LUISS

DEPARTMENT OF BUSINESS AND MANAGEMENT
BSc in Management and Computer Science

Course in Digital Business and Workplace Technology

# API INDUSTRIALISATION IN A HIGHLY REGULATED INDUSTRY: THE CASE OF EUROCONTROL NM B2B API

Supervisor:
*Chiar.mo Prof. Paolo Spagnoletti*

Candidate:
*Lorenzo Belli*
ID 258851

Academic Year 2022/2023

*"If we worked on the assumption that what is accepted as true really is true, then there would be little hope for advance."*

*Orville Wright*

Academic Year 2022/2023

# Table of Content

# List of Figures



# List of Tables

# 1 Introduction

## 1.1 What is an API?

It is important, for the purpose of this thesis, to have a common definition of API, not only from a general technical standpoint, but as a concept. API stands for *Application Programming Interface.* For the intents and purposes of this research we can introduce an API as *the contract of a software component in terms of the protocol, data format, and the endpoint for two computer applications to communicate with each other over a network* (De, 2017). Of course, this definition has its limits, but I will introduce later other, more technical, aspects.

The " API contract" at its essence includes three points:

- A clear description of what functionalities does the API provider offer.
- The location where the functionality can be accessed.
- Any condition and/or constraints to which the API usage is subject, both business and legal.

With regards of the last point, it is important to distinguish APIs in two main categories, depending on the final user:

**I. Private**

These types of API are developed from a company, to serve their purpose and we can estimate that they make up the majority of APIs in use (Daniel Jacobson, 2011). They itself are dived into two main subgroups, again depending on the final user.

a. Internal

These APIs are built from a company as a set of building block, and they use the APIs to access service and information they own.

b. Partner integration

These are built from a company to enhance the integration of their service with their data. This is a growing business strategy as it offers a number of benefits for all the parties involved in the "API contract", not least it strengthens the interdependencies, as if a user's software is built on a provider's API, it would require an effort to adapt it to a different API.

**II. Public**

They are developed from a company and exposed to the general public, for free or under compensation, but accessible.

## 1.2 Why are APIs important now?

As the definition on which it is based, offering APIs as a product, or as a service, it is really about providing a tool to sustain process, and those process will be developed around the tool in hand. Hence the interdependencies we observe between users and providers.

Similarly to the success of open-source software, the success of API is partly conducted to the ability to tailor the service to the needs of the customer (Daniel Jacobson, 2011). As a matter of fact, one of the APIs most valuable assets is the community surrounding it as it, even more if detached from the development company, as it could allow for "user-generated" customer support.

Large companies are moving to integrate APIs in their business and there are succeeding. Companies such as Google, Netflix and Twitter, are looking at an increase in traffic which can mostly be associated with API traffic (Daniel Jacobson, 2011).

API are becoming more relevant because of they are highly customizable and offer many advantages on both the side of the provider and on the side of the user.

This thesis seeks to understand and analyze the process of industrialization in the context of highly regulated environment, where the APIs are beginning to play a fundamental role and starting to influence also well-established, traditionally standardized industries.

In Chapter 2 I will discuss the literature I review, focusing on establishing what is the status quo on APIs and investigating the process of APIs industrialisation, focusing on highly regulated sectors what is the next step in the agenda.

In Chapter 3 I will leverage the experience I gained on the field to bring a concrete example of how an API can be structured in a highly regulated, safety critical, environment.

In Chapter 4 I extrapolate the relevant aspects of the case study and expand on the different facets of API industrialization, laying out the options for future API managers to choose from. In this chapter I focus on the areas of API pricing, enterprise architecture and the different aspects of service level management.

In Chapter 5 I draw my conclusions, summarizing what I establish thought the course of this thesis, and providing an overview of what are proven to be the critical aspect to consider in the process of industrializing an API in a highly regulated environment.

# 2 Literature review

As long as businesses continue to rely on APIs to facilitate interconnectivity between various apps and systems, API management will become an increasingly important component of modern software development. In this literature review, we'll look at the condition of API administration and consider its effect on the present technological environment.

### 2.1.1 Methodology

I reviewed a number of books and articles on the matter of API industrialisation, API architecture and API management. This material helped me understand what is the status quo on API.

While working on this thesis I had the opportunity to develop more in-depth, industry specific knowledge and gain on-the-field experience working as a trainee at EUROCONTROL, with a hands-on role on their API. I will dive deep in how this helped me shape my understanding and try to answer the questions arising from my research.

### 2.1.2 Findings

After reviewing the literature on API management and API industrialisation we highlight a clear interest on the matter. This is driven by the growing demand of different industries for expertise on the matter. However, as the API industries are expanding into traditionally highly standardized industries, there is concern for integration on critical systems. We note that where the API impacts relatively new sectors, the standardization process is fairly understood.

However, a "*critical gap is that most studies focus on standardization in nascent markets while established industries - where regulators can use standardization to boost competition - have been omitted"* (Dize Dinçkol, 2023).

I might add that standardisation and change in the regulatory framework could be used not only to increase competition, but also, in safety critical system, to ensure business continuity and increase reliability. New standards can improve and adapt APIs to resist new treat (i.e., cyberattacks) and ensure interoperability within the system.

On API industrialization the literature is extensive. We define the industrialization process as the process of systematizing and scaling the development, management, and utilization of Application Programming Interfaces within an organization or across multiple organizations, with the aim to enhance efficiency, productivity, interoperability and reliability, enabling connectivity and data exchange taking an active role in the development of interconnected ecosystems of application and services. It involves developing standardized practices to streamline the development phase, the deployment and the maintenance of the service provided.

This poses a numbers of challenges in highly regulated sectors, due to strict requirements on security, data governance policies, risk management and legal considerations. One of the sectors burdened with this task is the aviation sector. We consider how this sector requires high level of security and data segregation, dealing with safety critical systems and information. Industrializing APIs while adhering to these regulations can be challenging due to the need for data protection, access controls, auditability, and privacy. This thesis is looking to provide a framework for API managers to navigate this highly regulated industrialisation processes.

### 2.1.3 Research question

Making an API (Application Programming Interface) more dependable, effective, and scalable allows it to be used by more users and applications. This process is known as industrializing an API. This entails several processes, such as standardizing the API, enhancing its functionality, and ensuring its security.

The development of microservices architecture is one of the key factors driving API management. APIs are now a crucial part of application design as businesses increasingly adopt distributed systems.

It is with the intention of proposing a breakdown of what aspect to consider and understand the best choice for future APIs looking to affect highly regulated industry that I set out to answer the following:

***How to manage the process of API industrialisation in the context of a highly regulated industry?***

# 3 The case of the EUROCONTROL NM B2B API

## 3.1 About EUROCONTROL

### 3.1.1 The role of EUROCONTROL

*EUROCONTROL is a pan-European, civil-military organisation dedicated to supporting European aviation.* (EUROCONTROL, 2023)

There are a few adjectives that are worth discussing. Pan-European refers to the extension of the organisation, which operates on 41 Member states in European area, and 2 States with which special agreements are in place, namely Israel and Morocco. Civil-Military refers to the nature of the organisation, which acts as a coordination entity between civil aviation and military operations, and does so with the concept that airspace is a finite resource in mind.

*[EUROCONTROL] supports its Member States and its civil and military stakeholders (including air navigation service providers, airspace users, airports and aircraft/equipment manufacturers) in a joint effort to make aviation in Europe safer, more efficient, more cost-effective and with a minimal environmental impact.* (EUROCONTROL, 2023)

EUROCONTROL is nominated as the Network Manager[1] with Commission Decision of 7.7.2011 with a mandate that runs until the end of the Performance Scheme's second Reference Period (31 December 2019). The period of appointment of the Network Manager after the end of 2019 shall last at least two reference periods of the performance scheme (i.e., at least 10 years).

### 3.1.2 The NM B2B API

*The NM B2B Services constitute an interface provided by the EUROCONTROL Network Manager (NM) for system-to-system access to its services and data, allowing users to retrieve and use the information in their own systems.* (EUROCONTROL, 2023)

The NM B2B API is the API developed and managed by EUROCONTROL to provide its user with data relevant to their operation. It is a special form of private

[1] The Network Manager manages ATM(Air Traffic Management) network functions (airspace design, flow management) as well as scarce resources (transponder code allocations, radio frequencies), as defined in Regulation 677/2011 and Regulation 2019/123 (SKYbrary, s.d.).

partner, as at the moment its access is reserved to those eligible, after evaluation by the provider company. In the version 27.0 is divided in 6 service groups, 21 service and 187 single operations[2].

