields of documents stored on Elasticsearch. In addition to the central functionalities offered by the Elastic Stack, the custom modules were developed with JavaScript and run on Node JS environment.

For the integration, all there is needed to do is to send a POST request to the Logstash endpoint with a JSON body related with the related action. This will be enough for the Accountancy Service to capture the data and update the dashboards. Additionally, this simplicity also allows Accountancy Service to be integrated with an unlimited number of external marketplaces.

6.2.3 Automated Agent-based Marketplace and Online Bidding

The EFPF Automated Agent-based Marketplace and Online Bidding is a standalone service/tool accessible through EFPF portal. The tool enables (semi-) automated negotiations for specific goods and services. It is delivered as a web-based tool to end user and contains three main components:

1. Agent Ecosystem

- Bidding proxy / Agent Deployer: an entity that allows to deploy virtual agents and manage them. It is interconnected with the EFPF platform security framework. It enables registered users to transparently access the platform features without additional login (requires valid JWTs generated by the EFPF OAuth2 server).

- RabbitMQ: a dedicated AMQP broker, deployed within the ecosystem, enabling only agent-to-agent communications.

- Agent Management System (AMS): an entity that allows to register new agents and configure the infrastructure components to enable automated negotiations. The AMS also includes an instance of PostgreSQL dedicated for AMS purposes to persistently manage infrastructure related data.

- Requester/Supplier Agents: containerized software entities that are dynamically generated by the bidding-proxy and assigned to the user that run the deployment. They are autonomously configured through the quoted components and interconnected with a dedicated database to manage their persistence. They foresee direct communication thanks to the dynamic re-configuration of the nginx rules applied by the platform (during the deployment), secured with valid JWTs generated for each user owner (transparently inherited during the deployment by the bidding-proxy functionalities).

2. Matchmaker

The component is a complete semantic framework able to represent business entities, products etc. based on an ontology. Moreover, it supports the storing of business entities etc. as ontology instances/individual. In general it provides CRUD operations and a triple store for saving the data. The core functionality of the component is the matchmaking on 2 levels: (1) supplier/requester matching based on semantic rules/inference and (2) offers/request matching based on best score kind of algorithm. The Matchmaker has been developed in Java using Apache Jena Framework and it was deployed in a Tomcat server. Docker was used in order to create an image for the Matchmaker framework. The Matchmaker is finally deployed in a server as a Docker container. The Matchmaker services are provided as RESTful web services to both Agents and GUI. The ontology used by the Matchmaker has been developed in OWL2 using Protégé tool. The implemented ontology is based on existing ontologies related to manufacturing, supply-chain and e-commerce, like MSDL, MASON and GoodRelations Language.

3. Graphical User Interfaces

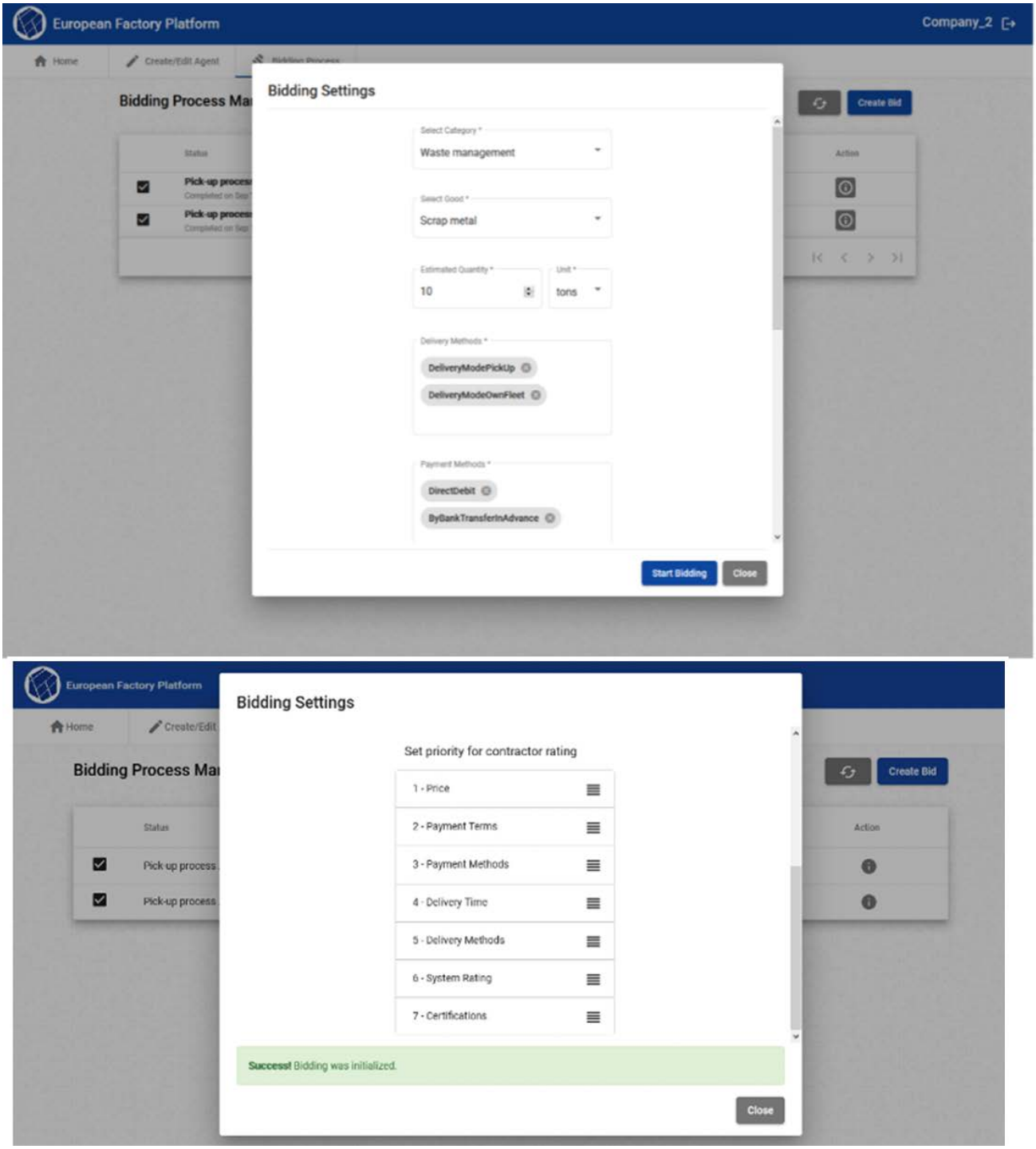


Figure 25 — Online Bidding Process Interfaces

It is a web based user interface, linked to the EFPF portal, able to interact with the APIs of the Online Bidding Process entities such as Agents APIs and Matchmaker APIs. The interface has been developed in Angular 9 and enable end-users to create an agent, to initialize a bidding process for a good, to participate in a bidding process as bidders, to manage the biddings history and to be notified for available requests etc.

6.3 Portal

The EFPF Portal component is divided in two components. The first component is the EFPF Portal frontend, which is a single-page application (SPA) based on the Angular web application framework4. The second component is the EFPF Portal backend, which is responsible for providing the required data for the frontend. It handles also different tasks like registration and event logging.

