H2020-SC6-GOVERNANCE-2018-2019-2020

DT-GOVERNANCE-05-2018-2019-2020

ACROSS

Towards user journeys
for the delivery of cross-border services
ensuring data sovereignty

D4.5 Micro Proxies and services catalogue - Intermediate

Project Reference No	959157 — ACROSS — H2020-SC6-GOVERNANCE-2018-2019-2020
Deliverable	D4.5 Micro Proxies and services catalogue – Intermediate
Work package	WP4 - ACROSS Modules Set-Up
Nature	OTHER
Dissemination Level	PU = Public
Date	31/01/2023
Status	Final 1.0
Editor(s)	Vincenzo Savarino (ENG)
Contributor(s)	Petros Christopoulos (Grnet), Enrique Areizaga (TEC), Valentín Sánchez (TEC), David Britnell (DATAPORT)
Reviewer(s)	David Britnell (DATAPORT), Thilo Ernst (FHG)
Document description	This report provides the first update of design and implementation of components of *Harmonization and connectors layer* of ACROSS architecture. It documents the requirements, functional specification, design, description of modules, and description of components APIs in line with the updated requirements. It also

	describes the respective initial PoCs delivered.

About

The project is co-funded by the European Commission's Horizon 2020 research and innovation framework programme. Spanning through three years, ACROSS consists of a consortium of 10 partners from 7 countries: Athens Technology Center (coordinator), Tecnalia, Dataport, Engineering, Fraunhofer, GRNET, TimeLex, The Lisbon Council, Waag and VARAM. The project kicked off its activities in February 2021, with an energising online meeting, where all partners took the floor to present their plans to make the project a great success.

Document Revision History

Version	Date	Modifications Introduced	
		Modification Reason	Modified by
V0.1	04/11/2022	Modification of ToC	ENG
V0.2	22/11/2022	Contribution on Service Catalogue model	TECNALIA, GRNET, DATAPORT, FHG
V0.3	09/12/2022	Added ACROSS Service model Annex I	ENG
V0.4	04/01/2023	Updated section 2.4	ENG

V0.5	10/01/2023	Updated Annexes II and III	ENG
V0.6	20/01/2022	Pre-release ready for internal review	ENG
V0.7	26/01/2022	Feedback and revision	DATAPORT, FHR
V1.0	30/01/2022	Final release	ENG

Executive Summary

The main objective of the ACROSS project is to provide the means (tools, methods and techniques) to enable user-centric design and implementation of interoperable cross-border (digital) public services compliant with the current European regulations (e.g. the Single Digital Gateway (SDG) and Once-Only principle (OOP), European Interoperability Framework (EIF)) where the private sector can also interconnect their services **while ensuring the data sovereignty of the citizens, who can set the privacy level that will allow the public and private sector to access to their data based on their requirements**.

This report documents the result of activities performed in Task 4.2 " Public & Private sector offerings management tool" describing the current version of Service Catalogue components and related service proxy adapters. This report is seen as a living document and an update of *D4.4 Micro Proxies and services catalogue-Initial* describing the evolution of design and implementation of Service Catalogue and Adapters solution by addressing the refinement of technical and user requirements of ACROSS platform.

Table of Contents

LIST OF FIGURES

List of Tables

List of Terms and Abbreviations

Abbreviation	Definition
API	Application Programming Interface
CH	Clearing House
CPSV-AP	Core Public Service Vocabulary Application Profile
DAPS	Dynamic Attribute Provisioning Service
DCAT	Data Catalog Vocabulary
DPV	Data Privacy Vocabulary
ECC	Execution Core Container
eIDAS	electronic Identification, Authentication and trust Services
EIF	European Interoperability Framework
ELI	European Legislation Identifier
EU	European Union
GDPR	General Data Protection Regulation
HTTPS	Hypertext Transfer Protocol Secure
ID	Identification
IDS	International Data Spaces
IDSCP	International Data Spaces Communication Protocol
ISA2	Interoperability solutions for public administrations, businesses and citizens
ISO	International Organization for Standardization
JPA	Java Persistence API

JSON	JavaScript Object Notation
JSON-LD	JavaScript Object Notation Linked Data
JWT	JSON Web Token
OAUTH2	Open Authorization 2.0
ODRL	Open Digital Rights Language
OOP	Once Only Principle
PA	Public Administration
PoC	Proof of Concept
RAM	Reference Architecture Model
REST	Representational state transfer
SDG	Single Digital Gateway
SME	Small Medium Enterprise
SSO	Single Sign On
ToC	Table of Content
UC	Usage Control
URI	Universal Resource Identifier
URL	Universal Resource Locator
WP	Work Package
WS	Web Socket

1 Introduction

1.1 Purpose and Scope

The main goal of ACROSS is provide a holistic solution that allows public administrations to deliver a user-centric interoperable cross-border mobility service compliant with the current European regulations where the private sector can also interconnect their services while ensuring the data sovereignty of the citizens. To this end one of the ACROSS objectives is to provide a **set of connectors and data harmonization tools** that will facilitate the actual interoperability of the cross-border mobility services through the connection of public services so that they can interoperate with services from other countries as well as with those of the private sector also to include their services and offerings.

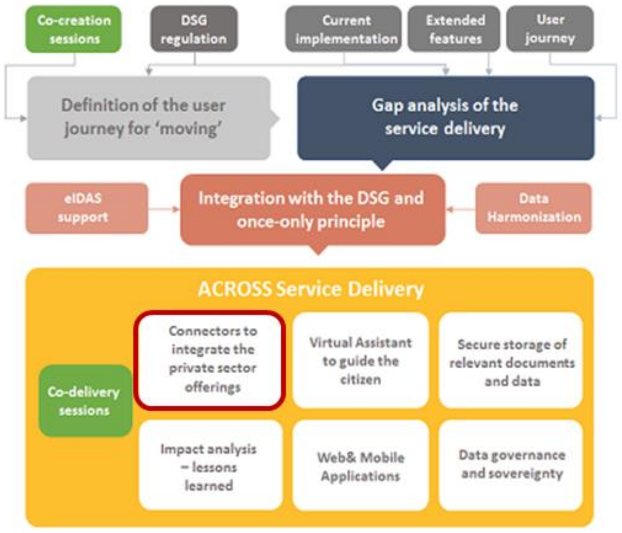

Co-creation sessions
DSG regulation
Current implementation
Extended features
User journey
Definition of the user journey for 'moving'
Gap analysis of the service delivery
eIDAS support
Integration with the DSG and once-only principle
Data Harmonization
ACROSS Service Delivery
Co-delivery sessions
Connectors to integrate the private sector offerings
Virtual Assistant to guide the citizen
Secure storage of relevant documents and data
Impact analysis – lessons learned
Web& Mobile Applications
Data governance and sovereignty

Figure 1 - ACROSS Conceptual approach and main components supporting user centric cross-border mobility service delivery.

The ACROSS solution aims to provide a common semantic and functional stratum enabling cross border access to data and processes backed up by functionalities and capabilities for data collection and harmonisation according to a set of common and shared data models based on European standards such as Core Public Service Vocabulary[1], Core Person Vocabulary[2] and DCAT for metadata[3].

The adoption of customizable connectors will provide proxy functionalities to connect and access public/private services offered by both the Public Administration and third parties and to get data from heterogeneous sources such as repositories, existing systems (e.g. owned by PA), etc. that could expose different interfaces.

[1]https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/core-public-service-vocabulary

[2]https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/core-person-vocabulary/release/100

[3] https://www.w3.org/TR/vocab-dcat-2/

To this aim, these customizable adapters will offer a set of software components ready to be used or tailored according to specific needs and requirements to allow the connection of data sources and public/third-party private services to ACROSS Platform for the provisioning of cross-border mobility service delivery (Figure 1).

1.2 Approach for Work Package and Relation to other Work Packages and Deliverables

WP4 aims to provide a set of tools and technological solutions that implements the borders of ACROSS Platform; in details, these tools and solutions will concern authentication aspects compliant with eIDAS, user support tools to facilitate both the interaction of the citizens with User Journey Services and connection to public and private sector services.

This report is seen as a living document, as evolution of the first one (D4.4) and the outcomes and the implementation of components presented in this deliverable are a subject to continuous refinements and modifications, based on the progress of all technical work packages (WP3, WP4, WP5), as well as the validation and evaluation phases performed in WP6 activities. In fact, the services and tools developed in this WP are integrated into the platform created in WP5, in line with the architecture described in [1] and adopted in the use cases in WP6.

1.3 Methodology and Structure of the Deliverable

This deliverable aims to report the updated design and implementation of data harmonization and connectors layer of ACROSS platform. To this end the deliverable has been structured in the following sections:

Section 2 describes the updated release of components of Data harmonization and connector layer of ACROSS solution and its relationship with ACROSS architecture components. The defined service model to support interoperability is described and updated to address Use case requirements and the current interaction with the other components of ACROSS platform. Besides, a detailed description of Service Catalogue is provided. Finally, the section provides an update of baseline technologies that can be used as needed connector implementation.

Section 3 describes the next steps towards the final release.

Annexes provides detailed information about the defined service model and the APIs exposed by the Service Catalogue. Finally, Annex III provides a mapping of covered requirements as identified in D5.2.

2 Data harmonization and connectors

The following sections provide the updated technical context and details of the design and implementation of the main components of Data harmonization and connectors layer in line with the ACROSS architecture as documented in D5.2.

2.1 ACROSS context and architecture overview

The following Figure 2 provides a conceptual view of ACROSS architecture by identifying the related layers and actors.

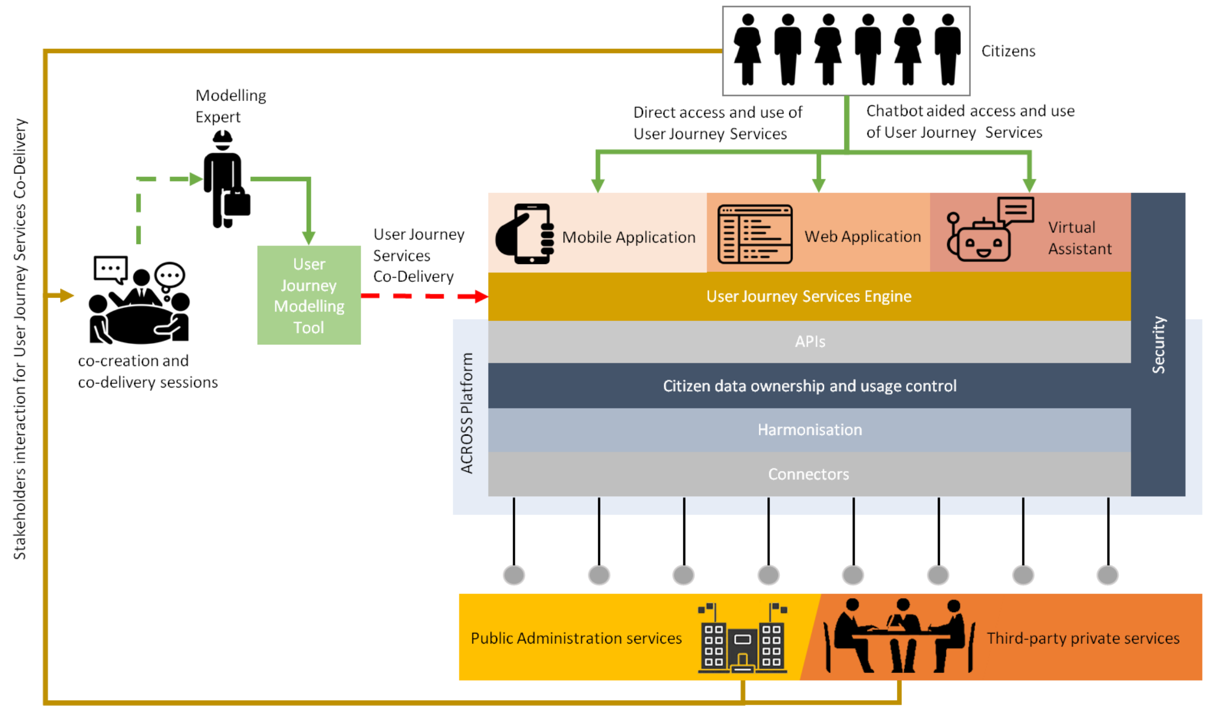

Citizens
Modelling Expert
Direct access and use of User Journey Services
Chatbot aided access and use of User Journey Services
Stakeholders interaction for User Journey Services Co-Delivery
co-creation and co-delivery sessions
User Journey Modelling Tool
User Journey Services Co-Delivery
Mobile Application
Web Application
Virtual Assistant
User Journey Services Engine
APIs
Citizen data ownership and usage control
Harmonisation
Connectors
Security
ACROSS Platform
Public Administration services
Third-party private services

Figure 2 - Conceptual architecture of ACROSS

In particular the **Harmonization and Connectors layers** provide the south bound connection with external systems and services from public and private sectors to provide a uniform description of the services and the related invocation. To this end these layers should provide for each identified service a service adapter (Figure 3), that is a ***single instance of adaptation*** of that service (public & private) for its use in ACROSS Platform from several points of view for its use in the upper layers: ***informational, data governance and service invocation***. To support that adaptation, Harmonization and Connectors layers should include a service catalogue with the aim to provides all functionality to register, model, map and publish and manage a uniform and harmonized machine-readable description of public and private services, according to the three above points of view, needed to support the uses of each service by the upper layers of ACROSS Platform.

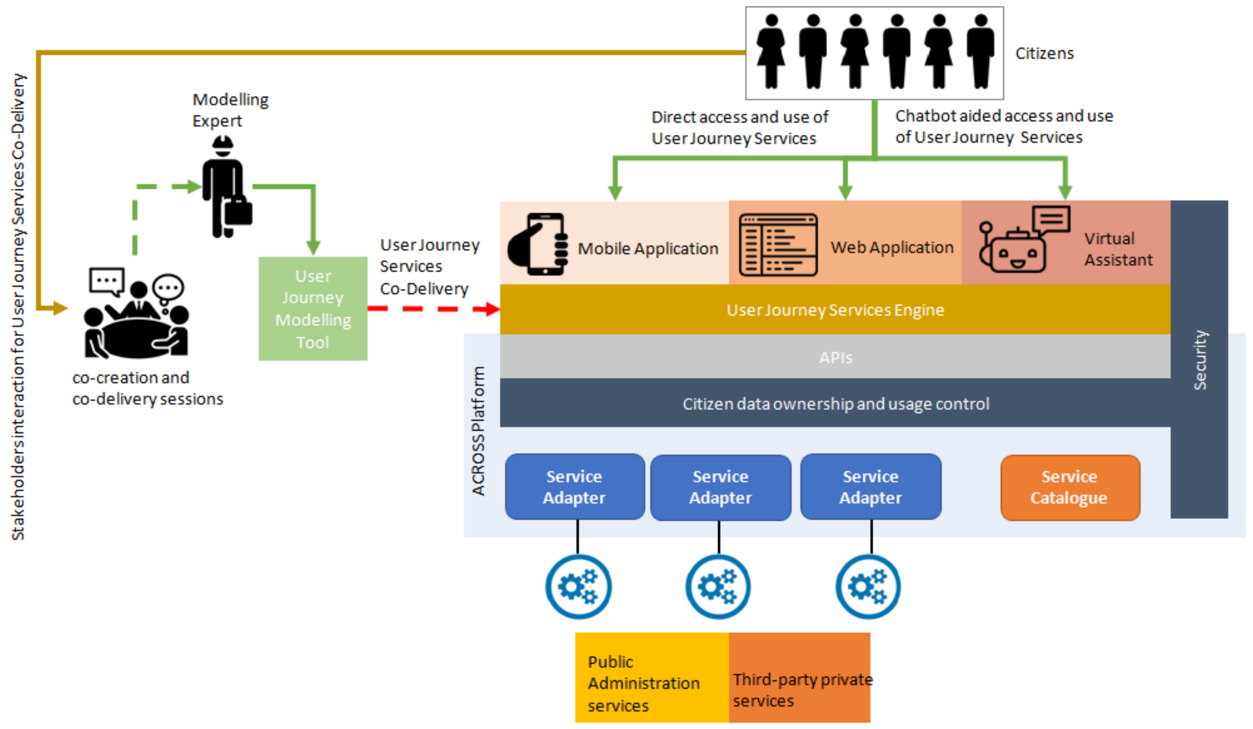

Citizens
Stakeholders interaction for User Journey Services Co-Delivery
Modelling Expert
Direct access and use of User Journey Services
Chatbot aided access and use of User Journey Services
User Journey Services Co-Delivery
User Journey Modelling Tool
co-creation and co-delivery sessions
Mobile Application
Web Application
Virtual Assistant
User Journey Services Engine
APIs
Citizen data ownership and usage control
Security
ACROSS Platform
Service Adapter
Service Adapter
Service Adapter
Service Catalogue
Public Administration services
Third-party private services

Figure 3 - Service Adapters and Catalogue. For each service used by ACROSS Platform a service adapter is defined and related information is stored in the Service Catalogue.