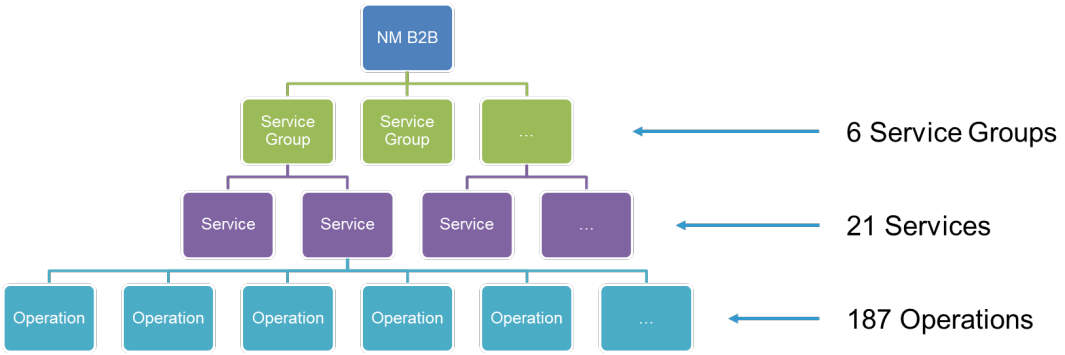


*Figure 1 - NM B2B API v27.0*

#### *3.1.2.1 Who are the users?*

Internal users, are those accessing the API from systems internal to EUROCONTROL, to rapidly access services and data, for the purpose of this research, the distinction is not particularly relevant as they are treated as external user for all intents and purposes.

Users can be classified in external or internal to EUROCONTROL. External are, inter alia:

- ANSP (Air Navigation Services Provider)
- Airports operator
- Handling companies (Cleaning, catering, fueling, etc...)
- CFSP (Computer Flight Plan Service Providers)

This last category is particularly relevant because they can help optimize the use of airspace and help facilitate the advancement to newer technologies, which is in line with the objectives of the Network Manager, such as:

- Manage flows of air traffic to ensure safety and efficiency
- Avoid hotspot[3]

[2] Operations are the single calls that can be made to the NM B2B API API
[3] A hotspot is intended as a point of high traffic and high risk, due to the number or complexity of flights passing through it.

- Protect downstream sectors[4]

User needs a certificate to access the B2B API. This certificate is represented by a token (a private/public key system) by Global Sign. The first two are included in the services provided by EUROCONTROL, while any further request will be charged at cost for the recovery of expenses (EUROCONTROL, 2023).

Currently the structure of the API is described as in *Figure 2.* While it might be outside the scope of this research to understand what each of the service groups does - or even more what each of the specific service or operation – it is important to note that the API offers a structured set of services. This is an important quality for scalability and user management, which will be of interest in later chapters.

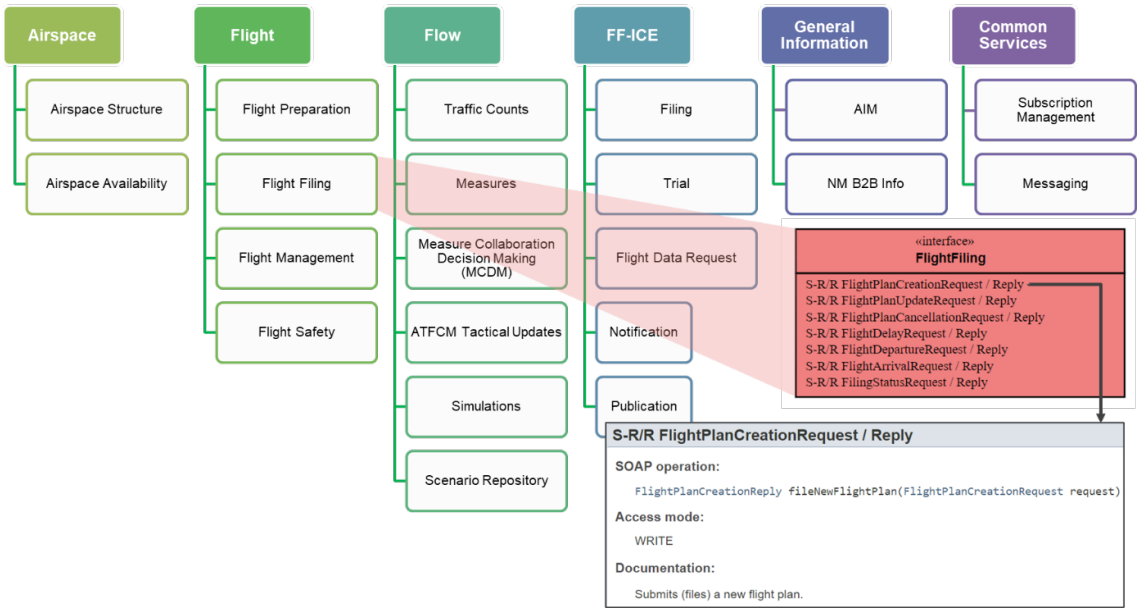


*Figure 2 - Service Groups (NM B2B API v. 27.0)*

### *3.1.2.2 Application Cooperation*

To better understand the complexity of an API, and in particular the API under analysis it is necessary to understand the complexity of the systems behind it. This allows me to underline how an architectural approach is essential to achieve a well-managed API and it is key in a very high industry such is the aviation industry.

[4]Portions of airspace which will be occupied at a later point in time over the course of a flight.

Figure 3 is an ArchiMate model of the Application Layer of the EUROCONTROL NM B2B API. In particular it is the Application viewpoint which "*shows the structure of one or more applications or components.*" (al., 2017).

This viewpoint is useful to understand and design the main structure of systems and applications, and identify the associated data and components. It is critical in planning a migration, an update or, more importantly in our case, to verify compliance of the system with specifications and standards.

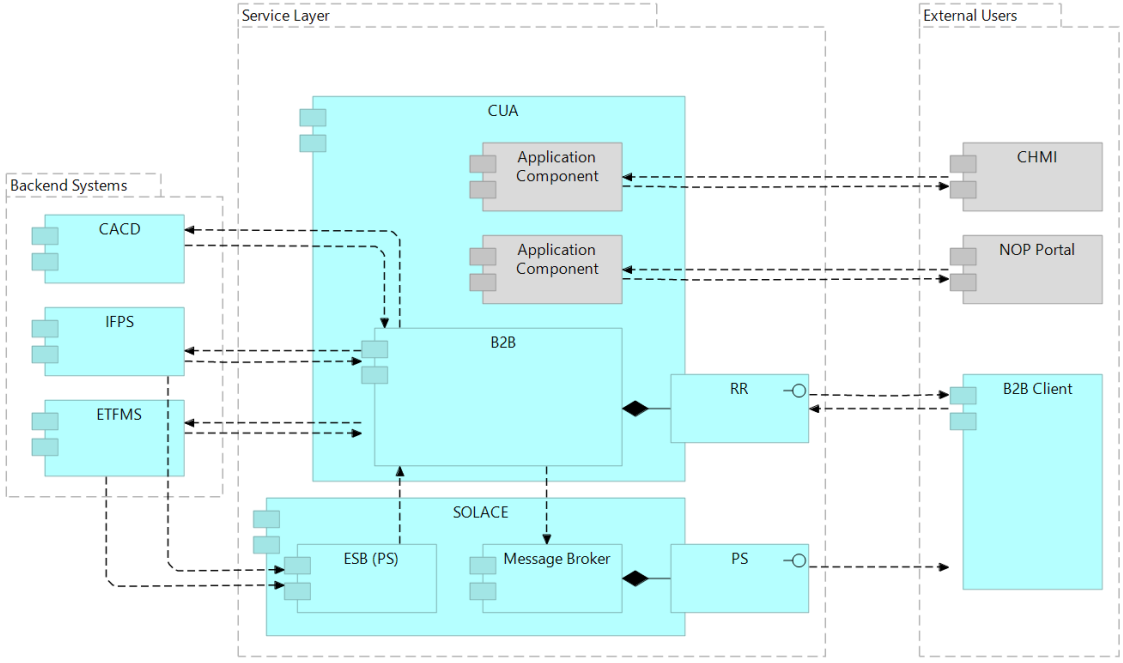


*Figure 3- Application Cooperation Viewpoint*

#### 3.1.2.2.1 Backend Systems

In general, the scope of the data contained and processed here is to balance capacity and demand.

I will briefly describe the component of this model, avoiding to dive deep in technicality outside this thesis' purpose, nevertheless, there will be the need to introduce some aeronautical concepts, which I will simplify.

