API4IPS essentials

API4IPS WG #6

eDelivery informal cooperation Network

28th June 2021

European Commission – Joint Research Centre

Monica Posada-Sanchez

Lorenzino Vaccari

Katarzyna Pogorzelska

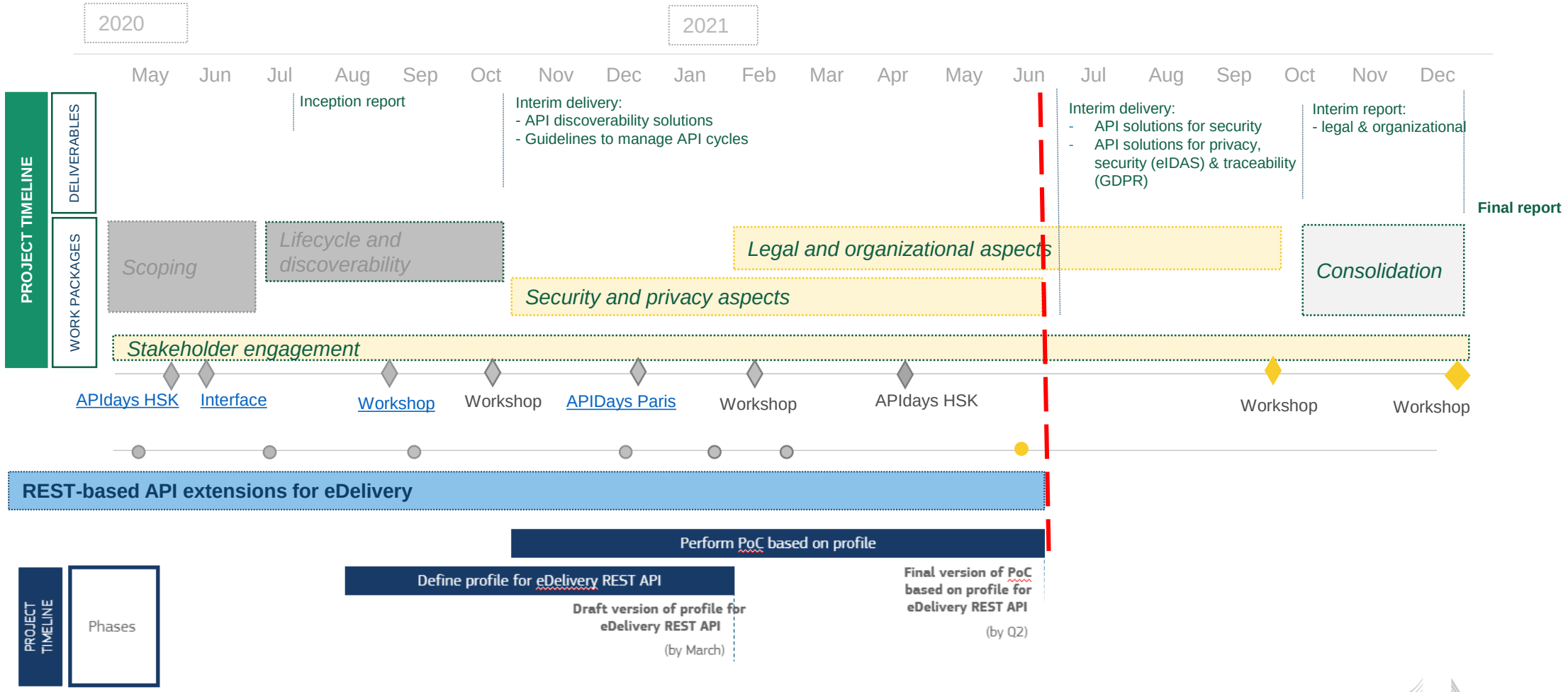
Update on JRC's work on API guidelines for government

API4IPS technical. legal and organisational essentials

Time	Items	Speakers
12.00 - 13.00	• Security & Privacy essentials highlights • Empirical analysis contractual conditions of APIs	Monica Posada, Katarzyna Pogorzelska (JRC B6) Lorenzino Vaccari

API4IPS timeline

Where are we?



European Commission

Accomplishments 2021
[Q1 & Q2]

2021 Q1:

1. Report published in Science Hub: Workshop on the role of APIs in governance processes
2. Stakeholder management: Uptake of our API reports by Member States
3. Framework of API adoption in the public sector published as ISA^2 solution

2021 Q2:

1. Publication of paper on MDPI Data special issue: APIs for EU Governments: A Landscape Analysis
2. Interim deliverable on API technical essentials: ***API security, traceability and privacy***
   consolidated draft due end of June
3. Interim deliverable on ***API legal & organizational essentials***
   first draft end of June

Technical essentials

SECURITY, PRIVACY & TRACEABILITY

Regulatory context (eIDAS)

