NIMBLE Federated Platform Launch Manual

Project Acronym	NIMBLE
Project Title	Collaboration Network for Industry, Manufacturing, Business and Logistics in Europe
Project Number	723810 (H2020)
Work Package	WP8: NIMBLE Platform Adoption – Communication - Exploitation
Responsible author	Alessio Gugliotta (INNOVA)
Dissemination Level	Public
Contractual Delivery Date	31.03.2019
Actual Delivery Date	31.03.2019
Version	V1.0

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 723810

Table of contents

Document Information

Project	NIMBLE (H2020-723810)
Identifier	NIMBLE-D8.10
Author(s):	INNOVA srl SRFG
Document title:	D8.10 NIMBLE Federated Platform Launch Manual
Source Filename:	NIMBLE_D8_10_Platform_Launch_Manual.docx
Dissemination level	Public

Document context information

Work package/Task	Task 8.5 NIMBLE Platform SEED Programme for Federated Platforms
Responsible person and project partner:	Alessio Gugliotta (INNOVA srl)

Quality Assurance / Review

Wernher Behrendt (SRFG)	Please address comments and changes in tracking mode. The document is of good quality and the final version will be fit for submission.

Citation information

Official citation	NIMBLE Platform Launch Manual V1.0 – March 2019

Document History

V	Name	Date	Remark
01	INNOVA	26/02/2019	Initial draft with structure
02	SRDC, SRFG, IBM	15/03/2019	Contribution about technical descriptions (section 4)
03	INNOVA	15/03/2019	Description of the framework for the platform strategy design (section 3)
04	INNOVA	21/03/2019	First draft with all sections completed
05	INNOVA	27/03/2019	Addressing internal reviews' comments
1.0	INNOVA / SRFG	31/03/2019	Final check and submission

1 Executive Summary

The objective of the NIMBLE Federated Platform Launch Manual is to provide a means for guiding potential new platform owners towards deploying their own digital platform in a specific manufacturing market / sector, by using the NIMBLE platform as baseline.

In this document we will first introduce an open source methodology tool - named *Digital Platform Toolkit* – that the NIMBLE project has adopted to guide a potential new digital platform owner to define and plan their own digital platform strategy. The toolkit offers a methodology and a set of design canvases specifically devised to help designers, founders and managers to design strategies, products and organizations "as a platform". Everything is supported by a facilitation guide (https://platformdesigntoolkit.com/toolkit/), so that anyone can use it. NIMBLE also plans to organize specific workshops where we will support one or more business entities interested to become platform owners in devising their business case with the help of the toolkit. As a result, the potential new platform owner will obtain:

- A vision and a design how the system can work, creating and exchanging value among different stakeholders in the target market;
- a first design for the validation by creating a prototype (Minimum Viable Platform) or assumptions for validation experiments.

On the basis of these results, the potential new platform owner can have a look at the NIMBLE platform capabilities and available services and decide to adopt it to rapidly implement the envisioned MVP.

Therefore, in the second part of the present document, we will focus on the technical aspects to create and customise a new NIMBLE installation, starting from the available open source components and services. The permissive open source approach and the standards (taxonomies/ontologies) at the core of NIMBLE will enable potential new platform owners to:

- benefit from a ready-to-use solution;
- use and customize NIMBLE as they wish;
- interoperate with other platforms/solutions using the same standards.

To this end, the document will briefly provide a description of:

- the NIMBLE core technologies and services;
- how to deploy a new NIMBLE instance;
- customization and extension levels.

Similarly to the platform strategy definition toolkit, a potential new platform owner can set up a new NIMBLE instance by simply referring to the provided materials. However, NIMBLE project partners will be available to support interested parties to deploy their NIMBLE installations.

In order validate the platform design toolkit materials, we did a workshop with use case partners covering the application of the Platform Design Toolkit (results of this workshop are reported in D8.12 – Business Plan). A technical workshop with use case and development partners to teach them how to launch the system is planned for April 2019.

2 Introduction

This document serves one of the two impact creation programmes of the NIMBLE project: while the AMBASSADOR programme is aiming at attracting manufacturers and suppliers to use a NIMBLE-based B2B platform, the **SEED programme** addressed in this deliverable supports the creation of a federation of NIMBLE platforms owned/managed by distinct organisations and serving distinct sectors and/or geographical areas. Hence, the SEED programme is targeted at any one of these potential platform providers.

As detailed in D8.8 (SEED Programme: Manual and Materials Package), the target audience of this programme is represented by the following classes of stakeholder:

- *Manufacturing B2B service providers and intermediaries* - i.e. all organizations (profit and non-profit) that facilitate companies (particularly SMEs) in growing their business.
- *Digital platform and infrastructure providers* - i.e. companies and organizations that offer open or private digital solutions for many classes of applications (marketplaces, supply chain management, IoT, etc.), business models (B2C, B2B and B2B2C) and verticals (manufacturing, transportation/logistic, smart cities, etc.).
- *Technology and Service Providers* – i.e. companies that can develop software services and modules on top of the core services of the NIMBLE platform to implement a new tool for platform customers.

For these stakeholders, it is very relevant to understand how the platform works and what type of (business) benefits it can bring to them and their customers.

In the previous deliverable of the SEED programme, namely D8.9 - Feasibility and Impact Assessment Toolkit, we focused on identifying metrics and methodology to demonstrate the potential business benefits of the NIMBLE platform.

We now assume that a potential new platform owner had a look at the NIMBLE platform (presentation, demo, current reference platform accessible through the project website) and to the existing business cases (project pilots and their validations). Based on the information, they are now interested to understand the next steps to develop their own digital platform by leveraging the NIMBLE solutions. The present deliverable provides insights and directions to address this question:

- *In the first part* (Section 3), we will introduce an open source tool the NIMBLE project has adopted to guide a potential new digital platform owner to define and plan their own digital platform strategy. As detailed later, the tool helps the potential new platform owner to identify the key actors, interactions and functionalities that the future digital platform should support and how. This is a preliminary step for motivating the interested party to become a digital platform owner and adopt the NIMBLE solutions to quickly start implementing their own digital platform strategy. In addition and as part of the SEED Programme, the NIMBLE project partners can set up joint or ad-hoc workshops to support interested parties in adopting such a tool and thus developing their platform strategy. As reported in D8.12 (Business Plan – v2) we adopted this tool for supporting project use case partners in defining the respective platform strategies.
- *In the second part* (Section 4), we will focus on the technical aspects to create and customise a new NIMBLE installation, starting from the available open source components and services. This will include a brief introduction of such components and services, their levels of configuration and customisation, the references to the respective software code repositories and additional materials. Similarly to the platform strategy definition tool, introduced in the first part, a potential new platform owner can set up a new NIMBLE instance

by referring to the provided materials. However, NIMBLE project partners will also be available to support interested parties to deploy their NIMBLE installations.

3 Becoming a Platform Owner: defining your own Platform Strategy

Platform strategies have become crucial to innovation. As introduced by John Hagel in "The Big Shift in Business Models"[1], business models in the past have been pretty simple: there was the vendor and the customer; the vendor provides the customer with products and services and the customer pays the vendor for those products and services. That's all changing. Increasingly, new business models are emerging with the opportunity to mobilize other parties (partners and other vendors) to deliver value to customers. There are also ways to connect customers with each other so that they can offer information and advice to each other.

In this context, platforms are becoming more and more central to value creation and value delivery. Platforms help companies and organization leverage the power of ecosystems to grow and reach outstanding results that cannot be reached independently. Even smaller and niche economic sectors could be organized in platform-powered markets, and made to grow by exploiting factors like internationalization, network effects generation, improvement of the quality of services provided and consumed. This process enables also smaller producers to join the market, as they can easily connect with larger pools of demand.

This is a pattern of the modern economy: as long as one organization emerges with a platform that is capable to aggregate supply and demand successfully, and to create great vertically bundled experiences to monetize, new market networks can get quickly organized, grown, developed. Platforms tend to offer customers far more choice and flexibility in moving from one product or service to another. As customers become more and more powerful and experience pressure to increase their own performance, they will see more value in accessing platforms that expand their array of choices. But, from a platform provider viewpoint, platforms definitely require an evolution of the business model. In particular, business-related networks (i.e. B2B and B2B2C) are today less evolved compared to consumer (B2C) services, and they hold a great potential of growth if organized, in the perspective we introduced above.

As part of supporting a company to launch a new instantiation of the NIMBLE solution and create its own B2B platform, the NIMBLE project decided to adopt an existing toolkit: **The Platform Design Toolkit** (https://platformdesigntoolkit.com). The toolkit is based on a methodology and provides a set of canvases specifically devised to help designers, founders and managers to design strategies, products and organizations "as a platform". Everything is supported by a facilitation guide (https://platformdesigntoolkit.com/toolkit/), so that anyone can use it, and all the necessary materials (canvas) are available here: https://stories.platformdesigntoolkit.com/releasing-platform-design-toolkit-2-1-6d0a973e0ea9.

In this document, we give an introduction to the main steps. As a result, the potential new platform owner will obtain:

- envision and design how the system can work, creating and exchanging value among different stakeholders in the target market;

[1] http://www.marketingjournal.org/the-big-shift-in-business-models-john-hagel/

- a first design for the validation by creating a prototype (Minimum Viable Platform) or assumptions for validation experiments.

On the basis of these results, the potential new platform owner can have a look at the NIMBLE platform capabilities and available services and decide whether to adopt it in order to rapidly implement the envisioned MVP.

3.1 A good context for a Platform Strategy

A Platform strategy should be run by a *shaper* (the potential new platform owner) with the aim of mobilizing an ecosystem that creates value in interaction, with the aim of capturing part of this value. In most cases, this is about either evolving an existing organization or product and service offering, or exploring a new market and its opportunities.

In the Platform Design Toolkit two main contexts of application are foreseen:

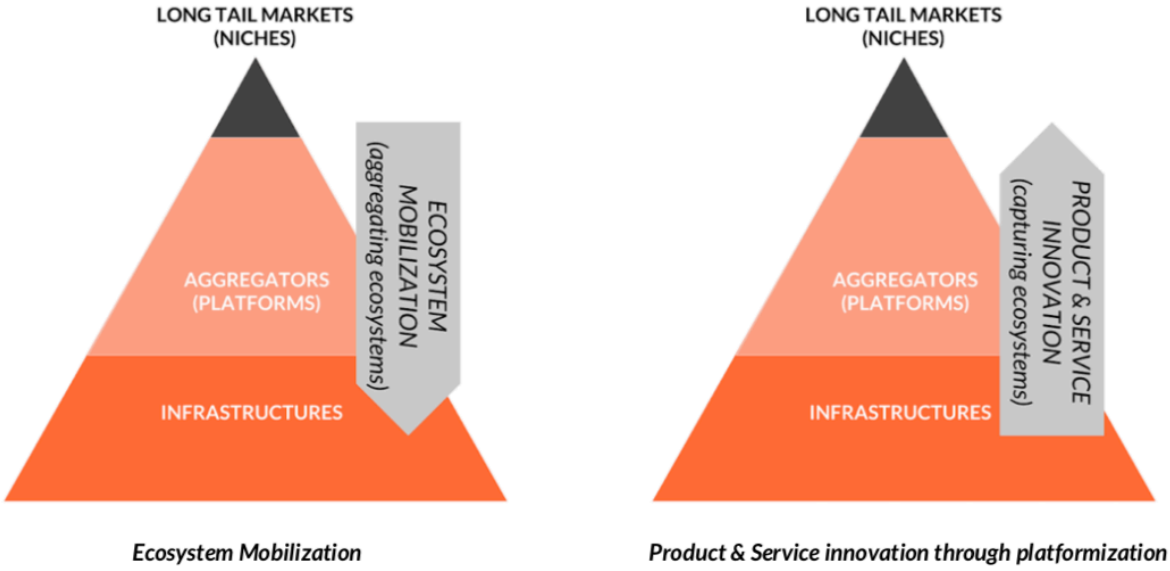


Figure 1 – Contexts of application of the Platform Strategy

- *Ecosystem Mobilization.* A common context of application of Platform Design is related to shaping and mobilizing ecosystems that already exist. If you see that value is being created and traded in a market; if you see producers and consumers that are self-organizing around value creation, and you think this market (context) is performing below potential, then this context is worth of organizing through **a platform strategy that amplifies its potential**.
- *Product & Service Innovation*. Another recurring case is that of a player trying to use a platform approach to organize a larger ecosystem of interactions that exists, or could exist, around existing products or services that the organization already provides. In this case there is already an ecosystem of entities using the product or service as a component of a value chain that leads to higher value services: the platform might better organize this ecosystem, facilitating higher value interactions.