I. **Central Airspace and Capacity Data (CACD)**

Contains the airspace environment data (i.e., sector configuration, from which capacity[5] can be computed)

II. **Integrated Initial Flight Plan Processing System (IFPS)**

The system, mostly automated, which manages flight plan submissions, validation and distribution.

III. **Enhanced Traffic Flow Management System (ETFMS)**

Computes the estimated time at each waypoint. The system provides enhanced tactical[6] data to all operational stakeholders, regardless of national boundaries, language, or equipment. ETFMS facilitates improvements in flight management from the pre-planning stage to the arrival of the flight (EUROCONTROL, s.d.).

#### 3.1.2.2.2 Service Layer

I. **Common User Access (CUA)**

a. B2B

II. **SOLACE** (this the service used as a message broker)

a. ESB (Enterprise Serial Bus)

It is a "components Off-the-shelf" (COTS)[7] data transfer system

b. Message Broker

It is an event driven message broker which allows for the subscription to topics (or subtopics) which are standard bundles of information which publication is triggered by defined events. The user can then customize the information to which they would like to subscribe and which they will the n be able to pull form a PS queue.

III. **Request – Reply pattern**

The use of this patter should be limited to occasional usage or the. Figure 4 describes the functioning of a Request Reply pattern. It shall not be used to

[5] The maximum number of aircraft that can be accommodated in a given time period by the system (airspace) or one of its components(i.e., airports, sectors) Can also be thought of as the throughput measure.
[6] Related to the day of operations.
[7] Ready-made hardware or software, which are adapted aftermarket to the needs of the purchasing organization, rather than the commissioning of custom-made.

request relevant information (due to risk of backend overload and inefficient usage of resources).

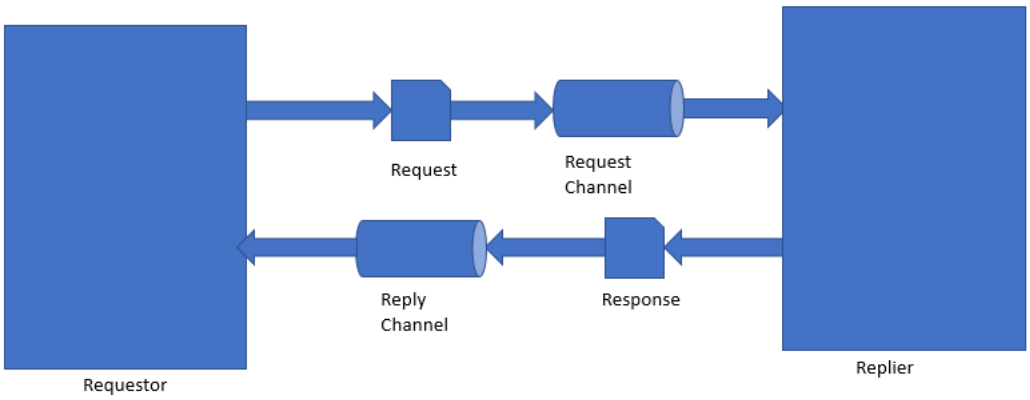


*Figure 4 - Request/Reply pattern (Pethuru Raj, 2017)*

## IV. Publish – Subscribe pattern

The Publish-Subscribe protocol is used to push messages in a queue from which the users can subscribe (different queues for different topics)

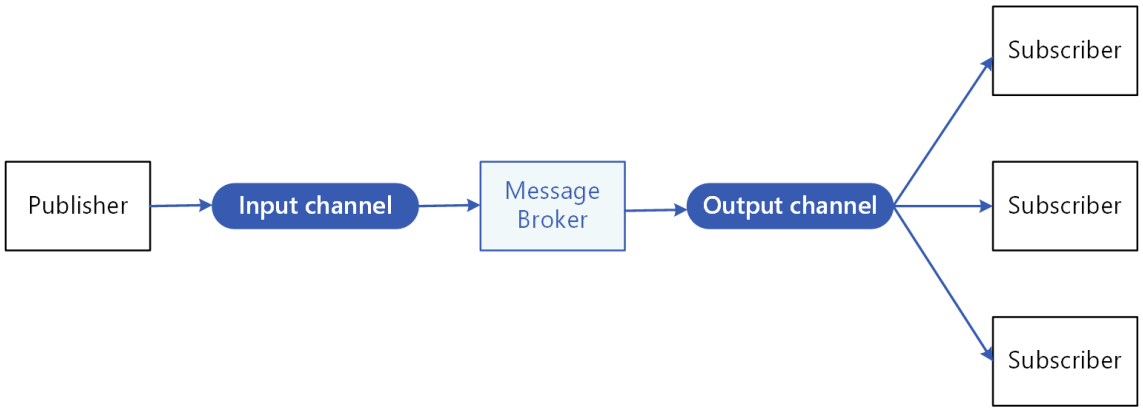


*Figure 5 - Publish/Subscribe Pattern (Microsoft , s.d.)*

#### 3.1.2.2.3 External - to NMOC (Network Manager Operations Centre)[8] systems - User

- **CHMI (Collaboration Human Machine Interface)**

[8] The NMOC researches, plans and coordinates [aviation] network activities. (EUROCONTROL, s.d.)

It is the terminal, the graphical interface for the network operations systems.

- **NOP (Network Operations Plan)**

  The NOP and in particular the NOP plan, benefit from the service layer to connect operational stakeholder to NMOC.

- **B2B Client**

  This refers to all those system – to – system connection established with the API.

## 3.2 An overview of the relevant processes

Now that the systems under study are clearer, we move to observe the process that manage those systems.

### 3.2.1 Standardization

The matter of standardisation is crucial for the purposes of this research. The process of standardisation can occur in multiple forms based on who takes part in initiating, implementing, and leading the standardization effort (Susanne K. Schmidt). These processes have been extensively studied by the literature and can be summarised as such:

- **Committee-led**
  Standards are developed and diffused through collaboration of stakeholders, through *"voluntary open and transparent, consensus-based standardization processes"* (Susanne K. Schmidt) (Knut Blind).
- **Market-led**
  Standards can be developed by anyone and coordination of actors during standards diffusion is ensured via competition of different standards, emerging naturally through market processes (Knut Blind).
- **Government-led**
  Anyone can develop standards and government hierarchy mandates standards diffusion (Paul Moritz Wiegmanna, 2017).

In the instance of EURCONTROL, the API, and more in general the systems are standardised over time in a collaborative effort of stakeholder and government entities, prioritizing safety above all else.

### 3.2.2 Data sharing in a safety-critical environment

Achieving a harmonious equilibrium between data sharing and data segregation is crucial for organizations. Data sharing involves the controlled and purposeful exchange of information among different entities, fostering collaboration, informed decision-making, and efficiency. Conversely, data segregation focuses on safeguarding data privacy, confidentiality, and compliance by separating sensitive information from more accessible data. By adhering to robust data governance frameworks, implementing secure sharing protocols, and enforcing strict access controls, organizations can strike the delicate balance between facilitating collaboration through data sharing and protecting sensitive data through segregation. This ensures that data is effectively managed, privacy is upheld, and the value of shared information is maximized without compromising security.

Data sharing is a critical aspect of APIs, even more in a context which directly impact the safety of human lives. In safety-critical APIs, data sharing plays a pivotal role in facilitating seamless communication and collaboration among different systems, devices, and stakeholders. It allows for real-time exchange of critical information, such as flight data, enabling timely decision-making and coordinated actions. However, stringent controls and protocols must be in place to ensure that only authorized parties have access to sensitive data and that data is transmitted securely to prevent any compromise of safety or regulatory compliance. This includes strict access controls, encryption techniques, and thorough audit trails to track and monitor data access and modifications.

### 3.2.3 Business Functions

Business functions are the specific areas or departments within an organization that are responsible for performing specific tasks or activities. Examples of business functions include marketing, sales, human resources, finance, and operations. Business functions are typically organized hierarchically, with senior management

overseeing the entire organization and lower-level managers responsible for specific areas or functions.

### 3.2.4 Business Processes

Business processes are the series of steps or activities that an organization uses to accomplish its objectives. These processes are typically analysed and documented using process models that show the inputs, outputs, and actions required to execute each step of the process. Business processes are often cross-functional, meaning they involve multiple departments or functions within an organization.