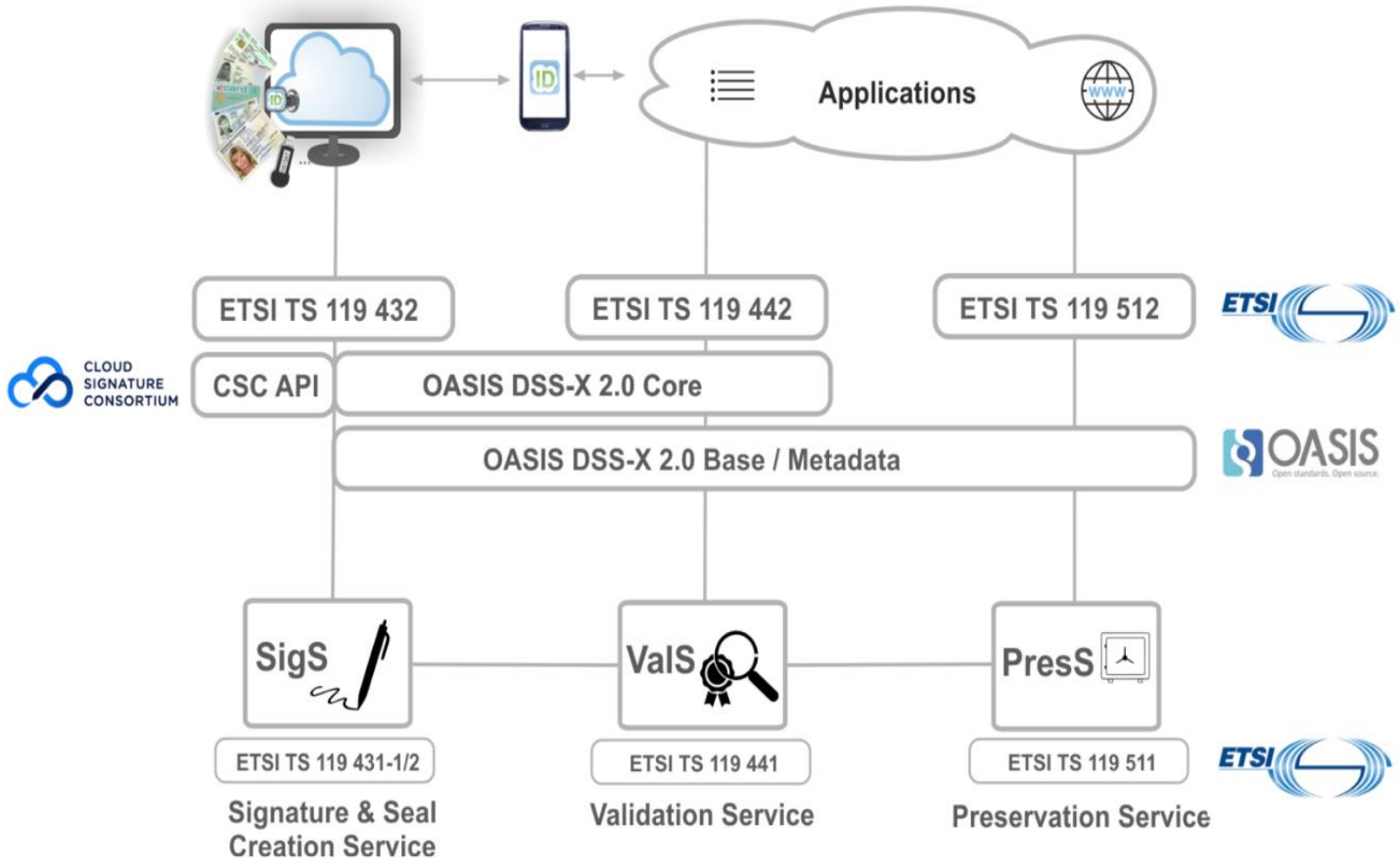
The **eIDAS** (EU Regulation Nr. 910/2014) provides a predictable regulatory environment to **enable secure and seamless electronic interactions** between businesses, citizens and public authorities.

eIDAS provides the regulatory environment for important aspects related to electronic transactions, including the ***Advanced electronic signature***, the ***Qualified electronic signature***, the ***Qualified digital certificate for electronic signature***, the ***Qualified website authentication certificate*** and the ***Trust service*** requirements.

eIDAS supporting tools include the following connecting European Facilities (CEF) building blocks: '**eID**', '**eSignature**', '**eDelivery**' and '**European Blockchain Services Infrastructure (EBSI)**'.

OASIS and ETSI ‘eIDAS-API’

- In order to facilitate the practical use of eIDAS of **trust services** in an interoperable way and to provide a solid foundation for a prosperous and sustainable eIDAS-Ecosystem, OASIS and ETSI have recently proposed ‘**eIDAS-API**’ which adopts **API-standards** for the **creation, validation, and long-term preservation of signatures, seals, timestamps and evidence records**.