The EFPF Portal component provides the main user interface of the EFPF platform. It is being connected to other components in the EFPF platform. The following list provides an overview of the relationship regarding connected components and a description of the functionality it enables:

- Identity and access management component EFS: This component secures access and communication of the EFPF portal.
- Smart contracting component: This component is being used at user registration time and enables the storage of registration information, i.e. does the user agreed to the EFPF Terms and Condition.
- Accountancy Service component: This component is being used to track user events like visiting an external platform or conducting a product search.
- External mail service: The EFPF Portal backend uses an external mail service to send out notification or confirmation emails, i.e. when a user registered successfully a welcome email will be sent out with the next steps.

The landing page provides a brief description of this project and explains the breadth of possibilities to a perspective user. The page allows a user to Login to EFPF portal or Register, if it is their first time on the platform. The landing page also provides the user with EFPF Terms & Conditions, so that they can read through them and ensure that it, aligns with their expectations before joining. The rest of the user journey is explained through separate sections below.

First time visitors can click on ‘Register’ button to sign up to the platform. Upon clicking on Register, a simple and quick registration form appears on the screen as shown in Figure 26 below.

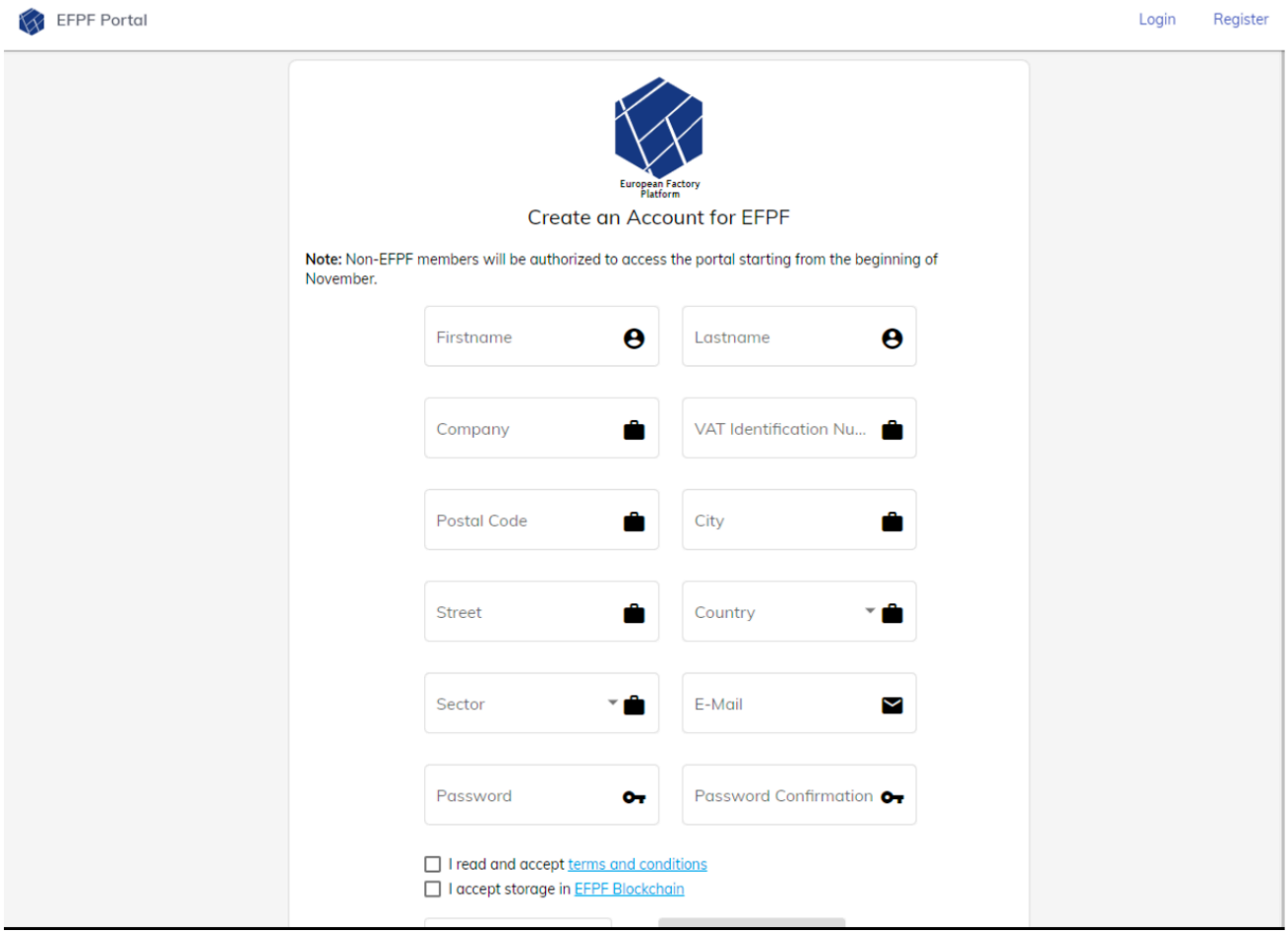


Figure 26 — EFPF Portal - Registration Form

Once this form is filled in, a user account will be set up and a welcome email will be sent to the new user providing general information about the EFPF portal and the next steps.

After the user enters the portal, they are greeted by the dashboard. The dashboard has been designed with a view to showcase the breadth of capabilities available in the federation. The presentation of the dashboard has been adapted to bring out customer benefits, so that they can understand the application and usefulness instantly, without going into the details of where this functionality is coming from in the EFPF federation.

A direct link to access the **Integrated Marketplace** is provided for those specifically looking to explore the integrated marketplace framework.

Further an array of tiles each pointing to a different value propositions of the EFPF federation are shown in the centre of the dashboard. The aim is to provide the user ease of navigation by bundling supplementary tools and services under high level value propositions. Figure 27 shows the overall layout of the dashboard.

