European Commission

# APIs for CPSV-AP based Catalogue of Services

**DIGIT**
**Directorate-General for Informatics**

**ISA² Programme**
**ec.europa.eu/isa2**

ISA²

EUROPEAN COMMISSION

Directorate-General for Informatics
Directorate B — Interoperability Solutions for public administrations, businesses and citizens
Unit B6 — ISA$^2$ Programme

*Contact:: isa2@ec.europa.eu*

*European Commission*
*B-1049 Brussels*

**DOCUMENT METADATA**

| Property | Value |
|---|---|
| Release date: | *04/09/2019* |
| Status: | *Published* |
| Authors: | *PwC EU Services* |
| Keywords (to increase the SEO presence) | *CPSV-AP, Core Vocabulary, API, Restful, Single Digital Gateway (SDG)* |

## Table of Contents

# 1. INTRODUCTION - USING APIS TO AUTOMATE DATA EXCHANGE

Over the past years, the use of APIs[1] has exploded exponentially and there are today thousands of public and private APIs[2] being used for a wide range of use-cases and a variety of purposes. In the public sector for instance, APIs have proven to be instrumental in fostering interoperability through data transfer. APIs are increasingly needed as the demand for transparency regarding the data provided and consumed by the public sector increases, which challenges traditional data management processes. Nowadays, citizens expect public administrations to provide fast and secure access to data while establishing proper connections with other public administrations.

## 1.1. Scope and objectives

The present document aims at identifying how public services can best be published by means of APIs. The scope of this study is limited to a desktop research on the usage of API in the public sector, interviews conducted with three Member States in order to identify different business cases, and the methodology on how to validate linked data through an API. The study is not intended to help the reader decide which type of API he should build or which security tool he should work with. This study rather wants to act as a point of reference for public administrations who wish to understand when to build an API and what important considerations need to be factored in when building an API.

## 1.2. Structure of this document

The remainder of this document develops as follows: the next section (section 2) aims to provide awareness to public administrations regarding the aspects that need to be considered before embarking on a new API journey. Section 2 is divided into three distinct parts: (i) understanding the role of the public administration; (ii) understanding the data at hand; and (iii) understanding the API lifecycle. Then, section 3 analyses the usage of APIs for public services. In particular, the JRC API study (section 3.1) and workshop (section 3.2) are looked at, along with the findings from interviews conducted with three Member States and the SDG business case (section 3.3). Next, section 4 focuses on how public administrations can ensure data quality when exchanging linked data. Finally, section 5 concentrates on the conclusions of the study.

---

[1] An Application Programming Interface, commonly referred to as an "API", is a software intermediary that, in simple terms, allows two applications to talk to each other.

[2] https://www.programmableweb.com/apis/directory

## 2. PREPARING FOR THE API JOURNEY

Building an API is a journey in itself and understanding what is required to move from the creation through the realization to the retirement stage is a challenge and struggle for many public organisations. Three key elements need to be carefully looked at before jumping into the making of an API.

First, public organisations should thoroughly consider and understand the role they intend to play: is the ultimate goal to *provide*, *consume*, or simply *publish* data?

Second, public organisations should have a sense of what makes up their data: is it structured, is it documented and can it be published?

Third, public organisations should understand how APIs mature and, in general, what the product lifecycle is — a series of predictable and useful stages. If understood properly, public organisations can then best anticipate and prepare themselves for each of the stages of the lifecycle.

### 2.1. Understanding your role within the API journey

Before building an API, a public administration will, more often than not, take the time to carefully understand the raison d'être of its API and whether it is worth building at all. In that regard, it is equally important for a public administration to understand and consider the role it intends and wishes to play with its API, an aspect that is often overlooked.

There exist **three main roles** that public administrations can choose from: *provider*, *consumer* and *publisher*.

First of all, public administrations can play an active role in *providing* data to citizens or other public administrations. For example, the French "API Particulier" helps citizens simplify some administrative procedures by allowing them to share their personal information with various French administrations.[3] Data is being *provided* to other public administrations through the API, as long as proper consent from citizens has been granted.[4]

Second, public administrations can *consume* data shared by other public organisations (e.g. a marriage certificate being transmitted from one municipality to another). For instance, the city of Lyon, located in France, is using the "API Particulier" developed by the DINSIC to compute the "family quotient" of its citizens.[5]

---

[3] https://api.gouv.fr/api/api-particulier.html

[4] Ed. Providing data to other public administrations is a concrete step towards the implementation of the "Once Only Principle" as multiple public administrations would not have to request the same information from citizens (see section 3.1).

[5] The calculation of the municipal "family quotient" is used to grant discounts on the price of school canteen meals in public or private schools and on the prices of some sport

Third, public administrations can *publish* data, generally on behalf of other public administrations, by providing discoverability services such as:

- a catalogue that gathers all available APIs of public services (see example Figure 1); and
- a catalogue of public services that is being provided as an API (see example Figure 2).[6]

The government of France has, for instance, built a portal that lists and provides access to all state-related APIs[7] under specific conditions (Figure 1), while the Finnish government provides a catalogue of available public services[8] by means of an API[9] (Figure 2).


**Figure 1 -** French catalogue of APIs of public services

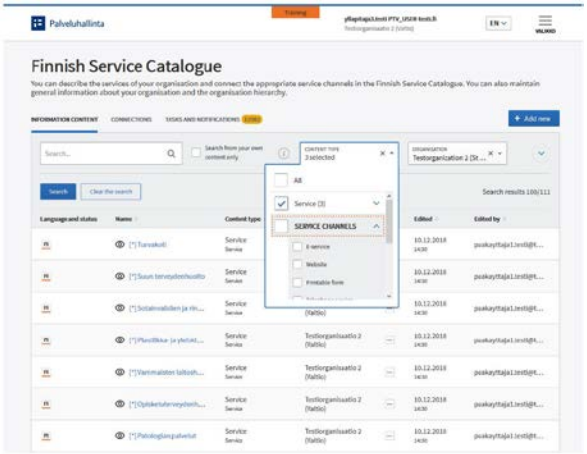


**Figure 2 -** Finnish catalogue of public services provided as an API

Regardless of the role chosen by a public administration, interoperability is of utmost importance and should always be taken into consideration. Data should, whenever possible, be provided in a smooth and fluid way (e.g. no data loss, automated transmission, etc.) as those principles lay down the premises to build proper APIs.

---

activities. The family quotient is determined by the income and composition of the family. (https://api.gouv.fr/service/calcul-quotient-familial.html)

[6] Ed. The catalogue is an API itself.

[7] https://api.gouv.fr/

[8] https://joinup.ec.europa.eu/solution/suomifi-finnish-service-catalogue/about

[9] https://api.palvelutietovaranto.suomi.fi/swagger/ui/index.html

## 2.2. Understanding your data before publishing it via API

In addition to understanding the roles it wishes to play, a public administration should understand the data at hand and whether it is *structured*, *documented* and can be *published*.

### *2.2.1. Creating an API requires data to be structured*

When providing data related to a public service by means of an API, a public administration should always first take a look at *how the data is structured*.

If it is not structured, it is recommended that said public administration strives to identify business rules and a data model to which the data should conform and against which the data could be validated (e.g. through the use of a SHACL validator for instance; see section 4 for more details). Generally speaking, the more similarities there exist between the chosen model and models of other public administrations, the better in terms of interoperability.

Developing an API that gathers public service descriptions, requires to define metadata of a public service, independently of how the public service itself is provided.

**Describing a public service through the CPSV-AP data model**

CPSV-AP, the Core Public Service Vocabulary Application Profile, is a data model provided by the ISA[2] Programme that is the result of a joint effort from different public administrations to reduce interoperability barriers.[10]

The CPSV-AP provides public administrations with *a common data model* for describing public services related to business and life events and to facilitate the set-up of catalogues of services oriented to businesses and citizens.

With the CPSV-AP, public administrations can (i) provide information on public services in a user-centric way, grouped logically around business or life events and other ways of classifying; (ii) map different data models to a common model requiring only one single description and (iii) federate and publish information on Points of Single Contact and eGovernment portals in a more efficient and interoperable way.

### *2.2.2. Documenting an API increases its adoption*

Data that cannot be understood by a citizen or an organisation is of no value. For instance, an organisation that plays the role of intermediary (e.g. portals that aggregate data to help citizens find what they need) might have trouble with the data at hand if it is not documented. Therefore, public administrations should

[10] https://ec.europa.eu/isa2/solutions/core-public-service-vocabulary-application-profile-cpsv-ap_en

always ensure that, in addition to having structured data, their data model and the related metadata are well documented (see for instance DCAT-AP[11]).