European Commission

ENISA general purpose recommendations

Security recommendations for eIDAS are proposed by the **European Union Agency for Cybersecurity (ENISA) that among its publications includes:**

- A "**Security framework for Trust Service Providers"** recommending risks assessment, analysis and evaluation and appropriate measures to mitigate the impact of security incidents as well as inform the stakeholders of the adverse effects of any such incidents (identification, preparation, detection, response and eradication)
- A comprehensive guideline on "**Good Practices for Security of IoT - Secure Software Development Lifecycle**" which, among its 81 security best practices, contains useful recommendations for APIs on:
  - **People**: security training, up to date of new security issues, assign security roles and privileges, establishing the correct governance
  - **Processes**: proper management of 3rd parties components, correct management of operations (incidents, change management, etc.), control access and authorization policies, security design, establish internal policies to prevent any disclosure of information
  - **Technologies**: Use libraries and 3rd parties components patched with last vulnerabilities, use secure and well known communication protocols, standards, web interfaces, session management, review security and use whitelists

Specific API security risks & mitigation measures

Most recurrent causes and mitigation measures regarding security aspects in APIs have been analysed, clustered and prioritised by the Open Web Application Security Project® (OWASP) foundation.

Each year OWASP identifies 10 most important security risks and mitigation measures. 2019's list:

1. **Broken Object Level Authorization**
2. **Broken User Authentication**
3. **Excessive Data Exposure**
4. **Lack of Resources & (missing) Rate Limiting**
5. **Broken Function Level Authorization**
6. **Mass Assignment**
7. **Security misconfiguration**
8. **Injection**
9. **Improper Assets Management**
10. **Insufficient Logging & Monitoring**

OWASP 1 & 2

1. Broken Object Level Authorisation

*"APIs tend to **expose endpoints that handle object identifiers**, creating a wide attack surface Level Access Control issue."*

- **Object level authorization checks should be considered in every function that accesses a data source using an input from the user**.
- System-provided IDs should be granted using a random ID generation rather than being sequentially allocated.
- IDs used in each session should be drawn from storage rather than those that the client sends.
- Use an authorization mechanism to check if the logged-in user has access to perform the requested action on the record in every function that uses an input from the client to access a record in the database.
- Implement a proper authorization mechanism that relies on the user policies and hierarchy.
- Write tests to evaluate the authorization mechanism. Do not deploy vulnerable changes that break the tests.

2. Broken User Authentication

*"**Authentication mechanisms are often implemented incorrectly**, allowing attackers to compromise authentication tokens or to exploit implementation flaws to assume other user's identities temporarily or permanently."*

- **Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes**.
- Multi-factor authentication, standard authentication protocols, token generation, and protected password storage should be used.
- API keys should not be used for user authentication.
- Password must be correctly maintained and protected
- Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login.
- Implement anti brute force mechanisms
- Implement account lockout / captcha mechanism to prevent brute force against specific users.

OWASP 3 & 4

3. Excessive Data Exposure

"*Looking forward to generic implementations, developers tend to **expose all object properties without considering their individual sensitivity**, relying on clients to perform the data filtering before displaying it to the user .*"

- **All personally identifiable information transmitted via the API should be identified and use of private data should be justified**. Classify sensitive and personally identifiable information (PII) that your application stores and works with, reviewing all API calls returning such information to see if these responses pose a security issue.
- Backend engineers should always consider the consumer of the data before exposing a new API endpoint.
- Avoid using methods to generally return all content of data (such as "to_json()" and "to_string()") and consider returning only specific information (i.e. selected properties of data).
- Implement a schema-based response validation mechanism as an extra layer of security. As part of this mechanism define and enforce data returned by all API methods, including errors.

4. Lack of Resources & Rate Limiting

*"Quite **often, APIs do not impose any restrictions on the size or number of resources that can be requested by the client/user**. Not only can this impact the API server performance, leading to Denial of Service (DoS), but also leaves the door open to authentication flaws such as brute force."*

- **Establish limits to the use of resources by API calls**.
- Implement a rate limit, i.e. a limit on how often a client can call the API within a defined timeframe.
- Checks on compression ratios should be added.
- Add proper server-side validation for query string and request body parameters, specifically the one that controls the number of records to be returned in the response.
- Define and enforce maximum size of data on all incoming parameters and payloads such as maximum length for strings and maximum number of elements in arrays.
- Notify the client when the limit is exceeded by providing the limit number and the time at which the limit will be reset.

OWASP 5 & 6

5. Broken Function Level Authorization

"***Complex access control policies*** *with different hierarchies, groups, and roles, and an unclear separation between administrative and regular functions,* ***tend to lead to authorization flaws****. By exploiting these issues, attackers gain access to other users' resources and/or administrative functions."*

- **Use strict identity access management processes and only allow users belonging to specific groups or roles to be enabled**.
- The enforcement mechanism(s) should deny all access by default, requiring explicit grants to specific roles for access to every function.
- Review your API endpoints against function level authorization flaws.
- Make sure that all of your administrative controllers inherit from an administrative abstract controller.
- Make sure that administrative functions inside a regular controller implements authorization checks based on the user's group and role.
- Applications should have a consistent and easy way to analyse authorization module that is invoked from all your business functions.

6. Mass Assignment

*"**Binding the client provided data** (e.g., JSON) **to data models**, **without proper properties filtering based on an allow list, usually leads to Mass Assignment**. Either guessing objects properties, exploring other API endpoints, reading the documentation, or providing additional object properties in request payloads, **allows attackers to modify object properties they are not supposed to**."*

- **Precisely define the schemas, types, and patterns accepted in requests at design time and enforce them at runtime**.
- Explicitly define all the input and output parameters and expected payloads.
- Use a multi-tier architecture and do not automatically bind incoming data and internal objects.
- Use the "readOnly" property set to true in object schemas for all properties

  that can be retrieved through APIs but that should never be modified.