#### *3.2.4.1 Onboarding process*

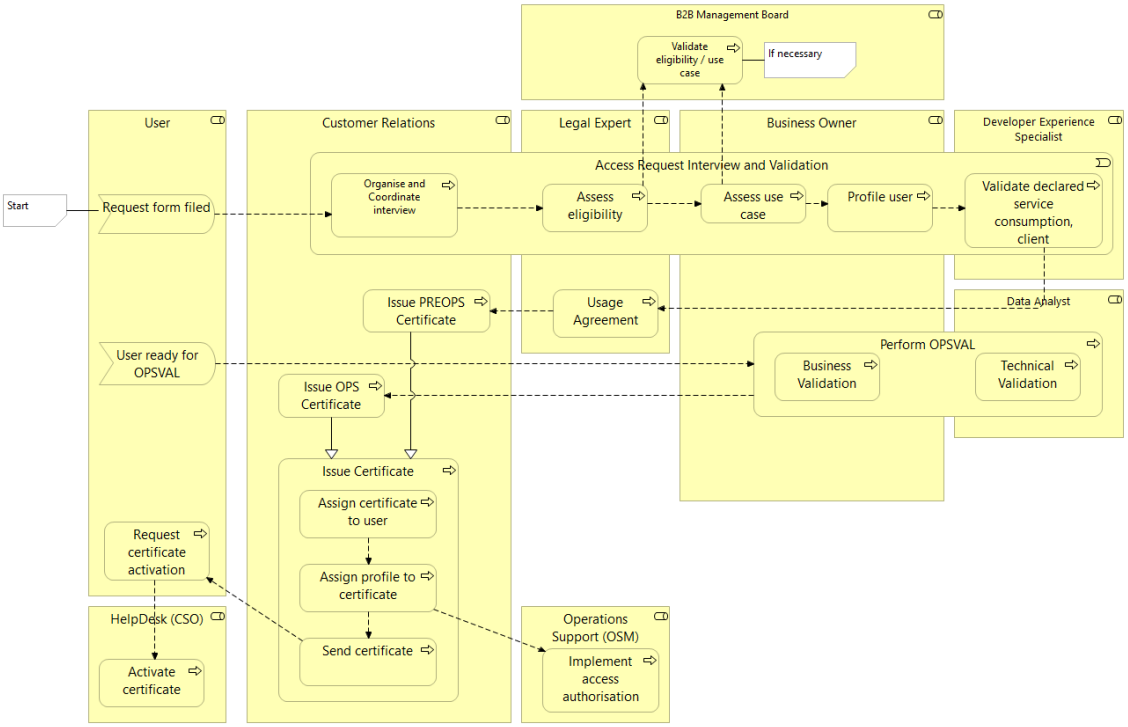


*Figure 6 - Onboarding Process to NM B2B API*

##### 3.2.4.1.1 Assessing eligibility

In most highly standardized industries, and for the principles of data sharing previously discussed, it is essential to verify eligibility of the user to access sensitive data (i.e., flight data). This is done via interviews and analysing the role of the “aspiring” user to evaluate its contributions to the industry and assessing its needs for the connection to the NM  B2B API, to minimize risk.

#### 3.2.4.1.2 Leveraging the regulatory environment to improve efficiency

Standardisation has some advantages to APIs too. Specially if the API you are managing **is** the standard. You can invest time and energy in developing a better service and scale your systems. However, with great power comes great responsibility and you need to insure an effective and efficient system usage. This starts at the moment of onboarding and is key to establish an appropriate usage of resources.

##### 3.2.4.1.2.1 The Operational Validation process (OPSval)

An operational validation is the process through which EUROCONTROL assesses that its users are fit to operate on a production environment.[9] It can be divided into two phases.

The first one is the *business validation* which assess the eligibility for a certain entity to access, read and in some cases write data on EURCONTROL system from a business perspective. More specifically it is verified that software using the NM B2B API act in respect of the regulations and procedures established by the industry. This is an important part of the validation, and should be implemented in a highly standardized environment, however it is very industry-dependent and not the focus of this research.

The second is the *technical validation.* During this process it is verified that the user is compliant and best using the API .The process consists of the following:

1. The user tests its application on the pre-production[10] platform.
2. The user runs the application on the PreOPS platform with a similar load and use to what it would be in the production environment for a period of minimum ten days.
3. The usage of the application during the trial period is logged.
4. The user communicates to the provider when the trial period has ended. This means when it had ten days of continuous usage with no relevant issues.

[9] The production environment, or platform, its denominated Operational, or OPS for short.
[10] The pre-production environment, or platform, its denominated Pre-Operational, or PreOPS for short.

5. At the end of the trial period, a report is produced, which will analyze various aspect of the usage, and will indicate whether an OPS certificate[11] can be issued or not.
6. If the OPS certificate can be issued, it will be released in the next window available (due to external constraints, this usually means one window available per week).

As an input to the operational validation, it is taken into account a form compiled at the moment of the connection request. This form is part of the "API contract", and specifies what API calls can be made by the user and to serve which business purpose. In other industries this is very unusual, but in the context of a pan-European civil-military organization offering services to specific stakeholders without having profit as a main driver it is the case.

During an operational validation, the aspects observed include, but are not limited to, the following:

- **General**
  - Usage coherence, completeness and the correctness of the use case is verified.
  - It is verified that the correct API version is used. Operational validations for versions near the end of their lifecycle should not be carried on.
- **Request Reply**
  - The operations used in the trial period should match the one the user requested in the use case.
  - It should be verified that the number of request and the data volumes are appropriate for the task in hand given the user operational needs.
  - It is verified that the amount and the distribution over time of user-caused errors is not prone to excessively loading the provider systems.
  - It is verified that overloads events are absent or the risk of them occurring is minimized.

[11] The OPS certificate is the token needed to access the Operational, or production, environment.

- **Publish-Subscribe**
  - It is verified the consistency with the use case form (i.e. the consumer is using the correct version and the subscriptions - and the topics[12] to which they are related - match your use case).
  - We verify that you put in place best practices for the subscription management (i.e., you do not let subscriptions systematically expire, you do not repeatedly create and delete subscriptions).
  - We look at the filtering you put on your queue, and we verify it matches your profile and your use case form.

### *3.2.4.2 Support Process*

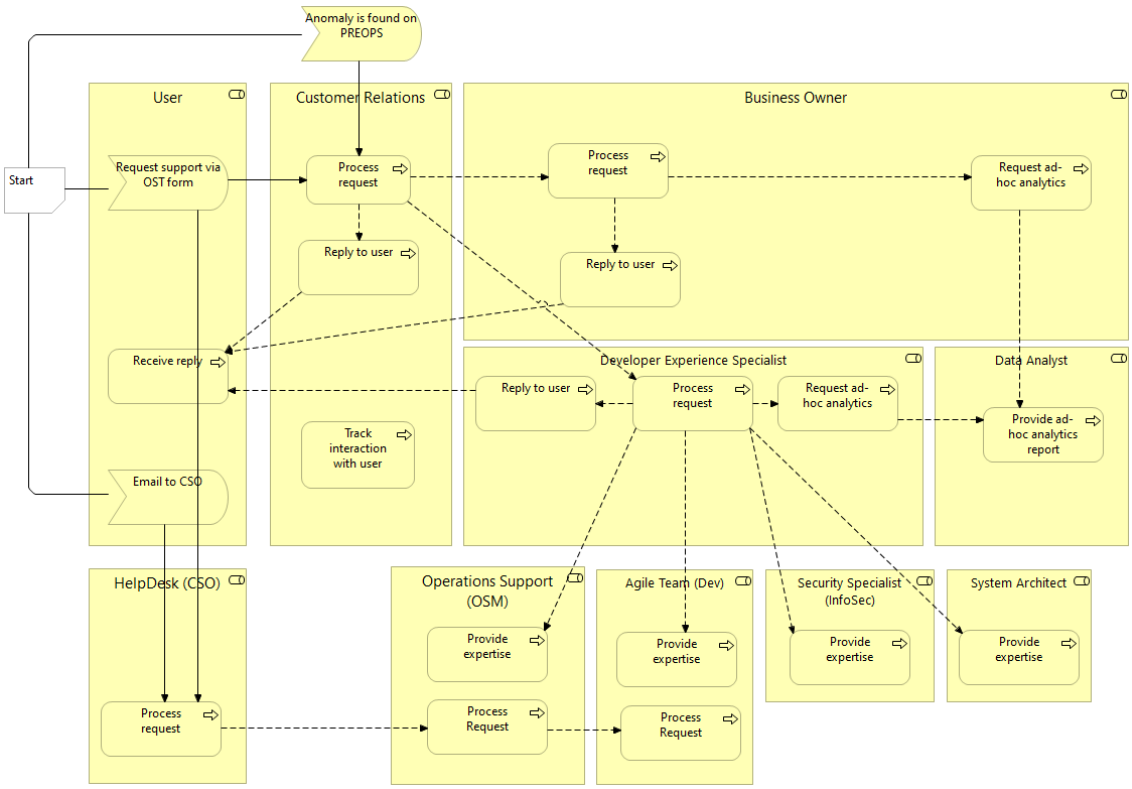


*Figure 7 - Support Process to user of NM B2B API*

#### 3.2.4.2.1 Types of support

API support can be broadly categorized in two groups, depending on who is providing it.