3.2 The four main phases of Platform Design

The work of a platform designer can now be divided into four macro phases:

- **Exploration**: understanding the context, and the strategic meaning and applicability of a platform strategy that impacts, shapes and influences the context;

- **Strategy Design**: mapping entities, understanding their individual context, their potential to exchange value, and imagining the two *key platform engines* (the *transactions* engine, the *learning* engine), plus designing the *experiences* one wants to create for participants;
- **Validation** and **Prototyping**: conducting ecosystem analysis and testing (this could also partially happen during the design phase, and is generally an iterative process), making the MVPs or the experiments dedicated to validate or invalidate the assumptions;
- **Growth Hacking**: applying tactics to help the strategy grow in the context (being it a market, or something different) and achieve niche enablement and network effects.

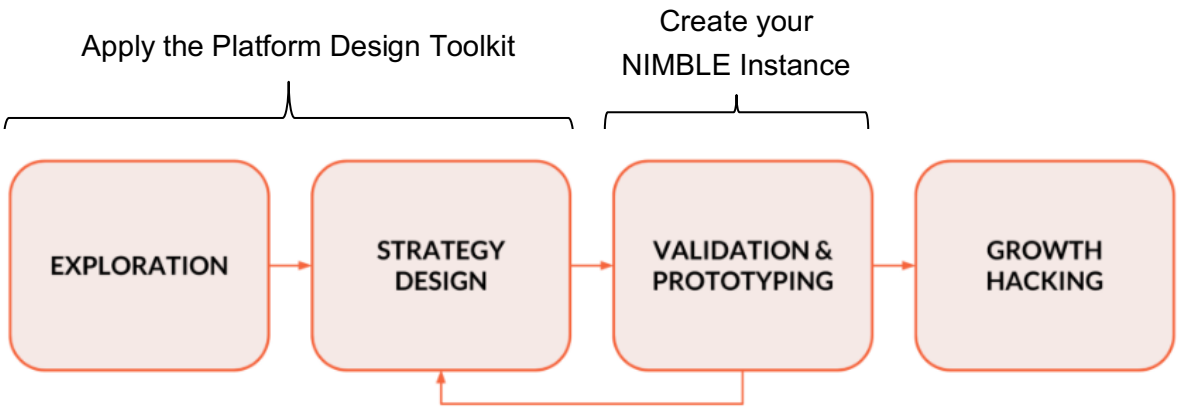


Figure 2 – The phases of Platform Design

The first two (exploration and strategy design) are the most relevant at this stage (and they will be covered in the following subsections), since they help the potential new platform owner to understand the context (market) to target, the services to activate and the values to capture.

The third step (validation & prototyping) is in fact the adoption of the NIMBLE platform for implementing and validating the devised platform strategy. How to do this will be covered in Section 4.

The forth step is related to a more specific market growth strategy, which is very sectorial/market-dependent and, thus, not covered in this document.

3.3 Exploration

Platform experiences are essentially a mix of direct interactions (transactions) *entity to entity,* and *platform to entity* provided services, designed to enable and empower, and to generate continuous learning and improvement.

In these regards, an organization that looks into a market to implement a, platform powered, ecosystem mobilization strategy, needs to understand what the ecosystem is trying to achieve, and in what ways. No matter whether an established platform (aggregator) player exists or not, in most cases four types of parties are involved:

- *peer producers* and *partners* (suppliers of value);
- *peer consumers* (consumers of value);
- *mediators* (like brokers, supporters, facilitators, aggregators);
- *infrastructures.*

Digital transformation makes the case for a dual behavior in markets: increasing fragmentation in marketplaces, and gradual consolidation in connecting layers. Producers and consumers normally inhabit the part of the market that is increasingly *fragmenting*: as long tails depend on personalized experiences, and niches, the market for any player living in the long tail is getting smaller and smaller, and more of these players are emerging every day.

Aggregators and infrastructures, on the other hand, are normally subject to concentration effects: *infrastructures* are naturally prone to concentration since they become ubiquitous utilities subject to economies of scale, and *aggregators* (platforms) as well—since they benefit from network effects—normally tend to concentrate.

Based on that, the first step is to map all entities and roles in the ecosystem to mobilize by means of the ECOSYSTEM SCAN CANVAS. A complete example is reported at the following link: https://stories.platformdesigntoolkit.com/exploring-ecosystems-the-patterns-of-platformization-6dd0eb6f95f3

This canvas has three "zones": the upper zone is about fragmented players and *peer relations*, while the middle and lower is about potentially concentrating players, aggregators and infrastructures. This is the space for a *"potential" platform strategy* to come into place, either as new or as displacing existing ones.

On this canvas, the aim is to quickly plot all the "situations" that make up the current experiences available in the target market.

3.4 Strategy Design

In this phase, the platform shaper maps and clusters existing entities, understands their individual context and explores the potential they have to exchange value among them.

Then, the platform shaper designs two key platform engines:

- the *Transactions Engine*, which is the set of channels and contexts specifically designed to facilitate interactions and exchanges between entities in the target context/market.
- the *Learning Engine*: which is the set of support services and contexts that the platform shaper provides and maintain for the participants so that they can learn, improve and evolve within the ecosystem.

Finally, the platform shaper selects the most, high-potential platform experience – along with its sustainability model (business model) – that can be brought to the context and iteratively validated with the ecosystem (following phase).

In order to achieve all of that, the strategy design phase is split into the following steps:

1. *Mapping the Ecosystem*. First, by using the Ecosystem Canvas the platform shaper will reflect on the ecosystem to shape, and organize with its platform strategy. The platform shaper will map the entities present in this ecosystem and will then understand what roles they might play, clustering them if necessary [ref. Platform Design Toolkit 2.1 – User Guide, page 16].

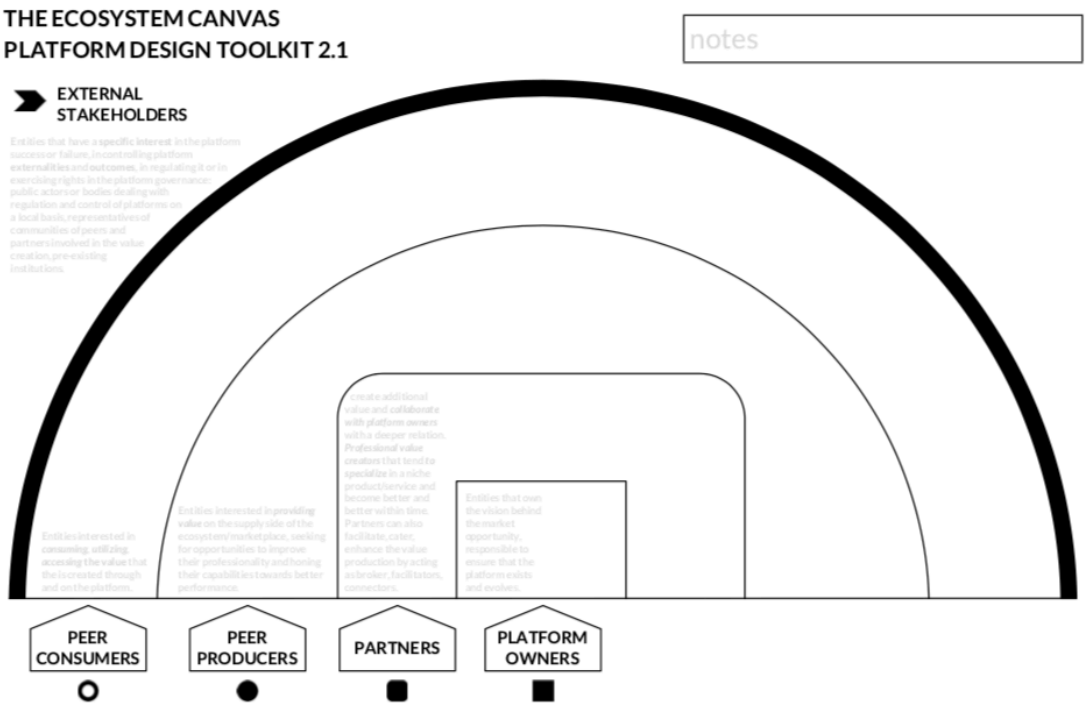


Figure 3 – Ecosystem Canvas

2. *Portraying Ecosystem's Entities*. In the Ecosystem Entity Portrait the platform shaper will make a consistent picture of the entities' context: what they're trying to achieve, with whom and how they're trying to connect, what potential they can express, and what kind of experience gains they're looking for - and therefore you should provide - as a platform shaper [ref. Platform Design Toolkit 2.1 – User Guide, page 18].

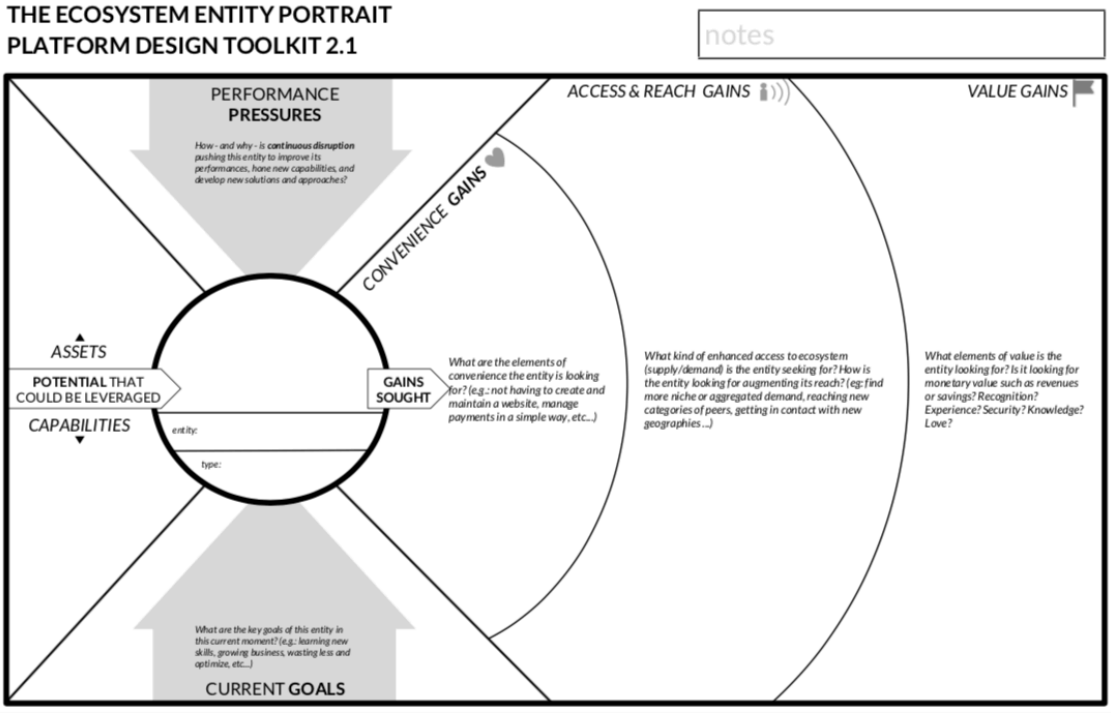


Figure 4 –Ecosystem Entity Portrait

3. *Analysing the potential to Exchange Value*. With the Ecosystem's Motivation Matrix the platform shaper will then analyse the potential to exchange flows of value: in other words it will map what kind of value exchanges the entities are performing already (or trying to), and what additional type of value they might exchange if properly enabled [ref. Platform Design Toolkit 2.1 – User Guide, page 20].