OWASP 7 & 8

7. Security misconfiguration

"***Security misconfiguration*** *is commonly a result of insecure default configurations, incomplete or ad-hoc configurations, open cloud storage, misconfigured HTTP headers, unnecessary HTTP methods, permissive Cross-Origin resource sharing (CORS), and verbose error messages containing sensitive information.*"

- Ensure **API can only be accessed by the specified HTTP verbs**.
- APIs expecting to be accessed from browser-based clients (e.g., WebApp front-end) should implement a **proper Cross-Origin Resource Sharing (CORS) policy**.
- To prevent exception traces and other valuable information from being sent back to attackers, if applicable, **define and enforce all API response payload schemas including error responses**.
- The **API life cycle** should include:
  - A repeatable hardening process leading to fast and easy deployment of a properly locked down environment.
  - A task to review and update configurations across the entire API stack.
  - A secure communication channel for all API interactions access to static assets.

8. Injection

*"**Injection flaws**, such as SQL, NoSQL, Command Injection, etc., **occur when untrusted data is sent to an interpreter as part of a command or query**. The attacker's malicious data can trick the interpreter into executing unintended commands or accessing data without proper authorization ."*

- **Keep data separate from commands and queries of the APIs**.
- Use safe APIs, reducing the use of an interpreter of data.
- Use positive or "whitelist" server-side input validation and for any residual dynamic queries, escape special characters using specific escape syntax for the interpreter.
- Strictly define all input data, such as schemas, types, and string patterns, and enforce them at runtime.
- Use LIMIT and other SQL controls within queries to prevent mass disclosure of records in case of SQL injection.

OWASP 9 & 10

9. Improper Assets Management

*"APIs tend to **expose more endpoints than traditional web applications, making proper and updated documentation highly important**. Proper hosts and deployed API versions inventory also play an important role to mitigate issues such as deprecated API versions and exposed debug endpoints."*

- **When newer versions of APIs include security improvements, perform risk analysis** to make the decision of the mitigation actions required for the older version.
- **Keep an up-to-date inventory all API** hosts and document important aspects of each one of them, focusing on the API environment, who should have network access to the host and the API version.
- **Document all aspects of your API**.
- Generate documentation automatically and make API documentation available to those authorized to use the API.
- **Limit access to anything that should not be public and to production data**, and segregate access to production and non-production data.
- Implement additional external controls, such as **API firewalls**.
- Properly **retire old versions of APIs** or backport security fixes to them.

10. Insufficient Logging & Monitoring

*"**Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response**, allows attackers to further attack systems, maintain persistence, pivot to more systems to tamper with, extract, or destroy data."*

- **Encourage full monitoring and logging of API traffic to enable behaviour modelling and detection of anomalies**, geared towards enabling investigations, improving security configurations and policies, and fixing bugs.
- **Log everything and meticulously**. Log failed attempts, denied access, input validation failures, or any failures in security policy checks.
- **Protect logs to not publish highly sensitive information**.
- Use a Security Information and Event Management (SIEM) system to aggregate and manage logs from all components of the API stack and hosts.
- Configure custom dashboards and alerts, enabling suspicious activities to be detected and responded to earlier.
- Ensure that logs are formatted so that other tools can consume them as well.

API security standards

- Application level
  - OAuth 2.0 (Authorisation)
  - OpenID Connect (Authorisation & Authentication)
- Transport level
  - **Transport Layer Security** (TLS 1.2, 1.3)
- Message level
  - **JSON Advanced Electronic Signature (JAdES)**
  - **JSON Web Signatures (JWS)**

APIs privacy and traceability

Regulatory context (GDPR)

The **General Data Protection Regulation (GDPR) Regulation (EU)** 2016/679 is about the protection of natural persons with regard to the processing of personal data and on the free movement of such data

It defines that: "*Natural persons may be associated with online identifiers provided by their devices, applications, tools and protocols, such as internet protocol addresses, cookie identifiers or other identifiers such as radio frequency identification tags.* ***This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them****."*

Traceability elements

APIs represents a common way to exchange digital assets, including private data about individuals and companies.

Traceability in APIs exchange of information is essential for the functioning of a digital system in an organisation or in a network of organisations interacting in a digital ecosystem.