In the definition of ACROSS architecture (Figure 4), the upper conceptual layers have been included in the Data Harmonization and connectors main components blocks, namely **Service adapter** and **Service Catalogue** (Figure 5), to provide the following features:

- Secure connection to internal system of the PAs and services.
- Secure connection to private services to support cross border services.
- Uniform API for data access.
- Exposure of sub-set of existing functionalities/services of PAs and private
- Data access management based on authentication, authorization and privacy management for requested information.
- Semantic adaption to support interoperability according to common vocabularies
- Catalogue of services instances, models and metadata registry

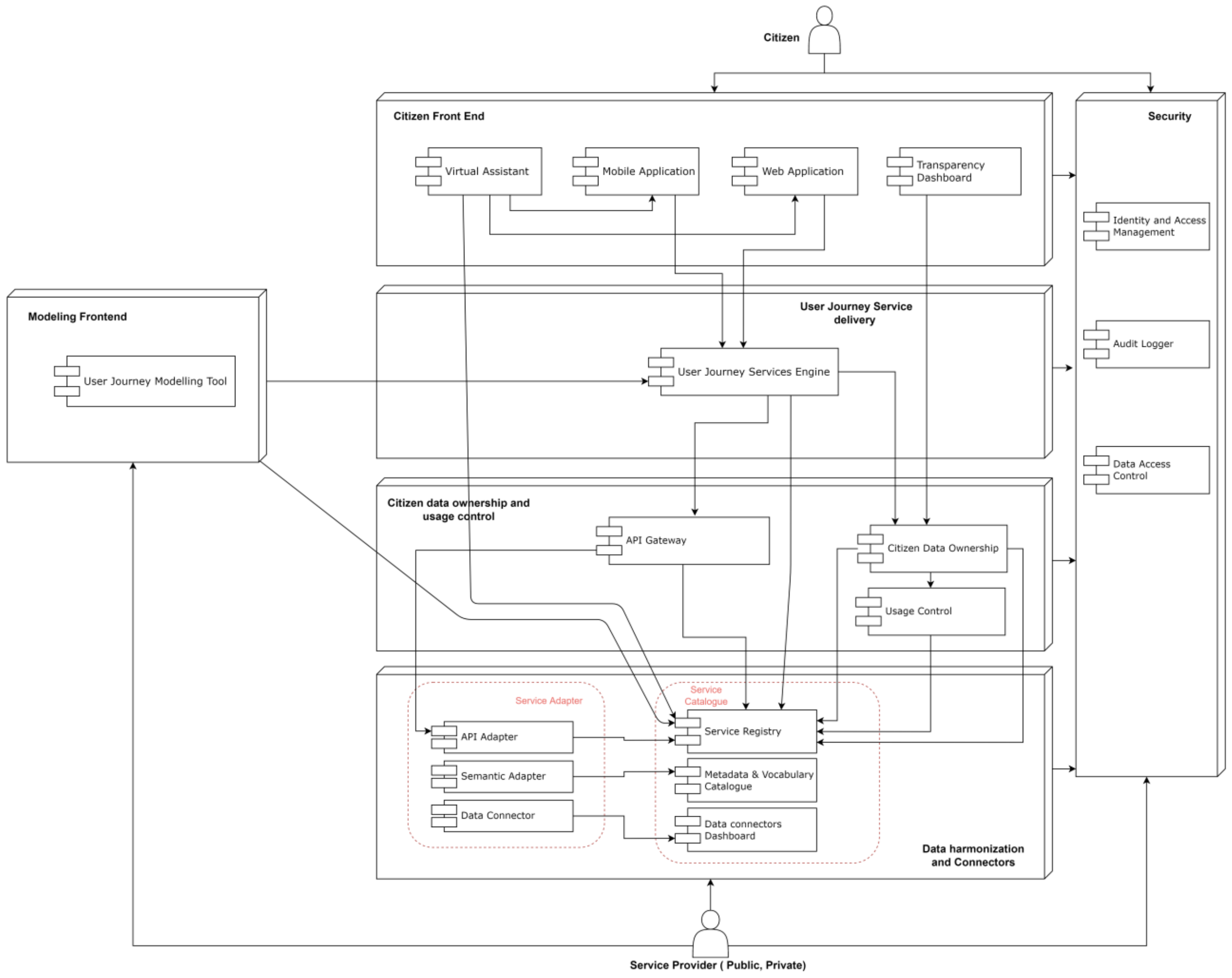


Figure 4 - Architecture of ACROSS Platform from [1]

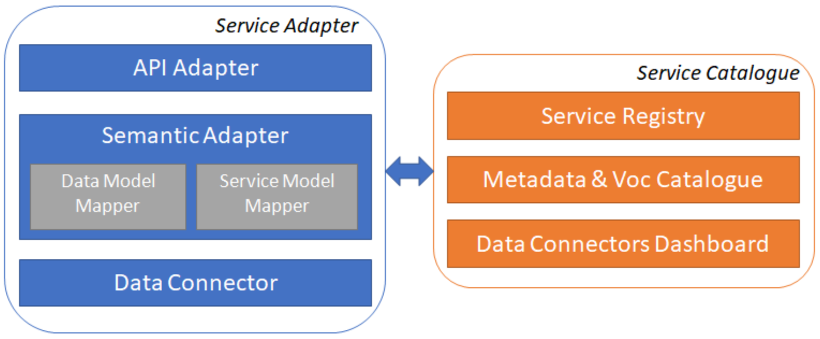


Figure 5 - Main components of Data harmonization and connectors. Each component is involved in the flow according to the type of adaptation performed

The **API Adapter** module is responsible to expose a standard set of APIs for data access (by means of data connectors) or functionalities supplied by internal or legacy systems of PAs and private service providers. API adapter is used by the API gateway to invoke all services included in the User Journey Service Engine. Moreover, it supports the enforcement of each request verifying, for instance, if a valid token is provided and if the token grants access to the requested resource. The **Semantic adapter** contributes in the service adaptation, through the Data Model Mapper, in order to supply data gathered from Data Connector in the expected format or, through the Service Model Mapper, to the semantic adaptation of service descriptions according to the standard and shared model adopted by the Service Catalogue. The **Data Connector** is actually the module that performs the technical integration of Legacy/Proprietary systems. Each instance of service adaptation has an ad-hoc developed data connector accordingly to the type of integration (e.g. read from API, read csv or json file or other files, read from SQL or NoSQL databases, etc.), covering also all the security aspects of authentication and authorization.

The above Service adapter modules are supported by the Service Catalogue by storing all the needed information or produced by the service adaptation modules to be consumed by the upper layers. The front-end and business layer of the Service Catalogue is the **Service Registry,** responsible to provide APIs for programmatically interaction and dashboards and graphical editors. The backend part is performed by the **Metadata and Vocabulary Catalogue.** Finally, the Data Connector Dashboards module provides a management cockpit of all available data connectors instances.

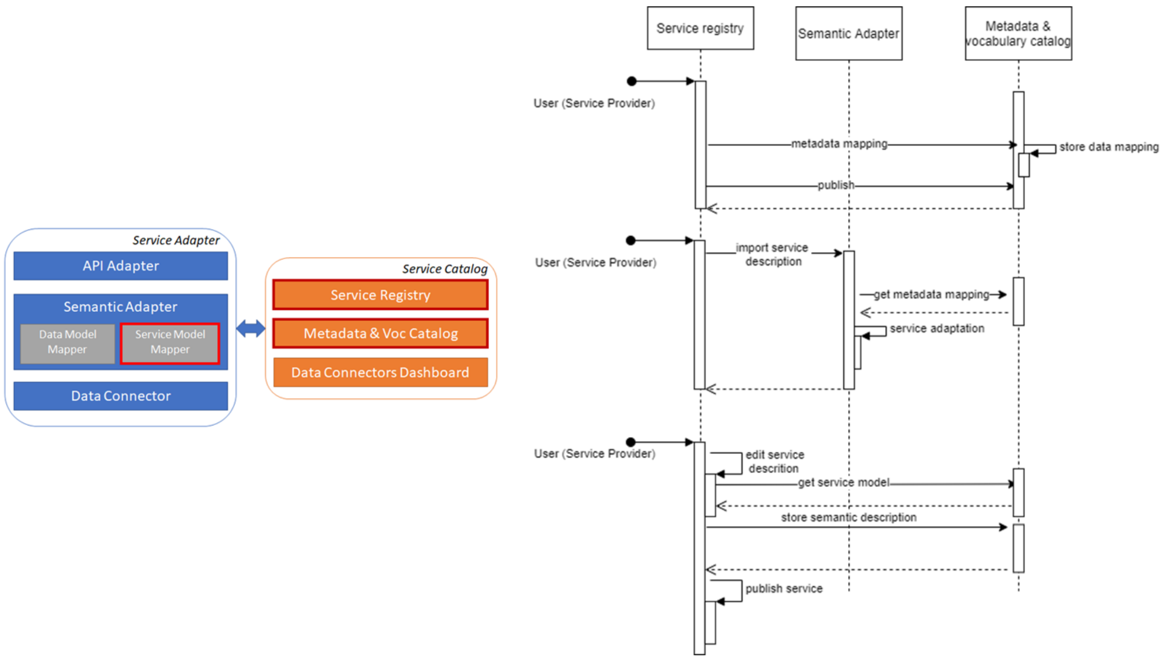
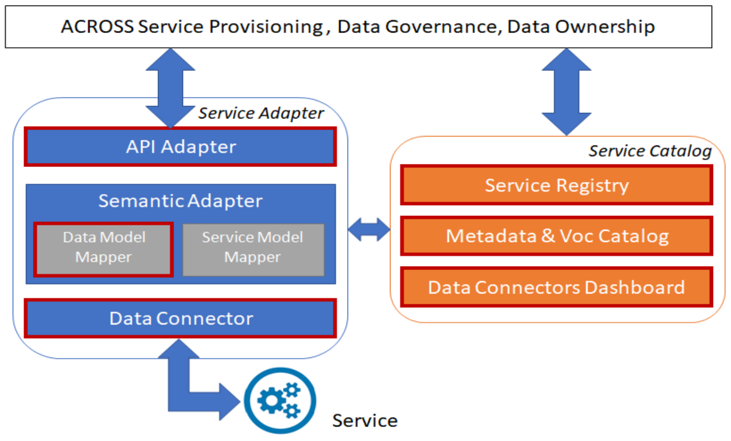


Figure 6 - Modules involved (and related flow) in Service model adaptation

Each component and related sub modules are invoked in relation to the type of service adaptation and the interaction with the upper layers. For example, the Semantic Adapter component can be invoked in order to provide functionalities for the provisioning of a common information model of the service registered to the platform. (i.e CPSV, see sections 2.2 and 2.3) (Figure 6), or used to support the interaction of ACROSS platform with an external service by defining a data connector and related data model adaptation (Figure 7).



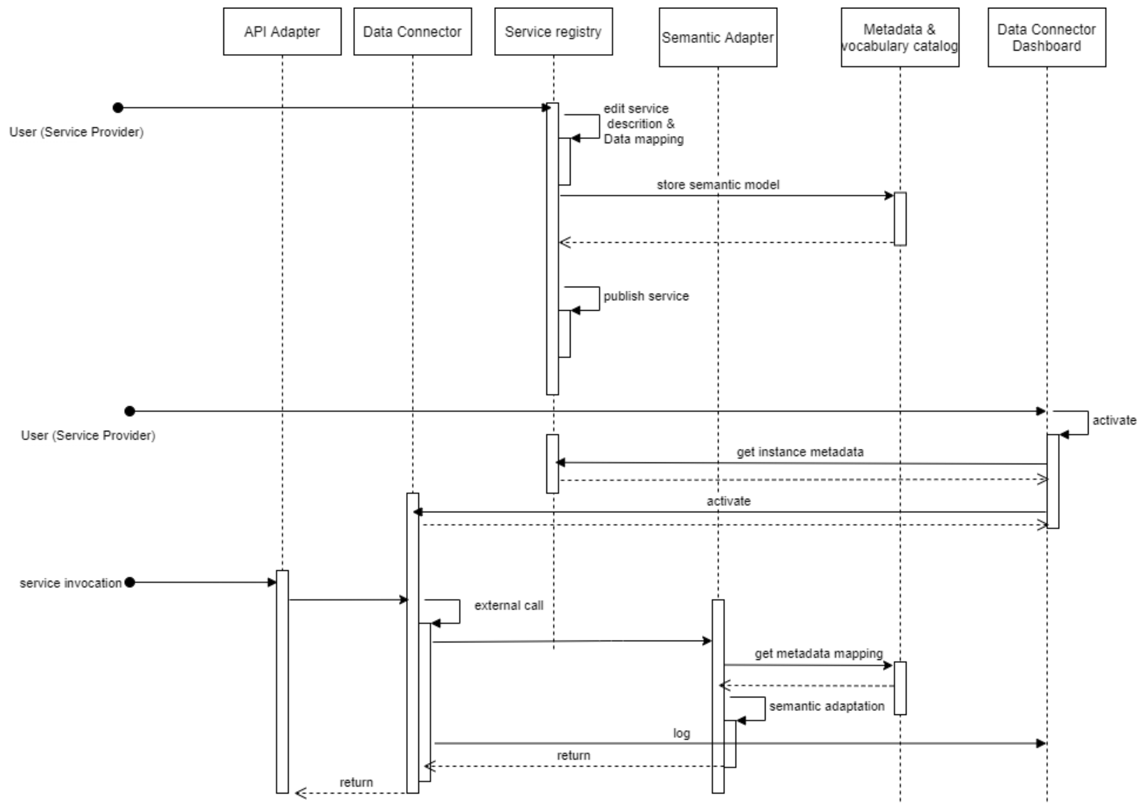


Figure 7 - Modules involved (and related flow) in Service invocation adaptation

In general, as depicted in the following Figure 8, each module interacts or it is invoked in order to provide a specific functionality or a piece of information, to be used also externally to ACROSS platform, for example to export service model description in a standard format.

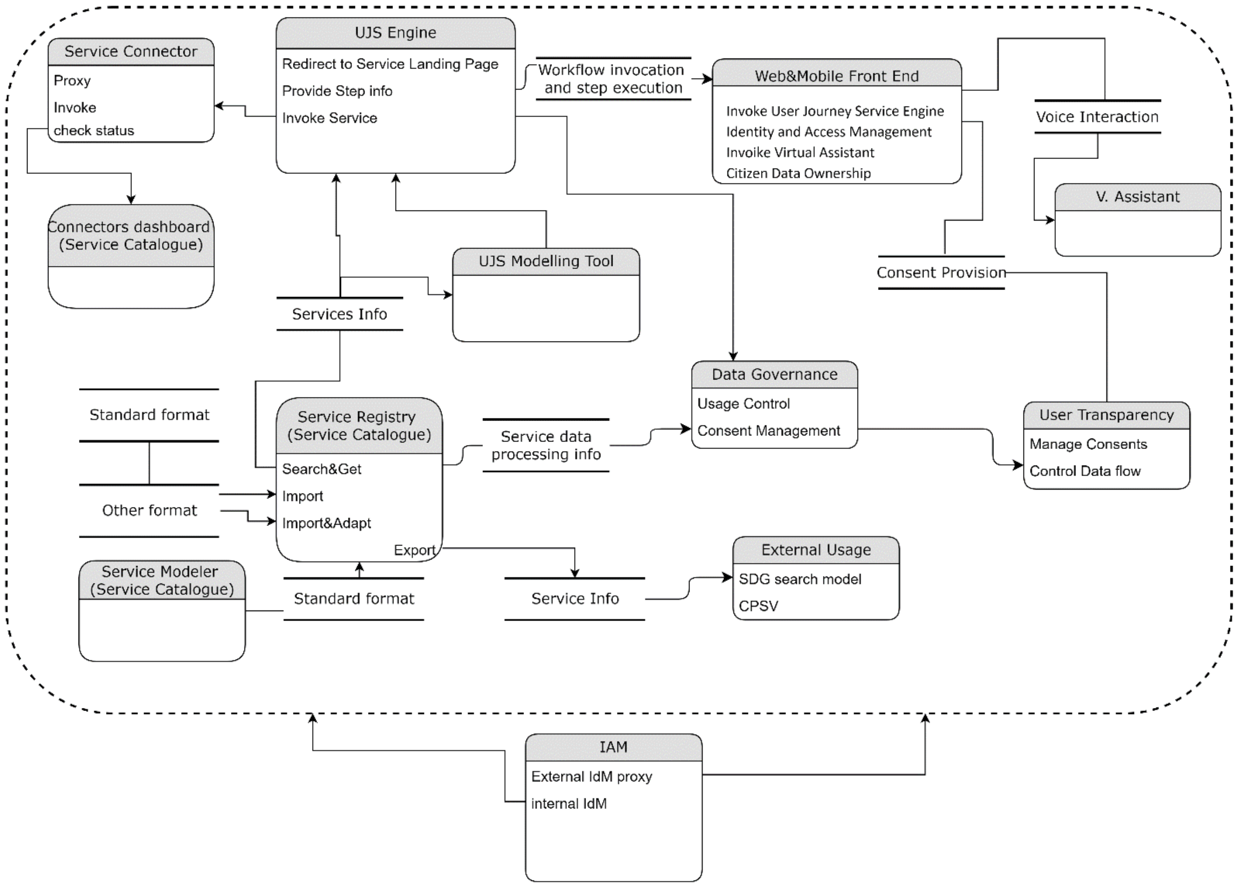


Figure 8 - Service Catalogue and Service Adapter modules interaction map with other modules of ACROSS platform.

2.2 Service Model

As described in the previous section, in order to support a uniform and harmonized machine-readable description of public and private services a service model has been defined to collect all information from the three point of view (Informational, Service invocation, Data Governance&Ownership) and managed in the Service Catalogue. The idea is to define this model by including and extending existing common models to describe each view (Figure 9).

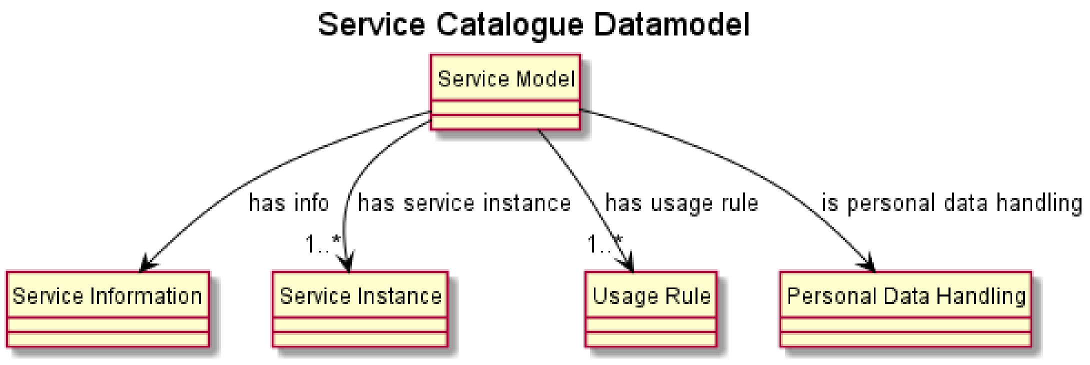


Figure 9 - Graphical representation of the Service Model main classes adopted in ACROSS

2.2.1 Information view

This section provides all information metadata of a service. This section follows the CPSV-AP, the Core Public Service Vocabulary Application Profile [2], a data model provided by the ISA2 Programme that is the result of a joint effort from different public administrations to reduce interoperability barriers. 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.

The definition of CPSV-AP model has followed the main objective to provide a lightweight and modular standard that can be reused. The CPSV-AP specifies only 2 mandatory classes (Public Service and Public Organisation) (Figure 10) and the data model itself is based on other standards such as the Core Public Organisation[4], the Core Criterion and Evidence[5] and the ELI Vocabulary[6] .

[4] https://joinup.ec.europa.eu/collection/semantic-interoperability-community-semic/solution/core-public-organisation-vocabulary/about

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

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

The CPSV-AP enables the description of public services and the associated life and business events, by standardising the semantics of personal milestones, including having a child, beginning education, looking for a new job, as well as professional changes such as starting or financing a company, hiring an employee. The descriptions will make data on these events structured, easier to capture and machine-readable. Public administrations and service providers can use this to guarantee a degree of cross-domain and cross-border interoperability between public service catalogues.