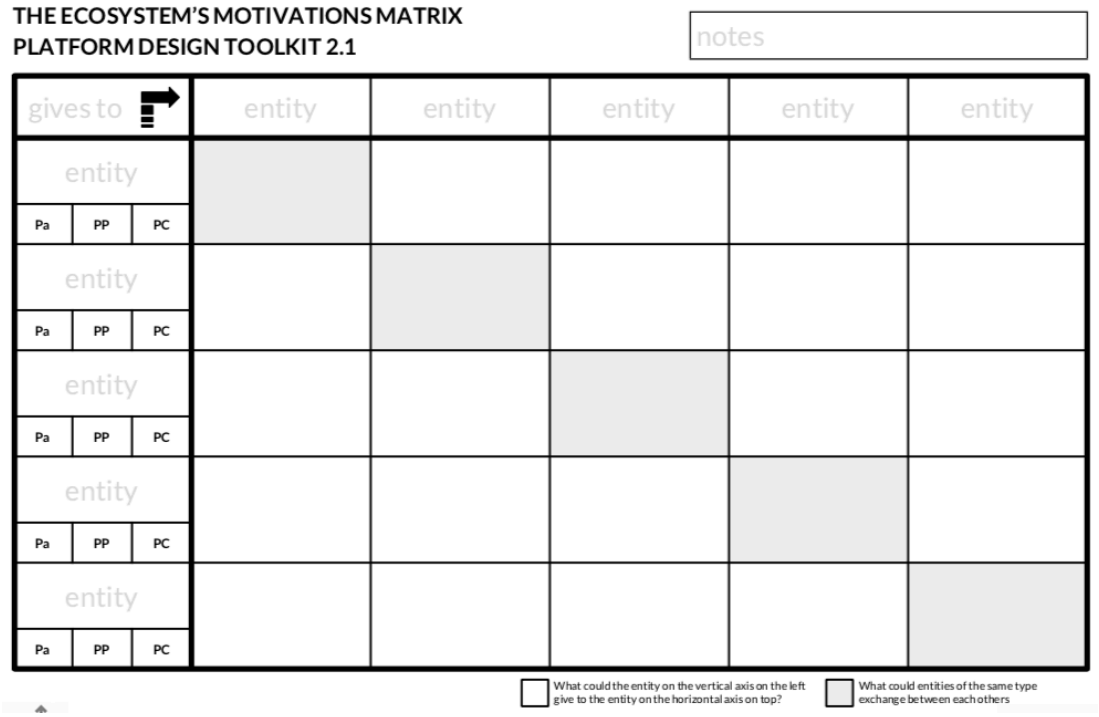


Figure 5 – Motivation Matrix Canvas

4. *Choosing the core relationships to Focus on*. At this point in the design process, it is important that the shaper identifies the focus: what are the entities to focus on in the ecosystem? What relationships are going to be the core of our design work (at least for a first iteration?). No specific canvas it used. Selection can be made by highlighting the target enties and the key relationships on the Ecosystem Canvas (created at step 1) [ref. Platform Design Toolkit 2.1 – User Guide, page 22].

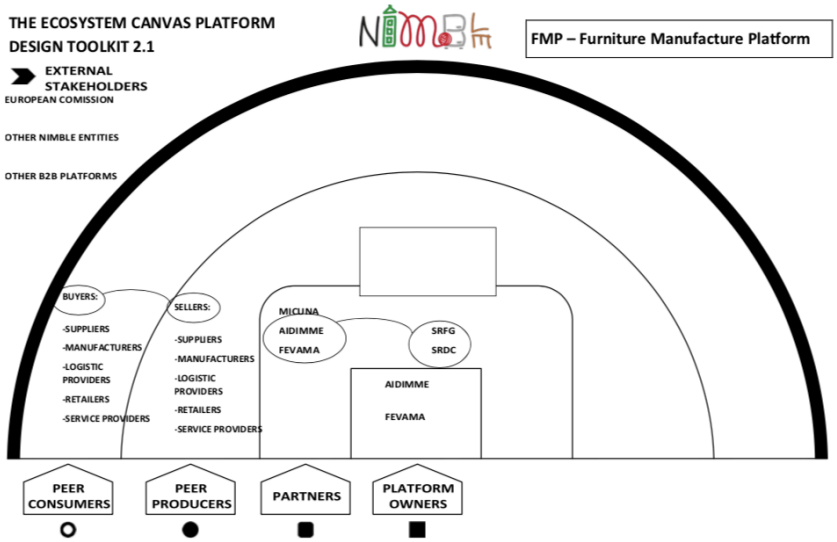


Figure 6 - Ecosystem Canvas with core relationships to focus on (example from the NIMBLE workshop, see D8.12)

5. *Identifying the Elementary Transactions*. With the Transactions Board, the platform shaper will map how the ecosystem is currently exchanging value (focusing on the entities and the relationships s/he decided to prioritize), and the platform shaper envisions how her/his platform strategy can help participants transact value in an *easier, cheaper* and *faster* way by providing, and curating channels and contexts that will make interactions and transactions more likely to happen [ref. Platform Design Toolkit 2.1 – User Guide, page 23].

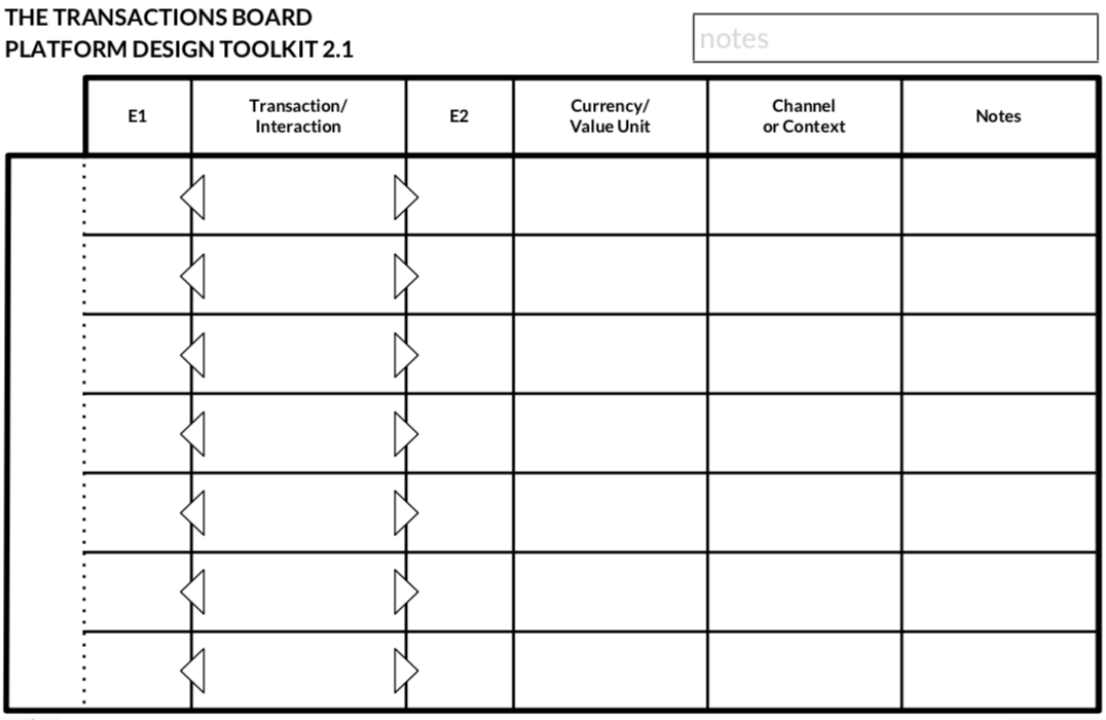


Figure 7 – Transaction Board Canvas

6. *Designing the Learning Engine*. With Learning Engine Canvas, the platform shaper will design a step by step process made of support/enabling services that will help her/his entities embrace your platform strategy. These services will help them evolve, emerge from the crowd, become better producers and consumers, and ultimately to undergo a radical evolution that will have them explore new opportunities, and behaviors not intended initially [ref. Platform Design Toolkit 2.1 – User Guide, page 25].

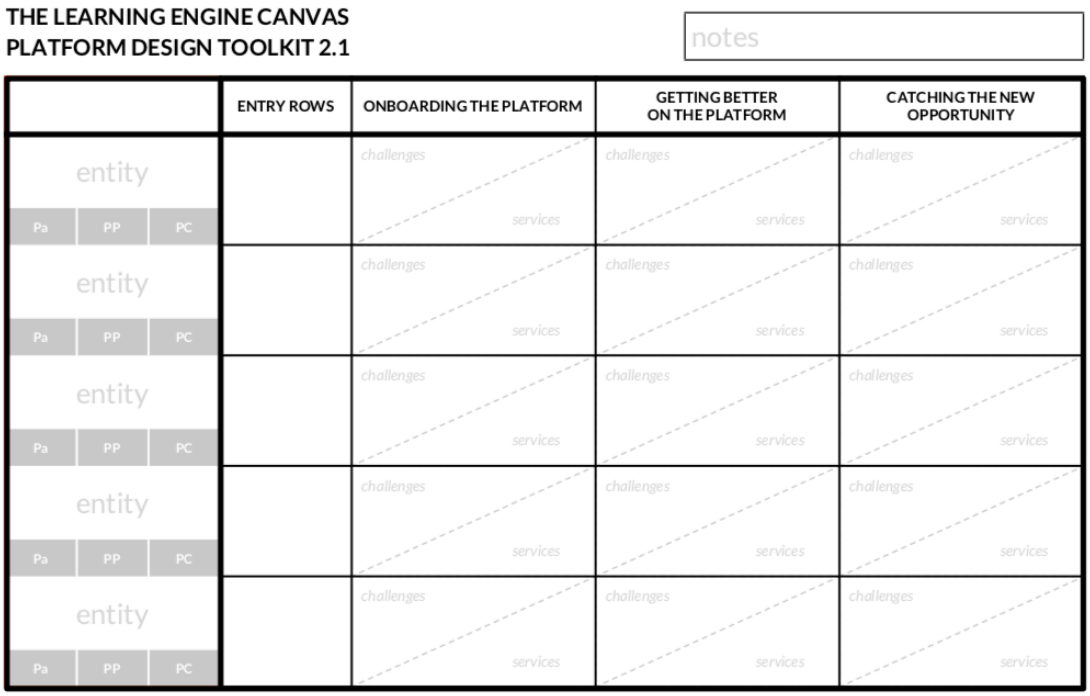


Figure 8 - Learning Engine Canvas

7. *Assembling the Platform Experiences*. With the Platform Experience Canvas, the platform shaper crafts an experience that synthesizes the core value proposition(s) arising from the Strategic Design phase and that - more than others - the platform shaper considers essential for the platform strategy. With this canvas, the platform shaper will assemble the elements emerged from the Transactions Board(s) and the ones emerged from Learning Engine Canvas. The platform shaper will then reflect on the sustainability model of this experience, covering the basic elements of Business Modeling, s/he will consider what resources and components will have to be set in place and managed in order to deliver this experience, and how the platform shaper will extract value from it [ref. Platform Design Toolkit 2.1 – User Guide, page 28].

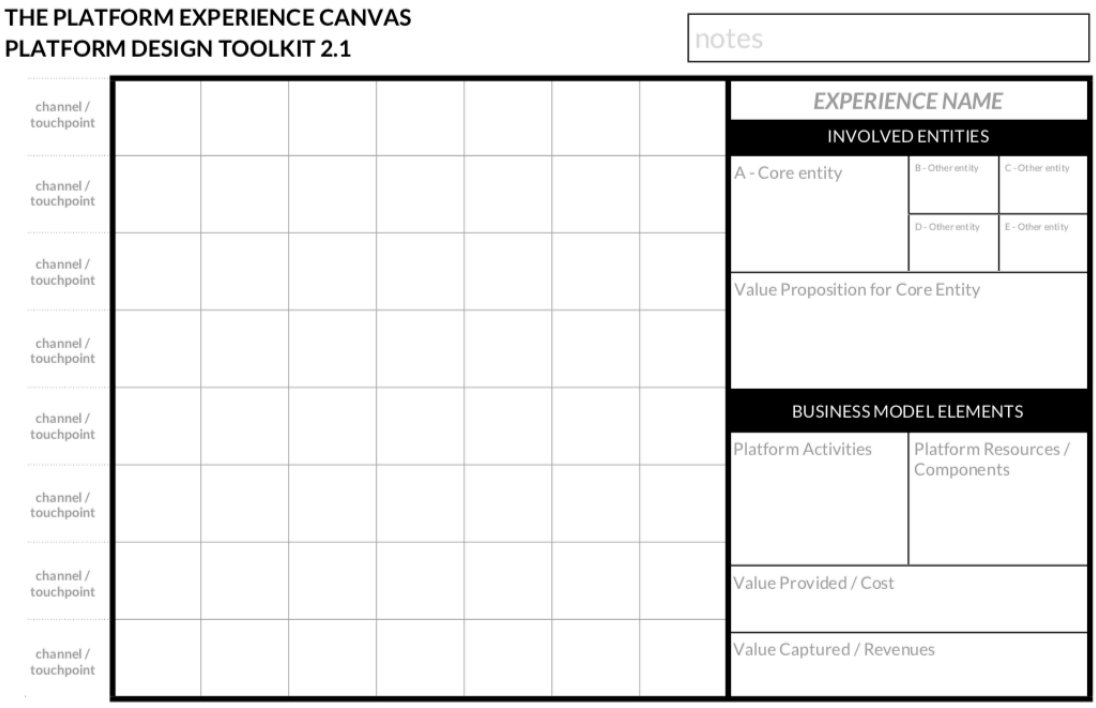


Figure 9 – Platform Experience Canvas

8. *Setting up the Minimum Viable Platform*. With the Minimum Viable Platform Canvas, the platform shaper finally *moves out of the building* to test in the real world if all the created design assumptions have a future or not. By looking at the design outputs, especially the Platform Experience Canvas(es), the platform shaper will extract the riskiest assumptions in her/his strategy, and s/he will set up experiments and metrics to validate them with the target ecosystem [ref. Platform Design Toolkit 2.1 – User Guide, page 31].