[12] The topics refer to the queue to which the user subscriptions are associated.

The first form of support is represented by the support directly from the provider. This type of support is generally available for a number of SaaS and It can take make form and varies from company to company, generally depending on the business model sustaining the API.

The second type of support, peculiar to APIs, and extremely beneficial for the API stakeholder, is the community support. This kind of support is more complex to develop, as it takes time and a community of active users for it to be effective and a viable option. However, the creation of the community support shall not be restricted to public APIs. It is, in fact, possible to build a strong community of partner, stakeholder or even internal user, the expertise might even be more of help as the industry will likely be the same.

#### 3.2.4.2.2 Defining the phases of support

The support offered by the provider is clearly organised in phases, depending on the criticality, the urgency and where the support should intervene. In particular we recognise the following:

a. L1 support

   This is the first line of support. It is active on h24 and seven days a week, due to the criticality of systems directly impacting safety.

b. L2 support

   This is non-priority, non-urgent, support. It is dealt with during office hours and mainly occurs during transition period (from a version to another or to brand new system). It also provides assistance with the configuration of the systems.

c. L3 support

   This is systemic, structural support. The teams operate directly on the code of the system. It is longer term and its aim is to solve issue intrinsic in the system.

### 3.2.4.3 *Auditing Process*

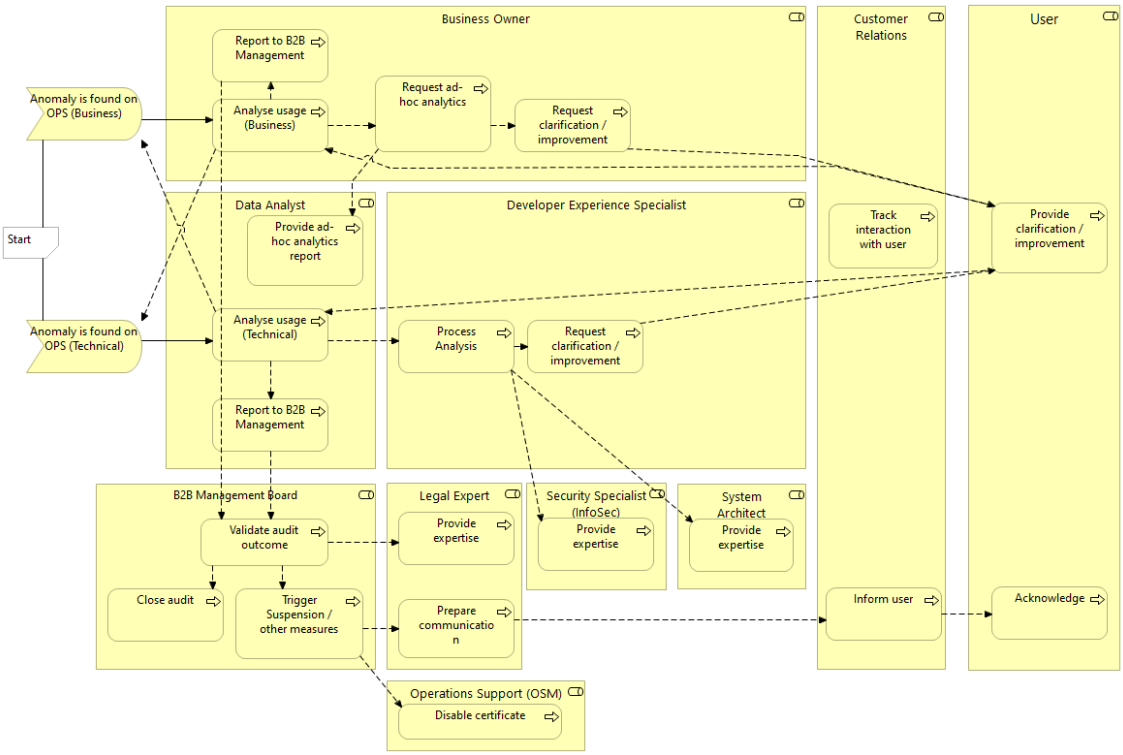


*Figure 8 - Auditing Process*

#### 3.2.4.3.1 Reporting

Reporting is key to digital products:

- **Periodic**

  Periodic reporting involves generating and sharing reports at regular intervals, providing a structured overview of API performance and usage trends.

- **Ad-Hoc**

  An efficient ad-hoc reporting methods is necessary to cater to on-demand requests, offering flexibility and real-time data access for troubleshooting and in-depth analysis.

A reporting tool is essential. These types of tools can be either “of-the-shelf” (e.g., Microsoft PowerBI and Celonis) or tailored.

#### 3.2.4.3.2 Punitive measures

In case the reporting process finds an anomaly that needs to be corrected, it is good practice to put punitive measures in place. These punitive measures can be of

increasing gravity based on the severity or on the number of occurrences of the infringement of the API agreement. We can distinguish punitive measure in two types, which can co-exist within the same reporting process.

**i. Automatic**

These are the punitive measures embedded in the code of the API itself. They are generally less severe punishment which entry into force when a certain threshold is reached (i.e., a certain percentage of errors caused by the over a rolling window, a certain threshold of data consumption breached over a rolling window, etc.)

**ii. Enforced**

These are generally more severe measures put in place after the user repeatedly breaches its use condition. Before putting in place this measure usually a deeper investigation is conducted, and a data is presented to a committee to evaluate the necessity and severity of the measures.

#### 3.2.5 Business Services

Business services are the capabilities or functionalities that an organization provides to its customers or stakeholders. These services are typically designed to support the organization's business processes and may be delivered through a variety of channels, such as web applications, mobile apps, or customer service centres. Examples of business services include order processing, payment processing, or customer support.

## 3.3 Digital business in a highly regulated industry

### 3.3.1 What is SWIM?

System Wide Information Management (SWIM) is a concept consisting of standards, infrastructure and governance enabling the management of ATM related information and its exchange between qualified parties via interoperable services.

*SWIM is the answer to achieving global interoperability and service orientation in air traffic management.* (EUROCONTROL, 2021)

#### *3.3.1.1 The SWIM framework*

SWIM has developed on the basis of three pillars:

- **Service Description**

  The information provided is described has a service, which under the SWIM framework are described in a standard format.

  - In the NM B2B API this is represented in the User Profile Template. This Template needs to be aligned with a *JSON*[13] schema to upload the service into the SWIM registry.

- **Information Definition**

  Information is defined under the standards of the SWIM registry.

  - In the NM B2B API the payloads are described in a standardized registry.

- **Technical Infrastructure**

  This pillar is a repository where the set of all technical standards that are deemed appropriate for the integration with SWIM are stored and classified.

#### *3.3.1.2 Compliance vs. Conformity*

The issue of systems being SWIM compliant or SWIM conformant is an issue of semantics, in general better left to the lawyers, but in this case, it allows for a deeper understanding of SWIM.

To be ***compliant*** a system must meet certain requirements of a standard defined in a framework.

To be ***conform*** a system must meet *all* requirements defined in a standard framework.

Having given this definition is clear that no system will be conform with the Technical Infrastructure pillar, as no system will need to be compliant with *all* the standards set out by the SWIM Technical Infrastructure pillar.

### 3.3.2 The importance of interoperability

Interoperability is defined as *"the ability of information and communication technology (ICT) systems and of the business processes which they support to*

[13] JSON Schema is a declarative language that allows you to annotate and validate JSON documents. JSON Schema enables the confident and reliable use of the JSON data format. It presents many benefits such as describing the existing data format, providing clear and readable human-machine documentation and validates data.

*exchange data and to allow the sharing of information and knowledge".* (EUROCONTROL, 2021)

This is particularly relevant in a highly regulated industry, where the ICT systems are safety critical. In fact, accepting the risk of cybersecurity threats or services degradations is not an option when dealing with a fast-paced environment where the stakes incredibly high. These led to the development of ad-hoc systems, software and infrastructures, built over time to different standards, to accommodate varying needs. The specificity of the requirements for which these have been built make the integration of the systems hard, if at all possible. The SWIM concept main objective is to solve this, by defining standards designed to be secure, yet expandable and scalable.