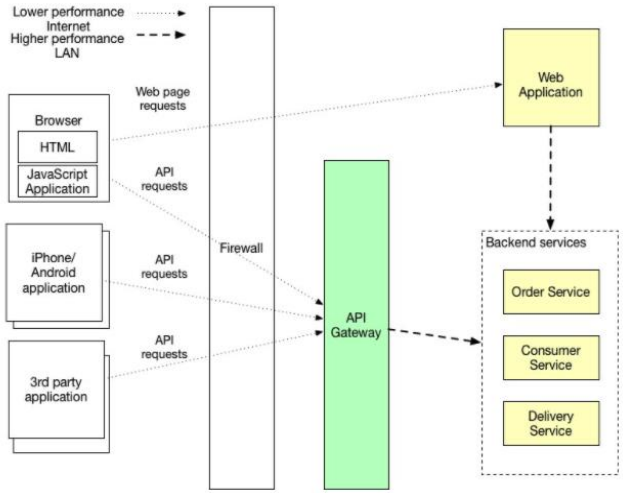
Traceability in APIs brings **risks of data privacy breaches**

Main elements of traceability are **actors** participating in the exchange of information and the **timestamp** of transactions.

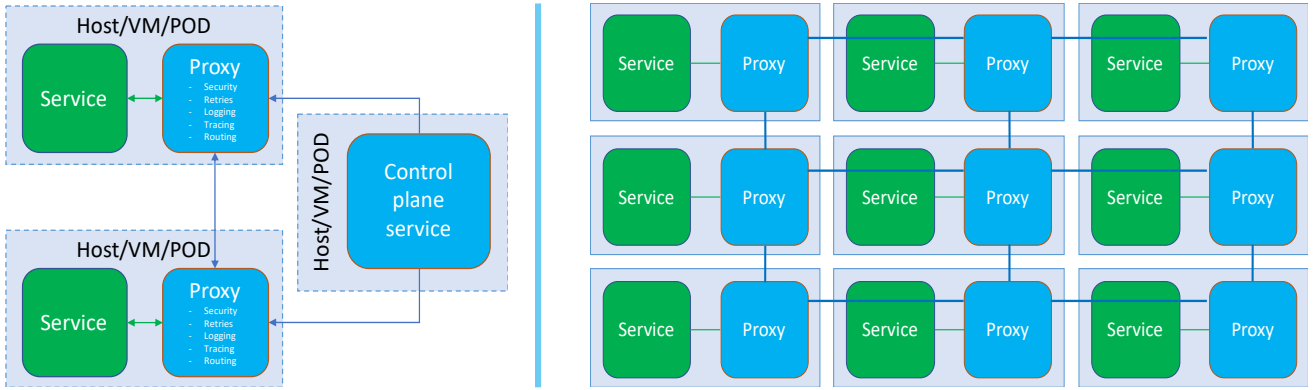
The **respect of the requirements of the GDPR** regulation must be considered with great attention during **APIs design, implementation and deployment**.

Governance of traceability

The main techniques and tools adopted to govern APIs in a distributed systems are **API gateways** and **service mesh**:



API gateway



Service Mesh

These two tools have to be configured to avoid data privacy breaches

Protecting traced information

- **OWASP** suggests best practices for logging & monitoring
- **Mydata** is an approach that was introduced in 2015 as '**a Nordic model for human-centred personal data management and processing**'
- The **Consent based Personal Data Suite (CaPe)** is a consent-based and user-centric platform targeted at Data Processors, in the private or public sector
- **Solid** is a mid-course correction for the Web by its inventor, Sir Tim Berners-Lee'. Solid gives people the power to store and control their data securely in decentralized **Personal Online Data Stores (Pod)**
- The **AMdEX** initiative aim is to **create a fair, open and reliable data market**. It contributes to the development of a fair and sustainable digital society and thus to a smart Amsterdam Metropolitan Area.
- Research in the field is also proposed by "***privacyTracker***" a framework to support the GDPR principles. The framework supports data traceability. Allowing a user to get a cryptographically verifiable snapshot of his/her data trail is also proposed

Legal & organisational essentials

- API LEGAL AND ORGANISATIONAL CONSIDERATIONS
- ANALYSIS OF THE LEGAL FRAMEWORK FOR APIS
- STRUCTURE AND ANALYSIS OF API TERMS OF SERVICE

API legal and organisational considerations

Why this analysis

For an organisation APIs are:

1. **Code** or **Software products** → digital assets
   - Intellectual Property Rights, Patents
2. **Operational services** → provided or consumed by organisations
   - Liability, accountability, ownership, rights of use.
3. **Enablers of the integration into digital ecosystems**
   - Coordination of responsibilities in all digital chain and ecosystem

API legal and organisational considerations

API service provider and consumer organizations

1. **Legal Framework**
   - laws, regulated topics, regulated sectors, regulations
2. **Capacity building**
   - Infrastructure
   - Skills
   - Ecosystem stakeholder management
3. **Operations**
   - Monitoring compliance, reporting obligations
   - Terms of Services (ToS, ToU, T&C)