One of the most common way for public administrations to document a data model is to publish on public portals their data model using different formats (HTML, PDF, PNG, etc.). For instance, standards such as CPSV-AP and DCAT-AP both provide on Joinup and GitHub their respective data models and related documentation.[12]

It is also wise to document the API itself, for instance through the use of a tool such as Swagger Codegen[13] that can automatically generate user documentation from the API contract. For an example, we refer to the API provided by the Finnish Catalogue of Public Services who provides the technical documentation of its API[14] using Swagger (see Figure 3).

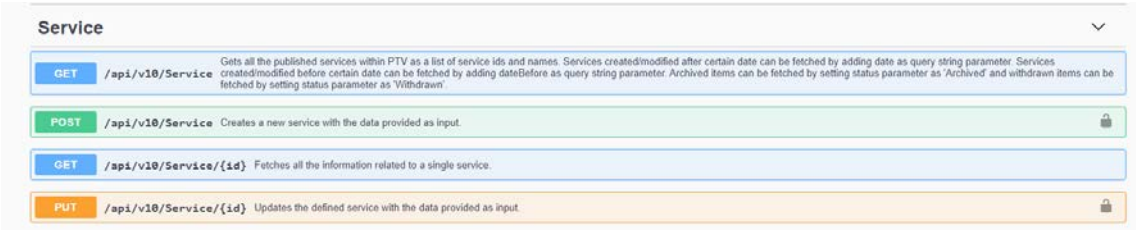


**Figure 3 -** Documentation of the API of the Finnish Catalogue of Public Services

Furthermore, public administrations are encouraged to publish on a central European or national repository their data and their documentation. For an example, we refer to the Service descriptions of the authorities in Bremen published on the European Data Portal[15] (see Figure 4).

---

[11] https://joinup.ec.europa.eu/solution/dcat-application-profile-data-portals-europe

[12] https://joinup.ec.europa.eu/solution/dcat-application-profile-data-portals-europe & https://joinup.ec.europa.eu/solution/core-public-service-vocabulary-application-profile

[13] https://swagger.io/tools/swagger-codegen/

[14] https://api.palvelutietovaranto.suomi.fi/swagger/ui/index.html

[15] https://www.europeandataportal.eu/data/en/dataset/https-www-transparenz-bremen-de-de-datensatz-bremen236-c-4401-de

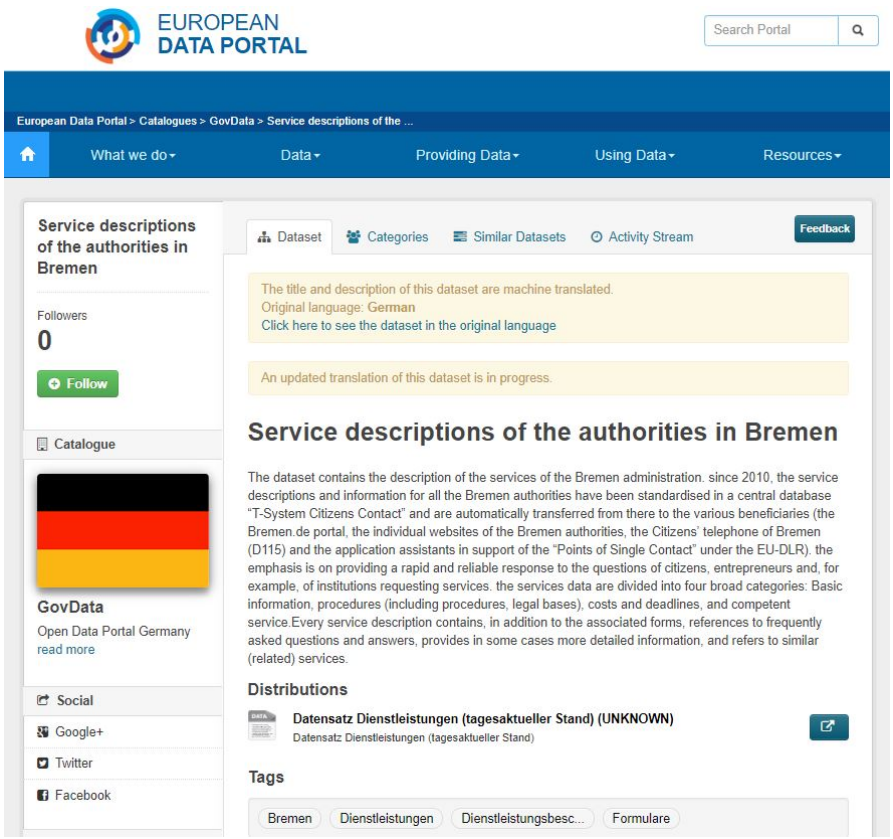


**Figure 4 –** Example of metadata description of public services published on the European Data Portal

### *2.2.3. Licensing and legal terms can affect the usage of an API*

Once data has been provided to a citizen, a public administration might be tempted to think it has done its part and its role is over. Unfortunately, if that public administration does not own the data it just published, it could face legal issues regarding the usage policy of the data. Therefore, as a preemptive measure, public administrations should always inform themselves on the different licenses that describe the rights associated with the consumption of the data. Generally, it is recommended to license the data supplied through an API as open data using, for example, the Creative Commons license family (e.g. CC4.0)[16] rather than creating a new license. In that regard, the European Data Portal is investing a lot of effort to raise awareness regarding the benefits of releasing data via open data licenses.[17]

Besides looking at the license, a study promoted by the European Commission[18] advises public administrations to document their API's terms and conditions, which

---

16 http://www.citysdk.eu/wp-content/uploads/2016/12/20171109_HarmonisedSmartCityAPIs_WEB.pdf & https://creativecommons.org/2019/04/02/european-commission-adopts-cc-by-and-cc0-for-sharing-information/

17 https://www.europeandataportal.eu/elearning/en/module4/#/id/co-01

18 http://www.citysdk.eu/wp-content/uploads/2016/12/20171109_HarmonisedSmartCityAPIs_WEB.pdf

usually cover critical aspects: Who can use the API? How can it be used? What are users allowed to do with the data? Are there limitations regarding the number of data queries? How is the API monitored (see section 2.3)? For instance, API Particulier specifies a limit on the number of data queries per second and per token and explains who can use the API.[19]

Finally, public organisations should also aim at achieving legal interoperability, as recommended by the EIRA[20], to ensure that APIs can effectively use other organisations' data.

### 2.3. Understanding the API lifecycle

Last but not least, public administrations should understand what the API product lifecycle[21] is about and what it implies for their APIs. The need for an API lifecycle stems from the fact that it is, more often than not, a struggle to measure in a generic way the success of an API and manage an API properly in each of the stages of its life.

Put simply, the API product lifecycle is composed of five stages of maturity that are applicable to most, if not all, APIs. As an API matures from creation to value generation to retirement, it will progress through different milestones.

To assess whether an API has moved from one stage to the next, some product milestones first need to be defined. To define a set of milestones which make sense to its specific needs, a public organisation should define clearly what the objectives of its API are and what tools it wants to use to measure its API's progress (e.g. KPIs and OKRs). Once this has been done, the public organisation can assess in which stage its API is and which actions to take to best manage its API.

**Five stages of the API lifecycle**

The first stage of the API product lifecycle is the *creation*. Every API starts with a commencement, a moment in time when someone decides to create an API and a team is put together because, for instance, there exists no API to address an identified need.

In the creation stage of an API, the contract between providers and consumers must be established and such contract must ideally meet the needs of consumers. It is important to note that the way a contract is described might change depending on how the API is to be developed (e.g REST - see OpenAPI[22] or SOAP API - see WSDL[23]). The contract can be based on a data structure defined directly by the data model (see section 2.2.1) and such data structure (as defined for example with an XML schema or XSD) can be reused by different contracts, thereby facilitating the creation of new APIs and automating the data

[19] https://api.gouv.fr/api/api-particulier.html

[20] https://joinup.ec.europa.eu/solution/eira

[21] M., Amundsen, R., Mitra, M., Medjaoui & E., Wilde (2018). Continuous API Management, Making the Right Decisions in an Evolving Landscape, *O'Reilly Media*.