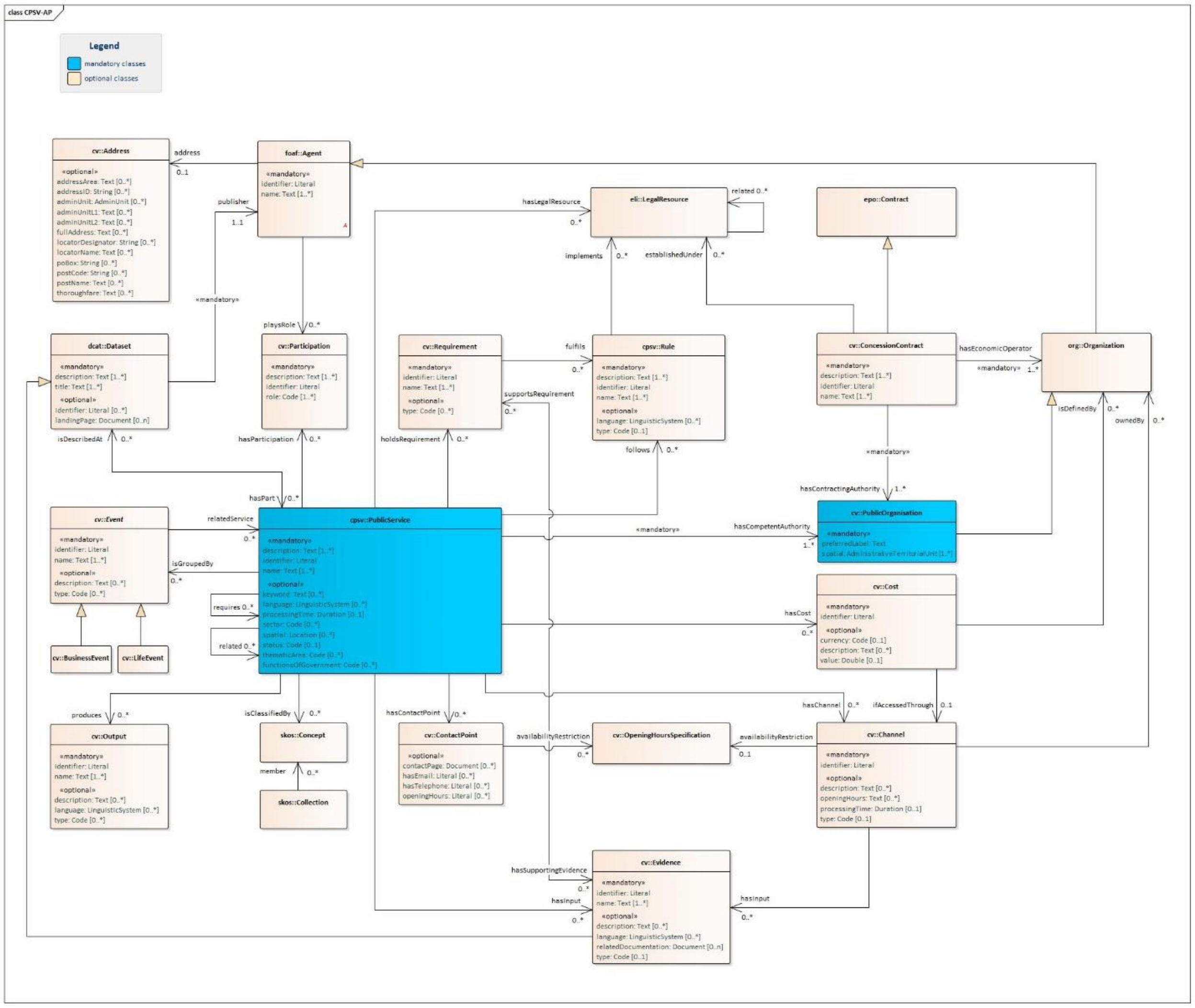


Figure 10 - Graphical representation of the relationships between the classes and properties of the full Core Public Service Vocabulary Application Profile (figure from [2])

These aspects contribute to address ACROSS requirements (section 7) for a Semantic and technical interoperability with SDG (single digital gateway) in order to "*Easy and user-friendly access to information means enabling the users to easily find the information, to easily identify which parts of the information are relevant for their particular situation and to easily understand the relevant information*"[4] and to address the recommendation to *"...use the Core Public Services Vocabulary (CPSV) to facilitate interoperability with national service catalogues and semantics. Member States should be encouraged to use the CPSV, but are free to decide to use national solutions. The information included in the repository for links should be made publicly available in open, commonly used and machine-readable format, for example by application programming interfaces (APIs), in order to enable its reuse."*

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. This information will be used to support the user journey modelling and service engine (see Figure 8).

2.2.2 Usage Rule and Personal Data Handling views

Usage Rule and Personal Data Handling views capture information about for handling data access rights. In particular Usage Rule view provides an interoperable information model, vocabulary, and encoding mechanisms for representing statements about the usage of content and services. This model, as described in [3] is based on IDS Usage Control Model [8] a specialization of the Open Digital Rights Language (ODRL)[7] to enforce usage restrictions for data, namely contract agreements, after access has been granted. Therefore, the purpose of usage control is to bind policies to data being exchanged and to continuously control the way how messages may be processed, aggregated, or forwarded to other endpoints. In the defined Service Model, a one-to-many relationship is defined (see section 5.8) by mapping a service description with one or more contract agreements defined in a separated model and not stored and managed by the Service Catalogue but managed by the components of Data Ownership and Usage Control layer.

[7] https://www.w3.org/TR/odrl-model/

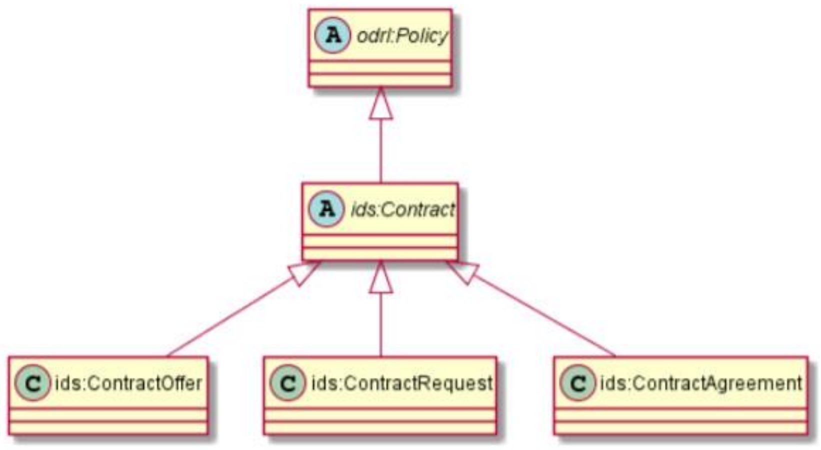


Figure 11 - IDS Contract Agreement

The personal Data Handling section is a specialized profile of The Data Privacy Vocabulary (DPV) [5] providing terms (classes and properties) to describe and represent information about personal data handling. In particular, the vocabulary provides extensible taxonomies of terms to describe the following components:

- Personal Data Categories
- Purposes
- Processing Categories
- Technical and Organisational Measures
- Legal Basis such as Consent
- Entities such as Recipients, Data Controllers, Data Subjects
- Rights
- Risks

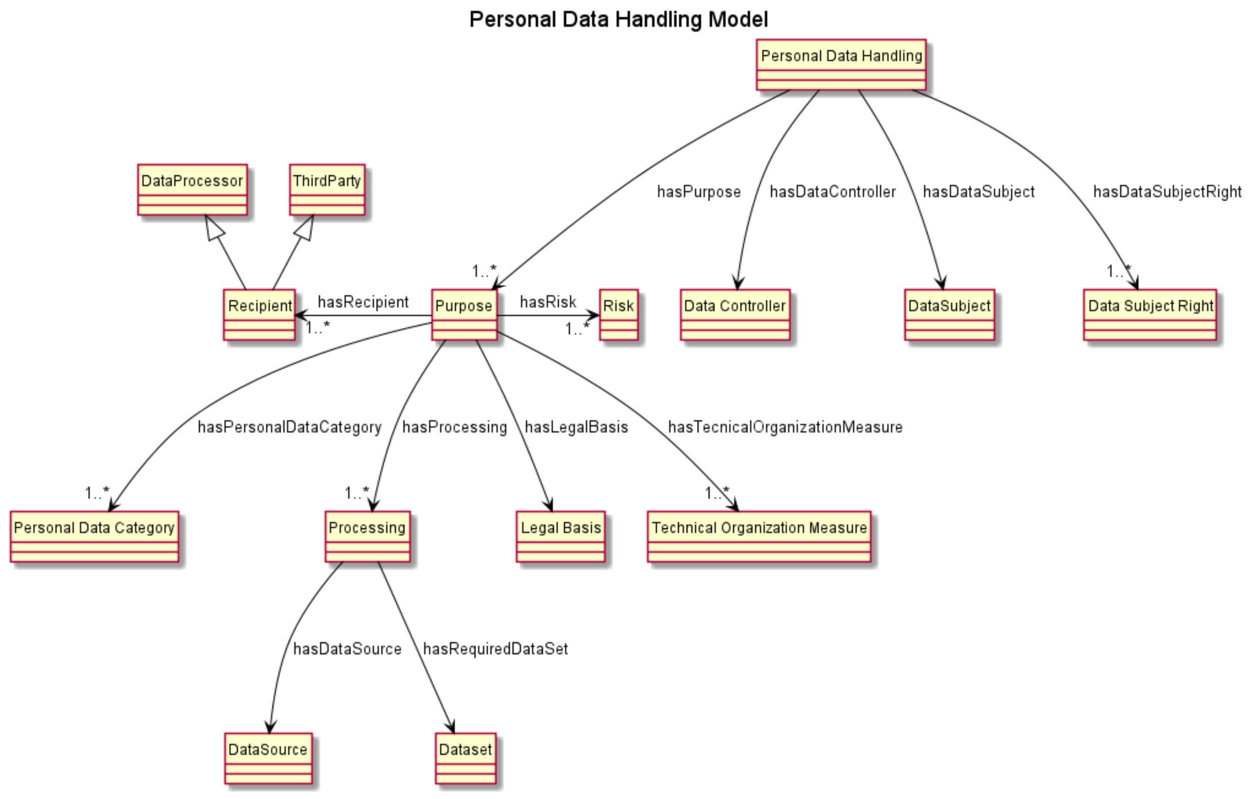


Figure 12 - Class diagram of Personal Data Handling model as profile of Data Privacy Vocabulary (DPV)

These terms are intended to represent personal data handling as machine-readable information by specifying personal data categories undergoing processing, its purpose(s), the data controller(s) involved, recipient(s) of this data, the legal bases or justifications used (e.g. consent or legitimate interest), involving technical and organisational measures and restrictions (e.g. storage location and storage duration), the applicable rights, and possibility of risks.

Examples of applications where the concepts provided by the DPV can be used are:

1. represent policies for personal data handling
2. represent information about consent e.g. provenance of consent
3. log/document personal data handling actions e.g. by a data controller
4. support automated checking of legal compliances of data handling ex ante (prior to processing), or ex post (i.e. check compliance after processing)

In accordance with the above application scenarios, the Personal Data Handling section uses the DPV vocabulary to collect all needed information that will consumed by the Consent Manager component of Data Ownership and Usage Control layer.

2.2.3 Service Instance view

This view provides all operational information to manage and invoke each service instance mediated by the related service adapter. It includes at least:

- Internal technical details to interact with internal components (e.g. data governance)
- Data/service connector invocation
- API Documentation (Open API/Swagger)
- Authentication and Authorization endpoints

The view will be extended with further information during the evolution of service adapter and all the component of ACROSS platform.

Detailed information about Service Model is provided in Annex I.

2.3 Service Catalogue

The following sections provides an overview of the current Service Catalogue implementation [6]. Service Catalogue provides all functionality to register, model, map and publish and manage all the information needed to support the uses of a service (public&private) according to the three points of view of Service Model described in the previous section:

- Informational
- Service Invocation
- Semantic interoperability & Personal Data Governance

The catalogue enables the storage and publishing of service by providing general, technical and data processing information based on standard models in particular based on CPSV-AP.

The Service Catalogue is a layered application implementing the Service Registry (front-end and backend) and Metadata Catalogue features, to provide APIs (see Appendix II) for programmatically interaction with other components of ACROSS platform and dashboards and Agraphical editors supporting users to manage service descriptions and related model adaptation (Figure 13).

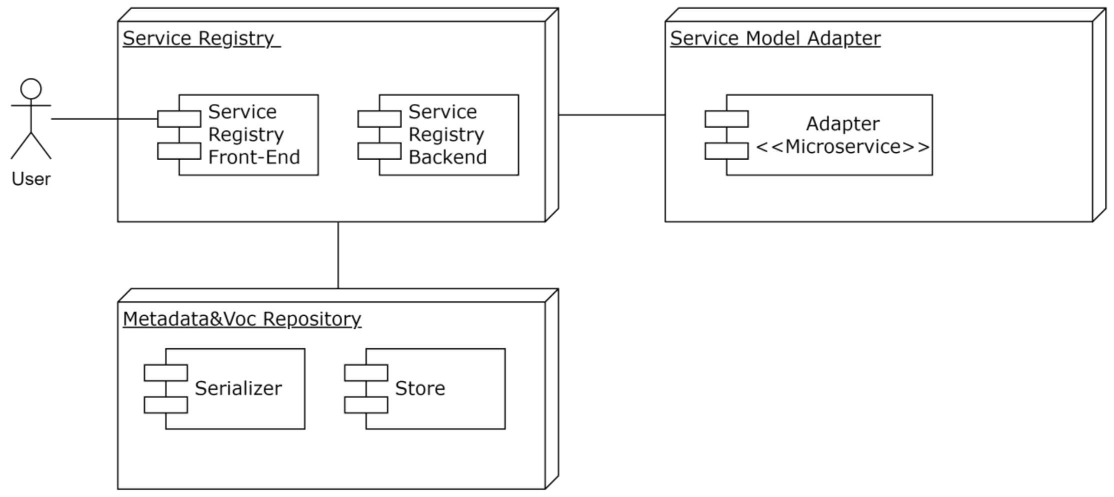


Figure 13 - Main modules implemented by the Service Catalogue

The Backend is implemented as Spring Boot[8] Java microservice, and will be deployed with a tightly coupled storage service (MongoDB[9] 4.2+). The Front-end, is an Angular portal based on Nebular[10] framework (Figure 14).

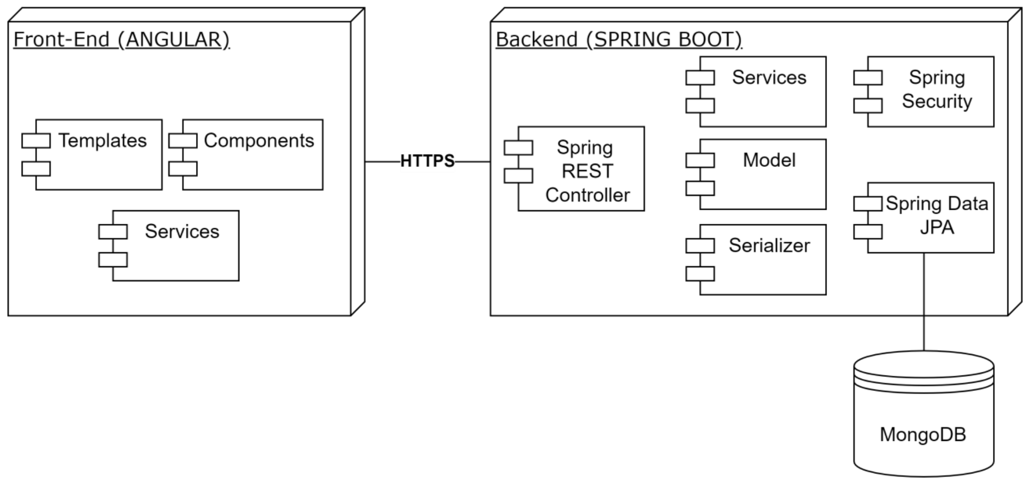


Figure 14 - Front-End and Backend of Service Catalogue implementation

[8] https://spring.io/projects/spring-boot
[9] https://www.mongodb.com/
[10] https://akveo.github.io/nebular/

The two layers can be deployed as Docker[11] containers, based on Tomcat Alpine image[12] and paired with a MongoDB container. This adoption of several reliable and production ready technologies (Figure 15) guarantees robustness and modularity of the solution.

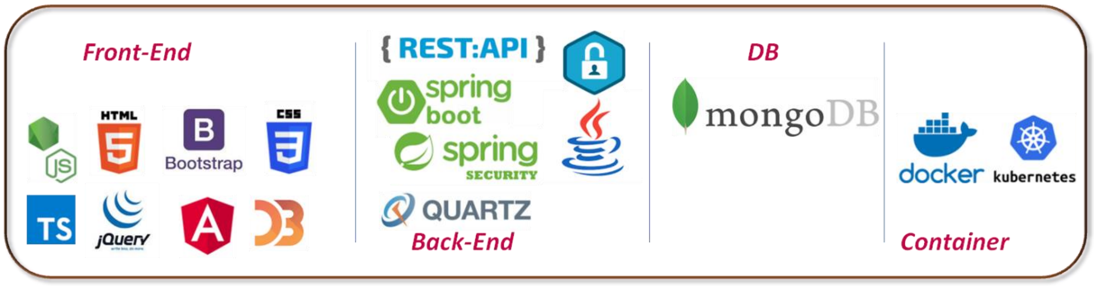


Figure 15 - Adopted technologies in Service Catalogue implementation

Service Catalogue architecture implementation is completed by integrating Spring Security and Keycloak[13] that supports OpenId Connect[14] and OAuth2[15] authorization framework. The Service Catalogue uses the Open Id Connect protocol upon the OAuth2 Authorization workflows, in order to perform User authentication and obtain an Access Token (JWT), which will be used to grant access.

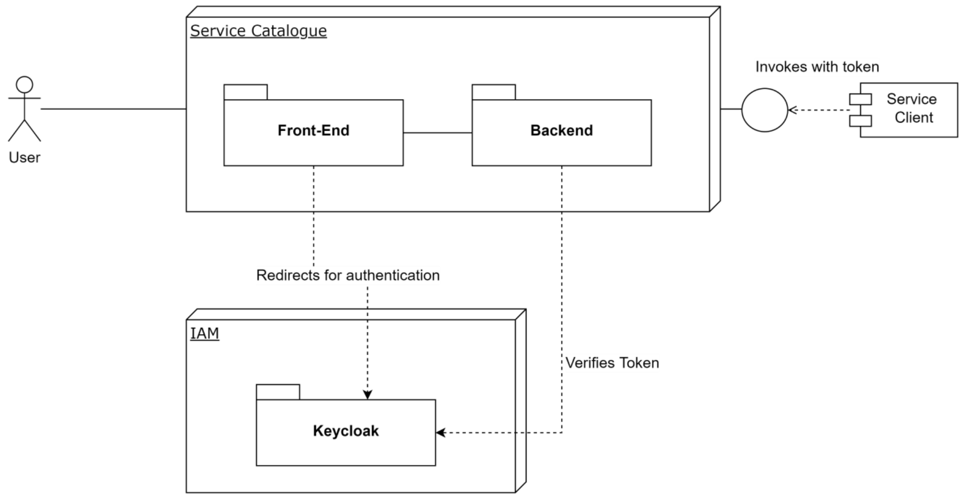


Figure 16 - Interaction of Service Catalogue with Keycloak for identity and access management

[11] https://www.docker.com/
[12] https://hub.docker.com/_/tomcat
[13] https://www.keycloak.org/
[14] https://openid.net/connect/
[15] https://oauth.net/2/

Similarly, a client application/service wanting to interact with the Service Catalogue, will perform OAuth2 Authorization, obtaining an Access Token to be used in the request to APIs (Figure 16).

The choice of Keycloak provides an out of box solution for a rapid security layer development of application with supporting features such as Single-Sign-On (SSO), Social Login, User Federation, Client Adapters, Admin Console and Account Management Console and finally Identity Brokering[16] (Figure 17).