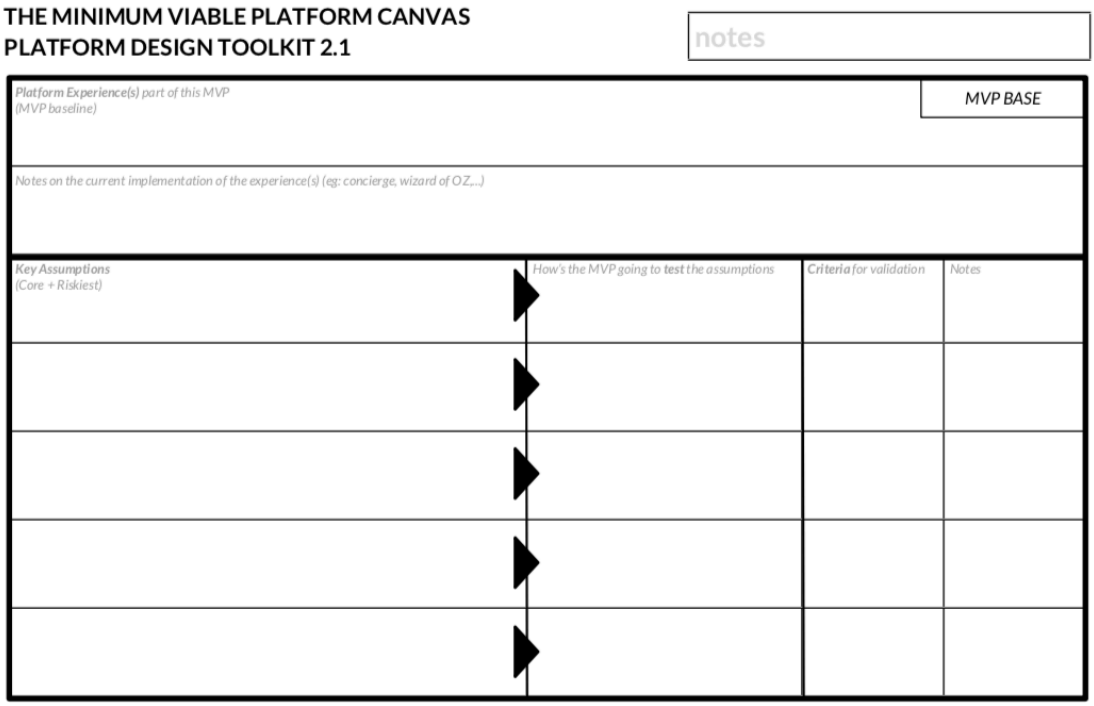


Figure 10 – Minimum Viable Platform Canvas

4 The NIMBLE Platform Opportunity

Following the steps reported in Section 3, a potential new platform owner has defined her/his own Platform Strategy and, specifically, has designed a target *Minimum Viable Platform* which needs to be deployed and validated. This implies development efforts and, thus, significant investment of resources.

The NIMBLE project has developed and started to validate, a novel, cloud based, real time and easy-access platform that will facilitate the establishing of dynamic supply networks for many classes of stakeholders in future collaborative manufacturing.

Specifically, the resulting platform is a *manufacturing B2B service delivery framework*, which is open source (Apache License 2.0), extensible and adaptable to multiple contexts. Moreover, developers may interact with the platform by using a comprehensive API set, giving them the possibility to extend the core services of the platform with valued added services and new tools for platform customers. The API set will mainly include access to back-end capabilities, but also business collaboration and federation interaction will be made possible via the APIs.

The following pictures provides a high-level view of the key NIMBLE elements.

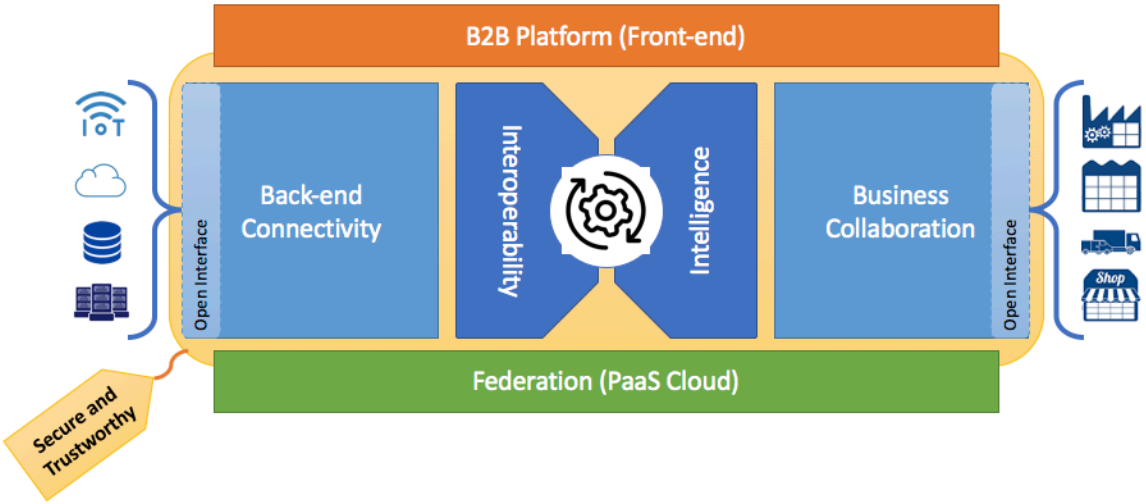


Figure 11 - NIMBLE Platfrom overview and main elements

Therefore, **NIMBLE represents a valid starting point for entities that aim to develop a next-generation digital platform for a target manufacturing sector**. The permissive open source approach and the standards (taxonomies/ontologies) at its core will enable potential new platform owners to:

- benefit from a ready-to-use solution;
- use and customize NIMBLE as they wish;
- to interoperate with other platforms/solutions using the same standards.

In the following subsections, we will detail the technical aspects of the NIMBLE platforms, in terms of: core technologies and services; how to deploy a new NIMBLE instance; customization and extension levels. We will cover:

- current and new releases,
- code repositories,
- and documentation

Please note that additional information is always available at the following page: https://www.nimble-project.org/software-documentation/

4.1 NIMBLE Platform Core Services and Technology

NIMBLE's technical architecture is built following the microservice paradigm, and individual microservices were designed according to the domain-driven design approach.

Several infrastructural components are necessary in order to run the application. The following figure depicts applied infrastructure services.

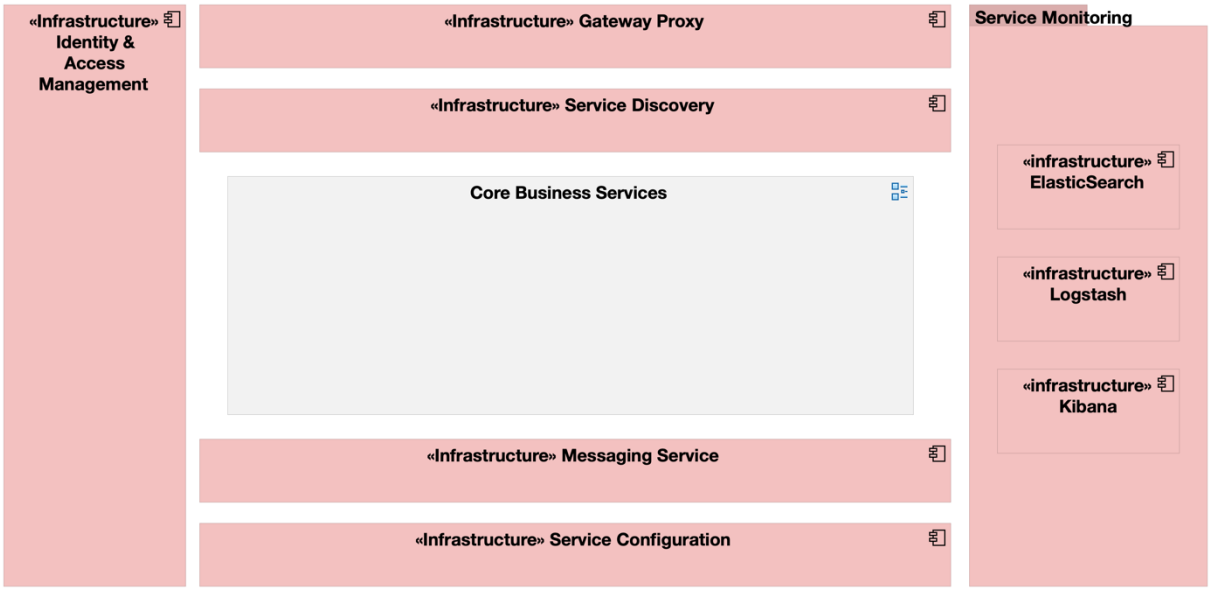


Figure 12: Infrastructure components for NIMBLE core services.

Spring Cloud is mainly used as technology and framework for building infrastructure services, which are further described in the table below.

Gateway Proxy	Single entry point for accessing individual microservices, which provides dynamic routing, security features and advanced filtering capabilities. It is based on Netflix's Zuul[2].
Service Discovery / Registration	Each microservice registers its instances to a central service registry during start-up. This registry can be used by each microservice for discovering other microservices in the same infrastructure. Spring Eureka[3] (Netflix OSS) is used as underlying technology.
Service Configuration	This component stores the configuration of the platform in a centralized manner. Each microservice fetches its respective configuration from the configuration service during start-up.

[2] https://github.com/Netflix/zuul

[3] https://github.com/spring-cloud/spring-cloud-netflix/tree/master/spring-cloud-netflix-eureka-server

Identity & Access Management	Security related identities on the platform are managed in a central Identity & Access Management service (based on RedHat's Keycloak[4]) in order to enable single sign on capabilities throughout the microservices.
Log Aggregation	Log streams for each microservice are aggregated using the Elasticstack[5] composition.

4.1.1 Core Business Microservices

The actual core services of NIMBLE consists of the following set of microservices.

Name	Description	Responsibilities
Frontend Service	This service provides the web-based graphical user interface. Each request from the user is delegated to other services (e.g. registration requests are delegated to the Identity Service).	• Provides GUI (HTML5 / Javascript) • Central receiver for user interactions/requests • Delegates requests to other services
Identity Service	Identities on the platform are administered by this service, which plays a vital role in terms of security. This service communicates with the Identity & Access Management stated in the microservice infrastructure above. Identities are defined as entities, which perform certain actions on the platform (i.e. users, companies and autonomous agents).	• Receives requests for CRUD operations of user entities • Receives requests for CRUD operations of company entities • Handles logins of users • Communicates with the UAA & Identity Management service • Stores company related data/configuration
Catalog Service	Stores products / services persistently and manages the underlying ontology.	• Stores products and user services persistently • Provides search capabilities • Provides functionalities for CRUD actions for products and user services
Business Process Service	Functionalities for collaborative execution of modelled business processes are provides by this service.	• Provides Business as a Service functionalities • Collaboration on business services

[4] https://www.keycloak.org/

[5] https://www.elastic.co/

Indexing Service	Search and indexing are based on this microservice.	• Consumes an OWL-Based ontology and provide insights w.r.t. the ontology by indexing OWL-Classes and OWL-Properties. • Consumes a product catalogue and provides the index information for (faceted) search of products. • Maintains an index for manufacturers including their trust score values
Trust Service	Trust rating for companies are computed and manged by this microservice.	• Defined computation model for trust score calculations • Updates trust score on requests according to updated data

4.1.2 Backing Services

Backing services in the applied microservice architecture are defined as services which are living on its own and are mainly used in the background by just one microservice. In addition to microservices the following backing services are part of the technical setup.