[22] https://www.openapis.org/

[23] https://en.wikipedia.org/wiki/Web_Services_Description_Language

validation process (see section 4).

Once the contract is established, all the roles (provider, consumer and publisher) can take advantage from it:

- the provider can generate the base code to get the input and provide an output from the API;
- the consumer can generate the base code to provide input and get an output from the API;
- the publisher can generate documentation and verify that the API is working correctly.

Note that if such contract is described in a standard way, such as OpenAPI or WSDL, all the roles can take advantage of existing tools (e.g the OpenAPI tools) to facilitate the creation stage.

Additionally, during the creation stage, the provider has to carefully approach details such as performance, versioning, language, errors handling, etc. As APIs can be part of a national/local strategy, the provider must also look at national/local guidelines (for instance, in Belgium[24] and in the Netherlands[25]) which might vary from one region to the next. Doing so is greatly beneficial from the consumer's point of view (e.g if the interface is the same for different APIs, when a new API is created, the client is already familiar with the interface).

Finally, the provider has to establish a service level agreement (SLA), which is a contract between a service provider and the end user that defines the level of service expected from the service provider.[26] SLAs are output-based and usually cover critical elements such as the service description, the reliability, the responsiveness, the performance monitoring, the constraints, etc.

The second stage of the API product lifecycle is the *publication*. In the publish stage, the API is discoverable, typically reachable via a dedicated URL[27], and is ready to be consumed. The consumer will need to understand how the data is structured – input and output (see section 2.2.1). The provider (or publisher) will ideally provide terms and conditions regarding the usage of the API and the data (see section 2.2.3).

In this stage, the publisher can also act as a proxy in between the API consumer and API provider. For instance, the publisher might provide an access point or API Gateway (see section 3.1 for an example) which redirects to the API of one or more providers.

The third stage of the API product lifecycle is the *realization*. In the realize stage, the provider has to get a sense of whether the API is reaching its goal or not (e.g. handling 500 requests per month). To do that, proper mechanisms to monitor the API have to be put in place, along with some feedback mechanism.

---

[24] https://www.gcloud.belgium.be/rest/

[25] https://standaarden.overheid.nl/sru

[26] https://tallyfy.com/service-level-agreement-sla/

[27] https://medium.com/@paulrohan/api-for-beginners-the-greatest-bridge-of-the-web-67eb4f64c5ad

For instance, API Particulier has defined a mechanism that checks whether the API is live or not[28] (one could imagine one of the goals of the API Particulier is to minimize downtime to x%).

The fourth stage of the API product lifecycle is the *maintenance*. In the maintain stage, the API provider makes changes to keep the API in its steady state for as long as possible. In case the API provider decides to change the contract of the API, a potential risk arises regarding the interoperability with previous versions. Accordingly, it should notify both the publisher and the consumer as a new version will impact the consumption of the data and the related documentation.

The fifth and final stage is called the *retirement*. In the retire stage, the API has come to the end of its life (e.g. due to a loss of demand) and the service providing the data will be completely replaced or shut down. If this is the case, it is strongly advised to notify publishers and consumers as soon as possible.

## Brief comparison - REST VS SOAP[29]

APIs can make use of different architectures to transfer data from the server to the client. *REST*[30] and *SOAP*[31] are today two of the preferred choices but there are some important differences that the reader needs to carefully understand before making a choice. Historically, SOAP was the most popular option. However, as it comes with strict rules, advanced security features and higher complexity (which often leads to slower page load times), it lost some popularity to its "rival" REST which has a more flexible architecture. Both of those APIs styles are today widely used and no one can claim one is better than the other, they are simply different and look at the question of data transmission from a different angle.

REST (**Re**presentational **S**tate **T**ransfer) is an architectural style that defines a set of recommendations for designing *loosely coupled* applications that use the HTTP protocol when transmitting data. Web services that are built following the REST guidelines are RESTful web services. There exist five (plus one optional) architectural constraints: (i) Uniform interface, (ii) Client-server separation, (iii) Statelessness, (iv) Cacheable resources, (v) Layered system, (vi) Code on demand. REST allows different messaging formats, such as HTML, JSON, XML, and plain text.

SOAP, which stands for **S**imple **O**bject **A**ccess **P**rotocol, is a messaging protocol for interchanging data in a decentralized and distributed environment. It works with all application layer protocols (HTTP, SMTP, TCP, UDP) and returns data in XML format. Features such as security, authorization, handling errors are built directly into the protocol.

The main difference between SOAP and REST is that REST is not a protocol but an

[28] https://api.gouv.fr/api/api-particulier.html#operations-tag-Ping
[29] https://raygun.com/blog/soap-vs-rest-vs-json/
[30] https://en.wikipedia.org/wiki/Representational_state_transfer
[31] https://en.wikipedia.org/wiki/SOAP

architectural style. SOAP is mostly used for enterprise-level web services where there is a need for high security and complex transactions (e.g. financial services, payment gateways, etc.). REST, with its more lightweight architecture, is more popular when it comes to mobile, as load time makes all the difference.

## 3. ANALYSIS ON THE USAGE OF API FOR PUBLIC SERVICES

This section highlights some of the business cases concerning the usage of APIs in the public sector. The main reference are the JRC work (study and workshop), three interviews conducted with Member States, and the business case of the Single Digital Gateway.

### 3.1. Takeaways from the JRC API study

In January 2018, the Joint Research Center of the European Commission launched a study on the usage of APIs by public administrations.[32]

The study, which lasts 2 years, intends to contribute to:

- The EU eGovernment Action Plan 2016-2020, in particular its open government approach to modernise public administrations[33];
- The implementation of the European Interoperability Framework[34] (e.g. actions of the focus area 4: "Develop, maintain and promote key interoperability enablers");
- The implementation of the Once Only Principle[35];
- The Building Block approach adopted by the Connecting European Facilities (CEF) programme[36].

The study provides findings regarding the usage of APIs by looking at different aspects (e.g. functionalities, governance, usage, technical architecture, etc.). Among others, there are 2 significant takeaways around the topic of interoperability:

(i) Standards enhance cross-border and cross-sector interoperability but need to be lightweight in order to be supported.

With regard to that, CPSV-AP aims to be a lightweight standard that can be reused easily by API providers without forcing them to adapt their data models when describing Public Services:

- The CPSV-AP specifies only 2 mandatory classes (Public Service and Public Organisation) out of 19 classes;
- The data model itself is based on other standards such as the Core Public Organisation[37], the Core Criterion And Evidence[38] and the ELI Vocabulary[39].

---

[32] https://joinup.ec.europa.eu/document/digital-government-benchmark-api-study

[33] https://ec.europa.eu/digital-single-market/en/news/communication-eu-egovernment-action-plan-2016-2020-accelerating-digital-transformation

[34] https://ec.europa.eu/isa2/eif_en

[35] https://ec.europa.eu/digital-single-market/en/news/eu-wide-digital-once-only-principle-citizens-and-businesses-policy-options-and-their-impacts

[36] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL2018/About+CEF+building+blocks

[37] https://joinup.ec.europa.eu/solution/core-public-organisation-vocabulary/about

[38] https://joinup.ec.europa.eu/solution/core-criterion-and-core-evidence-vocabulary/about

[39] https://publications.europa.eu/en/web/eu-vocabularies/model/-/resource/dataset/eli

(ii) API Gateways are needed as they are beneficial to expose APIs and mitigate interoperability issues.[40]

API Gateways can hide implementation details (such as URL changes), call automatically other public services supporting more complex services, and, at the same time, provide additional services such as load balancing and throttling. A practical example is the API Gateway[41] developed and maintained by Estonia and Finland within the context of the X-Road project. There exist off-the-shelf API gateways that can support publishers and providers in creating APIs.[42]

### 3.2. Takeaways from the JRC Workshop

In October 2018, JRC published the aforementioned study and invited Member States to a workshop to discuss current practices regarding the implementation of APIs.[43] There are three main takeaways concerning interoperability, discoverability and licensing:

First, a common data model is needed to facilitate the creation of APIs. Such data model makes it possible to create interoperability points when building frameworks to exchange data.[44] API providers can then build APIs based on common reusable components.[45] That being said, the creation of a common data model requires setting up data governance mechanisms.[46]

**Did you know?**

In the context of Public Services, the CSPV-AP data model is based on different reusable components[47] and a data governance process which is based on the decisions taken by a Working group[48] (+50 members and growing) has been established.