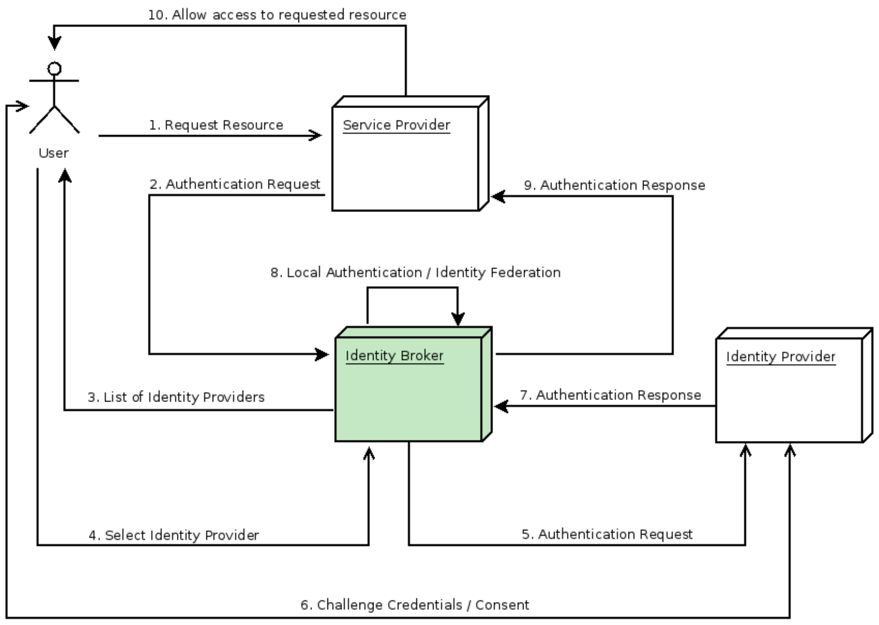

10. Allow access to requested resource
User
1. Request Resource
Service Provider
2. Authentication Request
9. Authentication Response
8. Local Authentication / Identity Federation
Identity Broker
Identity Provider
3. List of Identity Providers
7. Authentication Response
4. Select Identity Provider
5. Authentication Request
6. Challenge Credentials / Consent

Figure 17 - Identity brokering flow supported by Keycloak

This last aspect will facilitate the integration of the Service Catalogue with multiple and specific identity Systems.

2.3.1 Service Manager

The Service Manager is a multi-role, Angular[17] based admin dashboard implemented with the aim to include all module sections to interact with Service Catalogue but at the same time, according to the role of authenticated user, to manage the consents registry as Data Controller (Figure 18).

[16] https://www.keycloak.org/docs/latest/server_admin/#_identity_broker
[17] https://angular.io/

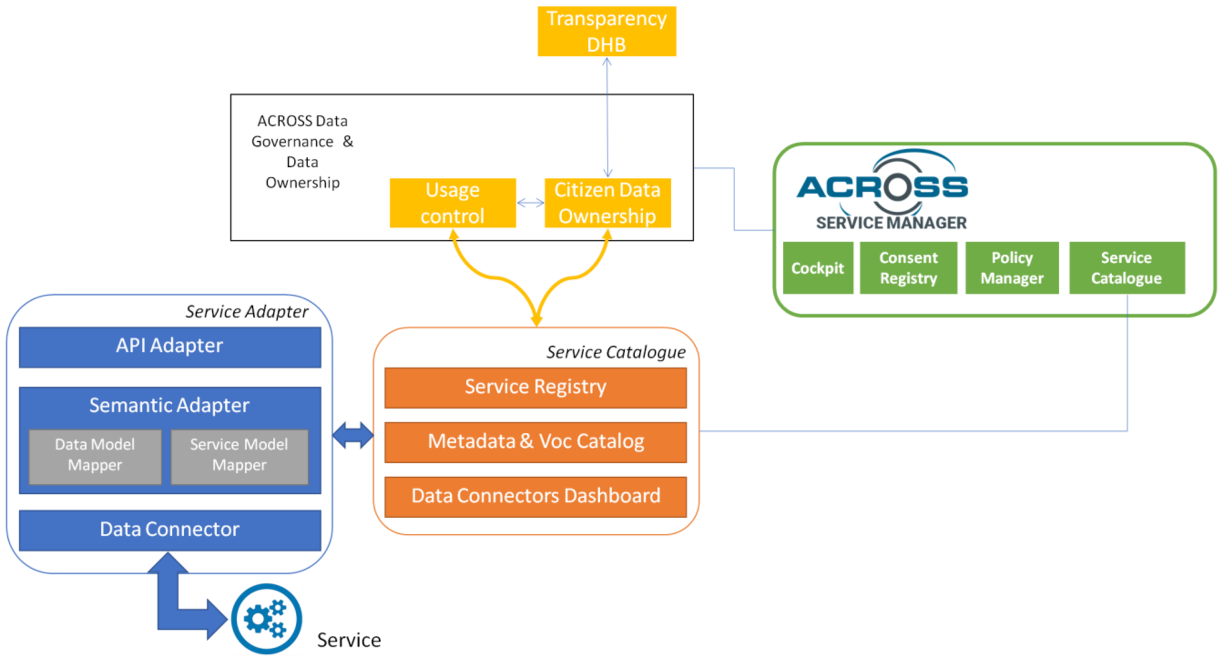
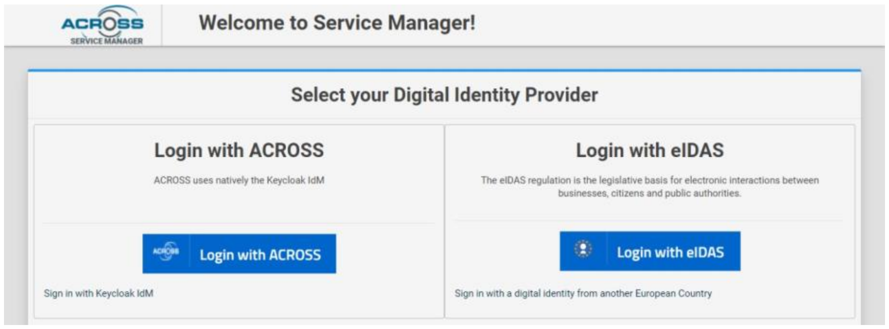


Figure 18 - The Service Manager admin dashboard provides a modular web interface to interact with several layers of ACROSS Platform.

The Service Manager uses Keycloak as identity broker providing at login phase an extensible page to select optional authentication systems (Figure 19).


Welcome to Service Manager!

Select your Digital Identity Provider

Login with ACROSS

ACROSS uses natively the Keycloak IdM

Login with ACROSS

Sign in with Keycloak IdM

Login with eIDAS

The eIDAS regulation is the legislative basis for electronic interactions between businesses, citizens and public authorities.

Login with eIDAS

Sign in with a digital identity from another European Country

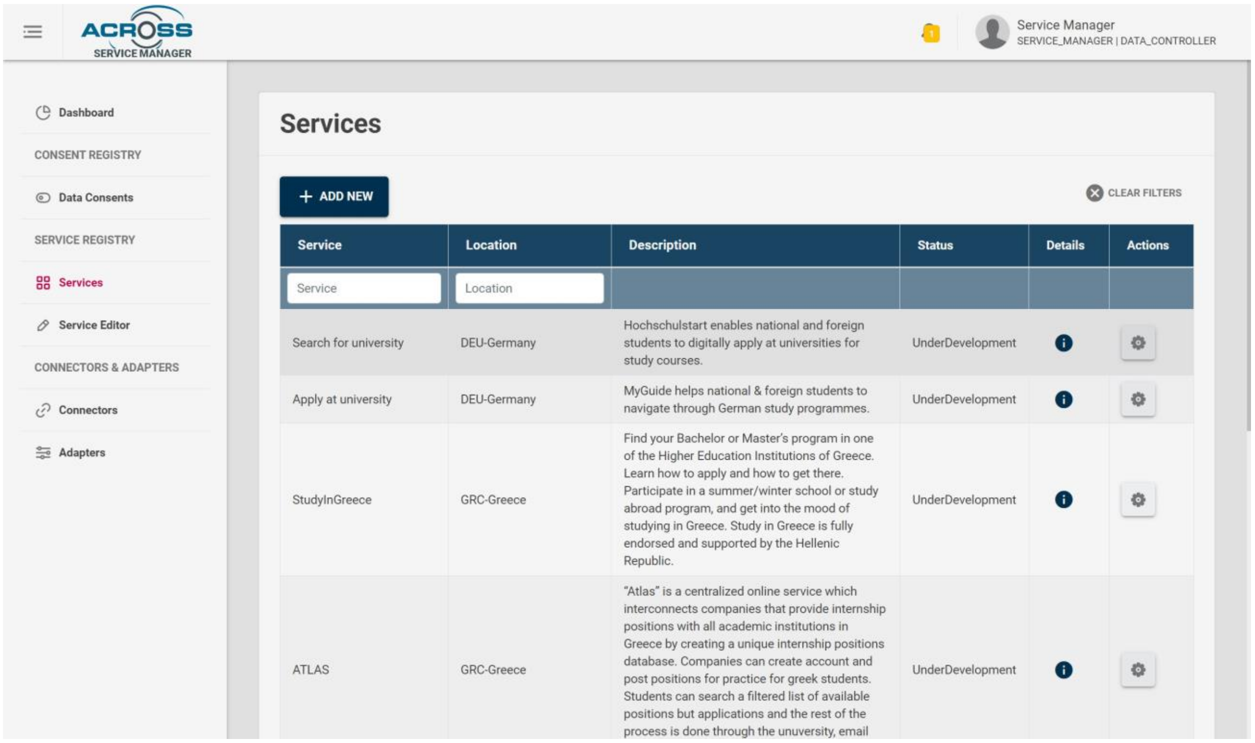
Figure 19 - Authentication page of Service Manager

In particular for eIDAS authentication, or national identity schemes, a dedicated adapter should be implemented, brokered by Keycloak.

Once authenticated, the user according to the roles assigned by means of Keycloak[18] the user can access to several sections. In particular can view the list of already inserted services by

[18] https://www.keycloak.org/docs/latest/server_admin/

having a first look about their basic information (name, status, description...), or to be redirected to the "detail" page (Figure 20).



Services

Service	Location	Description	Status	Details	Actions
Search for university	DEU-Germany	Hochschulstart enables national and foreign students to digitally apply at universities for study courses.	UnderDevelopment		
Apply at university	DEU-Germany	MyGuide helps national & foreign students to navigate through German study programmes.	UnderDevelopment		
StudyInGreece	GRC-Greece	Find your Bachelor or Master's program in one of the Higher Education Institutions of Greece. Learn how to apply and how to get there. Participate in a summer/winter school or study abroad program, and get into the mood of studying in Greece. Study in Greece is fully endorsed and supported by the Hellenic Republic.	UnderDevelopment		
ATLAS	GRC-Greece	"Atlas" is a centralized online service which interconnects companies that provide internship positions with all academic institutions in Greece by creating a unique internship positions database. Companies can create account and post positions for practice for greek students. Students can search a filtered list of available positions but applications and the rest of the process is done through the unuversity, email	UnderDevelopment		

Figure 20 - Service list page

From "Actions" the user can perform several actions in accordance with the status of service (Figure 21):

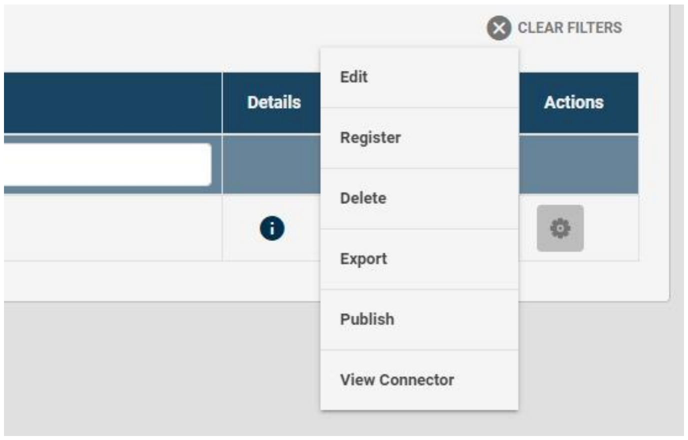
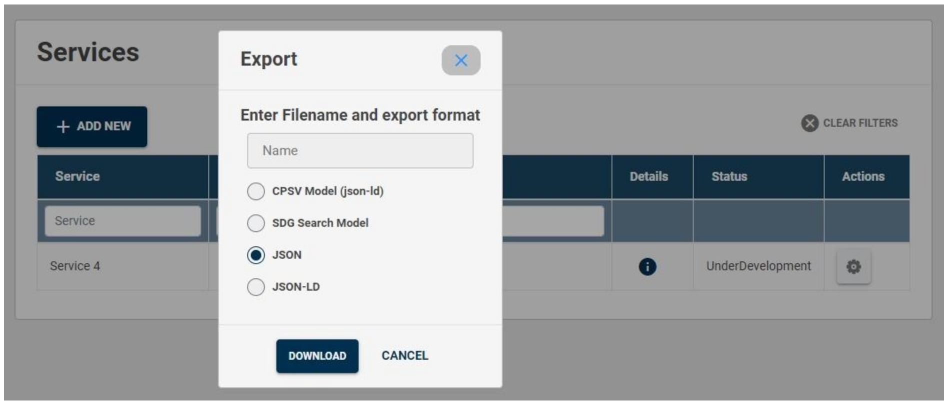


Figure 21 - Available actions in accordance to the status of service

- *Edit*. User can modify or complete service description, by entering in edit mode page

- *Register*. This action changes the status of service description into "completed". Once completed the service is searchable, by means of APIs exposed by the Service Catalogue, by ACROSS components.
- *Delete.* User can delete a service description. The action can be performed if the service is in the status of "UnderDevelopment" or after a de-registration.
- *Export.* User can export the description of Service by selecting different formats (Figure 22): JSON, JSON-LD, CPSV-AP Model (json-ld) and Single Digital Gateway Search Service model[19]
- *Publish.* By this action the Service Catalogue provides the availability to customize and manage multi publish actions. It lets to publish externally the service description at all or some information.
- *View Connector.* It lets to switch to the related connector adapter page.



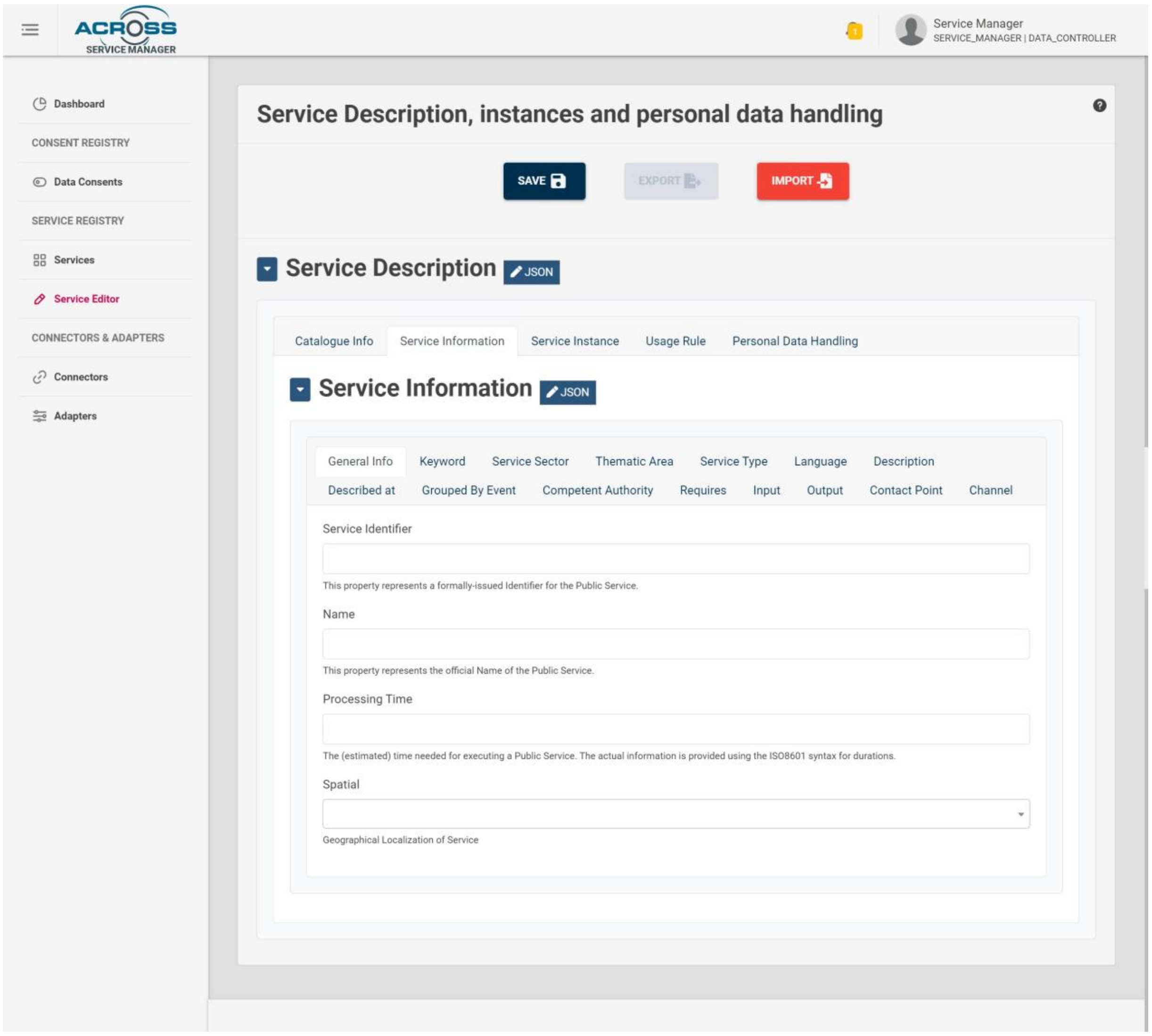
Service	Details	Status	Actions
Service 4		UnderDevelopment	

( ) CPSV Model (json-ld)
( ) SDG Search Model
[✓ JSON]
( ) JSON-LD

Figure 22 - Export dialog

From the list page the user can add a new service description by clicking on "Add new" button. The user is redirected to Service Editor page. It is composed in several tabs in relation to the Service model views described in section 2.2 and detailed in Annex I

[19] https://github.com/catalogue-of-services-isa/SDG-search-service-model



Service Identifier

This property represents a formally-issued Identifier for the Public Service.

Name

This property represents the official Name of the Public Service.

Processing Time

The (estimated) time needed for executing a Public Service. The actual information is provided using the ISO8601 syntax for durations.

Spatial

Geographical Localization of Service

Figure 23 - Service Editor page

Each property is documented with a description and by clicking the "?" button in the in the top right-hand corner a guide is provided. The user can import existing standard service models or non-standard/legacy descriptions by selecting the suitable registered service model adapter (Figure 24).