### 3.3.3 The impact of regulations

As stated many times, the aviation industry, in particular that relating to the support of operations and Air Traffic Flow and Capacity Management, is highly regulated due to the strong impact on safety.

*Effective ATM modernisation requires the timely implementation of innovative ATM functionalities. Those functionalities should be based on technologies that increase the levels of automation, cyber-secure data sharing, and connectivity in ATM.* (Commission Implementing Regulation (EU) 2021/116, 1 February 2021)

The previous quote of the Commission implementing regulation defines key strategic objective for stakeholders within the Air Navigation industry. It highlights the importance of features such as cyber-secure data sharing and connectivity, which are intrinsic properties of well-built APIs and the regulation even takes into account the increased level of automation, which should serve as a remark for us to understand the immense trust that is put in a digital product as this is.

#### *3.3.3.1 Estimating future demand*

This aspect is quite relevant in a standardised industry where there are strict services level to be respected and becomes critical when the API is meeting the requirement set out by the new standards. This forces stakeholders to pursue the onboarding process and increases the strain on the systems.

To estimate future demand is an exercise of inference which has to allow for generous margin errors. It is best to err on the side of caution, as it is best to consider that while the traffic network for which we estimate the data is the "run-state"[14] traffic, during the first development phase there will be a surge of tests, errors and interaction with the systems due to configurations processes and other phases of the onboarding process.

[14] Traffic occurring under normal condition on an average day of operation. Does not consider cyclical peaks such as updates on the date or version migrations, seasonal peaks and onboarding testing and development

# 4 The facets of API industrialization

Making an API (Application Programming Interface) more dependable, effective, and scalable allows it to be used by more users and applications. This process is known as industrializing an API. This entails several processes, such as standardizing the API, enhancing its functionality, and ensuring its security.

To achieve this, I will break down the issue into three main areas:

- API Management
- API Architecture
- API Pricing

## 4.1 API enterprise architecture

Enterprise architecture is a collection of documents helping establish effective communication between all relevant actors involved in strategic decision-making and implementation of IT systems (Kotusev, 2018).

In essence, the utilization of enterprise architecture brings an organization closer to achieving an ideal state where it operates as a decentralized network comprising various independent actors who collaborate and interact. This state resembles a single, highly intelligent "big brain" that possesses the ability to make globally and locally optimized business and IT decisions across all domains, drawing upon all available information resources.

## 4.2 API pricing

I will deviate for a moment from the case of EUROCONTROL B2B API, as at the moment there is no structure or decision that establishes pricing for the use of API. This is mainly due to the nature of the API and the role of EUROCONTROL within the industry. If EUROCONTROL should charge (at least certain categories of users) to access the NM B2B API is a discussion which I will leave out of my thesis. Nevertheless, this allows me to give the best perspective on this matter, which is: if a company were to charge for its API, how would the matter need to be approached?

Both API customers and API providers must take API pricing into account. In this examination of the literature, we'll look at the various pricing strategies employed by the API sector, as well as their benefits and drawbacks, and how they affect the API economy.

The two main types of API pricing models are transaction-based pricing and subscription-based pricing. Although subscription-based pricing costs a pre-set amount for access to the API for a certain period, transaction-based pricing charges API users for each transaction they perform.

I will not go in detail of the financial repercussion of each model, I will rather focus on the implication of choosing each of the models from the IT management, discussing the infrastructure required per each model, its complexity (and presumably cost).

### 4.2.1 Pricing Models

When it comes to which API pricing model is best, the answer varies based on the type(s) of API on offer, and the type of business offering it.

There are pros and cons to all API pricing models. Some offer uncertain revenue streams, some restrict access and use of the API, and some require more flexibility and fine-tuning than others.

It is important, when evaluating a pricing model to take into account the infrastructure needed to support it. It is in fact possible, as I will come to explain in illustrating some of the more common models, to tailor the pricing strategy to using countless variables and measures, such as, but clearly not limited to:

- Size of the payload
- Time of the day
- Location of the end user

These are only some of the more peculiar metrics that could be interesting when looking to put in place a pricing strategy. However, regardless of the metric chosen, there is the need to put in place metering system capable of the supporting the pricing strategy, and this infrastructure shall be taken into account for the costs involved.

#### *4.2.1.1 Pay as you go*

Pay as you go API pricing models charge developers based on how much they (or rather, their software product) use the API.

One example of a pay as you go API pricing model is pay per use/call. This is where you pay individually each time you make an API call. (i.e., when the API gets a request and retrieves the data required.)

Depending on what criterion is used to inform the pay as you go price, the pricing page may say '£X per download', '£X per call', etc. Prices here will typically be nominal — the idea is that the API will see a lot of use.

#### *4.2.1.2 Fixed pricing*

Fixed API pricing, is a form of subscription model. This would have developers and/or users pay a fixed amount per month or year to use the API.

This may or may not restrict the amount the developer or their users can make use of the API.

For example, take a fixed quota API pricing strategy. In a fixed quota strategy, the flat rate paid per month will cover the use of the API for a set amount of time/use. This is known as the quota. Once the quota is met, the API would not be usable until the next payment is taken.

#### *4.2.1.3 Combination Pricing*

The most popular of the API pricing models is an approach that combines the above two methods. They're often known as overage models.

An overage model has a base amount that is paid to use the API, and it comes with a quota. If this quota is met, the API will still work. However, you'll swap to a pay as you go method for all the API calls or downloads that exceed the quota for the specified time.

So, an API pricing page showing an overage model will read along the lines of '£X for Y calls per month + £Z per additional call'.

Unlike fixed-rate models, overage pricing allows for greater flexibility. All while retaining the predictable revenue that pay-as-you-go models cannot offer.

#### *4.2.1.4 Partnership based pricing*

There is one final type of API pricing to note. It's the type that relies on a kind of partnership between the developers using the API, and the API owners.

The API pricing model in question is known as revenue share. This is where the API provider gets a share of the revenue generated by the developer's use of their API.

#### *4.2.1.5 Freemium*

Freemium API packages boast free use of the software, but they also have limited usage or a lack of support. The idea of freemium API is to encourage the sale of a premium solution with more functionality later down the line. Freemium API pricing may also involve costs related to paying for support or for extra usage allowance. It allows for no upfront cost and makes it easy to try out the software and identify whether it holds value for the business, but it also presents some downsides such as:

- Freemium SaaS solutions will have limited functionality and usage allowances. So, not all features may be available without the upgrade
- The user may find itself locked out of using the software until your usage allowance renews. This means they can be disruptive to your daily workload.
- Freemium or free SaaS models will also come with less or no support (without incurring a cost). If the user encounters a problem, then, it can be difficult to get it resolved.

#### *4.2.1.6 Usage-based SaaS pricing*

Usage-based SaaS pricing is where the monthly bill is determined based on how much the service is used. The more you use the SaaS solution, the higher the bill. So, the price may be based on the number of certain actions completed or the amount of data used, for example.

There are a few variants of usage-based SaaS pricing. Where some are purely based on what you use, others will require a base subscription fee. You then incur extra charges either based on your total usage, or on any usage past a base allowance.

**Pros:**

- With this SaaS pricing model, the price you pay scales with the user. Only what is used is paid.
- It's adaptable to seasonal changes. If you have a slow month in terms of revenue, your use of the software may reflect this, meaning the price is adjusted accordingly.

**Cons:**

- It can be harder to predict the monthly cost of your SaaS solution. A month with more usage than normal can lead to a surprising bill.
- Most come with a base subscription charge, either with an allotted usage allowance or not. This can mean a higher comparative cost for low usage of the SaaS.

#### *4.2.1.7 Flat rate SaaS pricing*

*Flat rate* is probably the simplest of the SaaS pricing models, but also quite rare. A provider offering a flat-rate SaaS pricing option is offering one product, with a single set of features, at a single monthly price.

**Pros:**

- Flat-rate pricing is easy to understand. You don't have to worry about choosing the right setting or package to meet your needs. There's only one option.

- With flat rate pricing, you always know exactly how much your SaaS will cost each month. So, you can plan your resources and expenditure accordingly.

**Cons:**

- Flat-rate pricing tends to have a higher cost than other options. This can be a high barrier for smaller businesses, or those not looking for a big, heavy-duty SaaS solution.
- There's a lack of flexibility for flat-rate pricing model SaaS. You can't tailor the price to your usage, needs or required feature set, which can lead to surplus features and high costs.

#### *4.2.1.8 Per-User SaaS pricing*

Also known as pay-per-seat, pay-per-user pricing models come in two forms: pay per user, and pay per active user. In these SaaS pricing models, the monthly price you pay for your SaaS solution is based on how many users can or are using the software.