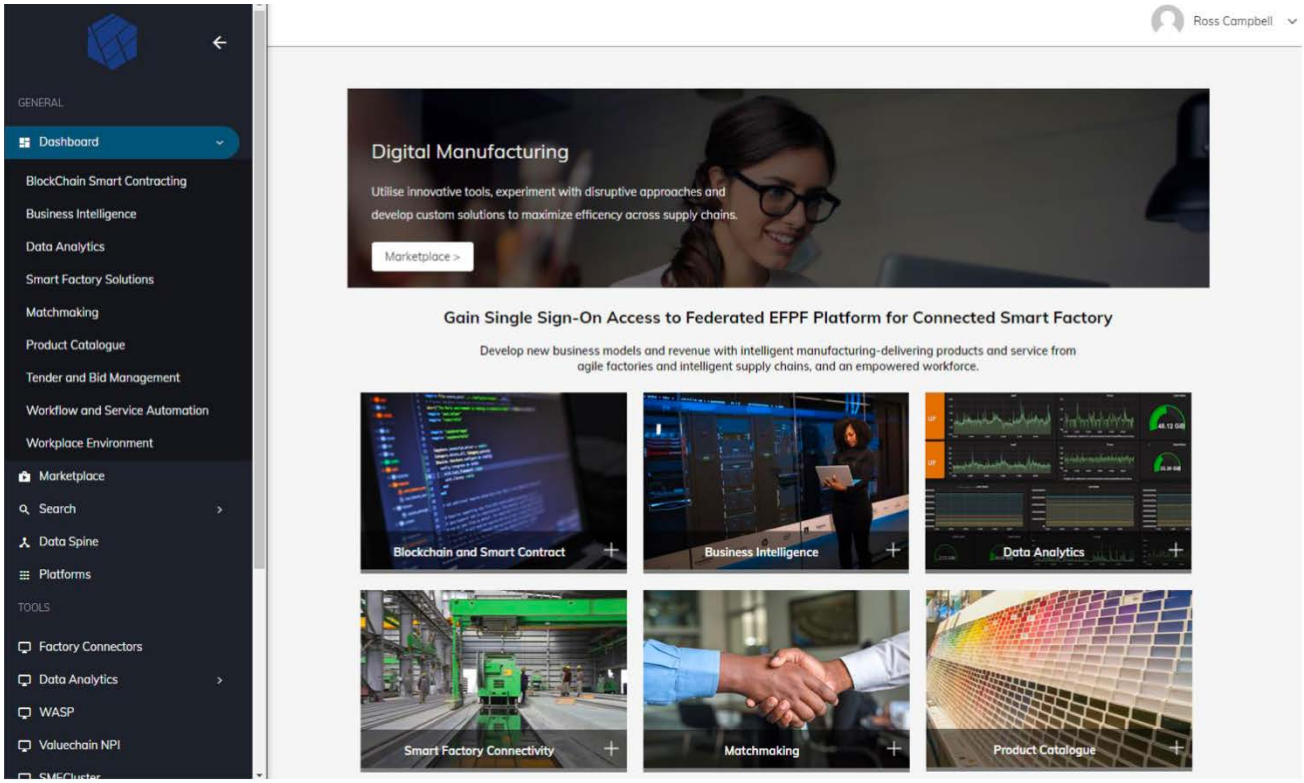


Figure 27 — EFPF Portal Dashboard

6.4 Matchmaking

6.4.1 Federated Search

The Federated Search component is a micro service that offers full-text search capabilities on data collections, as defined in the EFPF Matchmaking Ontology. The service collects data from various platforms in the EFPF ecosystem and provides a single search endpoint offering a unified view on the data in the distinct platforms.

The Federated Search implements the Matchmaking Ontology (see simplified version in Figure 28) as persistable data objects that are used to interact with the Apache SOLR search engine. The data objects are mapped to Apache SOLR collections automatically, thus the data objects are saved or retrieved from the index in a transparent way. To cover the requirements of arbitrary naming and categorization of indexed data elements, the Federated Search defines and extends the Core Concept, as shown in Figure 29. Each of the attributes is mapped with an indexed field in the respective data collection. For providing the basic collections for classification and properties, the service uses the following data classes: `ClassType`, `PropertyType`. Both classes inherit a common behaviour from the abstract class `Concept`.

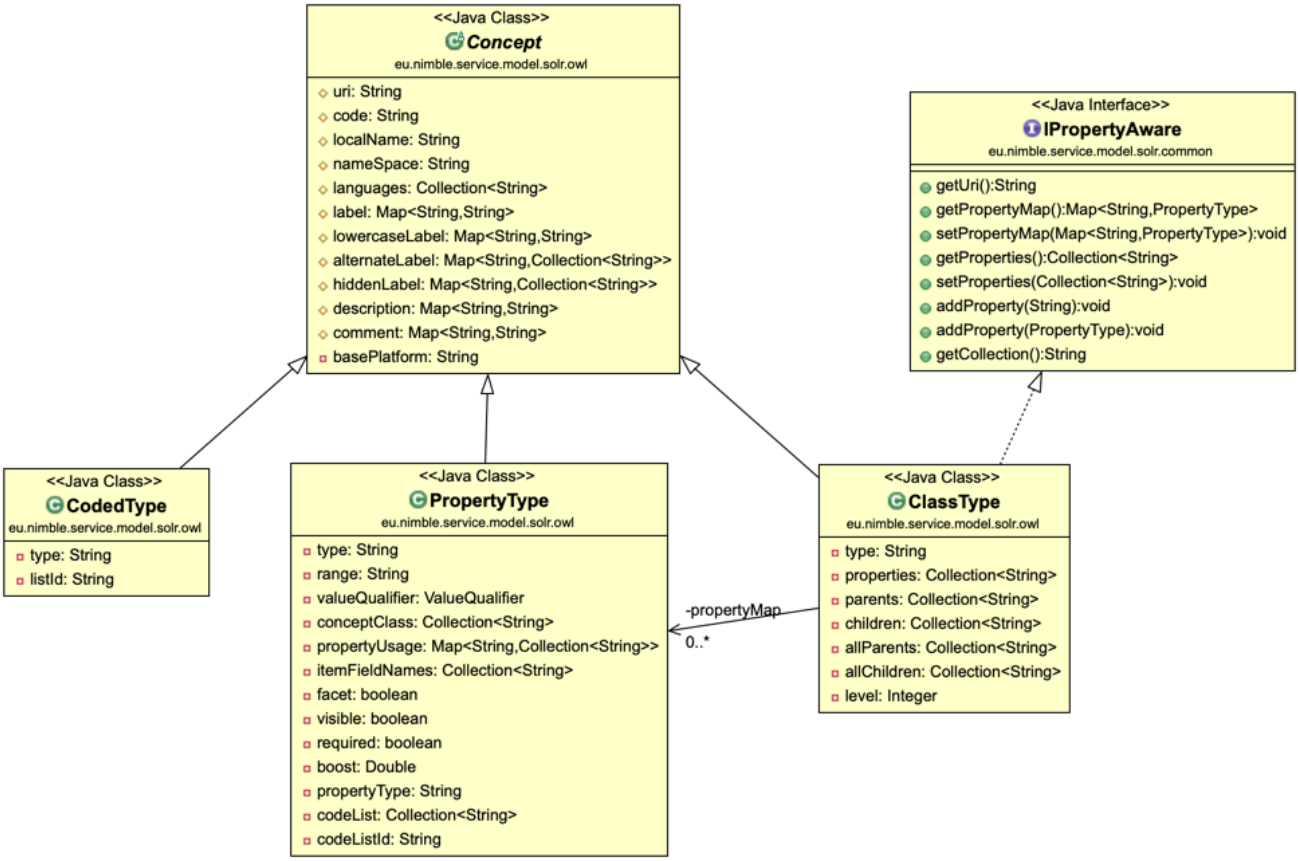


Figure 28 — Implementing Core Ontology Elements

The process of data mapping with the indexed fields is controlled by the following data types:

- Single value type: The underlying index field allows only a single value. The index field name is fixed.
- Collection of Strings, Numbers, Booleans: The underlying index field allows multiple values of the given data type. The index field name is fixed.
- Map with single values: The resulting index field is customized with the key value of the map. The index field name consists of a fixed and a variable part. Only one value per variable part is allowed. For example, each Concept can have only one preferred name per language.
- Map with multiple values: The resulting index field is customized with the key value of the map. The index field name consists of a fixed and a variable part. Multiple values per variable part are allowed. For example, each Concept may have multiple alternate names per language.