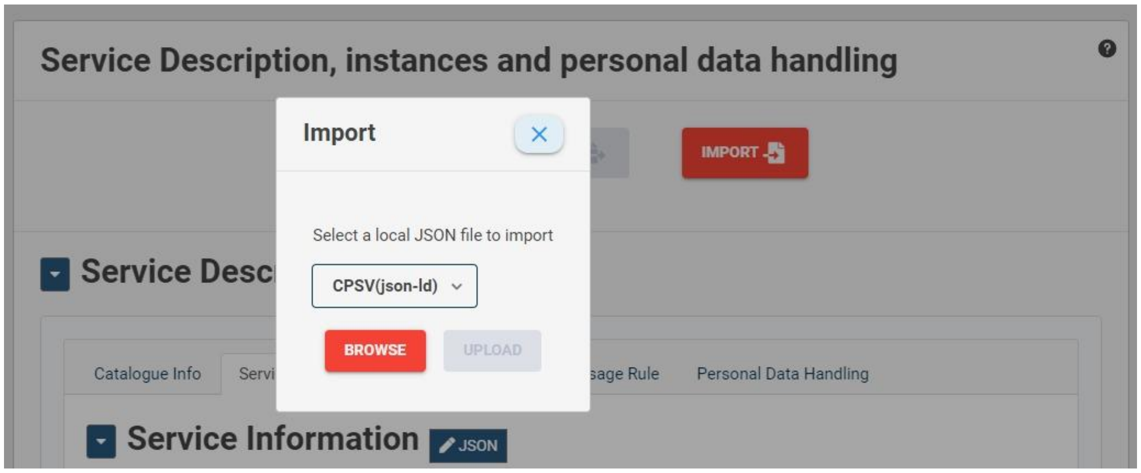
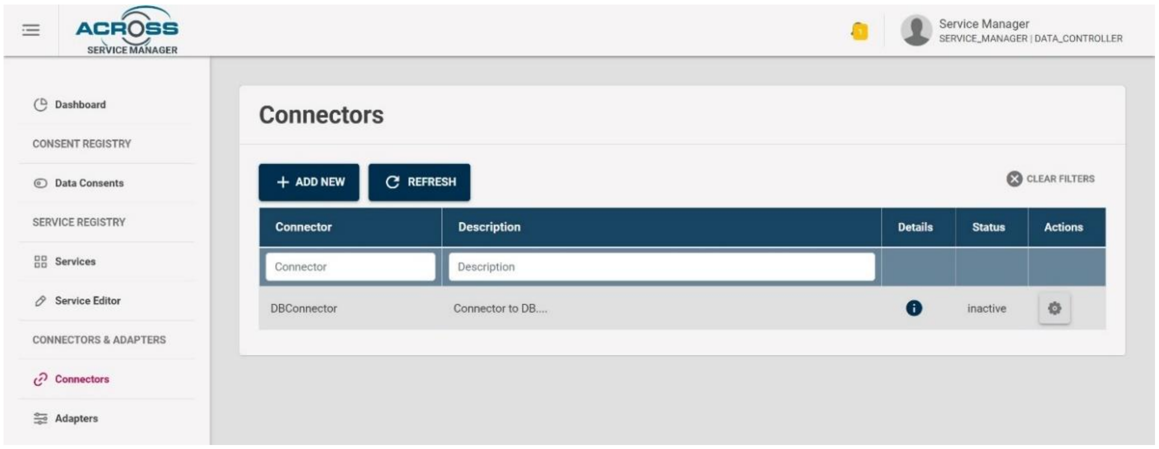


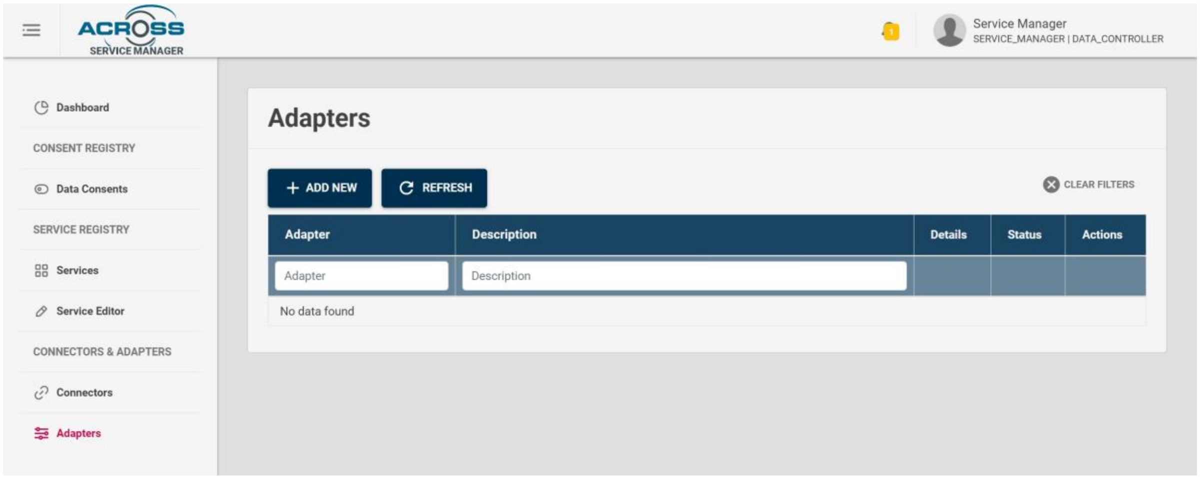
Figure 24 - Import dialog

The "Connectors" (Figure 25) and "Adapters" (Figure 26) sections provide quick information about the registered connectors and their status and logs. From these sections it is possible to edit their metadata or register new ones (Figure 27).



Connector	Description	Details	Status	Actions
DBConnector	Connector to DB....		inactive	

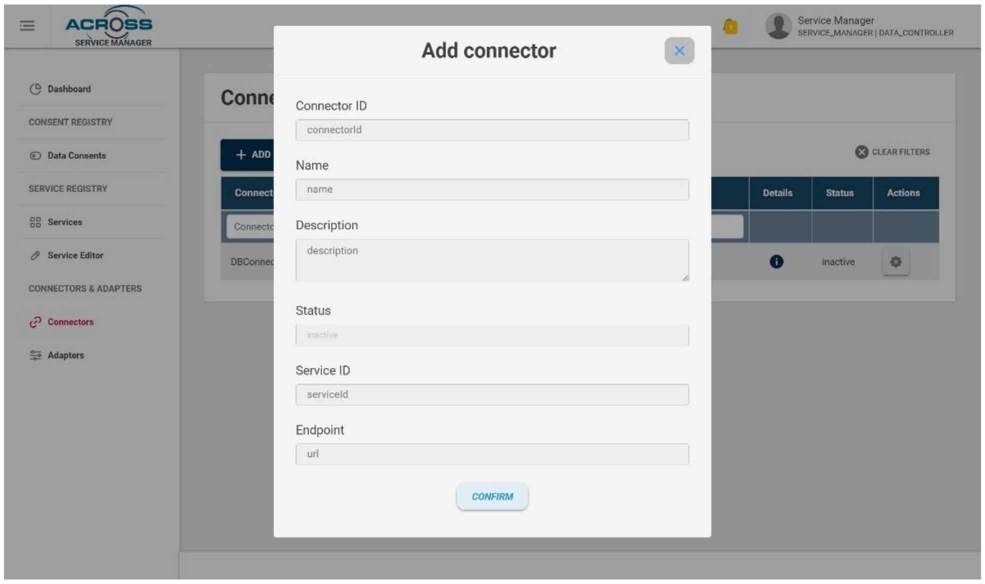
Figure 25 - Connectors list page



Adapters

Adapter	Description	Details	Status	Actions
Adapter	Description			
No data found				

Figure 26 - Adapters list page

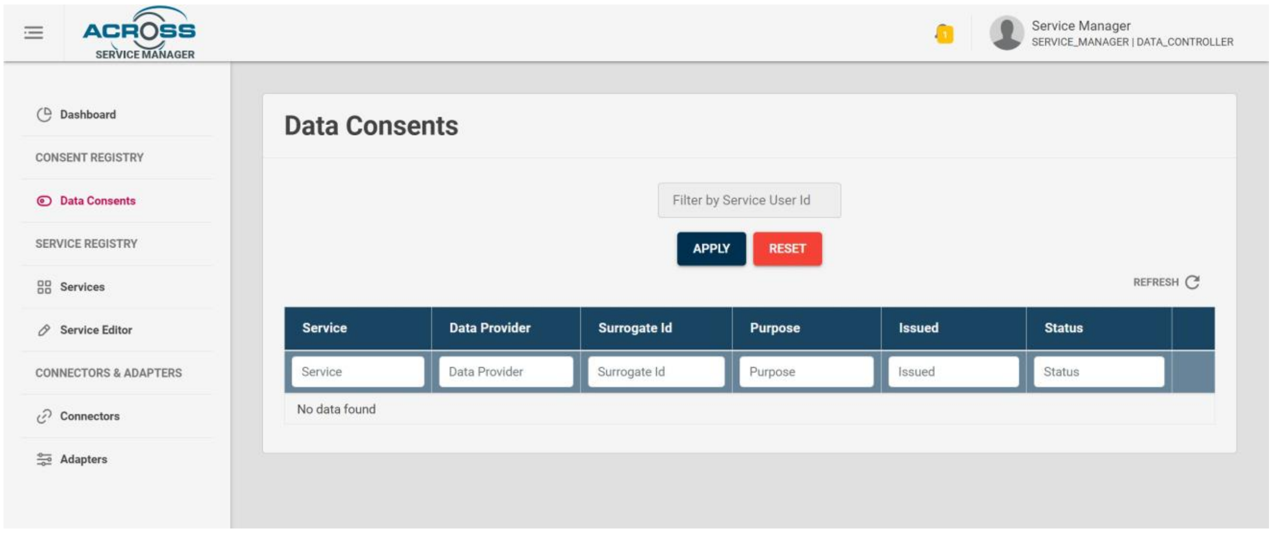


Add connector

Connector ID: connectorId
Name: name
Description: description
Status: inactive
Service ID: serviceId
Endpoint: url

Figure 27 - Connector metadata entry

The "Data Consents" page provides, is the authenticated user has "data-controller" role, a registry of consents collected. It is a front-end client of the APIs provided by the Consent Manager component of Data Governance & Data Ownership layer of ACROSS platform.



Service	Data Provider	Surrogate Id	Purpose	Issued	Status
Service	Data Provider	Surrogate Id	Purpose	Issued	Status
No data found					

Figure 28 - Consent Registry page

Finally, the section "Dashboards" provides and extensible page of graphical dashboard cards providing some summaries about the inserted services (Figure 29).

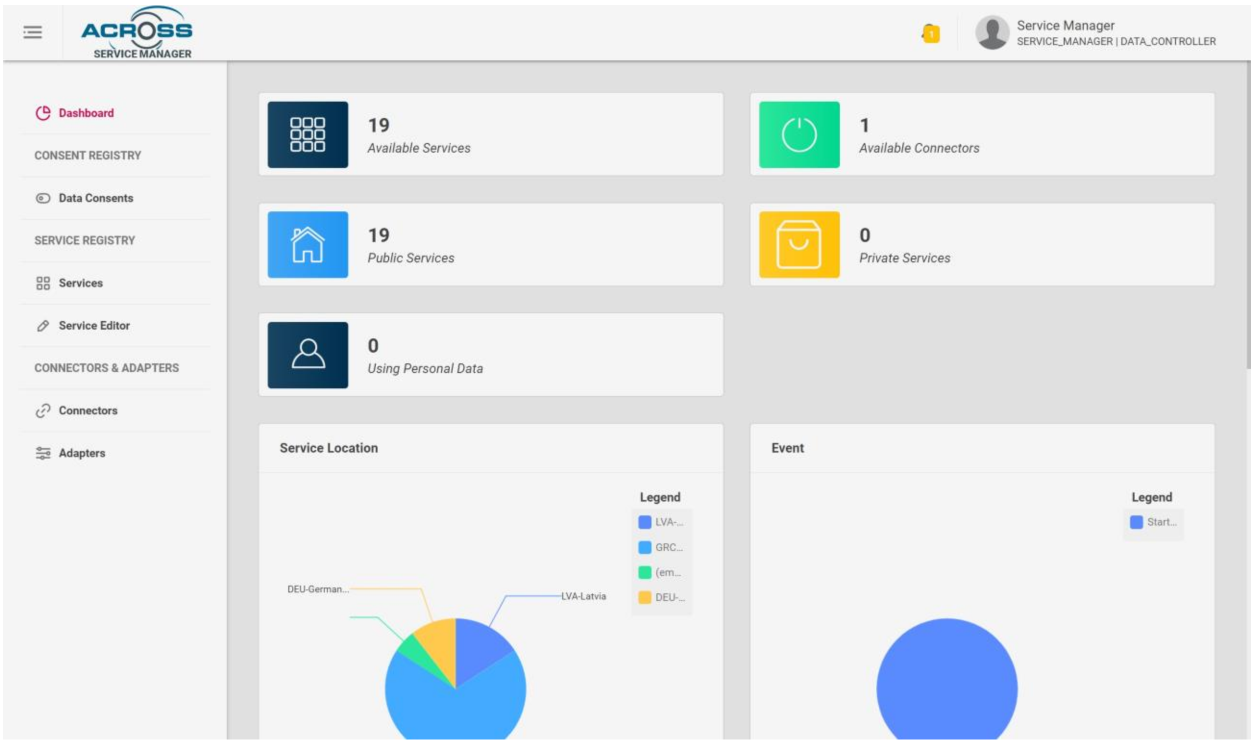


Figure 29 - Dashboard page

2.4 Service Adapter and baseline technologies

As defined in section 2 Service adapters are single instances of adaptation of services (public & private) for its use in ACROSS Platform useful to adapt heterogeneous service definitions into a common one and encapsulate generic communications-related logic required to use services and also to include logic that is quite specific for a given service. In particular connectors components make resources like database tables, stored procedures, domain objects, or files accessible to clients with a minimal amount of coding.

In the following section a description of technologies that can be adopted and extended to implement the custom implementation of a Service Adapter. The common approach is to implement, regardless the adopted technology to implement the specific adapter/connector, several *Enterprise Integration Patterns[13]* to provide a complete set of standards to integrate existing legacy application systems, company-developed host applications, and third-party vendor applications.

The choice of a specific technologies will be influenced to the type of integration patter to implement and to the technological background of the legacy system to integrate. Hence, the following technologies will be considered as some of the potential adoptions and the related metadata will be stored in the Service Catalogue in order to invoke the running instance of service adaptations (if needed) for the selected services included in the user journey processes.

2.4.1 Data Model Mapper

The Data Model Mapper tool[20] enables to convert several file types (e.g. CSV, Json) to different defined data models. The files in input can contain either rows, JSON objects or other structured data, each of them representing an object to be mapped to an entity, according to the selected Data Model.

In particular, it performs following steps:

1. *Parsing:*
    - Parse input file, by converting it into a row/object stream.
2. *Streaming:*

[20] https://gitlab.com/synchronicity-iot/data-model-mapper

   - Each row/object coming from the stream is converted to an intermediate object.
3. *Mapping:*
   - By using the input JSON Map, convert the intermediate object to an entity, according to a specific target Data Model.
4. *Validation and report:*
   - Validate resulting object against the JSON schema corresponding to a target Data Model.
   - Produce a report file with validated and unvalidated objects.

- *Writing: Context broker or File*
   - Validated objects can be sent to a configured context broker to the configured context broker[21] and/or to a local file.

The tool is developed in Node.js[22] and can be started as a command line tool. The current implementation is going to be extended to implement the Semantic Adapter as microservice module to be further customized for the specific service/data model adaptation.

2.4.2 Data Connectors

Data Connectors basically covers four application integration approaches from Enterprise Integration Patterns . The four integration approaches include File Transfer, Shared Database, Messaging, and Remote Procedure Invocation:

- File Transfer: One application produces data files for others to consume, and viceversa.
- Shared Database: Applications can store and share the information in a common database.
- Messaging: An application connects to a shared messaging system, exchanges data, and invokes s behaviour using messages.
- Remote Procedure Invocation: An application exposes its APIs so that they can be invoked remotely by other applications.

In the following section some easy to use integration frameworks are provided: Apache Camel Integration, Spring Integration. They are available in a JVM environment and offer a

[21] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/Context+Broker
[22] https://nodejs.org/

standardized, domain-specific language to integrate applications. Anyway, the used approach and integration patterns does not constrain the use of these specific frameworks at the expense of others.

2.4.2.1 Apache Camel

Apache Camel[9] is an integration framework, which implements all Enterprise Integration Patterns for easy integration of different applications using the required patterns. We can use Java, Spring XML, Scala, or Groovy. Almost all technologies are available, for example, HTTP, FTP, JPA, RMI, JMS, JMX, LDAP, JMS, EJB, and many more. Apache Camel is also used with Apache ServiceMix, Apache ActiveMQ, and Apache CXF in service-oriented architecture projects.

An Apache Camel can be deployed in a web container like Tomcat, in a JEE Application Server, and as a standalone application and in general as a microservice as well.

The Camel architecture consists of three components − Integration Engine and Router, Processors, and Components. This is illustrated in the following Figure 30:

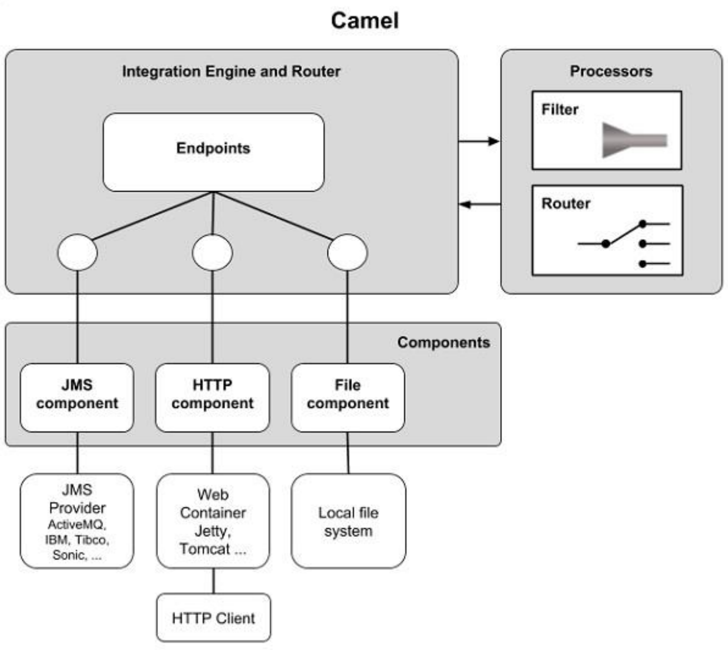


Figure 30 - Main components of Apache Camel

Apache Camel Architecture consists of a Camel Context that contains a collection of Component instances. A Component is a factory of Endpoint instances. We can explicitly configure Component instances in Java code, or they can be auto-discovered using URIs.

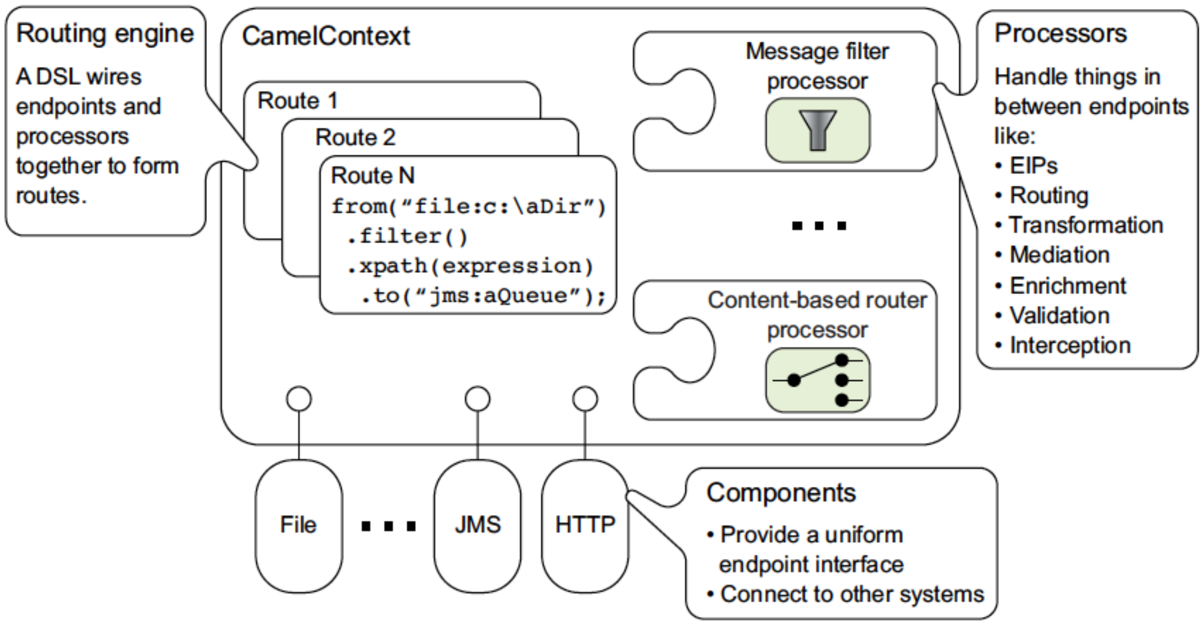


Figure 31 – Camel Context (figure from[9])

An Endpoint is either a URI or URL in a web application or a Destination in a JMS system. We can communicate with an endpoint either by sending messages to it or consuming messages from it. We can then create a Producer or Consumer on an Endpoint to exchange messages with it.