Pay per user means you can only have the number of user accounts you pay for. Pay per active user allows as many user accounts as you like, but only the specified number of users can be active at any given time.

**Pros:**

- Pay-per-user models allow you to tune the price of your SaaS solution to suit your size. If you only need a handful of users, you only pay for that handful of users. Plus, as you grow you can add (active) users to your plan to suit you.
- This pricing model also makes it easy to spread the use of your SaaS solution across your enterprise. You can use your solution anywhere if you have paid for enough seats to cover the usage.

**Cons:**

- Both variants of the pay-per-user model may require some trial and error before you find the best balance of users to support your needs.

- You may find yourself paying for non-active seats during months or seasonal times where business is slow.

#### *4.2.1.9 Pay-per-feature SaaS pricing*

Pay per feature is a pricing model that uses features and functionality as the value metric to base price on. So, the higher the price, the more features you have available to use.

**Pros**:

- The right pay-per-feature solution allows you to only pay for the features you need. You avoid non-useful functionality and shape the product around your own use case.
- It can save user money if you only want a few features.

**Cons**:

- It can prove frustrating to pay for a software program but be denied full use of all the features advertised.
- You could miss out on a key useful feature that you would get with other pricing models.

## 4.3 API management

The term "API management" refers to the collection of procedures and equipment used to plan, implement, maintain, and secure APIs. It covers a variety of tasks, such as producing documentation and enforcing access rules as well as enhancing API performance and guaranteeing scalability. I will expand on the management of an API that has reached its Run-State in its lifecycle.

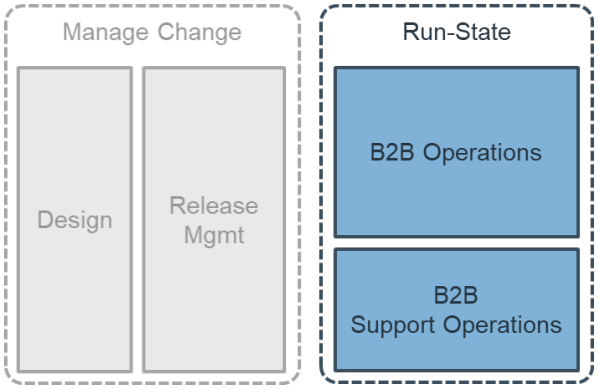


*Figure 9 - Phases of NM B2B API Management*

### 4.3.1 Service Level Management

The service level management is an integral part of the API management. Its objective is to "*maintain and gradually improve the alignment and the quality of IT services, through a constant cycle of agreement monitor reporting and revision of achievements of IT services and trough the implementation of action to eliminate unacceptable service levels"* (ITIL Italia, 2006).

Service level management. In various forms, is critical to ensure the business continuity and evaluating strategies to ensure site reliability. This per se is a complex topic which I will not expand on, but any API manager should deepen its knowledge on the matter to better comply with the service level that wants to achieve.

#### *4.3.1.1 Service Level Monitoring*

Service Level Monitoring (SLM) for APIs is a critical practice that involves actively measuring and tracking the performance and availability of APIs to ensure they meet predefined service level agreements (SLAs) and deliver a high-quality user experience.

SLM enables organizations to monitor key metrics associated with their APIs in real-time, allowing them to proactively identify and address issues that may impact performance and reliability. By implementing SLM for APIs, businesses can gain valuable insights into the health and performance of their API infrastructure. This information helps them identify bottlenecks, optimize resource allocation, plan for capacity needs, and prioritize improvements.

SLM typically involves setting up monitoring tools or platforms that collect and analyse API data. These tools may offer customizable dashboards, alerts, and reporting capabilities, enabling organizations to visualize and interpret API performance metrics effectively.

#### *4.3.1.2 Service Level Agreement*

The term Service level agreement refers to the contract with the users that includes consequences - most easily recognized when they are financial - of meeting (or missing) the performance objective they contain. It places a legal wrapper around the service level objectives, discussed later in more detail.

The following sections describe our understanding and considerations for defining and measuring the performance of the NM B2B API in order to ensure that it is meeting its performance and reliability goals.

API performance monitoring is necessary to ensure that the NM B2B API is meeting the required levels of performance, reliability, and availability. It should be automated and include the collection and calculation of service level metrics.

Regular reporting is also necessary to provide visibility into the NM B2B API's performance, to communicate them to our stakeholders and identify areas for improvement.

Here follows a proposed list of the main technical requirements:

- **Data collection infrastructure**: A reliable and scalable data collection infrastructure is necessary to gather data on the indicators being monitored. This can include tools for logging, tracing, and monitoring the API.
- **Data storage**: A data storage solution (e.g., DWH[15]) that can handle large amounts of data and provide fast access to the data is necessary in order to store and analyse the performance indicators.
- **Data processing**: A data processing solution that can handle large amounts of data in real-time and provide quick access to the processed data is

[15] Data Warehouse, one of the solution that can be implemented to log API data.

necessary in order to analyse the performance indicators. This can include stream processing, batch processing, and data warehousing.

- **Dashboards and visualizations**: A solution (e.g., PowerBI) for visualizing the data and providing insights into the performance indicators is necessary in order to monitor the performance of the API. This can include dashboards, graphs, and charts.
- **Alerting**: An alerting system (e.g., PowerBI notifications) that can send notifications when indicators fall outside of predefined thresholds is necessary in order to quickly identify and respond to issues with the API.
- **Site Reliability Infrastructure**: Site reliability describes the stability and quality of service that an application offers after being made available to end users. The site reliability infrastructure is key to ensure business continuity[16], in particular in such a highly regulated and safety critical environment as civil aviation.

#### 4.3.1.2.1 Service Level Indicators

Service level indicators (SLI) are carefully defined quantitative measure of some aspect of the level of service that is provided.

These indicators should be relevant, reliable, and easy to collect and measure.

Not every metric we can track in our systems should be consider as an SLI. We have to find a set of representative, meaningful, indicators of the system's health status.

The following SLIs would make sense, among others to be discussed with our stakeholders:

- **Availability**: This indicator measures the percentage of time the API is available and responding to requests. It can be measured by tracking the number of successful API responses over a given time period.
  - Time-based: How long the service was unavailable for a period of time.
  - Aggregate events: The proportion of valid requests served successfully.

[16] Business continuity is a business's level of readiness to maintain critical functions after an emergency or disruption

- **Latency**: The proportion of valid requests served faster than a threshold. This indicator measures the time it takes for the API to process a request and return a response. This is a key metric for ensuring the API can handle high levels of traffic and can help us understand the performance of the API under heavy load. Latency does not include network-related delays.
- **Throughput**: The proportion of time where the data processing rate is faster than a threshold.
- **Error Rate**: This indicator measures the number of unsuccessful API requests compared to the total number of API requests over a given time period. This can help us identify issues with the API and take corrective action as needed.
- **Concurrent Requests**: This indicator measures the number of simultaneous requests that are sent to a specific endpoint within the API at any given time. This is important for ensuring that the API can handle high levels of traffic and can help us identify when additional resources may be needed.

Several aggregation techniques can be used to gather and analyse SLIs in order to monitor the API's performance:

- *Average*: The mean value of the data over a specified time period.
- *Median*: The value that separates the data into two halves.
- *Percentiles*: The value at which a specified percentage of the data falls below.
- *Histograms*: This technique groups data into specified ranges, also known as "bins," and shows the frequency of data points within each bin. This can be useful for visualizing the distribution of data points.
- *Summaries*: This technique summarizes the data into key values, such as minimum, maximum, and average, for a specified time period.
- *Sampling*: This technique involves collecting a subset of data points to represent the entire dataset. This can be useful for monitoring SLIs in real-time, as collecting all data points may not be feasible.

Some examples of the KPIs mentioned above are reported in the following table.