The data objects in Figure 28 outline the basic data objects of the federated search. The Matchmaking Ontology is designed to support the federation of *Company* (`PartyType`) and *Product Item* (`ItemType`) data. The corresponding data objects in the Federated Search are shown in Figure 29**Fehler! Verweisquelle konnte nicht gefunden werden.**.

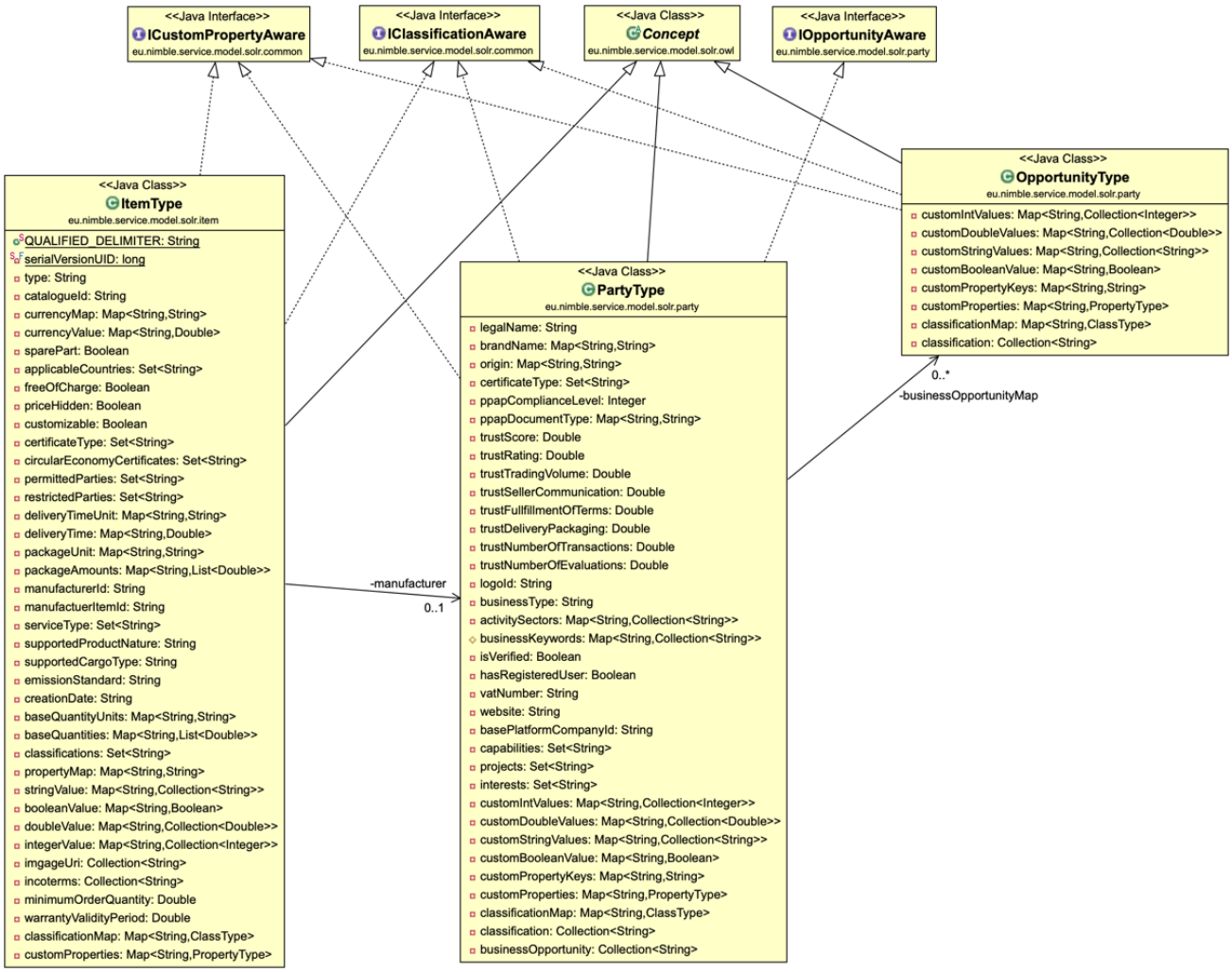


Figure 29 — Matchmaking Ontology - Federated Data Objects

The respective data elements are used to define the structure of the underlying data collections. Beside the company and items, the Federated Search provides the possibility of collecting Business Opportunities (`OpportunityType`). As shown, each of the data objects inherit from Concept, thus they may equipped with multilingual naming. Furthermore, the data collection implements behavioural interfaces such as

- `ICustomPropertyAware`: Unlike the known index fields with variable name-parts, the use of custom properties is independent of the predefined data model. Implementing Data collections may add/define custom values for indexing by providing the name and the value of the field. Those custom values may be accompanied with the corresponding Property-Meta-Data which is then automatically handled by the Federated Search, e.g., the meta data is stored in the respective collection and a link between the custom field and the describing property is maintained.

- `IClassificationAware`: Implementing Data Collections may be classified with Concept classes (see Figure 28). When updating data items, it is necessary to name the classification classes (`ClassType`) by its identifier. For easy usage, it is additionally possible to provide the complete Meta-Data Object in a single request. The Federated Search Service stores the meta-data object in the `ClassType` collection and maintains the link between the classified data and the `ClassType` collection.

- `IOpportunityTypeAware`: Similar to IClassificationAware. Implementing Data Collection maintain a link to `OpportunityType`.

The Federated Search Service handles incoming data and stores the data transparently in the Apache SOLR index. For retrieval, the Federated Search service contacts the Apache SOLR index and restores the data structure, according to the Matchmaking Ontology.

Furthermore, it resolves known relationships between collections and integrates the full meta data object with the requested data item(s).

6.4.2 Federated Search API

Conformant with the Matchmaking Ontology, the Federated Search maintains an Apache SOLR collection for each of the data objects to handle and provides REST methods with CRUD operations for each of the collections.

Like the SQL Query Language, querying of indexed data starts with the data collection of interest. Apache SOLR offers a sophisticated query interface which allows finetuning the search with a query expression, additional field query expressions, defining facets and the like. With paging information, it is possible to access the result set in a paged manner.

For getting insights of the data items in use, especially the finally constructed index field names, a API method provides the index field names and – when maintained – the linked multilingual naming of the index fields.

Annex A
(informative)

Best Practices / Lessons Learned

A.1 Real World Example – Furniture Pilot

User Story: Production optimisation. Real implementation in the industrial shopfloor of LAGRAMA (Vinaros, Spain).

Description

The **Error Proofing Production Optimisation solution** supports the optimisation of a production process in real-time by displaying key production guidance to Operator staff by interfacing between the product barcoding tracking system on each product, and the client ERP. This information aids the operator and ensures that they know the correct production actions and avoid unnecessary mistakes.