Overall, the architecture of Camel is simple. Camel Context represents the Camel runtime system, and it wires different concepts such as routes, components, or endpoints. Additionally, processors handle routing and transformations between parameters, while endpoints integrate disparate systems.

2.4.2.2 Spring Integration

Spring Integration[10] is an open source framework for enterprise application integration. It is built on top of Spring framework. Hence it is very easy to adopt Spring Integration in the projects which are already using the Spring framework.

In fact, Spring Integration provides an extension of the Spring programming model to support the well known Enterprise Integration Patterns. It enables lightweight messaging within Spring-based applications and supports integration with external systems through declarative adapters. Those adapters provide a higher level of abstraction over Spring's support for remoting, messaging, and scheduling.

As an extension of the Spring programming model, Spring Integration provides a wide variety of configuration options, including annotations, XML with namespace support, XML with generic "bean" elements, and direct usage of the underlying API. That API is based upon well-defined strategy interfaces and non-invasive, delegating adapters.

The basic concepts of a Spring Integration message-driven architecture are: message, message channel and message endpoint (Figure 32):

- A *message* is sent to an *endpoint*
- *Endpoints* are connected among them through *MessageChannels*
- An *endpoint* can receive *messages* from a *MessageChannel*

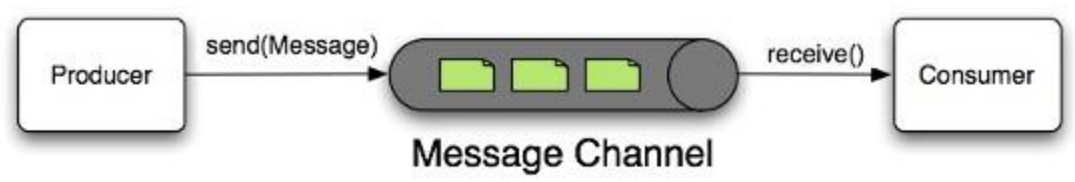


Figure 32 - Spring Integration message-driven architecture (figure from [10])

Below is a list of the available (see Figure 33 as example) message endpoints:

- *Channel adapter*: Connects the application to an external system (unidirectional).
- *Gateway*: Connects the application to an external system (bidirectional).
- *Service Activator*: Can invoke an operation on a service object.
- *Transformer*: Converts the content of a message.
- *Filter*: Determines if a message can continue its way to the output channel.
- *Router*: Decides to which channel the message will be sent.
- *Splitter*: Splits the message in several parts.
- *Aggregator*: Combines several messages into a single one.

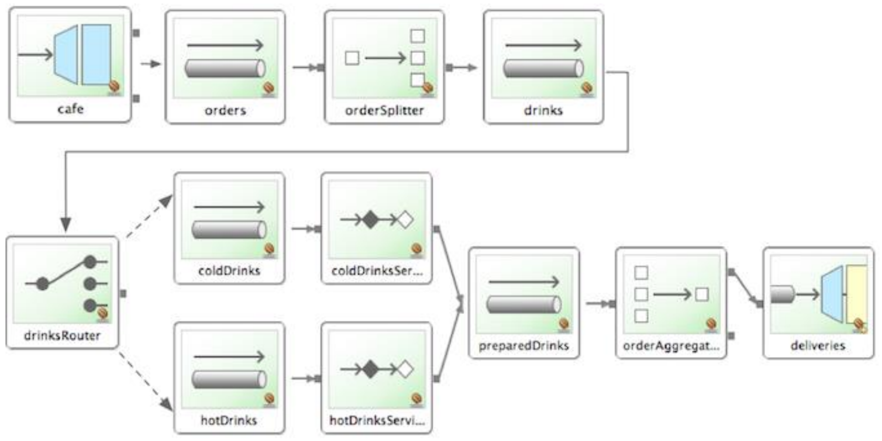


Figure 33 - Spring Integration sample from [10]

2.4.2.3 IDS Data Connector

In International Data Spaces (IDS) Reference Architecture Model (RAM)[11], the Connector is one of main technological building blocks. It is a dedicated software component allowing a Consumer and a provider to exchange, share and process digital content. At the same time, the Connector ensures that the data sovereignty of the Data Owner is always guaranteed.

It is a configurable component, providing several system services enabling secure bidirectional communication, enforcement of content usage policies, system monitoring, and logging of content transactions for clearing purposes. The functional range of a generic Connector may be extended by custom software (Data Apps), allowing data processing, visualization, persistence, etc. The Connector provides metadata to the Data Consumer Connector as specified in the Connector's self-description. For example, technical interface description, authentication mechanism, exposed data sources, and associated data usage. Following the peer-to-peer network concept, the data is transferred between the Connectors of the Data Provider and the Data Consumer (Figure 34).

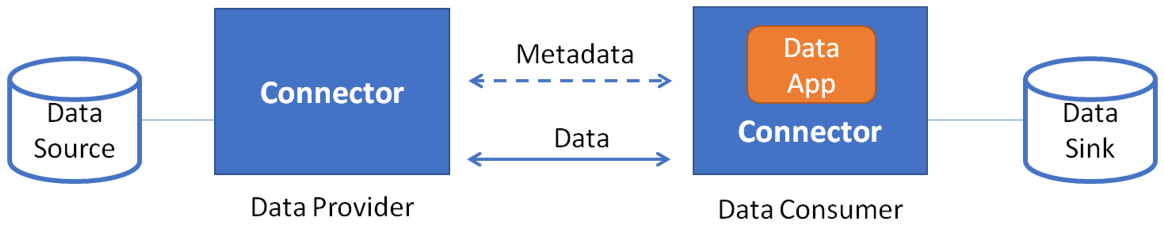


Figure 34 - IDS connector interactions

In the implementation of Service Adapter the open-source IDS connector TRUE (TRUsted Engineering) Connector[23] (Figure 35) will be leveraged in order to fit the specific Service Adapter needs. In particular the trivial Data APP application in order to adapt data on top of the Execution Core Container.

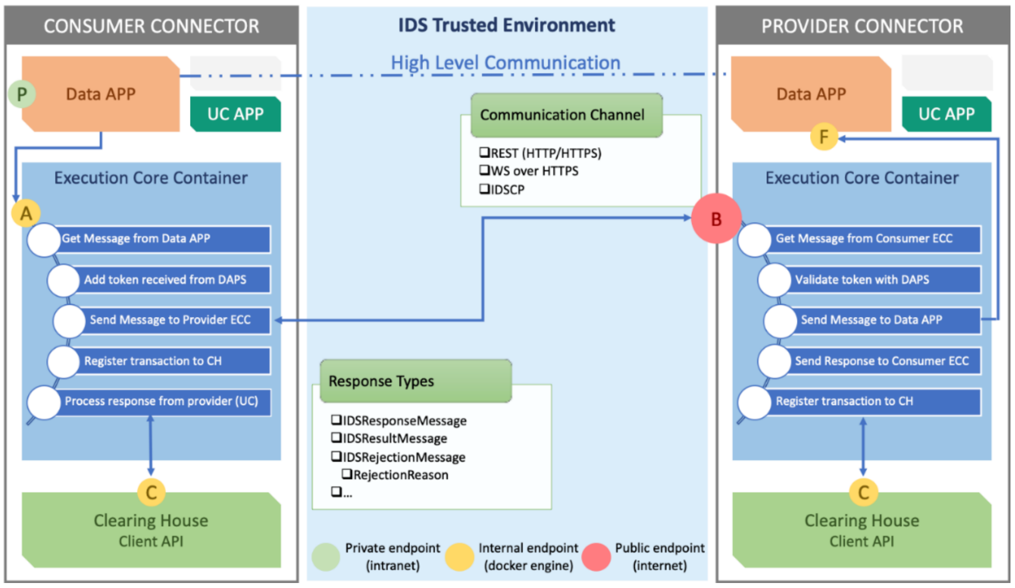


Figure 35 - TRUE Connector Architecture and Interactions

Considering the flows of interactions, the scenario depicted in Figure 35 shows how the Consumer connector accesses data from the Provider connector. Specifically, the Consumer receives a request to its P endpoint, it forwards the request to its internal Execution Core Container (A endpoint) which is in charge of establishing a secure communication with the Provider Execution Core Container to access the data. The Consumer's ECC receives the message from its Data APP, interacts with an external Identity Provider to retrieve the token of the Consumer and send the appropriate IDS message to the Provider's ECC using one of the provided communication channels (B endpoint). The Provider's ECC receives the message and validates the token against the Identity provider, then it retrieves the actual data from its Data APP (F endpoint), and returns it to the Consumer's ECC who, finally, processes the response, applies the usage control policies and forward the data to the original requester. All the transactions are logged into the Clearing House (C endpoints).

[23] https://github.com/Engineering-Research-and-Development/true-connector

3 Conclusions and next steps

This report has documented the update result of activities performed in Task 4.2 " Public & Private sector offerings management tool" taking into account the relations with the other activities involved in the definition and implementation of ACROSS Platform and related components. The report described the architecture of Data harmonization and connectors layer to support service adaptation and how each component is involved in the flow of according to the type of adaptation performed. Service catalogue, related service model, components and functionality have been described in line with the second step evolution. The report provides an update of technologies available to build ad hoc services adapters to be published and used in ACROSS Platform.

The service model has the goal to collect the multi-view of information of a service to be used in ACROSS Platform for the delivery of a user journey service. This model will evolve accordingly to the evolution of ACROSS Platform component, taking into account internal and external feedback and new requirements. This evolution will contribute into the continuous evolution of Service Catalogue functionality and its internal model. The layered implementation of Service Catalogue and the adopted technologies allow the service model evolution with a minimal amount of effort.

In the final phase, we will proceed to finalize the API ecosystem to support the upper components for user journey provisioning and by refining Service Catalogue in order to address the continuous collection of feedbacks provided by the use cases (WP6). Besides, some examples on how “build from scratch” potential service adapters taking into account the baseline technologies described in this report.

4 References

[1] ACROSS, D5.2 - System Architecture & Implementation Plan – Final

[2] Core Public Service Vocabulary Application Profile 3.1.0 (https://joinup.ec.europa.eu/collection/semic-support-centre/solution/core-public-service-vocabulary-application-profile/release/310)

[3] ACROSS, D3.1 - Design of the ACROSS Data Governance framework for data sovereignty – Initial

[4] Regulation (EU) 2018/1724 of the European Parliament and of the Council of 2 October 2018 establishing a single digital gateway to provide access to information, to procedures and to assistance and problem-solving services and amending Regulation (EU) No 1024/2012 (https://eur-lex.europa.eu/eli/reg/2018/1724/oj)

[5] Data Privacy Vocabulary (DPV) https://dpvcg.github.io/dpv/

[6] Service Catalogue source code: https://github.com/OPSILab/Service-Catalogue

[7] APIs for CPSV-AP based Catalogue of Services: https://joinup.ec.europa.eu/sites/default/files/news/2019-09/ISA2_APIs%20for%20CPSV-AP%20based%20Catalogue%20of%20Services.pdf

[8] Usage Control in IDS https://internationaldataspaces.org/wp-content/uploads/dlm_uploads/IDSA-Position-Paper-Usage-Control-in-the-IDS-V3..pdf

[9] APACHE CAMEL: https://camel.apache.org/manual/index.html

[10] SPRING Integration: https://docs.spring.io/spring-integration/docs/6.0.0/reference/pdf/spring-integration-reference.pdf

[11] IDS Reference architecture Model v3 - *https://internationaldataspaces.org/wp-content/uploads/IDS-Reference-Architecture-Model-3.0-2019.pdf*

[12] ACROSS, D6.2 Use Case Evaluation and Impact Assessment -Initial

[13] Enterprise Integration Patterns https://www.enterpriseintegrationpatterns.com/

5 Annex I - ACROSS Service Model

The following sections provides a deeper description of the ACROSS Service Model.

5.1 Service Model class diagram

The following Figure 36 provides the complete class diagram structure of Service Model adopted in Service Catalogue. Each class in the class diagram is described in the following sections

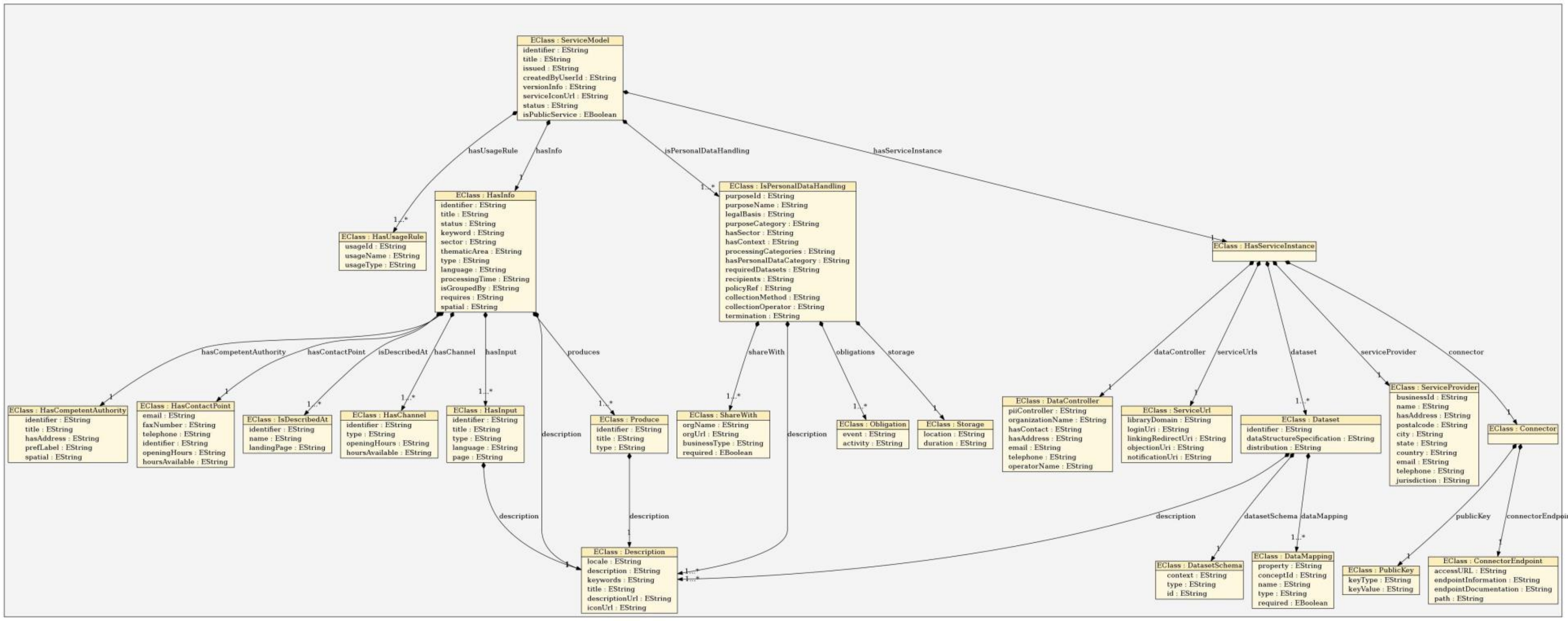


Figure 36 - Complete class diagram of Service Model

5.2 Service Basic Info

Each service to be registered in the Service Catalogue has to provide some basic information.

Table 1 Basic Info properties of Service Model class

Property	Type	Description
title	String (1..1)	Service Name
identifier	String (1..1)	Id of service or service URI if exists. This identifier will be used by the Service Catalogue to identify it and could be the same identifier provided in the information section 5.3.
issued	String (0..1)	When Service entry was created (system log data)
createdByUserId	String (0..1)	User Id (if any)of Service Editor
serviceDescriptionVersion	String (0..1)	Service description version number
serviceIconUrl	String (0..1)	URL pointing to service's icon (if available)
status	String (0..1)	Status of Service ["Completed", "Deprecated", "UnderDevelopment", "WithDrawn"]
isPublicService	Boolean (1..1)	If service is public or not
hasInfo	Object (1..1)	Object describing Service information section. See 5.3
hasServiceInstance	Object (1..1)	Object describing Service information section. See 5.4
isPersonalDataHandling	Object (1..n)	Object describing Personal data handling. See 5.7
hasUsageRule	Object (1..n)	Object describing Service usage Rules. See 5.8

5.3 Service Information section

Each service should provide basic information, identification, service classifications and locale descriptions. Such classifications are only related to public services, according to ISA² Core Public Service Vocabulary Application Profile (CPSV-AP). The following classes and properties belong to ISA² CPSV-AP v2.2.1 [2], and here are reported for completeness of this report.

Table 2 - Service Information Class

Property	Type	Description
Name	String (1..1)	It represents the official Name of the Public Service
Identifier	String (1..1)	This property represents a formally-issued Identifier for the Public Service.
Description	Object (1..1)	This property represents a free text Description of the Public Service. The description is likely to be the text that potential users of the Public Service see in any public service catalogue.

Property	*Type*	*Description*
locale	string	Language used in information, ISO 639-1 coded
description	string	Textual description

Keyword	String (0..n)	This property represents a keyword, term or phrase to describe the Public Service.
Sector	String (0..n)	This property represents the industry or sector a Public Service relates to, or is intended for. For example: environment, safety, housing. Note that a single Public Service may relate to multiple sectors. The possible values for this property are provided as a controlled vocabulary. See section 4 in [2].
Thematic Area	String (0..n)	This property represents the Thematic Area of a Public Service as described in a controlled vocabulary, for instance social protection, health, recreation, culture and religion, family, travelling economic affairs, tax, staff, environment. The recommended controlled vocabularies are listed in

		section 4 in [2].
Type	String (0..n)	This property represents the Type of a Public Service as described in a controlled vocabulary. For the indicating the Type, we are referring to the functions of government to indicate the purpose of a government activity, which the public service is intended for. The recommended controlled vocabularies are listed in section 4 in [2].
Language	String (0..n)	This property represents the language(s) in which the Public Service is available. This could be one language or multiple languages, for instance in countries with more than one official language. The possible values for this property are described in a controlled vocabulary. The recommended controlled vocabularies are listed in section 4 in [2].
Status	String (0..1)	Indicates whether a Public Service is active, inactive, under development etc. according to a controlled vocabulary.
Is Grouped By	String (0..1)	This property links the Public Service to the Event class (section 3.2.25 in [2]). Several Public Services may be associated with a particular Event and, likewise, the same Public Service may be associated with several different Events.
Requires	String (0..n)	One Public Service may require, or in some way make use of, the output of one or several other Public Services. In this case, for a Public Service to be executed, another Public Service must be executed beforehand.
Has Competent Authority	Object (1..1)	This property links a Public Service to a Public Organization, which is the responsible Agent for the delivery of the Public Service. Whether the particular Public Organization provides the public service directly or outsources it is not relevant. The Public Organization that is the Competent Authority of the service is the one that is ultimately responsible for managing and providing the public service. The CPSV-AP reuses the Core Public Organisation Vocabulary that defines the concept of a Public Organization and associated properties and relationships.