Second, when building APIs, API providers should aim to build a network of services rather than individual silos[49], thus the need for API discoverability to find out how services are related to each other.[50]

---

40 https://whatis.techtarget.com/definition/API-gateway-application-programming-interface-gateway

41 https://github.com/nordic-institute/REST-adapter-service

42 https://techbeacon.com/app-dev-testing/you-need-api-management-help-11-open-source-tools-consider

43 https://ec.europa.eu/jrc/en/event/workshop/assessing-government-api-strategies-across-eu

44 https://ec.europa.eu/jrc/sites/jrcsh/files/18_11_laura-rodriguez-de-lope_santander.pdf

45 https://ec.europa.eu/jrc/sites/jrcsh/files/18_08_dimitri-van-hees_netherlands.pdf

46 https://ec.europa.eu/jrc/sites/jrcsh/files/17_07_sven-rostgaard_denmark.pdf

47 https://joinup.ec.europa.eu/release/core-public-service-vocabulary-application-profile/22

48 https://joinup.ec.europa.eu/solution/core-public-service-vocabulary-application-profile/members

49 https://ec.europa.eu/jrc/sites/jrcsh/files/17_04_david-berlind_api101.pdf

**Did you know?**

In the context of Public Services, the CPSV-AP data model brings in the concepts of input (class Evidence) and output (actual result of executing a given Public Service) that might link to other Public Service's input and output. Furthermore the CPSV-AP data model provides properties (such as “required” and “related”) which allow to explicitly indicate other required and/or related public services.

Third, as already highlighted in section 2.2.3, API providers have to carefully consider the license rights governing the consumption of the data. In the case of Ireland, the legislation enforces the use of base registries which are the authoritative source of data.[51]

**Did you know?**

In the context of Public Services, the CPSV-AP data model takes into account, via the Legal Resource class, the legislation to which a Public Service relates, operates or has its legal basis.

## 3.3. Business Cases

In January 2019, three selected public administrations were interviewed to understand how APIs are used in the context of public services. The three public administrations are:

- The Autonomous Province of Trento acting both as an API consumer and API provider;
- The Flemish Information Agency, acting as an API provider;
- The Brønnøysund Register Centre, which provides a catalogue of public services, acting as an API publisher.

### *3.3.1. The Autonomous Province of Trento (Italy) – Establishing a common datastore of concepts and providing a mapping service*

The Autonomous Province of Trento is a good example of a public organization that has established a common datastore of concepts and provides a mapping service. The Province has indeed developed a distributed Content Management System (CMS), called ComunWeb, which is used by several local authorities. These CMSs[52] share a common data model to define structures such as a Public Service. The data, exchanged among local authorities, is shared by means of a REST API and published in JSON format.[53]

---

[50] https://ec.europa.eu/jrc/sites/jrcsh/files/18_07_mark-boyd_city-government-api-trends.pdf

[51] https://ec.europa.eu/jrc/sites/jrcsh/files/17_06_mark-warren-ireland.pdf

[52] http://ontopa.opencontent.it/openpa/classes

[53] If the readers search for the example (q = 'Abilitazioni relative ai prodotti fitosanitari') on the site http://ontopa.opencontent.it/opendata/console, the interface returns the link to the public service and the related JSON as a result.

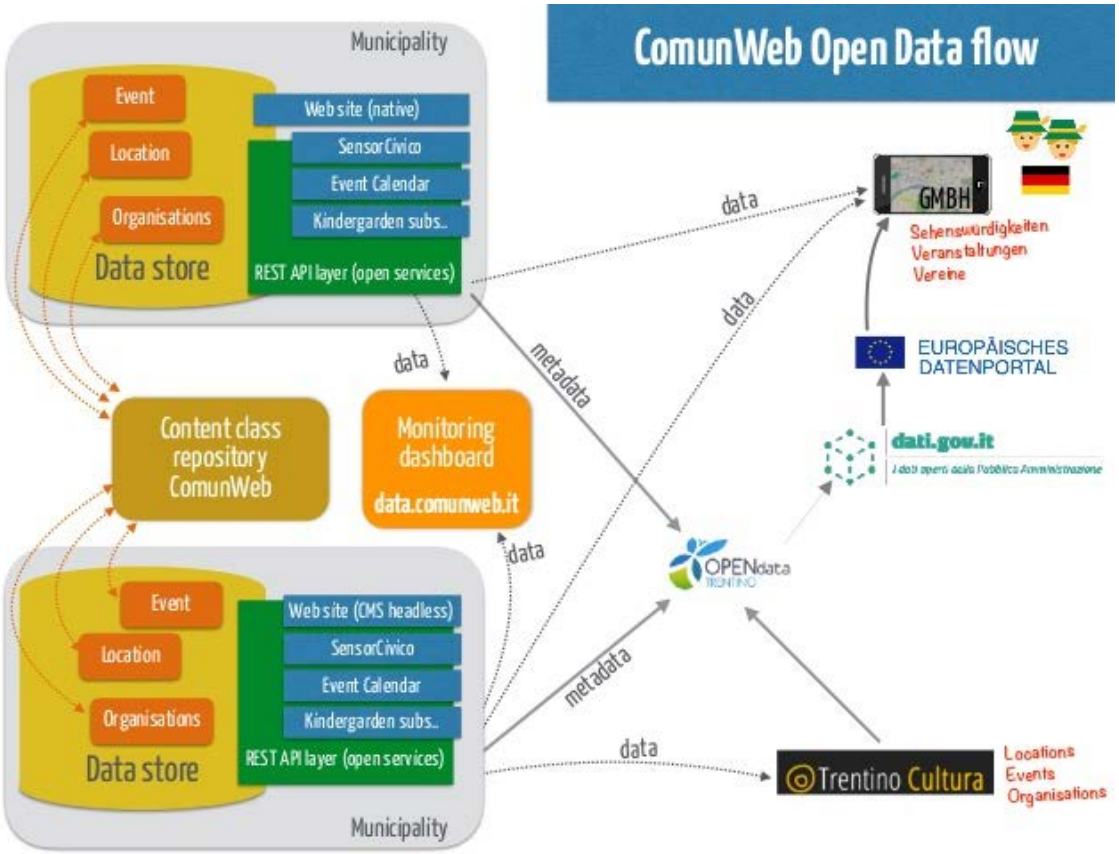


**Figure 5 –** ComunWeb Architecture (Trento)

In order to map the shared data structure to other data models, such as the CPSV-AP, a mapping tool has been developed so that data is transformed from JSON to JSON-LD format which is a way to express linked data (see for instance the Rovereto[54] municipality for the publication of events).

[54] http://www.comune.rovereto.tn.it/eventi/agenda/event/116108

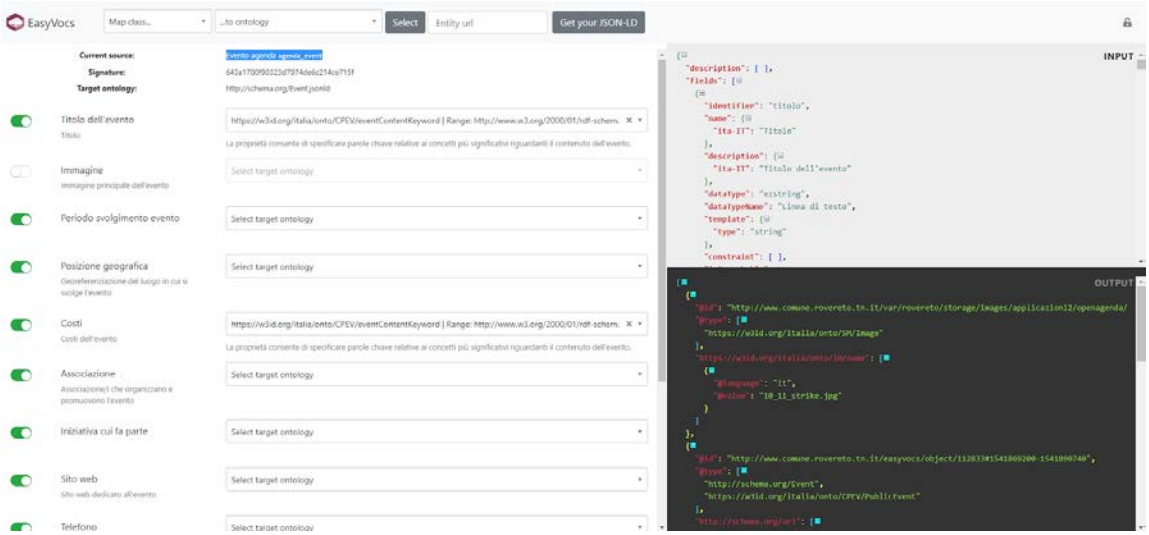


**Figure 6 –** Mapping tool provided by Trento to transform from JSON to JSON-LD

### *3.3.2. The Flemish Information Agency (Belgium) - Providing a JSON-LD context shared between APIs*

The Flemish Information Agency is a good example of an organization that invests in a common data model in order to uniform their public services, and provides a JSON-LD context that can be shared between APIs.