The solution involves both EFPF components and tools. The main EFPF component involved is the Data Spine – Message Bus, while the tools involved are:

- Industreweb Collect Factory Connector
- Industreweb Display Factory UI
- Client ERP
- Industrial Vision System

The primary components are installed within the factory with an *Industreweb Collect Factory Connector* located within an electrical cabinet connected to the Industrial vision system via Ethernet/IP to be able to capture the barcode values of products entering the machine. The barcode value is retrieved as a byte of strings and is converted to a single string to allow it to be passed to the next step. The ERP system is then interrogated with the barcode value to retrieve the Production schedule and associated metadata for the product. This is returned as a JSON object which is then checked to ensure is valid and complete and suitable for visualisation. This is then rendered asynchronously in the Industreweb Display asynchronous HTML operator screen located by the machine. Barcode and product metadata is passed to the data spine to validate the barcode format and to capture valuable production metrics from the process.

The overall solution optimises the production process by avoiding human error when attempting to interpret a barcode label on a product, which can cause mistakes and production delays which can impact on quality and performance production metrics.

The aim of this scenario is to improve the efficiency of a new edge banding machine at LAGRAMA by displaying clear instructions to the operator about how to proceed with the current piece, in order to avoid mistakes and detect any machine operation error.

High level architecture

The deployment initially involved three main hardware components: (i) the Industreweb Factory Connector, (ii) the sensor interface board, and (iii) the sensors of temperature, current and pressure.

To make clear to the operator how to proceed with the current piece, the barcode label attached to each piece is scanned with a barcode reader, which is connected to the Industreweb Factory which allows the Factory Connector to query the ERP in order to get the piece details back. These instructions are then

displayed in a clear way to the operator on a screen next to the machine. The elements involved can be seen in Figure A.1.

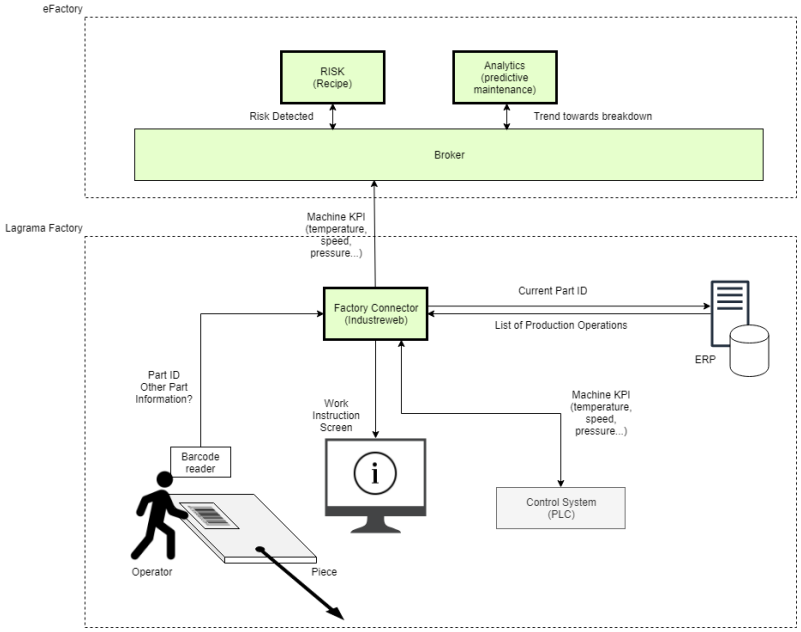


Figure A.1 — Elements involved in the deployment for predictive maintenance

From a sequence perspective, the main steps are the following:

- The operator loads the piece into the machine.
- The Factory Connector reads the label on the surface of the piece – that can be seen in Figure A.2 - via the barcode scanner head.
- The Factory Connector makes a request to the API of the ERP in order to get the information of the piece, which not only includes the current status but also the sequence of next operations that needs to be done by the worker.
- The data is returned from the API ERP.
- The Factory Connector interprets the data retrieved and displays the instructions in a clear format to the operator, via a display located next to the machine.
- The worker follows the instructions and moves the piece to the proper place for the next operation.

Figure A.2 — Detail of label attached to piece

The barcode is unique for each piece and it is represented in the Code 128 format (see ISO/IEC 15417).

Besides this, the text available in the label is also a unique identifier for each piece, and it could be used to get information about it. In particular:

- CSF represents that the piece above is a standard piece but unique inside the current lot
- The code 0820 represents the lot, and means week 8 of the year 20
- The codes 3094 and P14 represents the first and second part of an identifier
- The keyword Ruta represents the route that the piece traverses in the factory. For example, 1xN1 means one operation made in machine N1, while 4xB2 represents that 4 operations have been done in the machine B4.

All the information about the piece can be fetched from the barcode by querying the database through the API of the ERP. In Figure A.3 it is shown a diagram where the sensors and the barcode reader are placed in the edge-banding machine.

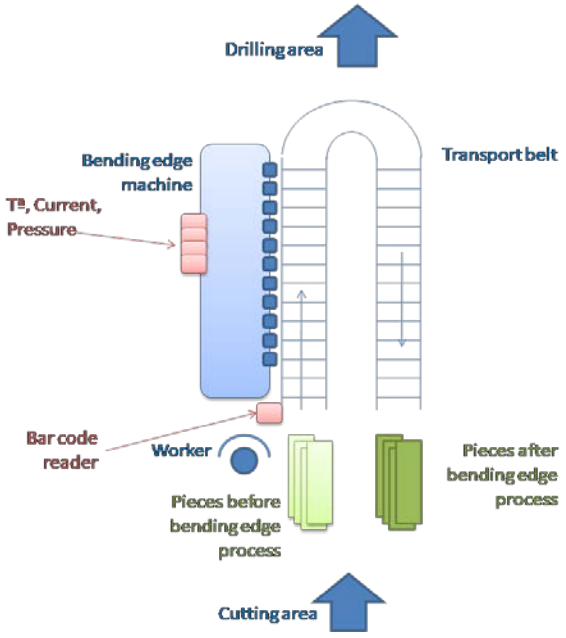


Figure A.3 — Diagram of the preliminary placement of the devices

The hardware required for the production optimisation solution is mainly:

- The Factory Connector
- A barcode reader
- A PLC

These components were integrated in an electrical panel that was in turn installed in a cabinet to be placed in the LAGRAMA premises near the edge banding machine. Several configuration activities were needed in order to achieve a successful recognition of the barcodes through the code of the PLC. To debug the PLC configuration and implementation, an instance of the development environment was installed in a computer, which was configured in the same IP range than the PLC in order to make this last one accessible from the computer. The camera of the scanner was able to recognise the barcodes and catch the information associated to them.

The test activities also involved the calculation of the maximum range that the camera is able to cover, as well as the orientation of the labels in the pieces. These parameters have been used to determine the height at which the scanner should be placed in the production line as well as the orientation of the pieces. Figure A.4 shows a label scanning process (left) and the information of the piece retrieved from the ERP in the dashboard of the tool (right).