Property	***Type***	***Description***
identifier	String	Public Organization identifier
title	String	Name of Public Organization
hasAddress	String	Address

		prefLabel / String / Preferred Label; **spatial** / String / Localization
Has Input	Object (0..n)	The Has Input property links a Public Service to one or more instances of the Evidence class. A specific Public Service may require the presence of certain pieces of Evidence in order to be delivered.
Produces	Object (0..n)	The Produces property links a Public Service to one or more instances of the Output class (see section 3.10 in [2]), describing the actual result of executing a given Public Service. Outputs can be any resource, for instance a document, artefact or anything else being produced as a result of executing the Public Service
Spatial	String	The area covered. The possible values for this property are described in a

Nested table (continued):

prefLabel	String	Preferred Label
spatial	String	Localization

Has Input:

Property	***Type***	***Description***
identifier	String	Identifier (URI if available)
title	String	Assigned Name
type	String	Category of input
language	String	Preferred Label
page	String (0..n)	Documentation
conformsTo	String(0..n)	Reference to characterization of the input. In particular can be linked to the specific dataset defined in section 5.4

Produces:

Property	***Type***	***Description***
identifier	String	Identifier (URI if available)
title	String	Assigned Name
type	String	Category of output

	(0..n)	controlled vocabulary. The recommended controlled vocabularies are listed in section 4 in [2].
Has Contact Point	Object (0..n)	The value of this property, the contact information itself, should be provided using schema:ContactPoint. Note that the contact information should be relevant to the Public Service which may not be the same as contact information for the Competent Authority or any Participant.

Property	***Type***	***Description***
email	String	Email
faxNumber	String	Fax
telephone	String	Telephone number
identifier	String	Id or URI (if available)
openingHours	String	Opening Hours Specification https://schema.org/OpeningHoursSpecification
hoursAvailable	String	Opening Hours restriction

Has Channel	Object (0..n)	This property links the Public Service to any Channel through which an Agent provides, uses or otherwise interacts with the Public Service, such as an online service, phone number or office

Property	***Type***	***Description***
identifier	String	Id or URI (if available)
type	String	Channel type from controlled vocabulary (see section 4 [2])
openingHours	String	Opening Hours Specification
hoursAvailable	String	Opening Hours restriction

Processing Time	String (0..1)	The value of this property is the (estimated) time needed for executing a Public Service. The actual information is provided using the ISO8601 syntax for durations.
Is Described At	Object (0..n)	The property links a Public Service to the Public Service Dataset(s) in which it is being described

Property	*Type*	*Description*
identifier	String	Id or URI (if available)
name	String	Dataset name
landingPage	String	Landing page URL where dataset is published

hasCost	Object (0..n)	The Cost class represents any costs related to the execution of a Public Service that the Agent consuming it needs to pay.

Property	*Type*	*Description*
identifier	String	Id or URI (if available)
code	String	The currency in which the Cost needs to be paid and the value of the Cost is expressed
hasValue	String	A numeric value indicating the amount of the Cost.
description	String[0..n]	A free text description of the Cost
ifaccessed through	String [0..1]	The costs created by the use of different Channels.

5.4 Service Instance

The following classes and properties collects information about the service instances to be used/invoked internally or externally the ACROSS platform. In particular this section collects information about Technical, Service Provider and Data Controller Descriptions of actual instance where service is deployed.

Table 3 - Service Instance class

Property	Type	Description
serviceProvider	Object (1..1)	Object describing service provider:

Property	*Type*	*Description*
businessId	String	Business ID
name	String	Name of service provider
hasAddress	String	Address
postalcode	String	Posta code
city	String	City
state	String	State
country	String	Country
email	String	mail
telephone	String	phone
jurisdition	String	Jurisdition

Property	Type	Description
dataController	Object (1..1)	Object describing Data Controller, the individual or organisation that decides (or controls) the purpose(s) of processing personal data:

Property	*Type*	*Description*
piiController	String	Name of Data Controller
organizationName	String	Organization name.
hasContact	String	Contact Person.
hasAddress	String	Address
email	String	Email Address.
telephone	String	phone

Property	Type	Description
connectorEndpoint	Object (0..1)	Object describing the referenced endpoint to proxy (if any) with a service connector. See 5.5

dataset	Object (1..n)	Object Describing the Service Data Description. See 5.6
serviceUrls	Object (1..1)	Object collects all information to interact with the internal components (e.g. Consent manager...):

Property	***Type***	***Description***
libraryDomain	String	Domain of library to interact with consent manager
loginUri	String	Link to login component.
linkingRedirectUri	String	Service link to interact with consent manager.
objectionUri	String	Service link to interact with consent manager for objection request
notificationUri	String	Service link to interact with consent manager for notification request

5.5 Connector Endpoint

Table 4 - Connector class

Property	Type	Description

endpoint	Object (1..1)	Object describing endpoint for Service Connector: *Property* | *Type* | *Description* **accessUrl** | String | Access URL of an endpoint. **endpointInformation** | String | Endpoint description **endpointDocumentation** | String | URI reference to a documentation of the endpoint, e.g., reference to an OpenAPI-based documentation. **path** | String | Relative path, topic or queue at which the content is published by the related host.
connectorId	String (0..1)	Id of registered connector instance associated to the service endpoint

5.6 Dataset

Table 5 - Dataset class

Property	Type	Description
identifier	String (1..1)	Dataset unique identifier.
datastructureSpecificati	String	URL pointing to further description of the data (e.g. to JSON

on	(0..1)	schema).
dataMapping	Object (1..n)	Array of objects describing the mapping of each data with a personal data concept belonging to a controlled vocabulary: *Property* | *Type* | *Description* **property** | string | Specific property of the datatset (for example a specific field in a form) **conceptId** | string | Reference (if any) to a concept of a personal data ontology (i.e. DPV) **type** | string | Type of data ["Text","Video","Image"," Audio"] **inputType** | string | Type of input data [file type, dropdown list,...] **required** | integer | If that property is required [True=1, False=0] **sensitive** | integer | If this personal data is also sensitive [True=1, False=0]
description	Object (1..n)	Array of localized textual descriptions. *Property* | *Type* | *Description* **locale** | string | Language used in information, ISO 639-1 coded **description** | string | Textual description **keywords** | Array[String] | Keyword tags related to textual des cription.

5.7 Service Personal Data Handling Section

This section It collects the different legal basis and requirements for personal data processing according to EU data protection Rules (Art. 6 GDPR). It describes describe different situations where a company or an organisation is allowed to collect or reuse your personal information: contract, legal obligation, vital interest, public interest, legitimate interest and consent. The following information are used by the consent manager component in the citizen data ownership layer.

Table 6 - Personal Data Handling class

Property	Type	Description
purposeId	String	Purpose's ID, must be unique within the service description
purposeName	String	Human readable Purpose's Name, Short name that identifies the purpose
legalBasis	String	Legal basis in the "processing" of personal Data according to the GDPR: ["Consent", "Contract", "Legal Obligation", "Vital Interest", "Public Interest", "Legitimate Interest"]
purposeCategory	String	Category of purpose from a controlled taxonomy.
hasSector	String	Purposes can be further restricted to specific sectors
hasSector	String	Purposes can be further restricted to specific contexts
processingCategories	String	Category of actions related to a specific purpose and

		from a controlled taxonomy.
description	Object (1..n)	Array of localized description of processing:

Property	Type	Description
locale	string	Language used in information, ISO 639-1 coded
title	string	Title of processing
description	string	Description of the service
descriptionUrl	string	Url of the document describing in detail the processing of personal data
iconUrl	string	Link of icon identifying the type of processing

hasPersonalDataCategory	String (1..n)	Indicates which category of personal data is processed from a controlled taxonomy.
requiredDatasets	String (1..n)	Array listing the required dataset (described previously)
storage	Object (0..n)	Object describing the type of storage:

Property	Type	Description

location	string	Storage Category
duration	string	duration

recipients	String (0..n)	List of type of recipients of personal data processing
shareWith	Object (0..n)	Array of objects describing with whom the consent permits to share data. Organisation identifies the organisation with whom the data is permitted to share:

Property	Type	Description
orgName	string	Organization Name
businessType	string	Business Type: Profit, No-Profit
orgUrl	string	url of organization page
required	boolean	It sets if optional or not

obligations	Object (0..n)	Obligations are the actions to be performed when an event occurs. Obligation defines the obligation related to consent, i.e. has an event and an activity and it defines what action to perform when an event related to consent occurs. For example, when the consent expires (event), then re-solicit consent (activity) or when the consent is revoked (event) then stop processing (activity).

Property	Type	Description
event	string	Event defines the event based on which an activity is

		required to do.
activity	string	Activity defines what activity is required to do when an event occurs.

policyRef	String	Reference to related Privacy Policy
collectionMethod	String	It indicates the method of collection of consent
CollectionOperator	String	It indicates Operator who collects Consents.
termination	String	Termination rule of legal basis under which personal data can be processed.

5.8 Service Data Usage Section

This section provides the reference of one or more usage rules associated to a specific service. The specification of usage rules is defined externally to the service Catalogue.

Table 7 - Data Usage class

Property	Type	Description
usageId	String (1..1)	Usage's ID, must be unique within the service description.
usageName	String (1..1)	Human readable Usage's Name, Short name that identifies the rule.
usageType	String (1..1)	Category of contract agreements from a controlled taxonomy.

5.9 Service Model JSON Schema

In the following the first level of JSON schema adopted by the Service Catalogue in line with the initial Service Model.

```
{
  "title": "ServiceModel",
  "$schema": "http://json-schema.org/draft-04/schema#",
  "type": "object",
  "properties": {
    "title": {
      "type": "string",
      "title": "Service Name",
      "description": "Service Name"
    },
    "identifier": {
      "type": "string",
      "title": "Service Identifier",
      "description": "Service URI if exists"
    },
    "issued": {
      "type": "string",
      "title": "Issued at",
      "description": "Timestamp of the Service creation"
    },
    "createdByUserId": {
      "type": "string",
      "title": "Created by User Id",
      "description": "User Id of Service Editor (e.g. Data Controller)."
    },
    "versionInfo": {
      "type": "string",
      "title": "Service Description version",
      "description": "Service description version number.",
    },
    "serviceIconUrl": {
      "type": "string",
      "title": "Service Icon Url",
      "description": "URL pointing to Service Icon file"
    },
    "status": {
      "type": "string",
      "title": "Service Description status",
      "description": "Status of Service Description (Allowed values: *Completed*, *Deprecated*
, *UnderDevelopment*, *Withdrawn*)",
      "default": "UnderDevelopment",
      "enum": ["Completed", "Deprecated", "UnderDevelopment", "WithDrawn"],
      "options": { "enum_titles": ["Completed", "Deprecated", "Under Development", "Withdrawn"
] }
    },
    "isPublicService": {
      "type": "boolean",
      "title": "Public Service",
      "description": "if public service or not",
      "default": "true"
    },

    "hasInfo": {
      "$ref": "./service-cpsv-entry.json"
    },
    "hasServiceInstance": {
      "$ref": "./service_instance.json"
    },
    "hasUsageRule": {
      "type": "array",
      "title": "Usage Rule",
      "format": "tabs",
      "description": "It collects contract and usage rules for data sharing",
      "items": {
        "$ref": "./usage_rule.json"
      }
    },
    "isPersonalDataHandling": {
      "type": "array",
      "title": "Personal Data Handling",
      "format": "tabs",
      "description": "It collects the different legal basis and requirements for personal data
 processing according to EU data protection Rules (Art. 6 GDPR). It describes describe differe
nt situations where a company or an organisation is allowed to collect or reuse your personal
information: contract, legal obligation, vital interest, public interest, legitimate interest
and consent",
      "items": {
        "$ref": "./isPersonalDataHandling.json"
      }
    }
  }
}
```