*Table 1 - KPIs proposed for NM B2B API*

| **SLI**[17] | **Definition** | **SLO**[18] |
|---|---|---|
| *Availability* | This measures the proportion of successful requests, as measured from the load balancer metrics. | Any HTTP status other than *X-X* is considered successful. |
| *Latency* | This measures the proportion of sufficiently fast requests, as measured from the load balancer metrics. | "Sufficiently fast" is defined as < *X ms* |
| *Uptime* | This measures the percentage of time that the server is available versus the time it is unavailable due to maintenance or other issues | The service should be available >*X%* of the time |
| *Downtime* | This measures the percentage of time that the server is unavailable versus the time it is available due to maintenance, or other issues | The service should be unavailable <*X%* of the time |
| *Throughput* | This measures the amount of data transferred from the server to the client per unit time. It indicates the capacity of the server to handle requests and serve content. | The server must maintain a minimum throughput of X amount of data transfer per second during peak hours (or throughout the day) with a Y% percentile of requests |
| *Server resource utilization* | This measures the usage of server resources such as CPU, memory, and disk space. It indicates the capacity of the server to handle resource-intensive processes and optimize resource allocation | The utilization of the resources should be below <*X% for Y% of the time* |
| *Capacity planning (?)* | Monitoring the usage of server resources over time to forecast the future demand | The server capacity must be reviewed and increased (if necessary) in advance |

[17] A quantitative measure of some aspect of the level of service that is provided
[18] A target value or range of values for a service level that is measured by an SLI

| | | |
|---|---|---|
| | and ensure that the server can handle the load | of a predicted X% increase in traffic or request volume to ensure the server can handle the increased load without impacting performance, with a Y% percentile of requests |

#### 4.3.1.2.2 Service Level Objectives

Service level objectives (SLO) are target value or range of values for a service level that is measured by an SLI.

$$SLI \leq target$$

or

$$lower\_bound \leq SLI \leq upper\_bound$$

These objectives should be specific, measurable, and achievable, and should consider the needs of the API users.

Once again, a categorisation of users can streamline the process, by tailoring the correct SLOs for the correct group of users and either organisation type or business case should be considered in order to define subset.

# 5 Conclusions

In my study I focused on the steps to be taken to set up an API for industrialisation success. The path to manage a successful API industrialisation may diverge down the road, but the preparation steps are shared across highly standardised industries and a correct view of the big picture at the beginning is common across industries and can be boiled down to the following, critical points

**i) Understand the audience**

Who are you building your API for? What features can you offer them? What are their priorities? Reliability, flexibility or scalability? These are only a few of the question you should ask yourself before even starting planning your API industrialization.

**ii) Use architecture to build coherence and stand the test of time**

. Enterprise architecture allows for the strategic alignment of APIs with business objectives, facilitating seamless integration and interoperability. It ensures consistency in design principles, data models, and security measures across the API landscape, enabling scalability and adaptability as needs evolve. By investing in a well-defined enterprise architecture framework, businesses can future-proof their API infrastructure, promote collaboration, and foster innovation for sustained success.

**iii) Collect data and learn from them**

Collecting data from APIs providers enables businesses to tap into a wealth of real-time insights, uncovering trends, patterns, and opportunities. To harness the power of data means to make informed decision and to grow organically. Successful organizations understand that data is not static: it is a dynamic asset that demands ongoing attention. Regularly monitoring and reviewing of API data enables the providers to detect anomalies, track key performance indicators, and stay informed about shifts in customer behavior.

**iv) Structure an efficient and appropriate service support, but complement with a dynamic community**

To ensure an efficient and appropriate service support for APIs, it is essential to establish a well-structured framework. This includes implementing robust documentation, comprehensive developer resources, and responsive customer support channels. However, to truly maximize the potential of APIs, it is equally important to complement this support structure with a dynamic community. Cultivating an engaged and active community around the APIs fosters collaboration, knowledge sharing, and innovation, and it allows organization to create an environment that nurtures API adoption, drives continuous improvement, and propels the success of their API

**v) An API is an agreement: define it clearly to tailor it for the business**

By establishing a clear definition, businesses can align the API with their unique goals and objectives, ensuring it seamlessly integrates with existing systems and processes. Clear API definitions also promote transparency, facilitate effective communication between stakeholders, and enable efficient development and integration efforts, ultimately driving successful outcomes for the business. By analyzing user behavior, usage patterns, and feedback, businesses can gain valuable insights into the features and functionalities that are most important to their users. This information can then be used to refine the API's definition, ensuring it aligns closely with the needs and expectations of the intended audience

To conclude my thesis, it is important that I summarize the answer to the original research question: *how to manage the process of API industrialisation in the context of a highly regulated industry?*

While there is no short answer to this, the initial approach to solving the question can go a long way, as demonstrated by the case of EUROCONTROL, and many of the companies overviewed in the literature, there is a correct approach that pays off in the long run: structuring your system and your process. Understanding the industry and building a modular architecture is crucial to adapt to everchanging standards. And in conclusion API is a business model driven by data. Collecting, analysing and making decision based on information extracted by data is a value added that any API manager should pursue to improve the service level provided.

APIs can revolutionise the IT services market, but to do so, the IT services need to learn and adapt to APIs, which will not take long in a dynamic, fast-paced industry, striving for standards improvement.

# 6 Bibliography

Commission Implementing Regulation (EU) 2021/116 (1 February 2021).

EUROCONTROL. (2021). *SWIM Reference*. Retrieved from https://reference.swim.aero/index.html

Weir, L. (2019). *Enterprise API Management: Design and deliver valuable business APIs.* Packt Publishing Ltd.

EUROCONTROL. (2023). *About Us*. Retrieved from https://www.eurocontrol.int/about-us

EUROCONTROL. (2023). *Network Manager B2B Web Services*. Retrieved from https://www.eurocontrol.int/service/network-manager-business-business-b2b-web-services

Jeanne W. Ross, P. W. (2006). *Enterprise Architecture as a strategy.* Boston: Harvard Business School press .

Daniel Jacobson, G. B. (2011). *APIs: a strategy guide.*

Dize Dinçkol, P. O. (2023). Regulatory standards and consequences for industry architecture: The case of UK Open Banking. *Elsevier*.

De, B. (2017). *API Management: An Architect's Guide to Developing and Managing APIs for Your Organization.* Bangalore, Karnataka, India: apress.

Biehl, M. (2015). *API architecture: The Big Picture for Buidling APIs.*

Clark, C. Y. (2000). *Design Rules: the Power of Modularity* . The MIT Press.

Lutkevich, B. (2023). *What is the API Economy?* Retrieved 04 17, 2023, from https://www.techtarget.com/searchapparchitecture/definition/API-economy

Kirchoff, L. (2017, 09 07). *The ultimate guide to pricing your API*. Retrieved from https://nordicapis.com/the-ultimate-guide-to-pricing-your-api/

Mehdi Medjaoui, E. W. (2019). *Continuos API Management: making hte right decisions in an evolving landscape.* O'Reilly.

SKYbrary. (n.d.). *Network Manager*. Retrieved from https://www.skybrary.aero/articles/network-manager

al., M. L. (2017). *Enterprise Architecture at work: Modelling, communications and Analysis.* Springer.

EUROCONTROL. (n.d.). *Network operations*. Retrieved from https://www.eurocontrol.int/network-operations

Microsoft . (n.d.). *Publisher-Subscriber pattern*. Retrieved from https://learn.microsoft.com/en-us/azure/architecture/patterns/publisher-subscriber

Pethuru Raj, A. R. (2017). *Architectural Patterns.* Packt.

EUROCONTROL. (n.d.). *Enhanced tactical flow management system*. Retrieved from https://www.eurocontrol.int/system/enhanced-tactical-flow-management-system

Susanne K. Schmidt, R. W. (n.d.). *Coordinating technology: Studies in the international standardization of telecommunications.* The MIT press.

Knut Blind, A. M. (n.d.). Motives to standardize: Empirical evidence from Germany. *Elsevier*.

Paul Moritz Wiegmanna, H. J. (2017). Multi-mode standardisation: A critical review and a research agenda. *Elsevier*.

ITIL Italia. (2006). *Service Level Management* . Retrieved from https://www.itil-italia.com/itilservicelevelmgt.htm

Kotusev, S. (2018). *The Practice of Enterprise Architecture: A Modern Approach to Business and IT Alignment* .

# Acknowledgments

*Thanks to Prof. Paolo Spagnoletti,*
*For believing in this project and giving me the tools and knowledge to make it happen.*

*Thanks to Alessandro,*
*For allowing this to happen, and guiding me every day through numerous obstacles trusting me and my ability to learn.*

*Thanks to my classmates,*
*For sharing these years, working through long hours, many laughs, and a fair share of fun.*

*Thanks to my colleagues at EUROCONTROL,*
*For sharing their knowledge (and quite a few coffees) with me.*

*Thanks to Federico,*
*For patiently explaining the fascinating intricacies of aviation.*

*To Mom,*
*For teaching me that not everything I should learn can be found on a book.*

*To Dad,*
*For teaching me how to fill the unforgiving minute with sixty seconds' worth of distance run.*

*To Leonardo,*
*For standing by my side every step of the way, supporting me and helping me as only a brother can do.*

*Thanks to all my Friends and Family,*
*For the unconditional support through countless hard times and at least as many joyful moments.*