Red Hat Keycloak

Keycloak is an open source identity and access management (IAM) tool that enables single-sign-on capabilities throughout the microservices. It is compliant with state-of-the-art security standards (e.g. OAuth 2.0 and OpenID Connect). This backing service is used solely by the identity service.

Apache Kafka

Apache Kafka[6] is a technology for stream-processing. It is developed under the permissive Apache 2.0 licence and provides real-time data feeds between participants. In the applied infrastructure Kafka is used to broadcast updates on between microservices. It is also the foundation of the messaging layer for data channel capabilities.

Elastic Stack

The Elastic Stack[7] (formerly known as ELK stack) combines the following three solutions in order to aggregate and analyse distributed log streams. Logstack is the receiver for individual log streams coming from single microservices or backing services. They are forwarded to ElasticSearch, which stores the logs persistently in a NoSQL database. Kibana provides visual tools for log exploration and analysis.

Camunda BPM

[6] http://kafka.apache.org/

[7] https://www.elastic.co/products/elasticsearch

Camunda[8] is an open source business process engine, which supports multiple standardised notations and execution for business and decision workflows. It is deeply integrated into the business process microservice via the provided Spring Boot starter package.

Apache Solr

Apache Solr[9] is an open source platform for enterprise search capabilities. It supports full-text search, faceted search and real-time indexing. Catalog and company related data is indexed using Solr in order to provide proper search functionalities. The indexing microservice is mainly using this backing service in the background.

4.2 Create a new NIMBLE platform instance

The high level technical architecture of the NMIBLE platform can be seen in Figure 13 below. The platform can be split into 4 layers:

- *Data Ingestion* – In the NIMBLE platform this is achieved mainly via the front-end, through which most of the data flows into the system. This could be achieved via a programmatic API as well.
- *Platform services* – Generic Back-end cloud based services to support the platform capabilities, such as storage and messaging
- *Business layer* – the components that implement the specific capabilities of the platform.
- *External layer* – the means for interaction between the platform internals and external users.

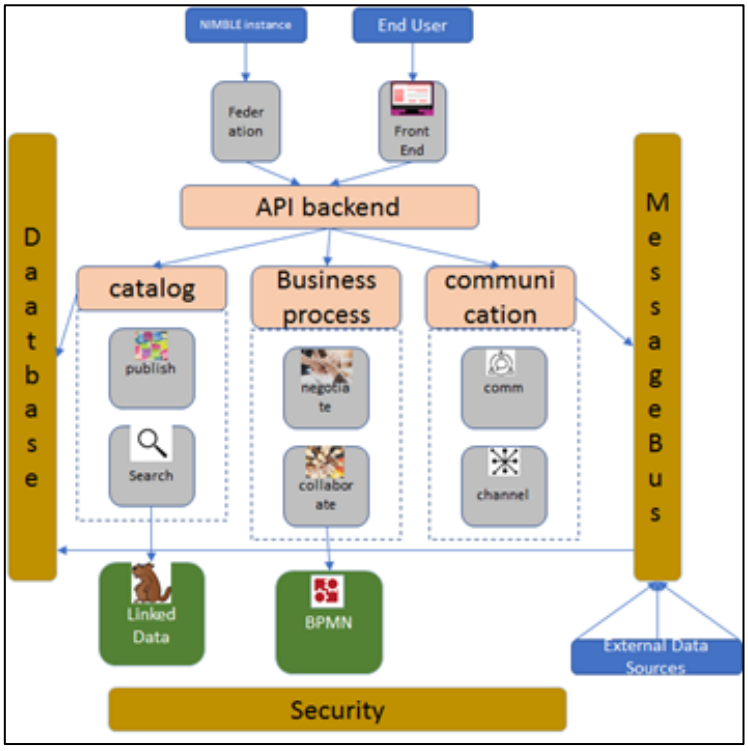


Figure 13: NIMBLE high-level architecture

[8] https://camunda.org/

[9] http://lucene.apache.org/solr/

As described earlier the main design pattern behind the architecture is that of microservices. Thus, the specific platform capabilities are developed and deployed as microservices. In NIMBLE, the microservices get deployed into a cloud based microservices orchestration cluster, namely Kubernetes. In Figure 14, we can see the overall deployment picture. At the heart lies a Kubernetes cluster deployed and currently supported over the IBM cloud. All the platform components are deployed independently within the Kubernetes cluster. In the background, but still on the cloud, are the back-end services supporting the microservices in storage and messaging services.

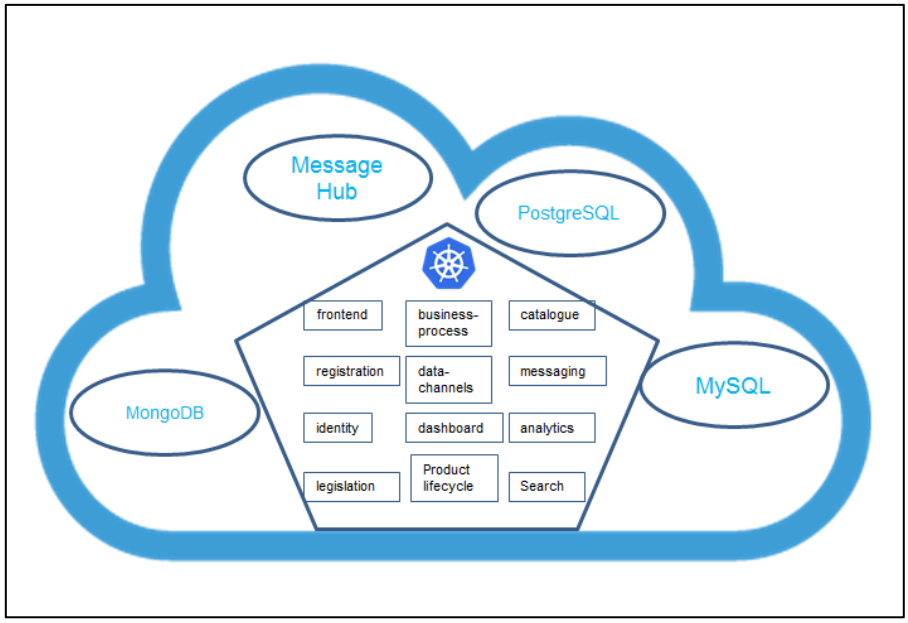


Figure 14: NIMBLE microservices deployment

For the NIMBLE instance to be deployed, first there is a need to have the back-end services up and running. In addition, a Kubernetes cluster should be set up to be able to host the individual microservices. Finally, the connection information to the back-end services should be made available from within the Kubernetes cluster, such that the deployed microservices can bind to the services they require for their proper operation. The code for all the microservices resides in GitHub (https://github.com/nimble-platform), and the system can be directly built from there.

An automatic Continuous Integration tool chain is in place, which kicks in upon code changes (or releases declared) in GitHub. Further down the chain, Jenkins (https://jenkins.io/) is used to further automate the build and deployment of the microservices. For the NIMBLE platform setup, an instance of Jenkins should be run in the target Kubernetes cluster, performing respective operations on the cluster itself.

The CI consists of:

- Triggered by changes in GitHub
- Followed by reading the respective Jenkins file and operating accordingly
- Executing a build on the Kubernetes cluster

Further operations either to build from source or to deploy existing docker images.

When building from source the following steps are executed by Jenkins on the Kubernetes cluster:

- Pulling code from GitHub
- Executing tasks based on Jenkins file which under normal circumstances include:
    - Building a Docker image from the course file (based on the Docker File)

- Push the created image to Docker Hub
- Pull the image from Docker Hub and deploy it on the cluster

Alternatively the platform can be deployed from pre-built Docker images (without building from source). In that case the required image file name should appear in the Kubernetes configuration file (under the spec / containers / image tag).

The important deployment and configuration files can be seen in Figure 15:

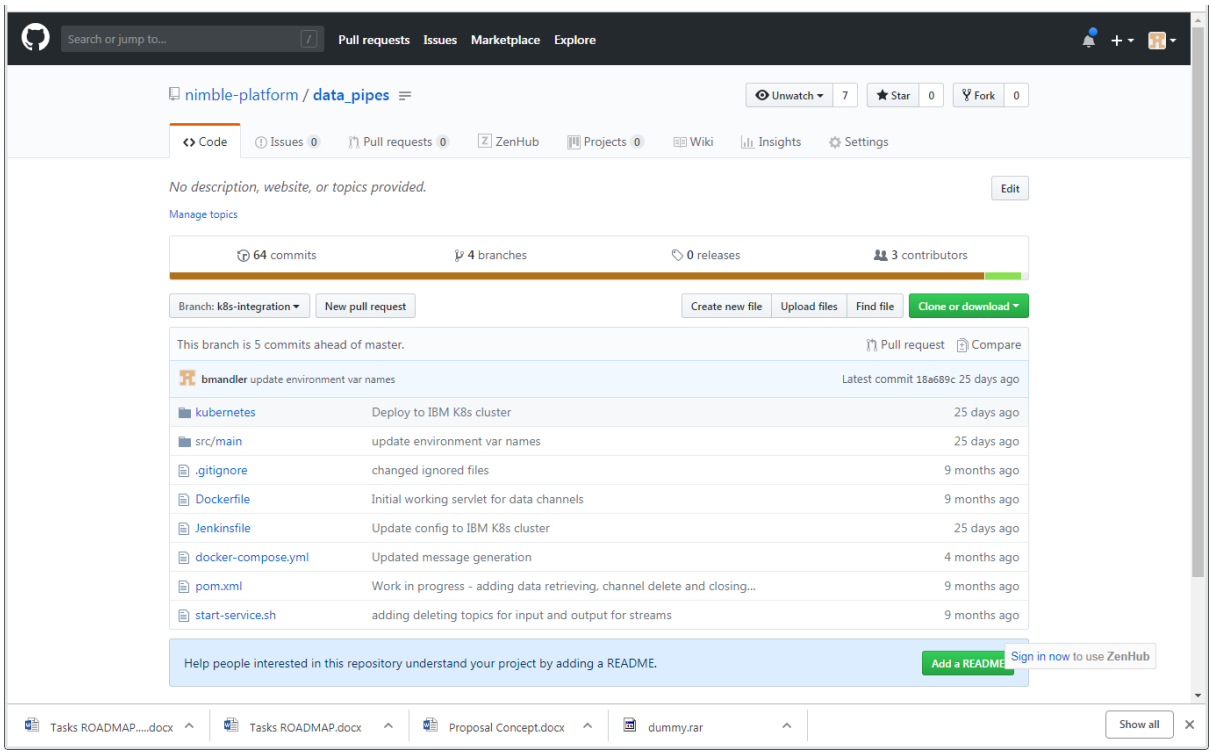


Figure 15: Deployment configuration files

1. Docker file – determines the manner in which the Docker image implementing the microservice shall be built
2. Jenkins File – determines the steps required to deploy the microservice on the cluster
3. Kubernetes
    a. Deploy file – describes the microservice to be deployed
    b. Service file – describes the Kubernetes service for the specific component.

4.3 NIMBLE Platform configurations and possible customisations

As anticipated at the beginning of Section 4, the NIMBLE platform can be adapted and customised/extended in order to better serve a specific ecosystem/market. In the following, we briefly report the main available configuration mechanisms.

4.3.1 Configurable Platform Settings

All created instances of NIMBLE (see Section 4.2) share the same code base, whereas instance specific adaptions can be enabled/disabled via the following configurable feature flags:

- Company registration required
- Definition and filtering of taxonomies
- Definition of available terms in registration and company settings
- Definition of supported user roles

- Company members visible to all company members
- Definition of platform logo
- Feature switch for
    - Explorative search
    - PPAP functionality
    - Track & Trace
    - Verification Info
    - Data channel integration

4.3.2 Integrating Additional Product Category Taxonomies

The NIMBLE platform could be extended with additional, domain-specific category taxonomies for proper annotation of the products.

The indexing service provides alternative ways for integrating such a taxonomy as described briefly as follows:

The taxonomy could be represented in OWL format and the */ontology* endpoint of the service could be used to import the taxonomy as a whole.