In the past few years, The Flemish Information Agency has invested a lot of effort in a data model, called OSLO-Dienstencataloog[55] aiming at easing the exchange of data and increasing the interoperability of Flemish government services. Such data model extends concepts from the European Commission's Core Vocabularies such as CPSV-AP.

In order for base registries to adopt such data model, the Flemish Information Agency has made the data model machine readable in a linked data format by means of custom made open source tools.[56] Base registries can then implement APIs which return structured data according to the OSLO[2] data model.[57]

In order to support base registries, the Flemish Information Agency has published a JSON-LD context[58] for each application profile in the data model so that REST API, currently returning JSON data, can be easily adapted by just adding the JSON-LD context in the response without interfering with the JSON data provided at the origin.[59] The JSON-LD context, which, in practice is a collection of classes and

---

[55] https://data.vlaanderen.be/doc/applicatieprofiel/dienstencataloog/

[56] https://github.com/Informatievlaanderen/OSLO-EA-to-RDF

[57] https://joinup.ec.europa.eu/sites/default/files/inline-files/SEMIC_Linked%20Base%20Registries%20as%20a%20key%20enabler%20in%20Flanders%20-%200_7.pdf

[58] https://w3c.github.io/json-ld-syntax/#the-context

[59] https://joinup.ec.europa.eu/sites/default/files/inline-files/SEMIC%202018%20_Buyle_1.pdf

properties for each application profile (see e.g based on CPSV-AP[60]), acts then as a shared vocabulary of terms that can be used by different APIs.[61]

As can be seen in Figure 7, a citizen has access to two different APIs (Grant Registry and Tax Calculation) maintained by two different organisations (A and B). The two APIs provide data in different data structures. However, by means of a shared context (which can be stored by another organisation), the meaning of the API response stays the same, so the web application used by citizens displays consistent information.

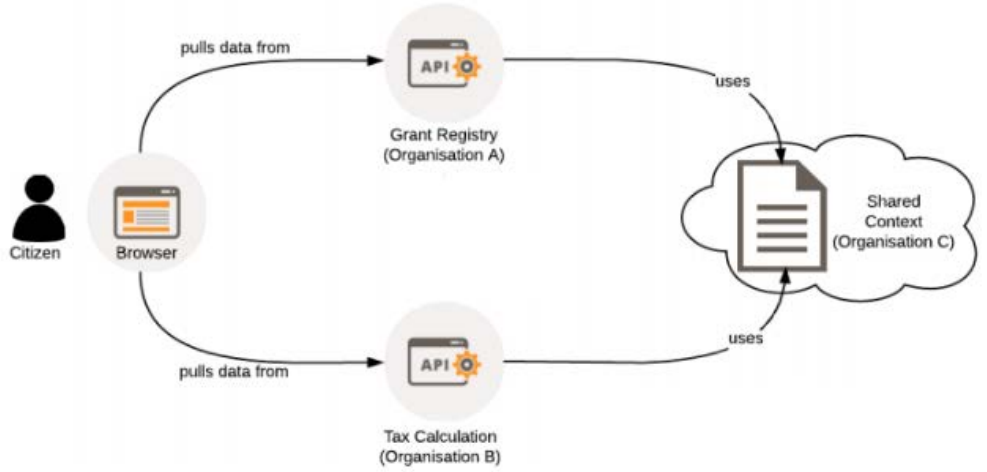


**Figure 7 –** Example of reusing a JSON-LD Context

Furthermore, the Flemish Information Agency has provided means to validate data against the OSLO² data model with the so-called OSLO² Validator[62], a web application based on the SHACL standard (Figure 8).

[60] https://data.vlaanderen.be/context/dienstencataloog.jsonld

[61] https://joinup.ec.europa.eu/sites/default/files/event/attachment/2018-01/SC508%20D03.02%20Webinar%202018-01-26%20v1.0.pdf

[62] https://data.vlaanderen.be/shacl-validator/

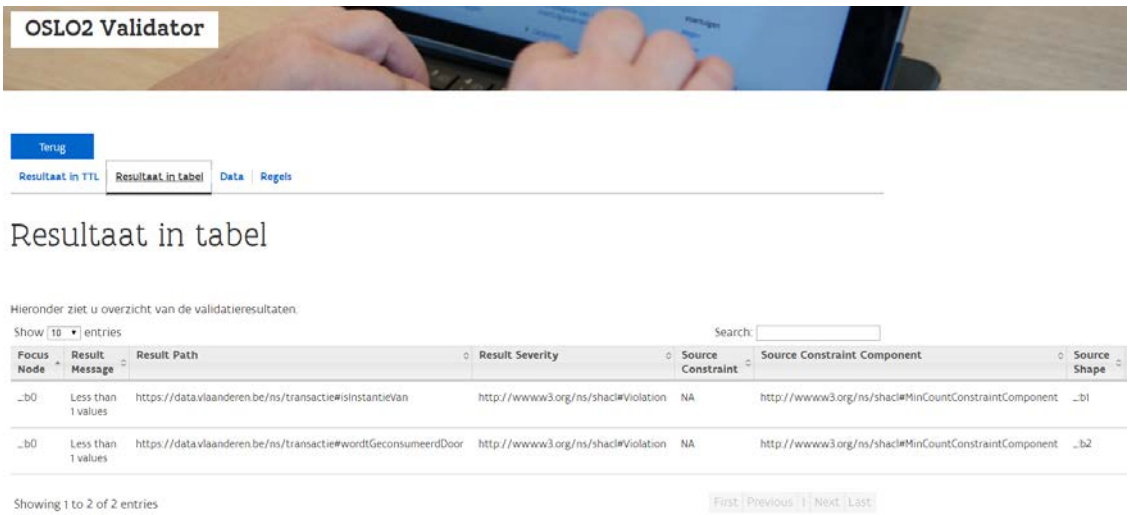


**Figure 8 –** OSLO[2] SHACL Validator

More recently, the Flemish Information Agency, has invested in hypermedia APIs[63], implemented via the Hydra Core Vocabulary[64] based on JSON-LD in order to standardize the operations executed by the REST API and not just the data returned.

### *3.3.3. The Brønnøysund Register Centre (Norway) - Standardizing API descriptions*

Finally, the Brønnøysund Register Centre is a good example of a public organization that publishes APIs described in an homogeneous way.

In Norway, public services' descriptions are still scattered on several portals maintained by different agencies despite the aim of the government to achieve the "Once Only Principle". The Brønnøysund Register Centre is responsible of creating one unique catalogue of public service descriptions bringing together 4 types of catalogues:

- Catalogue of concepts
- Catalogue of data models
- Catalogue of data sets
- Catalogue of APIs[65]

[63] https://sookocheff.com/post/api/on-choosing-a-hypermedia-format/
[64] http://www.hydra-cg.com/spec/latest/core/
[65] https://fellesdatakatalog.brreg.no/apis