Seqüència	Màquina	L1	L2	A1	A2
1	B4	X0,8			
2	B4		X0,8		
3	B4				X0,8
4	B4			X0,8	

Figure A.4 — Error proofing solution in LAGRAMA

The **User Value proposition** can be summarised in the following:

- Time saving in the redirection of the pieces to its proper destination at the production plant
- Reduction of mistakes in the component classification process

Technical development

Process/Machine Evaluation

In order to achieve the Pilot goal of Production Optimisation through Error Proofing the first stage was to understand the manufacturing process in detail, in order to identify where the Operator Errors are occurring. The Edge Banding machine applies edge trim to furniture board before the component part progresses to one of several subsequent production steps. In order for the operator to know what the correct production workflow is, they must read a small printer component label that can be applied somewhere on the board whilst manually loading it into the machine. Since the text can be hard to read

and interpret, mistakes by the operator can occur meaning that the component is incorrectly produced, or is delayed by entering the wrong next step.

The Error Proofing solution therefore must detect the appropriate flow for the product and ensure the Operator is clear in what the component is, and what must be done next. The system must therefore scan the component label on the part and retrieve information from the LAGRAMA ERP system in order to visualise and guide the operator. The characteristics of the movement of the part through the machine (speed, label size etc) are considered so that a suitable vision system can be selected.

API Definition

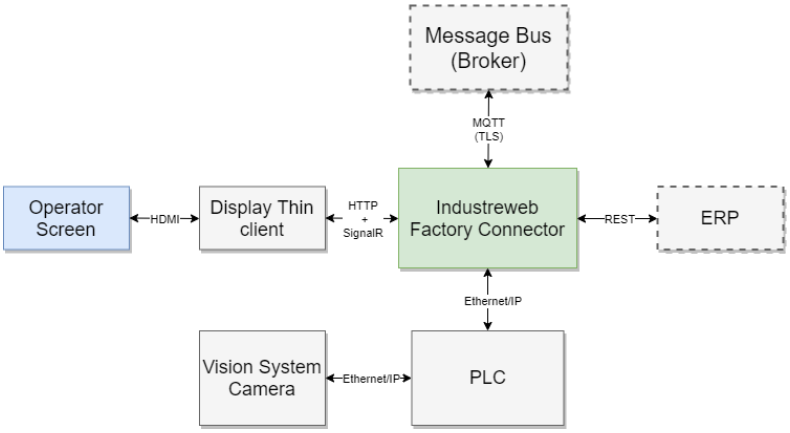


Figure A.5 — API specification for Production Optimisation

The hardware selected for the Pilot was an Omron NX PLC with an Omron V430 vision system camera which provides support for the barcode format used, based on the speed, distance and variation in position identified for the Edge Banding machine. The API available for this specific PLC model is Ethernet/IP Industrial Protocol, the camera scans the barcode and the value is saved as a string in the PLC, which can then be read by the Factory Connector.

The LAGRAMA ERP system provides a secure REST API which allows the information regarding the Production plan to be returned based on component part reference code.

The screen to be shown to the operator is delivered as a HTML page over HTTP which provides the page layout and aesthetics, whilst SignalR is used to dynamically load KPI values on the screen as they change in the process.

Architecture

The architecture for the Production Optimisation is shown in Figure A.6. At the factory level the Factory Connector (FC) incorporates at high-speed run time Engine which allows the FC to read and write to production data sources. This is achieved by a Connector based architecture, where a connector handles the protocol transformation to being data into the engine in a native common format. In the Engine the data can be evaluated and transformed based on an application specific configuration loaded into the FC. Changes to data inside Engine are reflected via the SignalR protocol in order for the Industreweb Display Operator Dash to subscribe and visualise the values.

Four Data Connectors are implemented which are described below:

- **MQTT Broker connector** allows Production events detected to be published to the EFPF Message bus over MQTT in order for the Analytics and Business Intelligence to be conducted.

- **EEIP Connector** allows the FC to detect changes in the barcode scanner reflected in the Omron PLC and to retrieve the barcode value.

- **REST Connector** allows the FC to call for Product information using a REST API call, by passing the barcode value retrieved from PLC. The JSON result includes the production details for that component and this is then used to visualise the Operator Dashboard

- **SignalR Connector** the JSON payload from the ERP REST response is passed to the HTML of the Operator dashboard and JavaScript is used to subscribe and render the values to guide the operator.

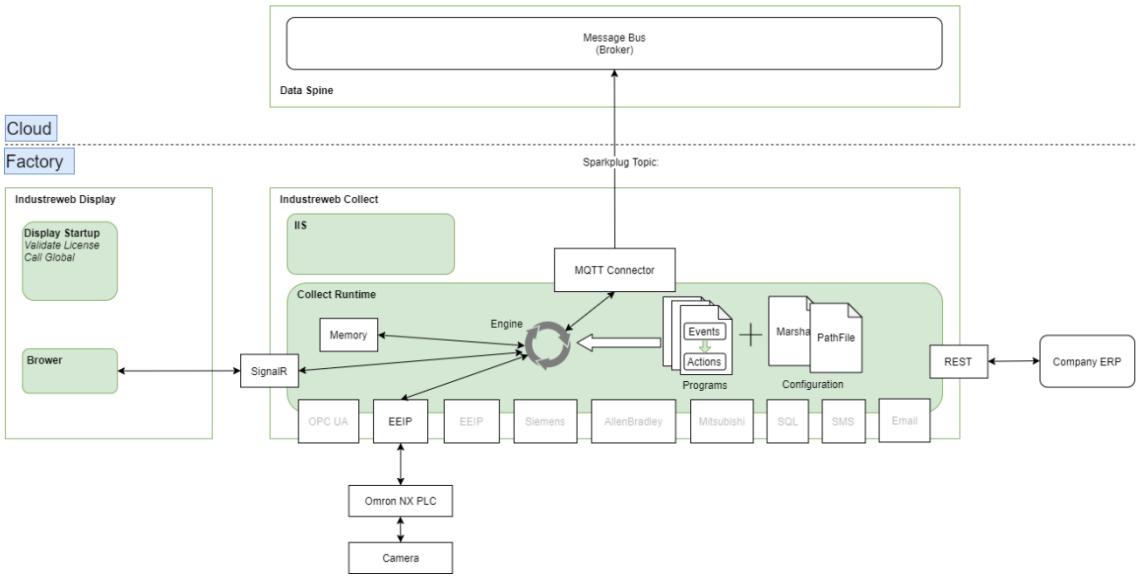
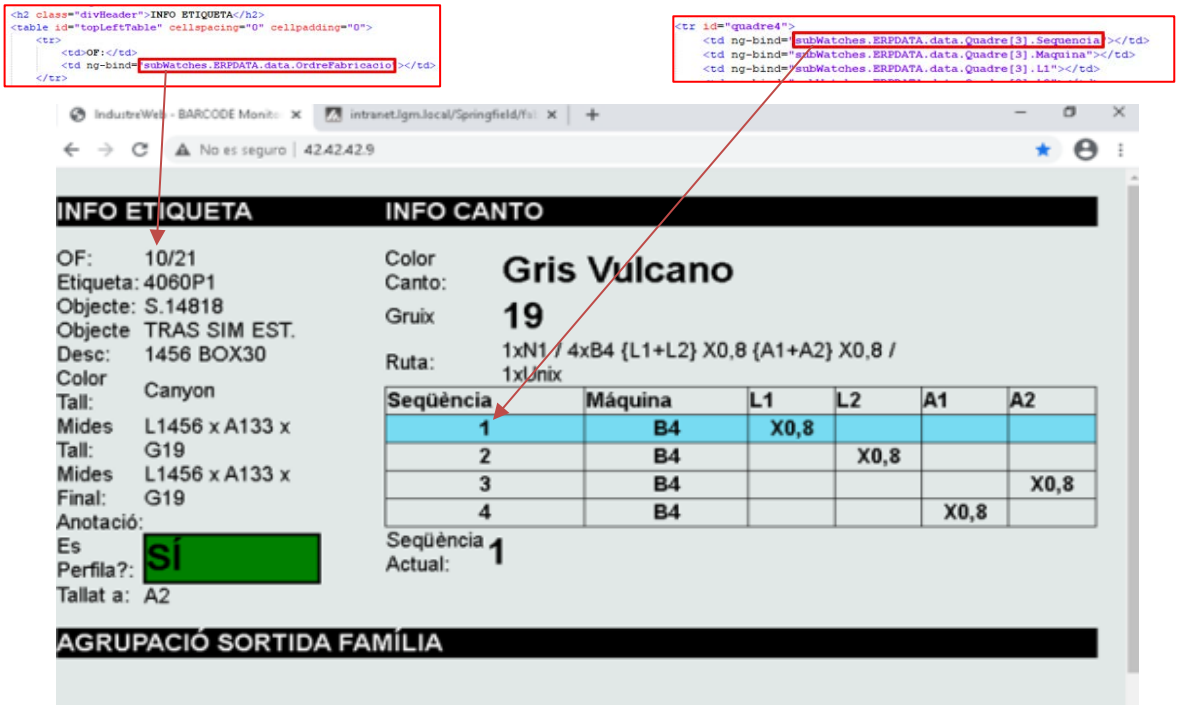