Analysis of API legal framework

European perspective

APIs do not operate in a legal vacuum

- EU law applicable to data sharing
- the landscape is changing by the day

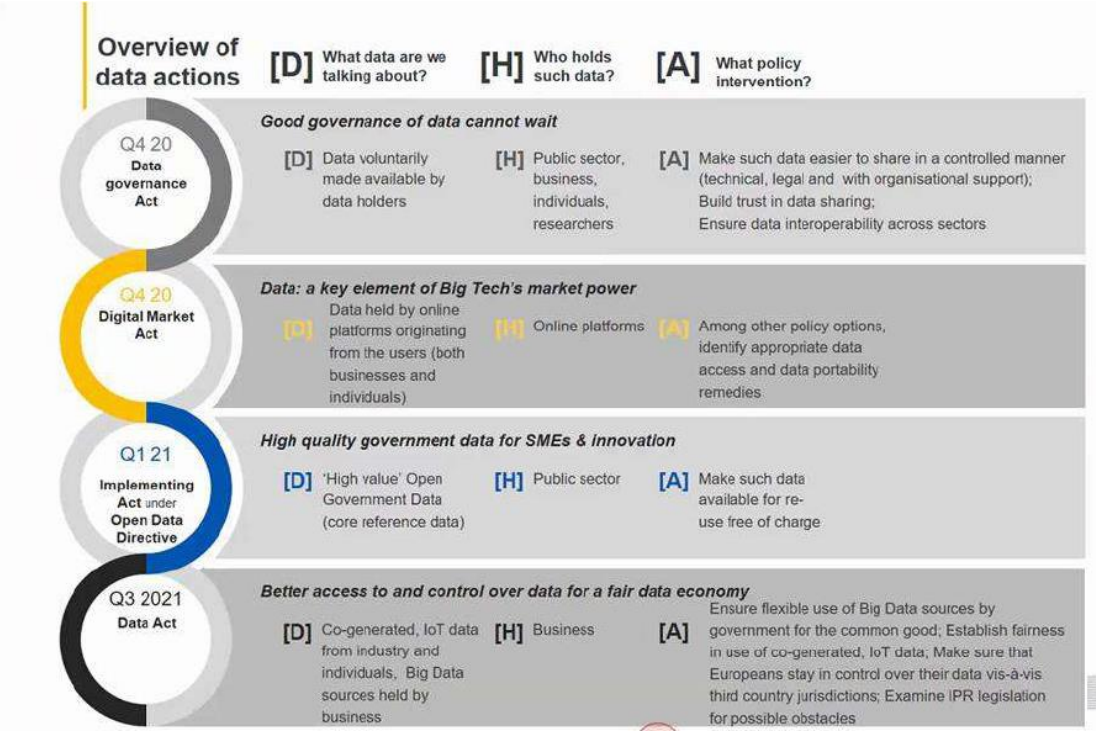
  DGA, DMA, DA (only in EU)

APIs can be:

- subject of regulation
- Means/tools for regulation

  Implementation, monitoring and reporting of data governance processes

European Data strategy roadmap



European Commission

Analysis of API legal framework

European perspective – data governance legal instruments

HORIZONTAL LEGAL INSTRUMENTS
Data Governance Act
Data Markets Act
Data Act
GDPR
Competition Law - TFEU
Data Base Directive
eCommerce Directive
Copyright DSM Directive
Regulation of the free flow of data
Trade secrets Directive
The Software Directive
The Digital Content Directive

SECTORAL LEGAL INSTRUMENTS
Open Data - Public Sector Information
PSD2 (banking)
MIFID framework
ePrivacy Directive & EU Electronic Com Code
INSPIRE Directive
Electricity Directive
Gas Directive
Regulation on Road safety (mobility)
Regulation on Vehicle Repair and Maintenance Info
REACH (chemicals)

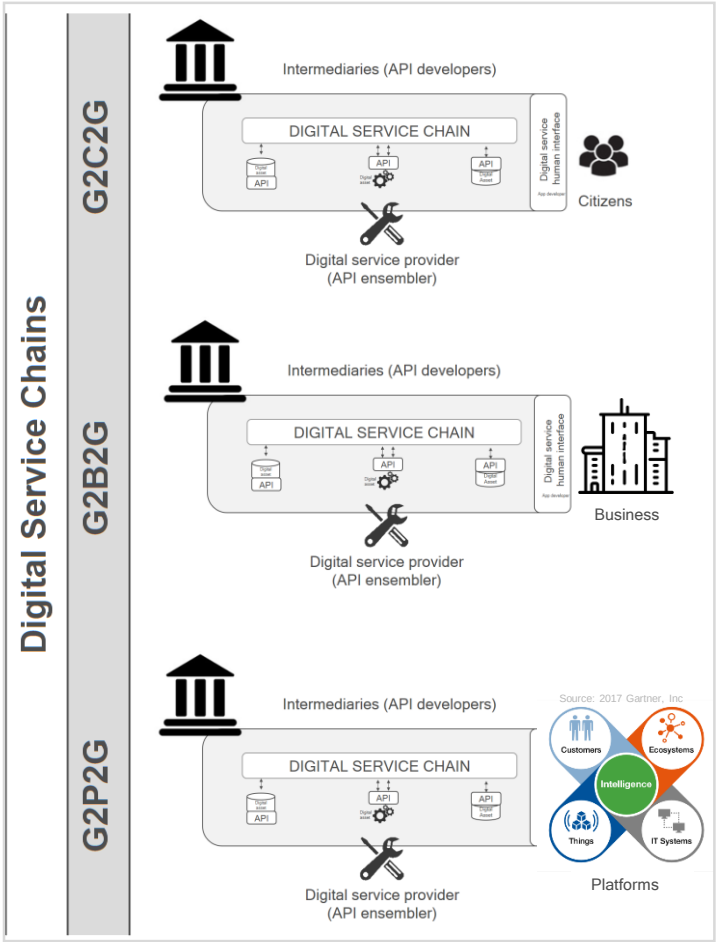
Analysis of API legal practices

API service agreements

WHAT

API service agreements → contracts

- ToS, ToU, T&C
- These documents define the conditions for interactions between actors (Citizens, Business,Platforms)