The categories (which correspond to class concept in the indexing service terminology) and properties can be imported one-by-one using the */class* and */property* endpoints respectively.

Once a taxonomy is indexed using one of the aforementioned ways, it automatically becomes usable in product publishing.

4.3.3 Business Process Settings

NIMBLE employs the Camunda business process engine (see Section 4.1.2) to design and execute business processes within the platform (e.g. product purchase negotiations).

Each core business process type supported in the default implementation of NIMBLE has been defined in a modular way; specifically, as a single bilateral message exchange between the trading companies. Accordingly, management of business process metadata as well as the front-end flows enabling the realization of these processes are tightly integrated with this approach.

As of writing this documentation, in order to extend the core business processes with additional ones, primarily the business process backend should be extended. If the new business process is intended to be used via the NIMBLE UI, the front-end should also be extended.

Extending the Business Process Service Backend

First of all, a BPMN representation for the new business process should be defined. Camunda Modeller is likely to be convenient for this. Once the process is designed in a way that describes the sequence of interaction between the involved actors, an implementation class for each regular task as a reference to the Java class is to be defined in the backend. Accordingly, the backend functionality should be extended with the specified task implementations. D3.4[10] provides further details about how built-in business processes are designed.

While designing the process, the REST services to start (https://nimble-platform.salzburgresearch.at/nimble/business-process/swagger-ui.html#!/start-controller/startProces-

[10] https://www.nimble-project.org/wp-content/uploads/2018/06/NIMBLE_D3_4.pdf

sInstanceUsingPOST) and continue (https://nimble-platform.salzburgresearch.at/nimble/business-process/swagger-ui.html#!/continue-controller/continueProcessInstanceUsingPOST) the business process should be taken into account. This mainly indicates that the process should be designed in a way that it allows step by step execution of the inquiry and response activities.

As the final step, the BPMN representation of the business process should be added to the "src/main/resources/bpmn" directory included in the business process service code base so that it can be fetched by the Camunda Business Process Engine.

The online documentation[11] includes details about the REST API to instantiate (i.e. start the endpoint referred above) instances of the new business process.

[11] https://www.nimble-project.org/wp-content/uploads/2019/01/Business-Process-Service-REST-API.docx

4.4 APIs for 3rd Party Services

Third party services can access the platform via the OpenAPI (see documentation below), whereas certain endpoints require authorised requests.

NIMBLE uses the OAuth2.0 standard for authorising requests throughout the platform, which follows a token-based authorisation schema.

The access token can be retrieved from the identity service (login endpoint) or directly from Keycloak. If the token is fetched from Keycloak standard OAuth2.0 workflows can be used.

API Documentation

All documentation is provided in Swagger[12], which provides structured API definitions (JSON) and an interactive web-based user interface for fast access. Each microservice has a separate API documentation. The following section lists the public API endpoints for the MVP instance.

- **Identity Service**:
  https://nimble-platform.salzburgresearch.at/nimble/identity/swagger-ui.html
- **Catalog Service**
  https://nimble-platform.salzburgresearch.at/nimble/catalog/swagger-ui.html
- **Business Process Service**
  https://nimble-platform.salzburgresearch.at/nimble/business-process/swagger-ui.html
- **Trust Service**
  https://nimble-platform.salzburgresearch.at/nimble/trust/swagger-ui.html
- **Data Channel Service**
  https://nimble-platform.salzburgresearch.at/nimble/data-channel/swagger-ui.html
- **Aggregation Service**
  https://nimble-platform.salzburgresearch.at/nimble/data-aggregation/swagger-ui.html

4.4.1 Identity Service API - Overview

Figure 16 shows the web UI of the identity service for the MVP instance.

https://nimble-platform.salzburgresearch.at/nimble/identity/swagger-ui.html

[12] https://swagger.io/

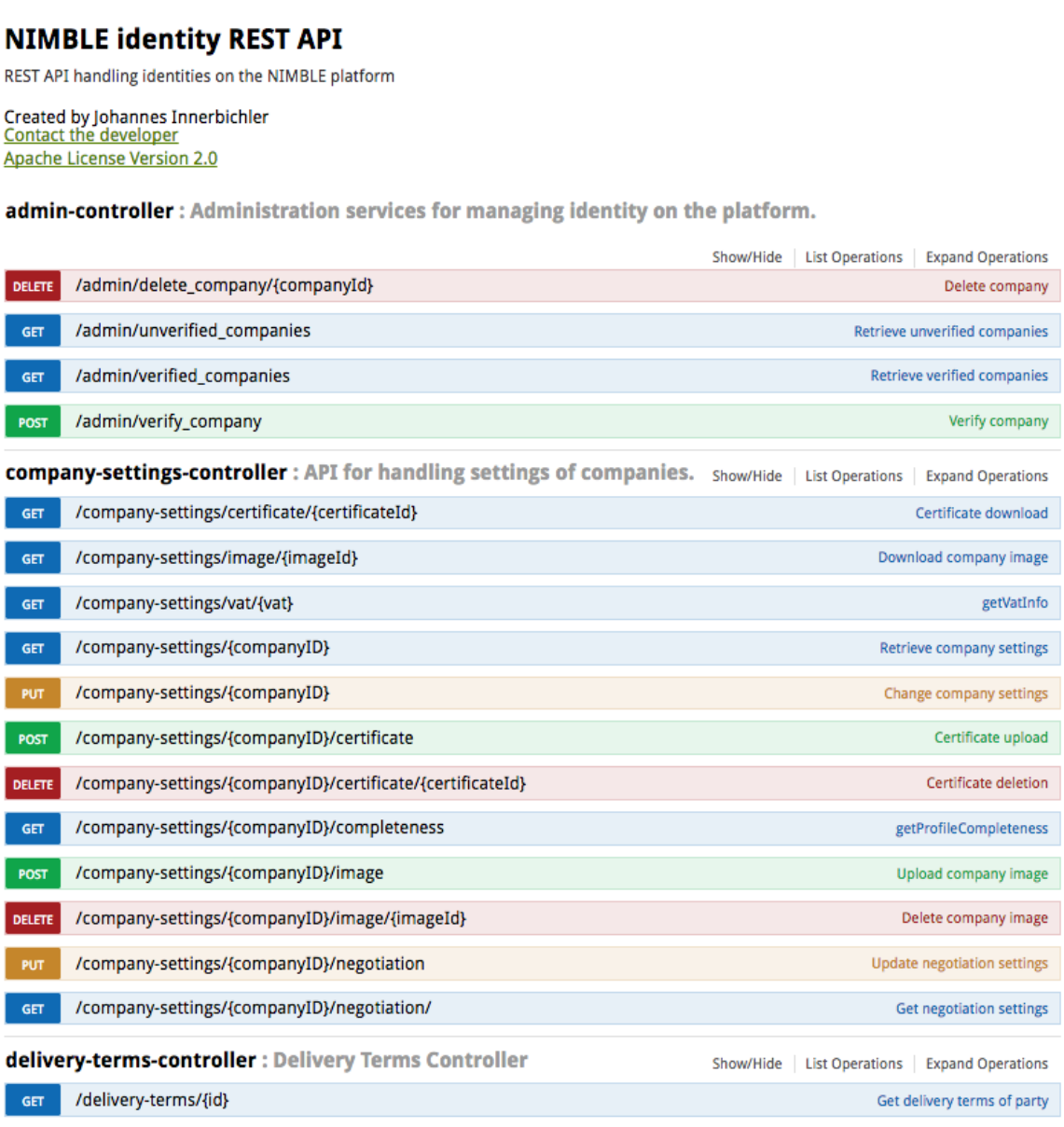


Figure 16: Swagger-based API documentation of the identity service (part 1 of 3)

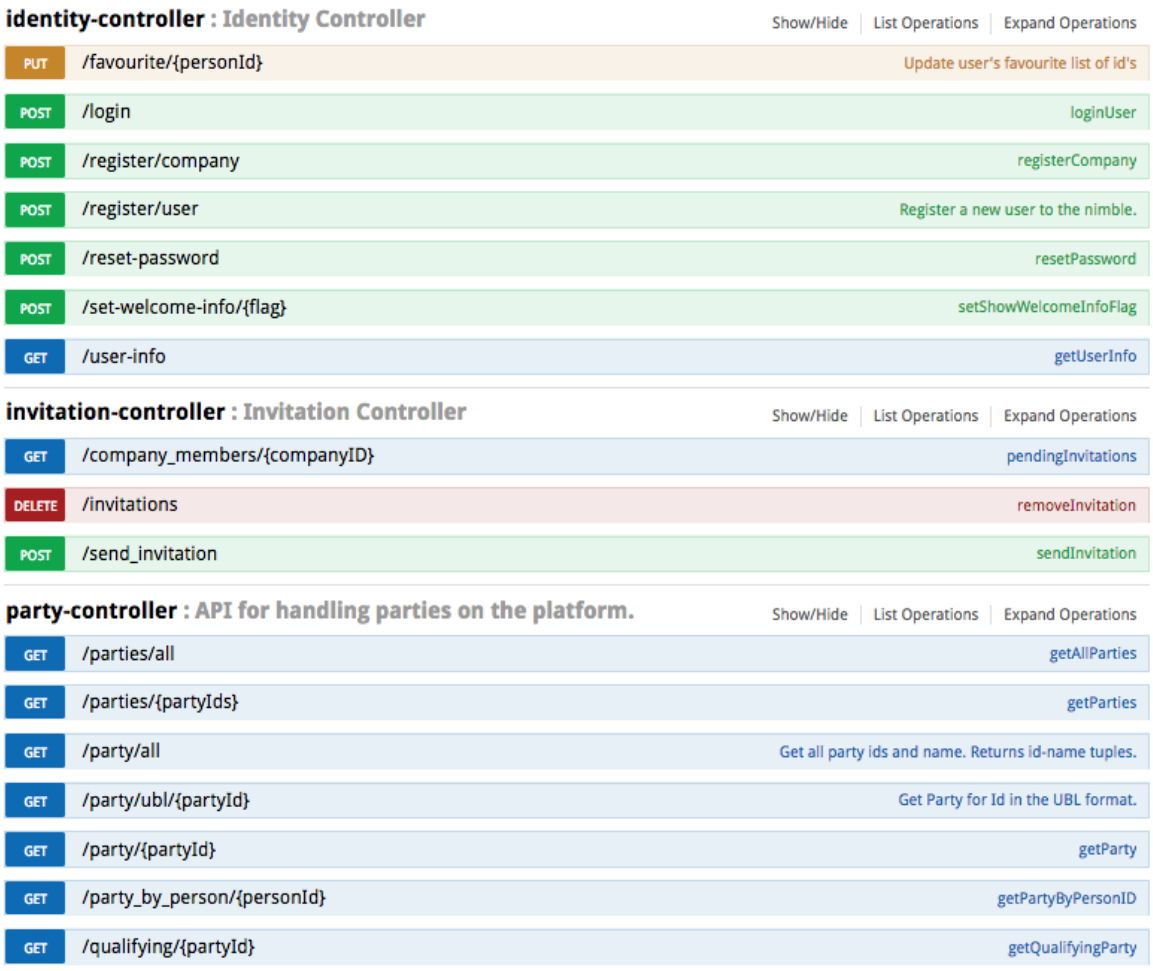
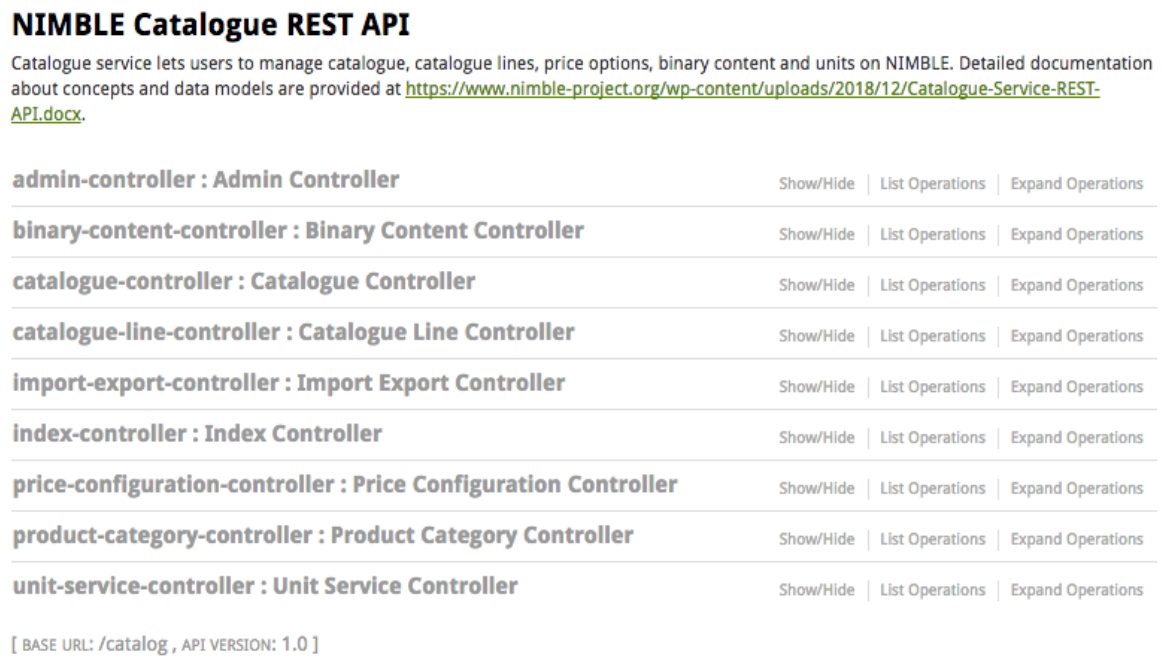