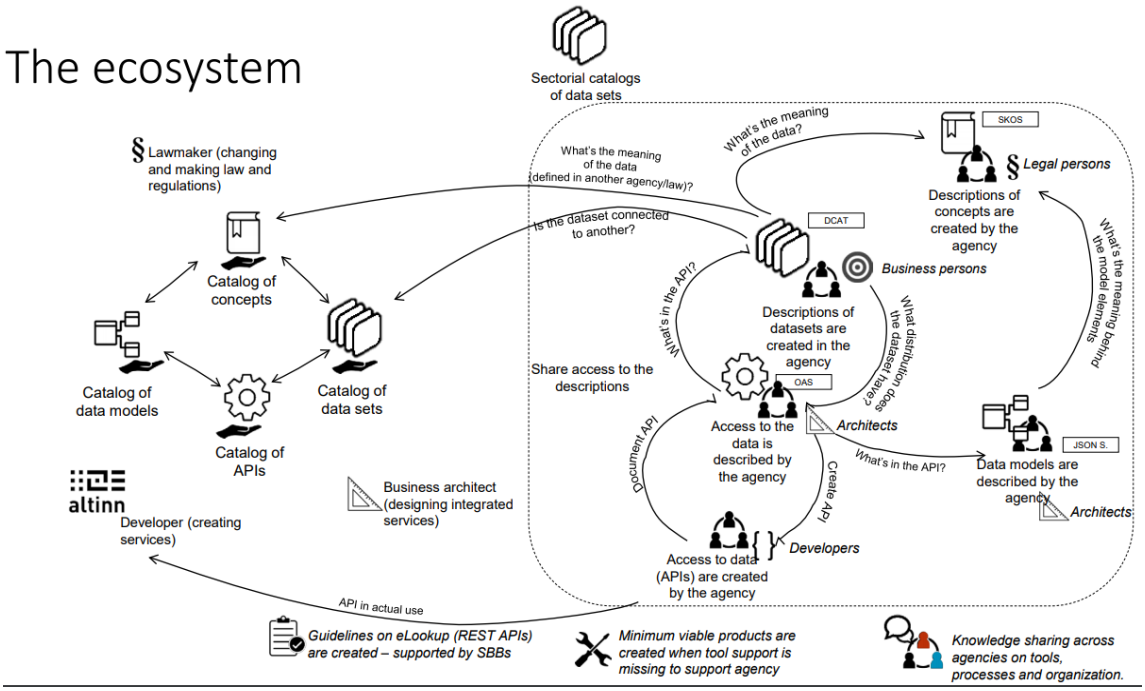


**Figure 9 –** Ecosystem of the catalogues published by the Brønnøysund Register Centre

As shown in the picture above (Figure 9), each API described in the API catalogue is linked to its respective data models and data sets. Therefore, if APIs are described

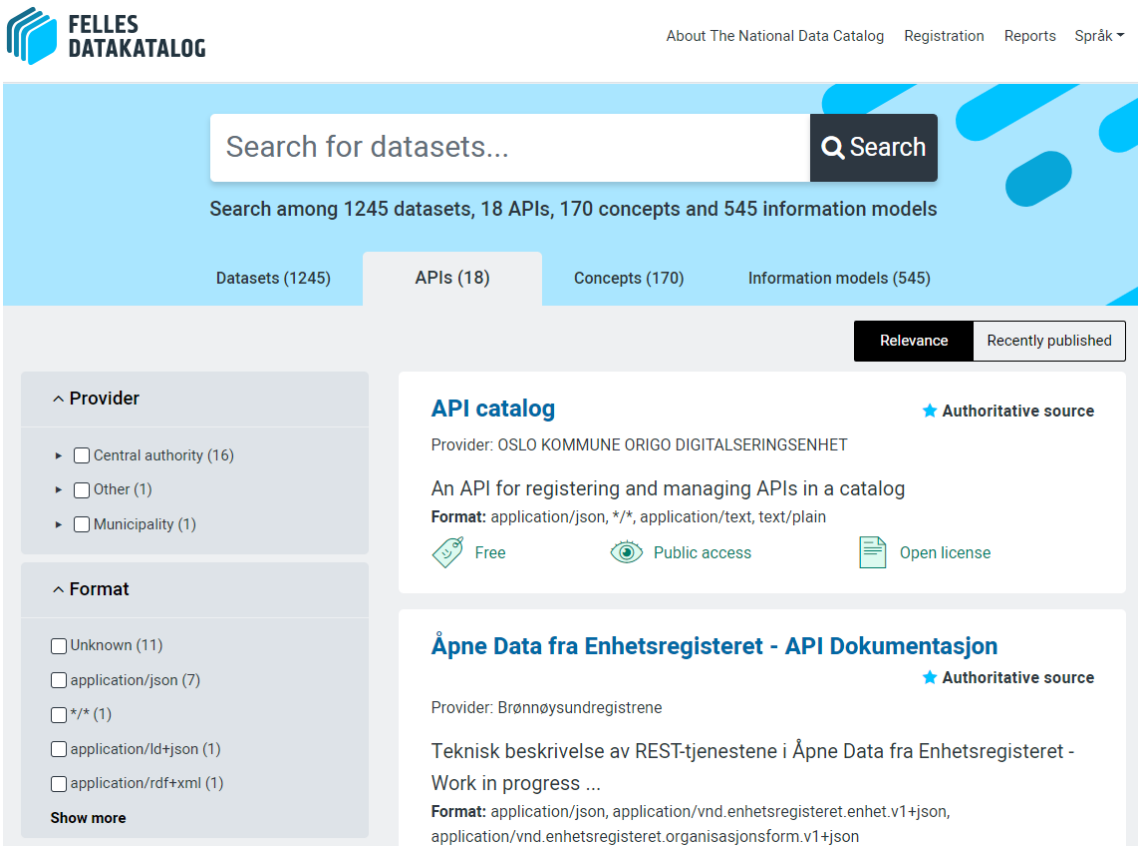


**Figure 10 –** Norway API Catalog

in a consistent manner (the OpenAPI specification is recommended), it is possible to harvest those APIs automatically and create links between the different types of catalogues. Furthermore, thanks to the taxonomy of public services, data sets and APIs can be searched through the catalogues.

As an API publisher, the Brønnøysund Register Centre needs to standardize APIs' descriptions as well as the related data models. Thus, by adopting CPSV-AP (currently planned), public service descriptions can be homogenised.

#### *3.3.4. SDG Business case*

The Single Digital Gateway (SDG) regulation[66] implies some fundamental changes in the way that public services are delivered and information about those services are exchanged and made available publicly. The European coordinator of the Single Digital Gateway has to collect the descriptions of public services from European public administrations in one unique portal. This requirement has different consequences:

- The European coordinator must ensure that a minimal set of information is made available to the end-users;
- With 28 Member States and more than 21 services to be delivered according to the regulation, collecting the information manually represents a considerable effort;
- The maintenance of the information at the European level should be automated to avoid human mistakes and having to update the information manually every time there is a change from a European public administration.

Automating the collection of the public service descriptions with a certain frequency can be done by implementing a network of APIs between the different European catalogues of services as shown in Figure 11.

[66] http://data.consilium.europa.eu/doc/document/PE-41-2018-INIT/en/pdf

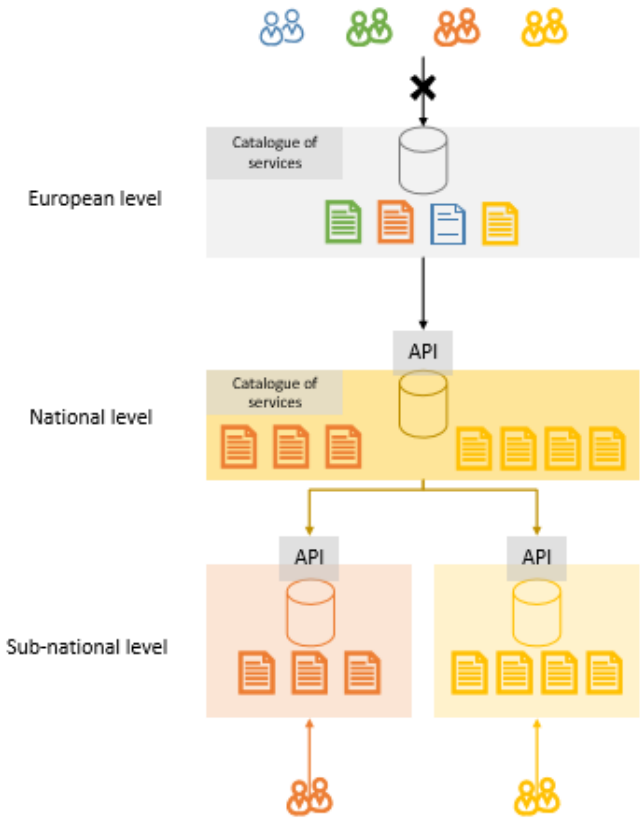


**Figure 11 -** Automation of the collection of data between catalogues of services

The data descriptions or metadata of the public services are stored in the catalogues of services databases at subnational level. For each database, there is an API exposing the descriptions to the related catalogue of services or to any other catalogue of services, for example at national level. A similar logic applies between the national and European levels.

However, without a common data structure, or in other words, without a common way of describing public services, the European coordinator will need to understand the data model applied at the national level and map it to its data model used for displaying the descriptions in the SDG portal. This hinders the findability and understandability of the services described. Figure 11 highlights with different colors, the fact that at European level, there would be need of different APIs, one for each catalogue published at national level.