Figure A.6 — Factory Connector Architecture

UI Development

The Operator UI is based on a responsive HTML5 and JavaScript technology. On the page load event the Dashboard subscribes to the ERP result data point which contains a JSON data model. AngularJS is then used to navigate the data model and subscribe to the specific data fields from the ERP.

A sample of the source HTML for this page is shown in Figure A.7. The subscription is updated every 1 second which is sufficiently responsive to allow the operator to react to the correct operation for that component once it has been scanned when passing beneath the camera.



Seqüència	Máquina	L1	L2	A1	A2
1	B4	X0,8			
2	B4		X0,8		
3	B4				X0,8
4	B4			X0,8	

Figure A.7 — HTML5 Operator Dashboard

Testing and Validation

The Production Optimisation pilot was tested in two phases:

The first phase involved manually passing component barcode labels beneath the vision system camera, monitoring the REST response from the ERP and visually checking the dashboard updated to reflect the result. This testing resolved a number of issues where the barcode was not fully scanned or was misinterpreted and there was an empty or error response from the ERP. An empty response from the dashboard would cause the dashboard to become empty. This was mitigated by configuring the Factory Connector detecting for malformed patterns in the barcode string, and also by filtering out any empty results returned by the ERP.

The second phase involved the camera being mounted above the bed plate of the machine set at a height optimised to scan the barcodes as they pass beneath attached to the furniture board component. Initial testing proved successful, however ongoing testing in a live production environment is ongoing. The largest challenge is to ensure consistency of scan despite the furniture board varying in dimensions, and the position of the barcode label varying in its position.

A.2 Real World Example – Circular Economy Pilot

User Story: Predictive Maintenance Solution for Lift Manufacturer KLEEMANN's Polishing Machine, Kilkis – Greece

Description: This user story focuses on the early detection of machine failures in the polishing machine at KLEEMANN's shopfloor. The sensors installed are used to capture vibration data. In order to achieve this, a solution is needed to predict potential machine defects, so that the Maintenance Manager can improve maintenance operations and procedures.

The Predictive Maintenance solution was implemented by using EFPF components such as

- Vibration sensors connected over Wi-Fi in KLEEMANN production line

- Data Spine's Message Bus is used as the secure broker for publishing vibration sensors data in order to be used by analytics components
- Data Spine's Security Framework is used in order to provide access to the solution and its data only to authorized users from KLEEMANN
- EFPF Portal is used in this case as well, in order to direct user to the solution's interfaces
- Visual and Data Analytics tool is used for the delivery of the predictive maintenance solution in terms of both data analysis and real time monitoring

Architecture:

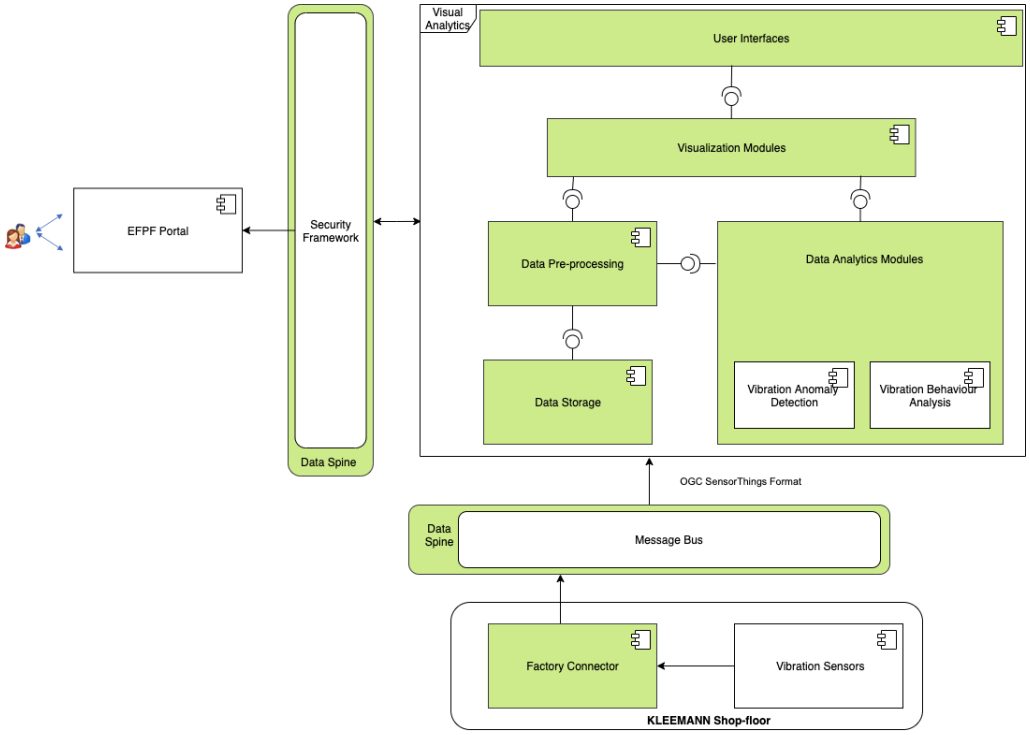


Figure A.8 — KLEEMANN Predictive Maintenance Solution Architecture

Technical Development and Main Components:

Vibration Sensors available on KLEEMANN's polishing machine

Each deployed sensor consists of a feather ESP32 SoC board which provides the micro-controller unit and WiFi connectivity, and a Lis3DH3 axis Mems accelerometer (see Figure A.9). As soon as the motors are operating, the sensors are activated and send continuously vibration samplings over WiFi. The samplings are received through MQTT by the analytic platform and they are being processed as time series of accelerations in 3-axis. The time-stamps between two successive sampling windows, which last 1 second each, are 1.5 –2.5 seconds apart, depending on the quality of wireless communication. Sensitivity of the sensor is 0.038 m/s2 or 3.87 mg at 1344 kHz sampling rate and +2g measurement limit, which is the configuration applied on our case. The data was transmitted by using OGC Observation and Measurement JSON format.