Figure 17: Swagger-based API documentation of the identity service (part 2 of 3)

Figure 18: Swagger-based API documentation of the identity service (part 3 of 3)

4.4.2 Catalogue Service API - Overview



NIMBLE Catalogue REST API - Details

NIMBLE Catalogue REST API

Catalogue service lets users to manage catalogue, catalogue lines, price options, binary content and units on NIMBLE. Detailed documentation about concepts and data models are provided at https://www.nimble-project.org/wp-content/uploads/2018/12/Catalogue-Service-REST-API.docx.

admin-controller : Admin Controller — Show/Hide | List Operations | Expand Operations

GET	/admin/add-class	getDefaultCatalogue

binary-content-controller : Binary Content Controller — Show/Hide | List Operations | Expand Operations

GET	/binary-content	getBinaryContent
GET	/binary-content/raw	getBase64BinaryContent
GET	/binary-contents	getBinaryContents

catalogue-controller : Catalogue Controller — Show/Hide | List Operations | Expand Operations

GET	/catalogue/semantic/{uuid}	getCatalogueInSemanticFormat
GET	/catalogue/standards	getSupportedStandards
GET	/catalogue/template	downloadTemplate
POST	/catalogue/template/upload	uploadTemplate
GET	/catalogue/{partyId}/default	getDefaultCatalogue
GET	/catalogue/{partyId}/pagination/default	getDefaultCataloguePagination
POST	/catalogue/{standard}	addCatalogue
PUT	/catalogue/{standard}	updateCatalogue
DELETE	/catalogue/{standard}/{uuid}	deleteCatalogue
GET	/catalogue/{standard}/{uuid}	getCatalogue
GET	/catalogue/{uuid}/delete-images	deleteImagesInsideCatalogue
POST	/catalogue/{uuid}/image/upload	uploadImages

catalogue-line-controller : Catalogue Line Controller — Show/Hide | List Operations | Expand Operations

POST	/catalogue/{catalogueUuid}/catalogueline	addCatalogueLine
PUT	/catalogue/{catalogueUuid}/catalogueline	updateCatalogueLine
DELETE	/catalogue/{catalogueUuid}/catalogueline/{lineId}	deleteCatalogueLine
GET	/catalogue/{catalogueUuid}/catalogueline/{lineId}	getCatalogueLine
GET	/catalogueline/{hjid}	getCatalogueLineByHjid
GET	/cataloguelines	getCatalogueLinesByHjids

import-export-controller : Import Export Controller — Show/Hide | List Operations | Expand Operations

GET	/catalogue/export	exportCatalogue
POST	/catalogue/import	importCatalogue

index-controller : Index Controller — Show/Hide | List Operations | Expand Operations

DELETE	/catalogue/index/item	clearItemIndex

price-configuration-controller : Price Configuration Controller — Show/Hide | List Operations | Expand Operations

POST	/catalogue/{catalogueUuid}/catalogueline/{lineId}/price-options	addPricingOption
PUT	/catalogue/{catalogueUuid}/catalogueline/{lineId}/price-options	updatePricingOption
DELETE	/catalogue/{catalogueUuid}/catalogueline/{lineId}/price-options/{optionId}	deletePricingOption

product-category-controller : Product Category Controller — Show/Hide | List Operations | Expand Operations

GET	/categories	getSpecificCategories
GET	/taxonomies/id	getAvailableTaxonomyIds
GET	/taxonomies/{taxonomyId}/categories	getCategoriesByName
GET	/taxonomies/{taxonomyId}/categories/children-categories	getChildrenCategories
GET	/taxonomies/{taxonomyId}/categories/tree	getCategoryTree
GET	/taxonomies/{taxonomyId}/root-categories	getRootCategories

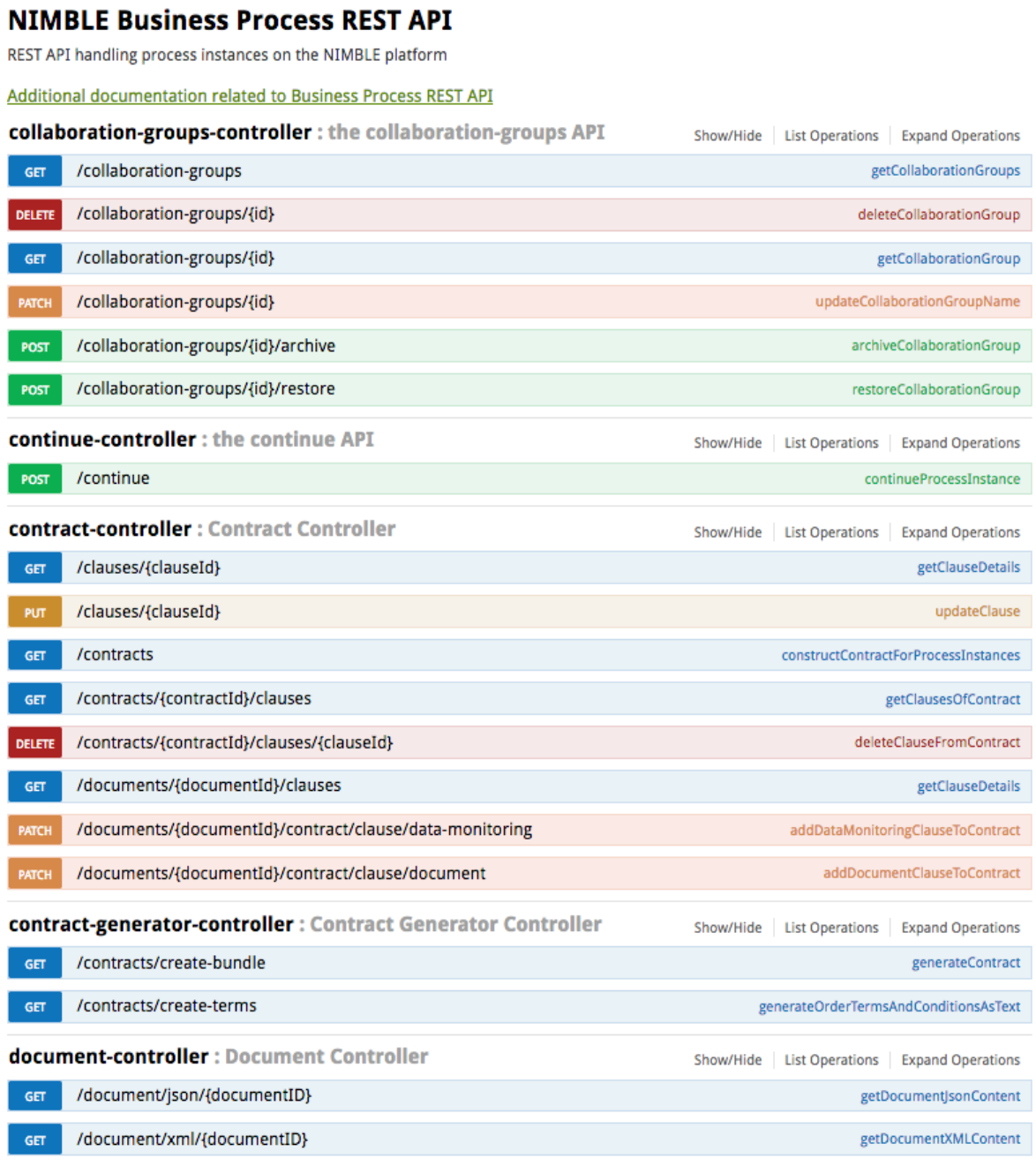
unit-service-controller : Unit Service Controller — Show/Hide | List Operations | Expand Operations

GET	/unit-lists	getAllUnitLists
POST	/unit-lists	addUnitList
GET	/unit-lists/{unitListId}	getValues
PATCH	/unit-lists/{unitListId}	addUnitToList
DELETE	/unit-lists/{unitListId}/unit/{unit}	deleteUnitFromList

[BASE URL: /catalog , API VERSION: 1.0]

4.4.3 Business Process Service

https://nimble-platform.salzburgresearch.at/nimble/business-process/swagger-ui.html



NIMBLE Business Process REST API

REST API handling process instances on the NIMBLE platform

Additional documentation related to Business Process REST API

collaboration-groups-controller : the collaboration-groups API — Show/Hide | List Operations | Expand Operations

GET	/collaboration-groups	getCollaborationGroups
DELETE	/collaboration-groups/{id}	deleteCollaborationGroup
GET	/collaboration-groups/{id}	getCollaborationGroup
PATCH	/collaboration-groups/{id}	updateCollaborationGroupName
POST	/collaboration-groups/{id}/archive	archiveCollaborationGroup
POST	/collaboration-groups/{id}/restore	restoreCollaborationGroup

continue-controller : the continue API — Show/Hide | List Operations | Expand Operations

POST	/continue	continueProcessInstance

contract-controller : Contract Controller — Show/Hide | List Operations | Expand Operations

GET	/clauses/{clauseId}	getClauseDetails
PUT	/clauses/{clauseId}	updateClause
GET	/contracts	constructContractForProcessInstances
GET	/contracts/{contractId}/clauses	getClausesOfContract
DELETE	/contracts/{contractId}/clauses/{clauseId}	deleteClauseFromContract
GET	/documents/{documentId}/clauses	getClauseDetails
PATCH	/documents/{documentId}/contract/clause/data-monitoring	addDataMonitoringClauseToContract
PATCH	/documents/{documentId}/contract/clause/document	addDocumentClauseToContract

contract-generator-controller : Contract Generator Controller — Show/Hide | List Operations | Expand Operations

GET	/contracts/create-bundle	generateContract
GET	/contracts/create-terms	generateOrderTermsAndConditionsAsText

document-controller : Document Controller — Show/Hide | List Operations | Expand Operations

GET	/document/json/{documentID}	getDocumentJsonContent
GET	/document/xml/{documentID}	getDocumentXMLContent

Business process service API: continued on next page

Business process service API: continued from previous page

epc-controller : EPC Controller — Show/Hide | List Operations | Expand Operations

GET	/t-t/epc-codes	getEPCCodesBelongsToProduct
GET	/t-t/epc-details	getTTDetails

process-instance-controller : Process Instance Controller — Show/Hide | List Operations | Expand Operations

PATCH	/processInstance	updateProcessInstance
POST	/processInstance/{processInstanceId}/cancel	cancelProcessInstance
GET	/processInstance/{processInstanceId}/details	getDashboardProcessInstanceDetails
GET	/processInstance/{processInstanceId}/isRated	isRated

process-instance-group-controller : the process-instance-groups API — Show/Hide | List Operations | Expand Operations

GET	/process-instance-groups/filters	getProcessInstanceGroupFilters
GET	/process-instance-groups/order-document	getOrderDocument
DELETE	/process-instance-groups/{id}	deleteProcessInstanceGroup
GET	/process-instance-groups/{id}	getProcessInstanceGroup
POST	/process-instance-groups/{id}/cancel	cancelCollaboration

start-controller : the start API — Show/Hide | List Operations | Expand Operations