One solution (see Figure 12) consists of combining the network of APIs with a common data model which would enable each public administration to directly describe his services in a harmonised manner, improve the quality of the information retrieved by the APIs and simplify the process for retrieving it. As CPSV-AP is an existing standard data model that is already used in some Member States, implementing CPSV-AP within the API at a national level would be beneficial at a European level to have one unique API.

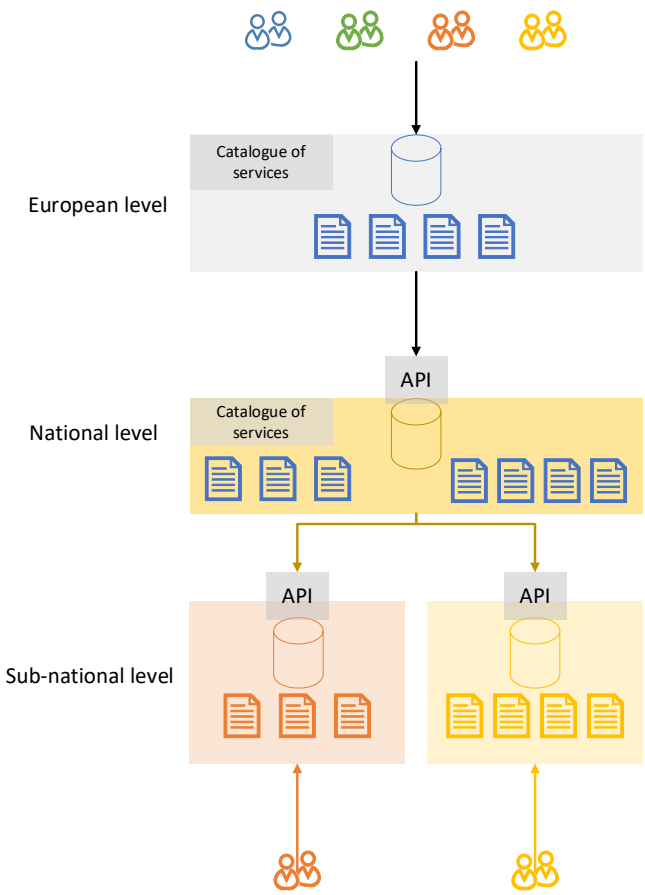


**Figure 12 –** A common data model allows to gather description of catalogue of public services with one API only

## 4. ENSURING DATA QUALITY WHEN EXCHANGING LINKED DATA

There are two mains safeguards that can ensure data quality when exchanging linked data.

First, when designing an API, features such as URL definition, content negotiation, performance, authorization and error management ought to be, in some cases, defined. During the creation stage of the API, the API provider should put in place mechanisms to ensure that the API is working in conformance with the designed features, which is generally achieved by setting up a *test suite* (see the "API Landscape" for a list of tools).

Second, once the API functions properly, the API provider should ensure that the data provided by the API itself is valid. A validation mechanism is therefore desired any time some data is transferred. In case the data is structured and a schema has been created out of the data model, it is possible to validate the data against the schema, for example:

- data transmitted as XML could be directly validated against its XML schema or an XML validator such as Schematron;
- data transmitted as linked data (RDF) could be validated against its SHACL constraints.

Such validation process could be automated through the use of an API. In that regard, the ISA² interoperability test bed (Action 2016.25)[67] provides a central, standards-based reusable service to facilitate interoperability and conformance testing by means of APIs.

**Linked data & SHACL**

In the linked data world, the most dominant standard is RDF (Resource Description Framework), a standard that allows to reduce interoperability barriers by defining concepts such as entity's relations at the semantic level first.

In July 2017, the W3C published the SHACL specification to have, once and for all, a mechanism to validate RDF data. The SHACL constraints are expressed in RDF and can be derived directly from a data model (see for example section 3.3.2).

The CPSV-AP has a well defined data model that is published as RDF and whose constraints are published as SHACL constraints.[68] For instance, the CPSV-AP SHACL constraints specify that a Public Service should be provided by *one and only one* Competent Authority.

In addition, the CPSV-AP SHACL constraints can be reused by public

[67] https://joinup.ec.europa.eu/solution/interoperability-test-bed/about

[68] https://github.com/catalogue-of-services-isa/CPSV-AP/tree/master/releases/2.2.1

administrations and third-party tool providers when validating public services' descriptions. For instance the ISA² interoperability test bed[69] provides a SHACL Validator as an API. The SHACL Validator is provided with documentation of the API (as recommended in section 2.2.2) and terms and conditions (e.g. not available during the weekend and overnight), which can be adjusted on demand.

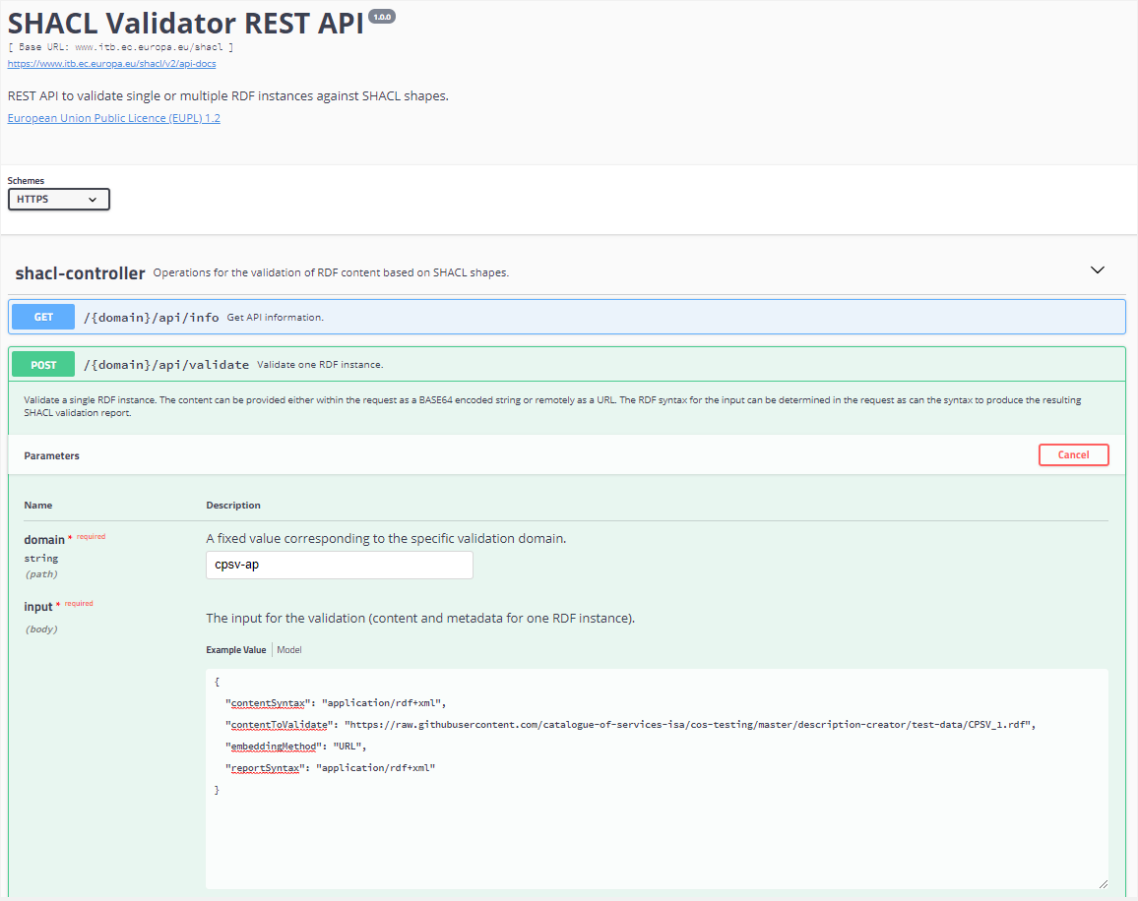


**Figure 13 –** Example of validation using the API SHACL Validator for CPSV-AP

[69] https://www.itb.ec.europa.eu/shacl/swagger-ui.html#/shacl-controller

## 5. CONCLUSIONS

In the public sector, APIs have proven to be instrumental in enabling public administrations and citizens to benefit from interoperable cross-sector and cross-border public services. Given the undeniable importance of APIs in the lives of most public organisations, the present document looked in detail at how public services can best be published through APIs.