Examples of ToS in government

GEO - CONTEXT	SCOPE	LINKS
Swedish API licence	National	https://apilicens.se/en/
Singapore	National	https://www.mas.gov.sg/terms-of-use/api-terms-of-service
Canada	National	http://www.ca.gov/Use
Cataluña	Regional	https://www.idescat.cat/dev/api/?lang=en#cdu
Seattle	Local	https://data.seattle.gov/stories/s/Data-Policy/6ukr-wvup/

European Commission

Analysis of API legal practices

API service agreements

HOW

Description of Dataset

- snapshot in 2019 provided
  - self-declared ToS Documents: 4287
  - downloads succeeded with content: 2753

NLP analysis on practices of active players

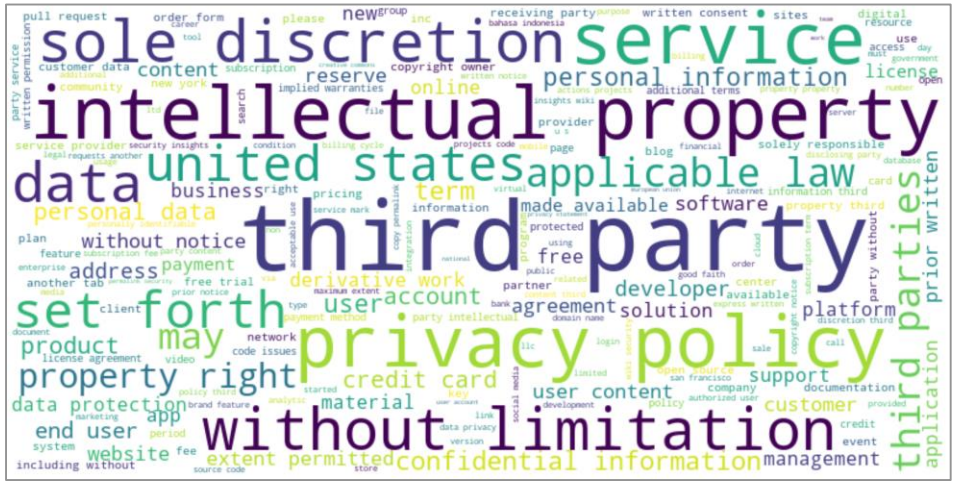
- Analysis of the structure
- Analysis of the conditions

Analysis of API Terms of Services

Legal empirics: Structure

Elements and conditions of ToS:

- Contracting parties to the agreement
- Start of the contract
- Termination/suspension/modification/restriction of service provision
- Payment
- Governing Law
- Liability
- Indemnification
- Warranty
- Privacy
- Severability
- License of the generated content (IPR)



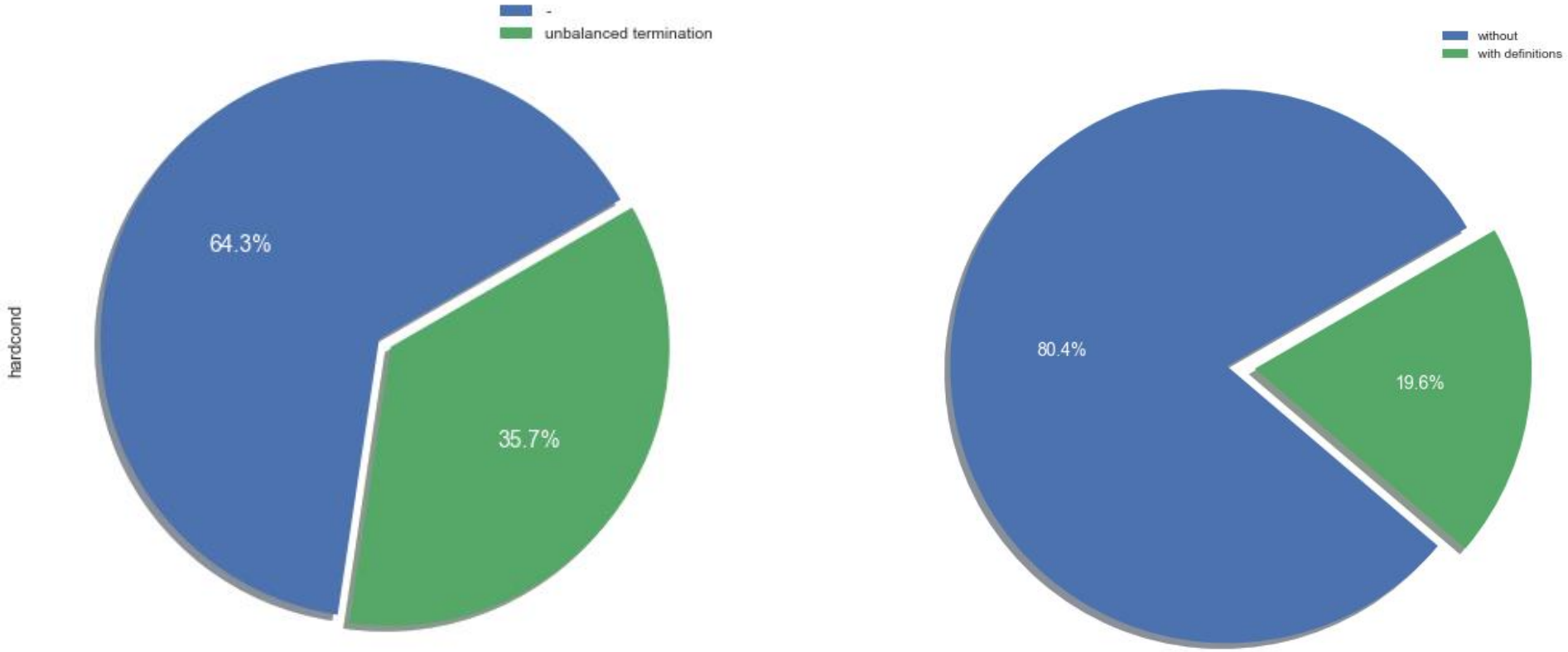
[1] Word cloud on collocations in the corpus