Figure A.9 — SP32 board and Lis3DH on a breakout board

The solution initially tested in lab conditions by using the same setup as described above:

Figure A.10 — Lab-testing Setup

Besides the vibration sensor, the lab test system was powered by a VMARK IPS-2000C power supply and the monitored rotating machine is RS-555SH-6513 with nominal voltage 12V.

Machine's vibration monitoring for Anomaly Detection

The machine's vibration monitoring for Anomaly Detection is a dynamic solution based on real-time data coming from deployed vibration sensors. The real-time vibrations monitoring and analysis result in an anomaly detection pipeline towards early fault detection predictive maintenance. The method uses the historical data to calculate the MAHALANOBIS distance of each point and modelling normal behaviour. A threshold is defined at this point, too, in order to detect outliers when new incoming sensor data overcome this threshold. The maintenance manager is visually informed via the Visual Analytics when the machine's activity surpasses the abnormal vibration threshold and can check for potential faults in the machine operation.

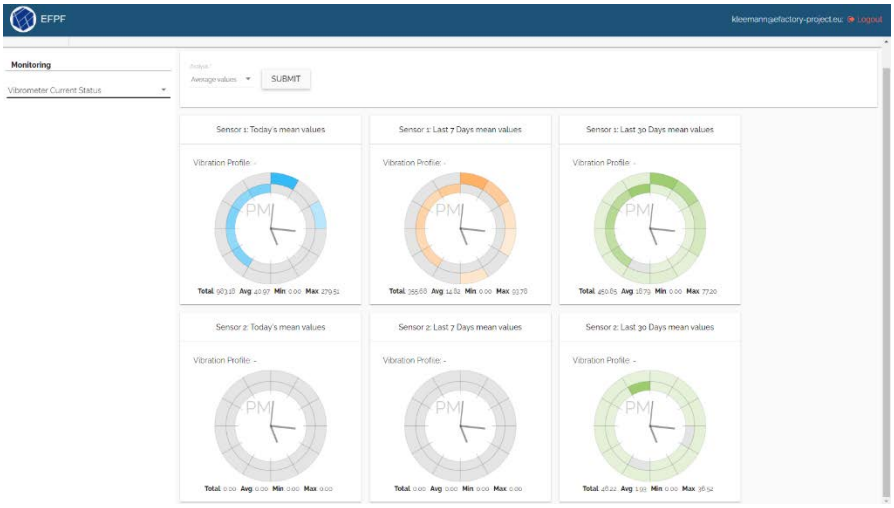


Figure A.11 — Monitoring Interface

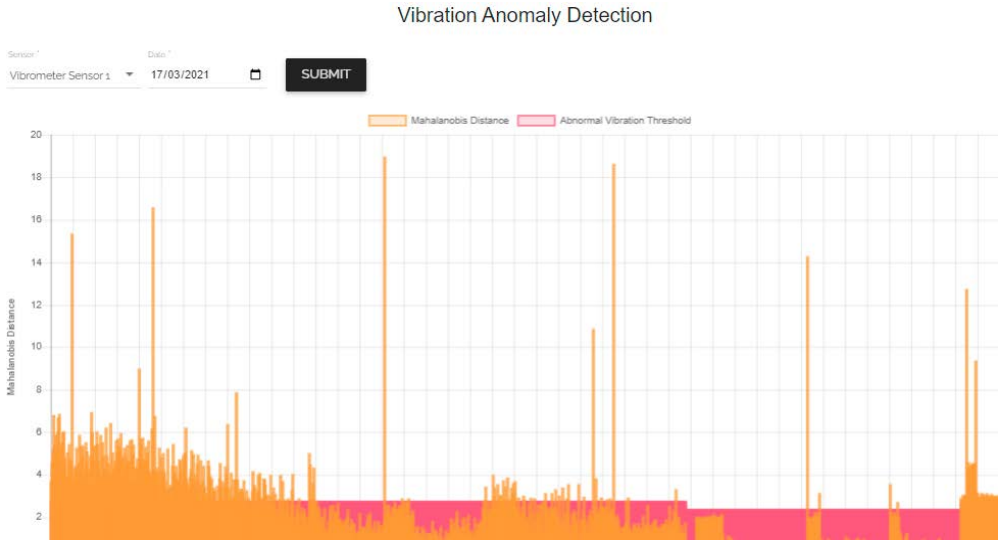


Figure A.12 — Anomaly Detection Interface

For the monitoring purposes the Visual and Data Analytics tool was used. The tool provides a complete solution for analysing and visualizing various types of data. It is a web-based framework to analyse and visualize both industrial and supply chain data. The tool was developed based on web technologies such as AngularJS, JavaScript and ChartJS. The backend algorithm was developed in Python. The deployment was based on micro services logic and Docker is used as the containerization technology. Both REST/HTTP and MQTT protocols are utilized by the analytics tool to communicate with other tools and IoT devices.

Figure A.13 — Installed sensor on polishing machine

After the lab testing period, the solution was deployed to KLEEMANN and become available through the EFPF portal interfaces and Visual and Data Analytics CERTH's tool.

Bibliography

[1] EN ISO 11354-1, *Advanced automation technologies and their applications – Requirements for establishing manufacturing enterprise process interoperability – Part 1: Framework for enterprise interoperability*

[2] ISO/IEC 15417, *Information technology – Automatic identification and data capture techniques – Code 128 bar code symbology specification*

[3] Deshmukh, R.A.; Jayakody, D.; Schneider, A.; Damjanovic-Behrendt, V. Data Spine: A Federated Interoperability Enabler for Heterogeneous IoT Platform Ecosystems. Sensors 2021, 21, 4010. https://doi.org/10.3390/s21124010

[4] Selvanathan, N.; Jayakody, D.; Damjanovic-Behrendt, V. Federated Identity Management and Interoperability for Heterogeneous Cloud Platform Ecosystems. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; ACM: Canterbury, UK, 2019; pp. 1–7.

[5] D3.2: EFPF Data Spine Realisation – I: https://www.efpf.org/deliverables, Direct link: https://www.efpf.org/_files/ugd/26f25a_9304e422a2ef421f8913c52ec4bf60b1.pdf, Retrieved 2022-06-09