As building an API is no small feat and represents a journey for which public administrations are often underprepared, the first part of the report looked at three key aspects that public administrations should consider:

(i) public organisations should make sure they have built a proper understanding of the role they want to play with their API (to *provide*, *consume* or *publish* data);

(ii) public organisations should have a sense of what makes up their data: is it *structured*, is it *documented* and can it be *published*?

(iii) public organisations should understand how APIs mature over time.

The report also analysed the usage of APIs in the public sector. More specifically, the JRC API study and workshop were looked at; three interviews with Member States were conducted; and the SDG business case was covered extensively.

Finally, two safeguards to ensure data quality when exchanging *linked data* were discussed in section 4.

The first takeaway from this study is that public administrations who wish to *provide* data should consider investing in defining common data models as it allows to speed up the API lifecycle as reusable data structures. API providers could make use of the CPSV-AP data model (whose goal is to describe Public Services) to define data structures on which their APIs can be based. As was seen in the business case of the Flemish Information Agency, in order to support base registries, the Flemish Information Agency has published JSON-LD contexts so that they can be reused by REST APIs. Likewise, by making a public JSON-LD context, CPSV-AP could be reused by REST APIs publishing public services in linked data format. Furthermore, API providers are encouraged to take into account existing policies that determine how data can be published and the terms and conditions that govern how said data can be provided.

The second takeaway is that public administrations who wish to *publish* data should make use of common data models (that can be expressed in different forms, e.g. XML schema or OpenAPI description) as it would not only help them to automatically generate their documentation, but also help them act as a catalogue of APIs or an API Gateway. By doing so, discoverability is enhanced and interoperability barriers are reduced.

The third and final takeaway is that public administrations who wish to *consume* data should verify that the data they have received is of high quality through an effective validation process.

## 6. REFERENCES

| Name | Link |
|---|---|
| The API landscape tools | http://theapilandscape.pagedemo.co/ |
| An API published by the Government of France | https://api.gouv.fr/api/api-particulier.html |
| A Service provided on top of an API in France | https://api.gouv.fr/service/calcul-quotient-familial.html |
| The API Catalogue of France | https://api.gouv.fr/ |
| The API Catalogue of Norway | https://fellesdatakatalog.brreg.no/apis |
| The CPSV-AP data model | https://ec.europa.eu/isa2/solutions/core-public-service-vocabulary-application-profile-cpsv-ap_en |
| The DCAT-AP data model | https://joinup.ec.europa.eu/solution/dcat-application-profile-data-portals-europe |
| An example of data published on the European Data Portal | https://www.europeandataportal.eu/data/en/dataset/https-www-transparenz-bremen-de-de-datensatz-bremen236-c-4401-de |
| Learning about licensing on the European Data Portal | https://www.europeandataportal.eu/elearning/en/module4/#/id/co-01 |
| The Harmonized Smart City APIs study | http://www.citysdk.eu/wp-content/uploads/2016/12/20171109_HarmonisedSmartCityAPIs_WEB.pdf |
| OpenAPI | https://www.openapis.org/ |
| OpenAPI tools | https://github.com/OAI/OpenAPI-Specification/blob/master/IMPLEMENTATIONS.md |
| The REST guidelines published by the Belgian Government | https://www.gcloud.belgium.be/rest |
| The SRU standard adopted by the Netherlands | https://standaarden.overheid.nl/sru |
| The X-road API Gateway | https://github.com/nordic-institute/REST-adapter-service |
| The JRC API Study | https://joinup.ec.europa.eu/document/digital-government-benchmark-api-study) |
| The EU eGovernment Action Plan 2016- | https://ec.europa.eu/digital-single- |

| | |
|---|---|
| 2020 | market/en/news/communication-eu-egovernment-action-plan-2016-2020-accelerating-digital-transformation |
| The European Interoperability Framework | https://ec.europa.eu/isa2/eif_en |
| The Once Only Principle | https://ec.europa.eu/digital-single-market/en/news/eu-wide-digital-once-only-principle-citizens-and-businesses-policy-options-and-their-impacts |
| The CEF Programme | https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL2018/About+CEF+building+blocks |
| The Core Public Organisation Vocabulary | https://joinup.ec.europa.eu/solution/core-public-organisation-vocabulary/about |
| The Core Criterion and Evidence Vocabulary | https://joinup.ec.europa.eu/solution/core-criterion-and-core-evidence-vocabulary/about |
| The ELI data model | https://publications.europa.eu/en/web/eu-vocabularies/model/-/resource/dataset/eli |
| API Management tools | https://techbeacon.com/app-dev-testing/you-need-api-management-help-11-open-source-tools-consider |
| JRC API workshop | https://ec.europa.eu/jrc/en/event/workshop/assessing-government-api-strategies-across-eu |
| Santander presentation (JRC Workshop) | https://ec.europa.eu/jrc/sites/jrcsh/files/18_11_laura-rodriguez-de-lope_santander.pdf |
| Netherlands presentation (JRC Workshop) | https://ec.europa.eu/jrc/sites/jrcsh/files/18_08_dimitri-van-hees_netherlands.pdf |
| Denmark presentation (JRC Workshop) | https://ec.europa.eu/jrc/sites/jrcsh/files/17_07_sven-rostgaard_denmark.pdf) |
| ProgrammableWeb presentation (JRC Workshop) | https://ec.europa.eu/jrc/sites/jrcsh/files/17_04_david-berlind_api101.pdf |
| The Working Group of CPSV-AP | https://joinup.ec.europa.eu/solution/core-public-service-vocabulary-application-profile/members |
| CPSV-AP 2.2 | https://joinup.ec.europa.eu/release/core-public-service-vocabulary-application-profile/22 |

| | |
|---|---|
| Platformable presentation (JRC Workshop) | https://ec.europa.eu/jrc/sites/jrcsh/files/18_07_mark-boyd_city-government-api-trends.pdf |
| Ireland presentation (JRC Workshop) | https://ec.europa.eu/jrc/sites/jrcsh/files/17_06_mark-warren-ireland.pdf |
| The GitHub repository of the Flemish Government to generate SHACL shapes | https://github.com/Informatievlaanderen/OSLO-Generated |
| The CPSV-AP SHACL shapes | https://github.com/catalogue-of-services-isa/CPSV-AP/tree/master/releases/2.2.1 |

## 7. LIST OF ACRONYMS

| Name | Meaning |
|---|---|
| API | Application Programming Interface |
| API Gateway | Application Programming Interface Gateway |
| CEF | Connecting European Facilities |
| CMS | Content Management System |
| CPSV-AP | Core Public Service Vocabulary Application Profile |
| DCAT-AP | Data Catalogue vocabulary Application Profile |
| DINSIC | Direction Interministérielle du Numérique et du Système d'Information et de Communication de l'État |
| EIRA | European Interoperablity Reference Architecture |
| ELI | European Legislation Identifier |
| HTML | Hyper Text Markup Language |
| ISA$^2$ | Interoperability solutions for public administrations, businesses and citizens |
| JRC | Joint Research Center |
| JSON | JavaScript Object Notation |
| JSON-LD | JavaScript Object Notation for Linked Data |
| KPI | Key Performance Indicator |
| OKR | Objectives and key results |
| OSLO | Open Standaarden voor Linkende Organisaties |
| PDF | Portable Document Format |
| PNG | Portable Network Graphics |
| RDF | Resource Description Framework |
| REST | Representational State Transfer |
| SDG | Single Digital Gateway |
| SHACL | Shapes Constraint Language |

| | |
|---|---|
| SOAP | Simple Object Access Protocol |
| URL | Uniform Resource Locator |
| XML | eXtensible Markup Language |
| XSD | XML Schema Definition |
| WSDL | Web Services Description Language |
| W3C | World Wide Web Consortium |

## An action supported by ISA²

ISA² is a EUR 131 million programme of the European Commission which develops digital solutions that enable interoperable cross-border and cross-sector public services, for the benefit of public administrations, businesses and citizens across the EU. ISA² supports a wide range of activities and solutions, among which is the Semantic Interoperability Community (SEMIC) action.

ISA² solutions can be used free of charge and are open source when related to IT.

## More on the programme

ec.europa.eu/isa2

## Contact ISA²

isa2@ec.europa.eu

## Follow us

@EU_ISA2
@Joinup_eu

isa2 programme