POST	/start	startProcessInstance

statistics-controller : The statistics API — Show/Hide | List Operations | Expand Operations

GET	/statistics/collaboration-time	Gets average collaboration time for the party in terms of days
GET	/statistics/inactive-companies	Gets the inactive companies. (Companies that have not initiated a business process)
GET	/statistics/non-ordered	getNonOrderedProducts
GET	/statistics/overall	Gets statistics (average collaboration time,average response time,trading volume and number of transactions) for the party
GET	/statistics/response-time	Gets average response time for the party in terms of days
GET	/statistics/total-number/business-process	getProcessCount
GET	/statistics/total-number/business-process/action-required	getActionRequiredProcessCount
GET	/statistics/total-number/business-process/break-down	getProcessCountBreakDown
GET	/statistics/trading-volume	getTradingVolume

trust-service-controller : Trust Service Controller — Show/Hide | List Operations | Expand Operations

GET	/ratingsAndReviews	listAllIndividualRatingsAndReviews
POST	/ratingsAndReviews	createRatingAndReview
GET	/ratingsSummary	getRatingsSummary

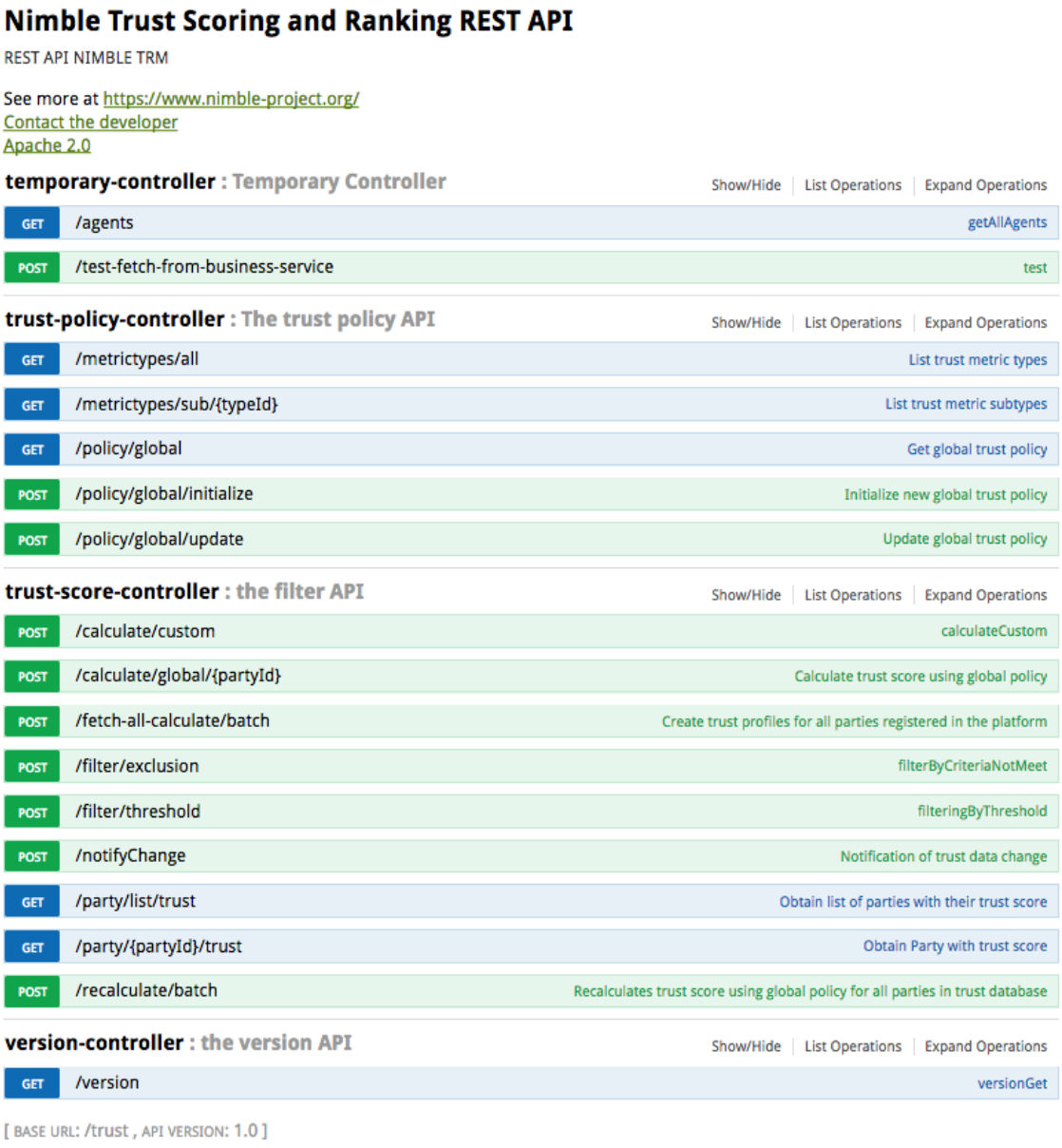
version-controller : the version API — Show/Hide | List Operations | Expand Operations

GET	/version	get the name and version string

[BASE URL: /business-process , API VERSION: 1.0]

4.4.4 Trust Service

https://nimble-platform.salzburgresearch.at/nimble/trust/swagger-ui.html



Nimble Trust Scoring and Ranking REST API

REST API NIMBLE TRM

See more at https://www.nimble-project.org/
Contact the developer
Apache 2.0

temporary-controller : Temporary Controller — Show/Hide | List Operations | Expand Operations

GET	/agents	getAllAgents
POST	/test-fetch-from-business-service	test

trust-policy-controller : The trust policy API — Show/Hide | List Operations | Expand Operations

GET	/metrictypes/all	List trust metric types
GET	/metrictypes/sub/{typeId}	List trust metric subtypes
GET	/policy/global	Get global trust policy
POST	/policy/global/initialize	Initialize new global trust policy
POST	/policy/global/update	Update global trust policy

trust-score-controller : the filter API — Show/Hide | List Operations | Expand Operations

POST	/calculate/custom	calculateCustom
POST	/calculate/global/{partyId}	Calculate trust score using global policy
POST	/fetch-all-calculate/batch	Create trust profiles for all parties registered in the platform
POST	/filter/exclusion	filterByCriteriaNotMeet
POST	/filter/threshold	filteringByThreshold
POST	/notifyChange	Notification of trust data change
GET	/party/list/trust	Obtain list of parties with their trust score
GET	/party/{partyId}/trust	Obtain Party with trust score
POST	/recalculate/batch	Recalculates trust score using global policy for all parties in trust database

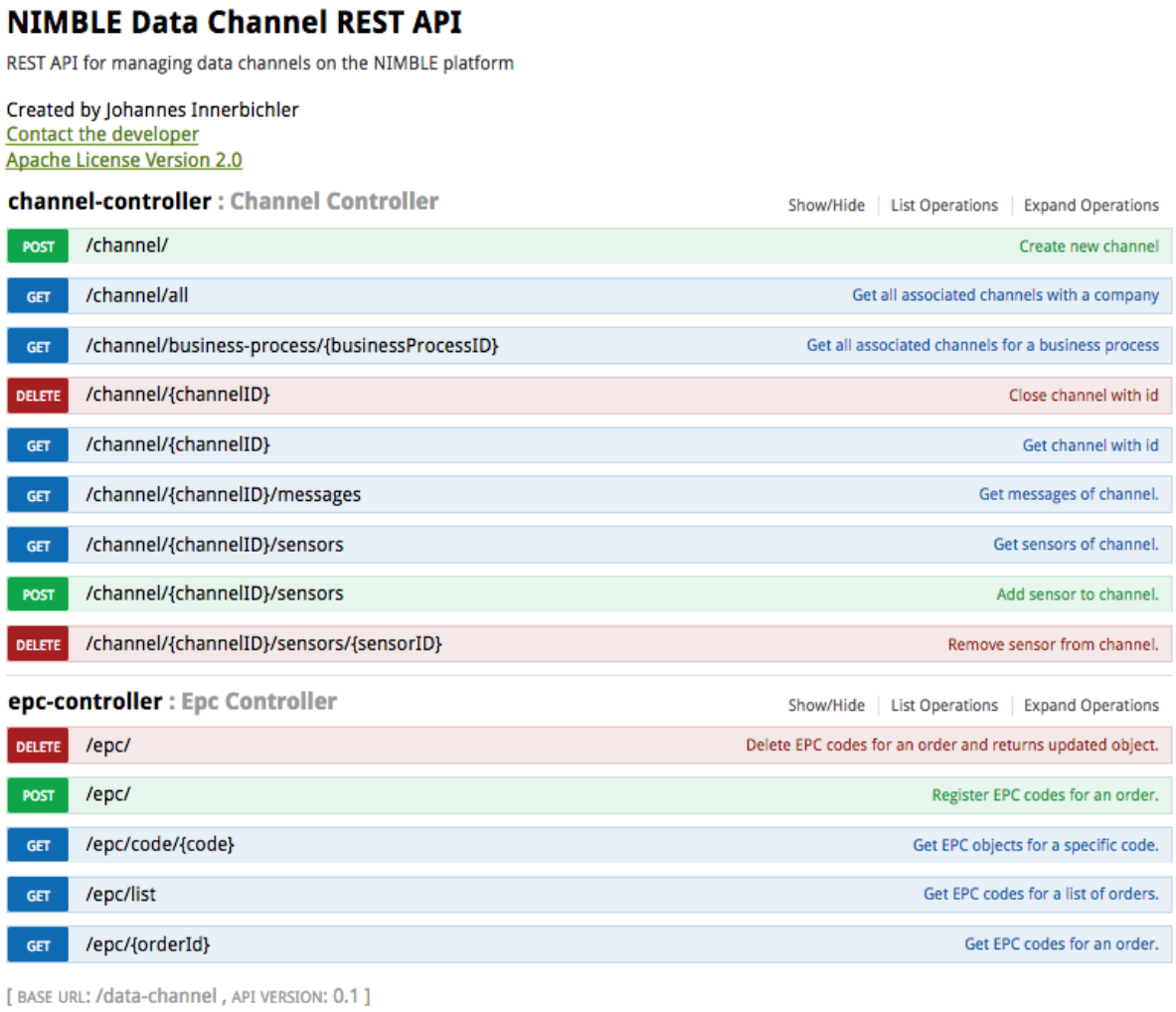
version-controller : the version API — Show/Hide | List Operations | Expand Operations

GET	/version	versionGet

[BASE URL: /trust , API VERSION: 1.0]

4.4.5 Data Channel Service

https://nimble-platform.salzburgresearch.at/nimble/data-channel/swagger-ui.html



4.4.6 Aggregation Service

https://nimble-platform.salzburgresearch.at/nimble/data-aggregation/swagger-ui.html

The data aggregation service API is still under development. Its purpose is to support governance of the platform, by showing the dynamics of transactions between the platform participants, e.g. trade volume, successful vs unsuccessful transactions, etc.

5 Conclusions

As part of the NIMBLE SEED programme, the present deliverable targets potential new platform owners and offers them two main contributions:

1. A structured framework (methodology and canvas), based on an existing toolkit named Platform Design Toolkit (http://platformdesigntoolkit.com), to help potential new platform owners in defining a platform strategy to target a specific ecosystem (i.e. market/sector). The final result is the design of a Minimum Viable Platform that could be then implemented by exploiting the NIMBLE Platform solutions.
2. An overview of the technical NIMBLE architecture (its core and backing services) and the main steps to deploy and customise/extend a new NIMBLE platform instance. Technical elements are briefly reported, and links to additional supporting materials, code repositories and external links (e.g. to backing service descriptions) are provided.

The present document will be made available on the project Website and advertised via the project dissemination channels (newsletter and Twitter) to attract the attention of potential new platform owners. If necessary (e.g. due to changes and relevant extensions in the core services), revisions of the document will be performed before the end of the project.

In addition, the reported contents and materials (for both main contributions) will be used in dedicated workshops and/or hands-on sessions with potential new platform owners to support them in adopting the NIMBLE solutions.

The tools were tested in a workshop with use case partners covering the application of the Platform Design Toolkit (results of this workshop are reported in D8.12 – Business Plan). Furthermore, the required steps related to the creation of a new NIMBLE installation were presented to all partners and following a “dry-run” internally, to disseminate platform launch expertise, there will be workshops and webinars in the final year of the project, to spread the word of NIMBLE to potential platform providers .