Word clouds obtained from:
[2] ToS headings
[3] ToS words in bold
[4] capitalized words

European Commission

Analysis of API Terms of Services

Legal empirics: examples of termination conditions, complexity



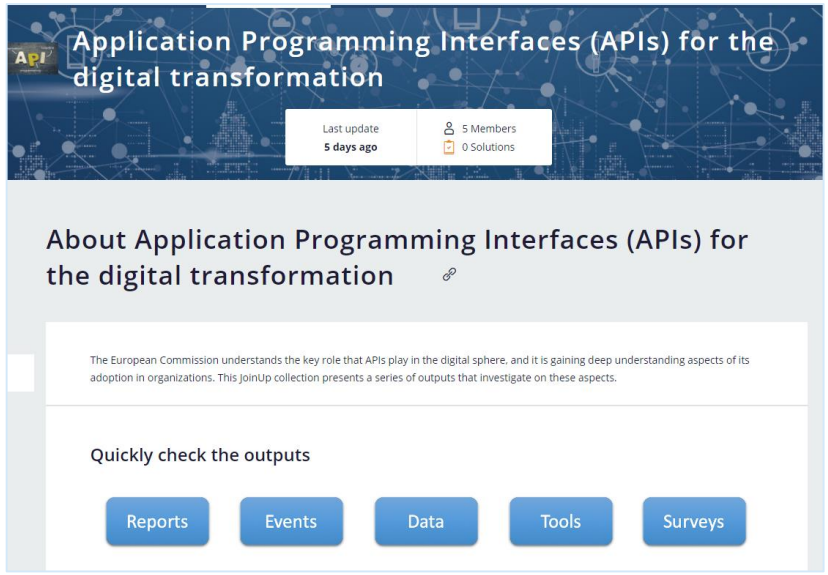
Ongoing analysis of practices:

- big players practices?, ownership practice?, legal framework observed?

What’s next

- Delivery of consolidated interim report on Security, Privacy (Q2)
- Delivery of first draft of legal and organizational essentials (Q2)
- Consolidation analysis of legal and organisational essentials (Q3)
  - Analysis of practices, mapping of actors, infrastructures, rights and responsibilities
  - Expert validation (workshop, case studies)

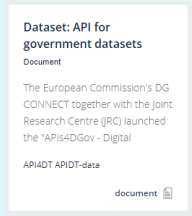
API4DT joinup collection – stay tuned!



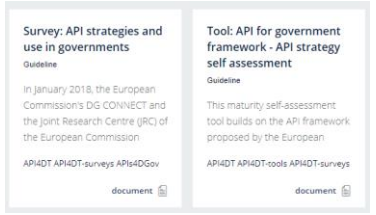
Join our community and stay tuned:
https://joinup.ec.europa.eu/collection/api4dt

REPORTS

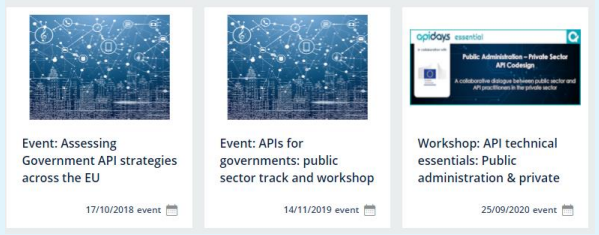
DATASETs



TOOLS & SURVEYS



EVENTS



Contact us directly jrc-apis4dgov@ec.europa.eu

European Commission

Thank you!

EU Science Hub: ec.europa.eu/jrc

@EU_ScienceHub

EU Science Hub – Joint Research Centre

EU Science, Research and Innovation

Eu Science Hub

European Commission