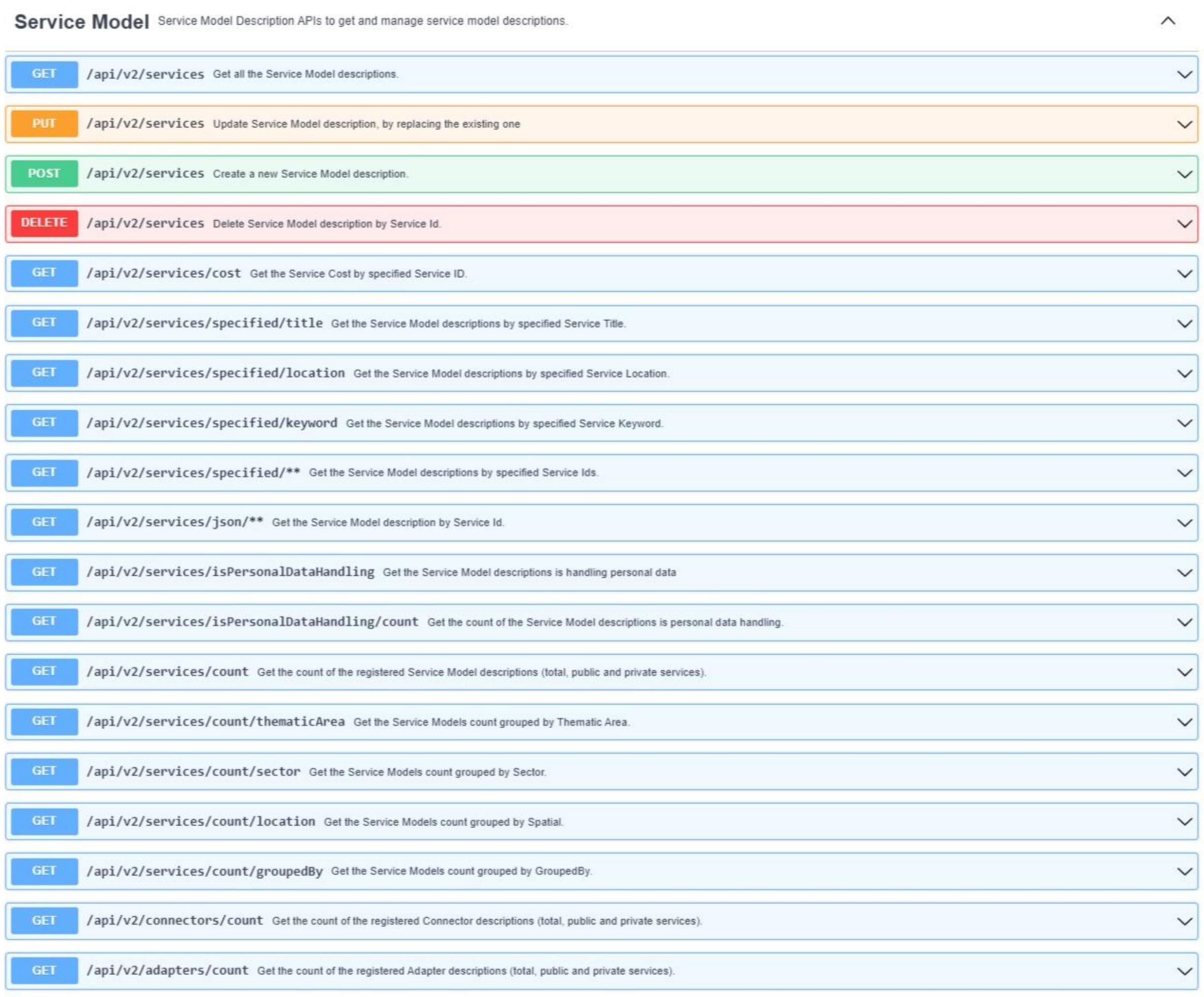
5.10 Service Model JSON-LD Context

The following context definitions are used to export Service Description into the JSON-LD semantic model.

```
{
    "@context": {
        "isTypeOf": "@type",
        "id": "@id",
        "acr": "https://across-h2020.eu/ns/serviceModel",
        "adms": "http://www.w3.org/ns/adms#",
        "cpsv": "http://purl.org/vocab/cpsv#",
        "cv": "http://data.europa.eu/m8g/",
        "dcat": "http://www.w3.org/ns/dcat#",
        "dct": "http://purl.org/dc/terms/",
        "dpv": "https://w3.org/ns/dpv",
        "eli": "http://data.europa.eu/eli/ontology#",
        "foaf": "http://xmlns.com/foaf/0.1/",
        "ids": "https://w3id.org/idsa/core/",
        "locn": "http://www.w3.org/ns/locn#",
        "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
        "schema": "https://schema.org/",
        "skos": "http://www.w3.org/2004/02/skos/core#",
        "xsd": "http://www.w3.org/2001/XMLSchema#",
        "owl": "http://www.w3.org/2002/07/owl",
        "prov": "https://www.w3.org/ns/prov#",
        "vcard": "http://www.w3.org/2006/vcard/ns#",
        "Agent": "dct:Agent",
        "BusinessEvent": "cv:BusinessEvent",
        "Channel": "cv:Channel",
        "Collection": "skos:Collection",
        "Concept": "skos:Concept",
        "ContactPoint": "schema:ContactPoint",
        "Cost": "cv:Cost",
        "CriterionRequirement": "cv:CriterionRequirement",
        "Event": "cv:Event",
        "Evidence": "cv:Evidence",
        "LegalResource": "eli:LegalResource",
        "LifeEvent": "cv:LifeEvent",
        "LinguisticSystem": "dct:LinguisticSystem",
        "Location": "dct:Location",
        "OpeningHoursSpecification": "schema:OpeningHoursSpecification",
        "Output": "cv:Output",
        "Participation": "cv:Participation",
        "PublicOrganisation": "cv:PublicOrganisation",
        "PublicService": "cpsv:PublicService",
        "PublicServiceDataset": "cv:PublicServiceDataset",
        "Rule": "cpsv:Rule",
        "accessURL": {
            "@id": "dcat:accessURL",
            "@type": "@id"
        },
        "endpointInformation": {
            "@id": "ids:endpointInformation",
            "@type": "rdfs:Literal"
        },
        "endpointDocumentation": {
            "@id": "ids:endpointDocumentation",
            "@type": "rdfs:Literal"
        },
        "path": {
            "@id": "ids:path",
            "@type": "rdfs:Literal"
        },
        "createdByUserId": {
            "@id": "acr:createdByUserId",
            "@type": "rdfs:Literal"
        },
        "currency": {
            "@id": "cv:currency",
            "@type": "@id"
        },
        "connector": {
            "@id": "ids:connector",
            "@type": "@id"
        },
        "connectorEndpoint": {
            "@id": "ids:hasDefaultEndpoint",
            "@type": "@id"
        },
        "description": {
            "@id": "dct:description",
            "@type": "rdfs:Literal"
        },
        "email": {
            "@id": "schema:email",
            "@type": "rdfs:Literal"
        },
        "fax": {
            "@id": "schema:faxNumber",
            "@type": "rdfs:Literal"
        },
        "follows": {
            "@id": "cpsv:follows",
            "@type": "@id"
        },
        "follows": {
            "@id": "cpsv:follows",
            "@type": "@id"
        },
        "format": {
            "@id": "dct:format",
            "@type": "@id"
        },
        "hasAddress": {
            "@id": "cv:hasAddress",
            "@type": "@id"
        },
        "hasChannel": {
            "@id": "cv:hasChannel",
            "@type": "@id"
        },
        "hasCompetentAuthority": {
            "@id": "cv:hasCompetentAuthority",
            "@type": "@id"
        },
        "hasContactPoint": {
            "@id": "cv:hasContactPoint",
            "@type": "@id"
        },
        "hasCost": {
            "@id": "cv:hasCost",
            "@type": "@id"
        },
        "hasCriterion": {
            "@id": "cv:hasCriterion",
            "@type": "@id"
        },
        "hasDataController": {
            "@id": "dpv:hasDataController",
            "@type": "@id"
        },
        "hasInput": {
            "@id": "cpsv:hasInput",
            "@type": "@id"
        },
        "hasInfo": {
            "@id": "acr:hasInfo",
            "@type": "@id"
        },
        "hasLegalResource": {
            "@id": "cv:hasLegalResource",
            "@type": "@id"
        },
        "hasPart": {
            "@id": "dct:hasPart",
            "@type": "@id"
        },
        "hasParticipation": {
            "@id": "cv:hasParticipation",
            "@type": "@id"
        },
        "hasServiceInstance": {
            "@id": "acr:hasServiceInstance",
            "@type": "@id"
        },
        "hoursAvailable": {
            "@id": "schema:hoursAvailable",
            "@type": "@id"
        },
        "iconUrl": {
            "@id": "schema:url",
            "@type": "rdfs:Literal"
        },
        "identifier": {
            "@id": "dct:identifier",
            "@type": "rdfs:Literal"
        },
        "ifAccessedThrough": {
            "@id": "cv:ifAccessedThrough",
            "@type": "@id"
        },
        "implements": {
            "@id": "cpsv:implements",
            "@type": "@id"
        },
        "isClassifiedBy": {
            "@id": "cv:isClassifiedBy",
            "@type": "@id"
        },
        "isDefinedBy": {
            "@id": "cv:isDefinedBy",
            "@type": "@id"
        },
        "isDescribedAt": {
            "@id": "cv:isDescribedAt",
            "@type": "@id"
        },
        "isGroupedBy": {
            "@id": "cv:isGroupedBy",
            "@type": "@id"
        },
        "isPublicService": {
            "@id": "acr:isPublicService",
            "@type": "xsd:boolean"
        },
        "issued": {
            "@id": "dct:issued",
            "@type": "rdfs:Literal"
        },
        "keyword": {
            "@id": "dcat:keyword",
            "@type": "rdfs:Literal"
        },
        "publicKey": {
            "@id": "ids:publicKey",
            "@type": "@id"
        },
        "keyType": {
            "@id": "ids:keyType",
            "@type": "rdfs:Literal"
        },
        "keyValue": {
            "@id": "ids:keyValue",
            "@type": "rdfs:Literal"
        },
        "landingPage": {
            "@id": "dcat:landingPage",
            "@type": "@id"
        },
        "language": {
            "@id": "dct:language",
            "@type": "@id"
        },
        "libraryDomain": {
            "@id": "dct:identifier",
            "@type": "rdfs:Literal"
        },
        "loginUri": {
            "@id": "dct:identifier",
            "@type": "rdfs:Literal"
        },
        "linkingRedirectUri": {
            "@id": "dct:identifier",
            "@type": "rdfs:Literal"
        },
        "objectionUri": {
            "@id": "dct:identifier",
            "@type": "rdfs:Literal"
        },
        "notificationUri": {
            "@id": "dct:identifier",
            "@type": "rdfs:Literal"
        },
        "member": {
            "@id": "skos:member",
            "@type": "@id"
        },
        "onBehalf" : {
            "@id": "prov:actedOnBehalfOf",
            "@type": "@id"
        },
        "openingHours": {
            "@id": "schema:openingHours",
            "@type": "rdfs:Literal"
        },
        "operatorName": {
            "@id": "acr:operatorName",
            "@type": "rdfs:Literal"
        },
        "organizationName": {
            "@id": "vcard:organization-name",
            "@type": "rdfs:Literal"
        },
        "ownedBy": {
            "@id": "cv:ownedBy",
            "@type": "@id"
        },
        "page": {
            "@id": "foaf:page",
            "@type": "@id"
        },
        "playsRole": {
            "@id": "cv:playsRole",
            "@type": "@id"
        },
        "prefLabel": {
            "@id": "skos:prefLabel",
            "@type": "rdfs:Literal"
        },
        "processingTime": {
            "@id": "cv:processingTime",
            "@type": "rdfs:Literal"
        },
        "produces": {
            "@id": "cpsv:produces",
            "@type": "@id"
        },
        "publisher": {
            "@id": "dct:publisher",
            "@type": "@id"
        },
        "related": {
            "@id": "dct:relation",
            "@type": "@id"
        },
        "requires": {
            "@id": "dct:requires",
            "@type": "@id"
        },
        "role": {
            "@id": "cv:role",
            "@type": "@id"
        },
        "sector": {
            "@id": "cv:sector",
            "@type": "@id"
        },
        "serviceProvider": {
            "@id": "acr:serviceProvider",
            "@type": "@id"
        },
        "serviceUrls": {
            "@id": "acr:serviceUrls",
            "@type": "@id"
        },
        "businessId": {
                "@id": "dct:identifier",
                "@type": "rdfs:Literal"
            },
            "name": {
                "@id": "foaf:name",
                "@type": "rdfs:Literal"
            },
            "postalcode": {
                "@id": "schema:postalCode",
                "@type": "rdfs:Literal"
            },
            "city": {
                "@id": "vcard:locality",
                "@type": "rdfs:Literal"
            },
            "state": {
                "@id": "vcard:region",
                "@type": "rdfs:Literal"
            },
            "country": {
                "@id": "vcard:country-name",
                "@type": "rdfs:Literal"
            },
            "jurisdiction": {
                "@id": "eli:jurisdiction",
                "@type": "rdfs:Literal"
            },
            "spatial": {
                "@id": "dct:spatial",
                "@type": "@id"
            },
            "status": {
                "@id": "adms:status",
                "@type": "@id"
            },
            "telephone": {
                "@id": "schema:telephone",
                "@type": "rdfs:Literal"
            },
            "thematicArea": {
                "@id": "cv:thematicArea",
                "@type": "@id"
            },
            "title": {
                "@id": "dct:title",
                "@type": "rdfs:Literal"
            },
            "type": {
                "@id": "dct:type",
                "@type": "@id"
            },
            "value": {
                "@id": "cv:value",
                "@type": "xsd:double"
            },
            "versionInfo": {
                "@id": "owv:versionInfo",
                "@type": "http://www.w3.org/2001/XMLSchema#string"
            }
        }
    }
```

6 Annex II - Service Catalogue APIs

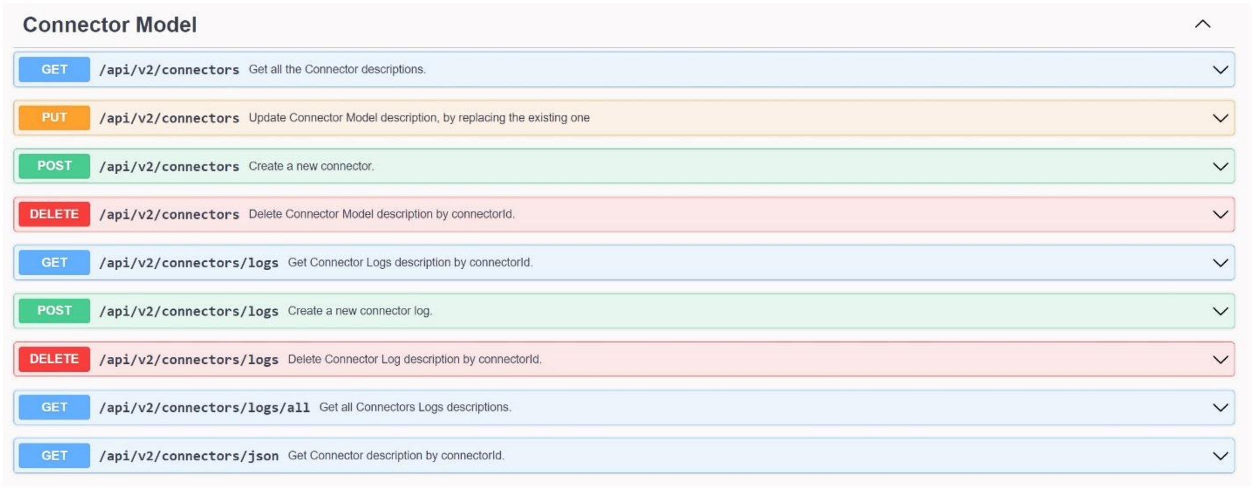
Several Swagger-UI screenshot are reported below, in order to summarize the current APIs exposed by the Service Catalogue for the management of Service descriptions (Figure 37) and related connectors (Figure 38) and adapters (Figure 39).



Service Model Service Model Description APIs to get and manage service model descriptions.

GET /api/v2/services Get all the Service Model descriptions.
PUT /api/v2/services Update Service Model description, by replacing the existing one
POST /api/v2/services Create a new Service Model description.
DELETE /api/v2/services Delete Service Model description by Service Id.
GET /api/v2/services/cost Get the Service Cost by specified Service ID.
GET /api/v2/services/specified/title Get the Service Model descriptions by specified Service Title.
GET /api/v2/services/specified/location Get the Service Model descriptions by specified Service Location.
GET /api/v2/services/specified/keyword Get the Service Model descriptions by specified Service Keyword.
GET /api/v2/services/specified/** Get the Service Model descriptions by specified Service Ids.
GET /api/v2/services/json/** Get the Service Model description by Service Id.
GET /api/v2/services/isPersonalDataHandling Get the Service Model descriptions is handling personal data
GET /api/v2/services/isPersonalDataHandling/count Get the count of the Service Model descriptions is personal data handling.
GET /api/v2/services/count Get the count of the registered Service Model descriptions (total, public and private services).
GET /api/v2/services/count/thematicArea Get the Service Models count grouped by Thematic Area.
GET /api/v2/services/count/sector Get the Service Models count grouped by Sector.
GET /api/v2/services/count/location Get the Service Models count grouped by Spatial.
GET /api/v2/services/count/groupedBy Get the Service Models count grouped by GroupedBy.
GET /api/v2/connectors/count Get the count of the registered Connector descriptions (total, public and private services).
GET /api/v2/adapters/count Get the count of the registered Adapter descriptions (total, public and private services).

Figure 37 - Documentation of the API of Service Catalogue (Service Model)



Connector Model

GET /api/v2/connectors Get all the Connector descriptions.

PUT /api/v2/connectors Update Connector Model description, by replacing the existing one

POST /api/v2/connectors Create a new connector.

DELETE /api/v2/connectors Delete Connector Model description by connectorId.

GET /api/v2/connectors/logs Get Connector Logs description by connectorId.

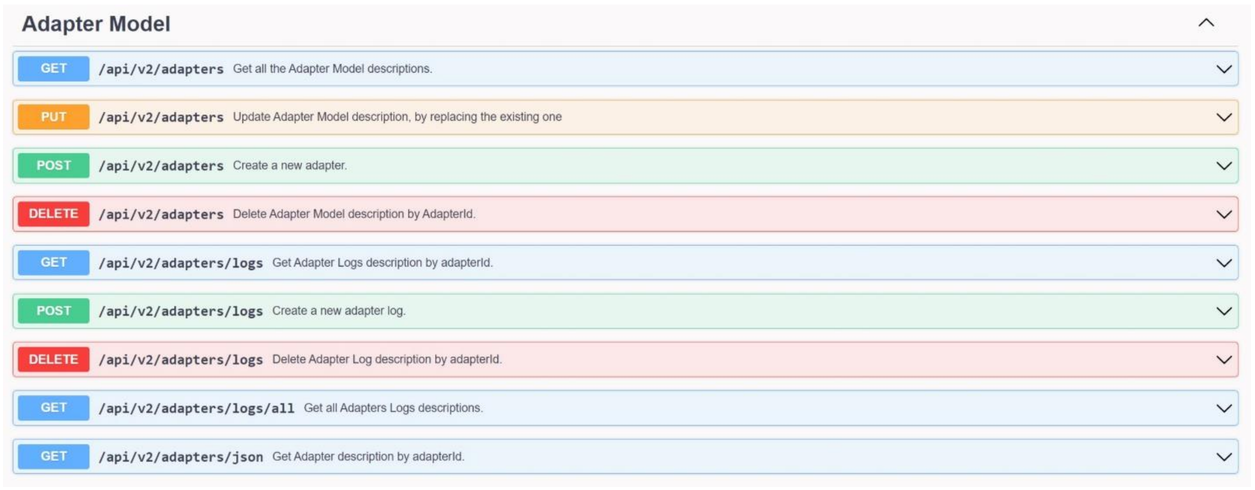
POST /api/v2/connectors/logs Create a new connector log.

DELETE /api/v2/connectors/logs Delete Connector Log description by connectorId.

GET /api/v2/connectors/logs/all Get all Connectors Logs descriptions.

GET /api/v2/connectors/json Get Connector description by connectorId.

Figure 38 - Documentation of the API of Service Catalogue (Connector Model)



Adapter Model

GET /api/v2/adapters Get all the Adapter Model descriptions.

PUT /api/v2/adapters Update Adapter Model description, by replacing the existing one

POST /api/v2/adapters Create a new adapter.

DELETE /api/v2/adapters Delete Adapter Model description by AdapterId.

GET /api/v2/adapters/logs Get Adapter Logs description by adapterId.

POST /api/v2/adapters/logs Create a new adapter log.

DELETE /api/v2/adapters/logs Delete Adapter Log description by adapterId.

GET /api/v2/adapters/logs/all Get all Adapters Logs descriptions.

GET /api/v2/adapters/json Get Adapter description by adapterId.

Figure 39 - Documentation of the API of Service Catalogue (Adapter Model)

Service Catalogue APIs are protected by the Keycloak Oauth2 authorization server. An external client application/service that wants to interact with Service Catalogue by using the APIs, must perform one of the available OAuth2 flows (Authorization Code, Client Credentials and Password grants) against the Keycloak IdM, in order to get an Access Token and then use it in the API requests.

7 Annex III - ACROSS Requirements Mapping

Table 8 - Functional and Technical Requirement

Id	Title	Description	Type	Category
Req_01	Semantic and technical interoperability with SDG	The system should ensure an alignment of semantic and technical interoperability with SDG IT Tools	non functional	Platform architecture and interoperability
Req_09	Free access to other countries' e-services	As a user I want to be able to access other countries' services	non functional	Platform architecture and interoperability
Req_13	Interoperability with legacy systems	It has to be possible to connect the ACROSS platform with the existent PA legacy systems (e.g. databases, web services). Secure and reliable communication with the existing public administration information systems have to be provided without requiring changes in these systems. The platform should also provide tools and predefined components to facilitate the interoperability.	non functional	Platform architecture and interoperability
Req_15	Easy to use service integration and orchestration tools	In order to create cross border services the platform has to support Public and Private providing a set of tools and applications that will help them to easily implement service integration.	functional	Connectors to integrate the private and public sector offering
Req_16	Open API access	Data and services available in the ACROSS platform have to be accessible via a set of APIs using standardized approaches(e.g. RESTful API).	functional	Platform architecture and interoperability

Req_17	Service Registries	ACROSS platform has to maintain registries of all available services offered by different PAs, SMEs and by the platform itself. Every service should be well-described using standard metamodels	functional	Connectors to integrate the private and public sector offering
Req_18	Cross Border Authentication	The services deployed and executed in ACROSS platform should have the possibility to be integrated, if needed, with eIDAS system. The platform can optionally support single-sign-on mechanism to simplify authentication on multiple applications and services internally to the platform.	functional	Security and Privacy
Req_19	Reliability and Integrity	The implementation of ACROSS should follow open standards and use well-known and widely accepted technologies in order to ensure integrity. The ACROSS platform has to be reliable assuring integrity of the components/tools that are part of it.	non functional	Platform architecture and interoperability
Req_20	Security access	Access to services and data has to be available to authorized users/applications only. Only audited applications are allowed to be deployed to ensure compliance with the security policies. Every security violation should be reported and the necessary actions to protect information and applications present in the platform has to be performed.	functional	Security and Privacy
Req_27	Catalogue of services (public/private) data model	The Catalogue of services data model will follow the common public core vocabularies coming from ISA2 and the EIF implementation regulation and will support interoperability with SDG	functional	Platform architecture and interoperability

Req_28	Catalogue of services (public/private) objective	The catalogue of services will take care of harmonisation of the private and public services and related data enabling semantic interoperability and supporting the selected common vocabularies should be used to express the metadata.	functional	Platform architecture and interoperability
Req_29	No vendor lock-in	I want the ACROSS reference architecture to be technologically agnostic to avoid vendor lock-in.	non functional	Platform architecture and interoperability
Req_30	Open source	I want the ACROSS reference architecture to reuse already available open source solutions and only create or improve those aspects that are not covered by the existing solutions	non functional	Platform architecture and interoperability
Req_35	Usability and adaptability	The provided solutions in the platform should be user-friendly and easy to use and should be multilingual. No piece of text that might be displayed to a user shall reside in source code and solution and user should be able to select the preferred language . The implementation of the system should follow open standards and use well-known and widely accepted technologies in order to ensure ease of use.	non functional	Platform architecture and interoperability
Req_36	Minimal browser support.	The component user interface (where available e.g. dashboards, forms, etc...) should provide support for the wide range of widely used browsers.	non functional	Web&Mobile applications

The following table provides additional requirements and recommendations that are/will addressed by the Service Catalogue, selected from the user requirements coming from the initial use case evaluation [12] also reported in [1]:

Table 9 - User Requirements and Recommendations from [12]

No.	Title	Description
Req_3	Tutorials & examples tool	As user I want a place where I can access examples on how to perform services, fill in forms, and access other relevant information on services depending on the country.
Req_4	Information tool	As user I want a place where general information on migration to other countries is stored and constantly updated. It must be written in simple and understandable language.
Req_5	Connections to outer sources	As user I want to be able to view relevant informational links – national platforms, suggestions on job search portals, housing market, education portals, etc. while I access services.
Rec_C	Integrated application forms' features must be extended and standardized	As I service provider that I would like to integrate an application form in ACROSS platform, I want to give the same experience to the end user as with the original application form
Rec_L	Facilitating trust in the process and product	It has to include official contacts of each service owner and possibly their social media accounts for ability to reach out for support (contacting a human being, not an AI solution).
Rec_M	Reusability of Components and Technologies	As a developer I would like to download the components developed for ACROSS (like Transparency Dashboard, User Journey Engine, eIDAS proxy, ect) well documented and ready-to-use by other platforms.
Rec_N	Estimated processing time for applications done in ACROSS	As a user, I would like to have an overview of the expected processing time by the authorities for my applications, in order to simplify my time planning for the preparation of my stay abroad.
Rec_O	Overview about all costs and fees for administrative services	As a user, I would like to have an overview of the expected direct and upcoming costs, especially for mandatory official notices from authorities, in order to be able to plan my stay abroad financially.