EJP RD
European Joint Programme on Rare Diseases

H2020-SC1-2018-Single-Stage-RTD
SC1-BHC-04-2018
Rare Disease European Joint Programme Cofund

Grant agreement number 825575

AD36

Virtual Platform Specification (VIPS)

Organisation name of lead beneficiary for this deliverable:
Partner 03 - AIT

Due date of deliverable: month 30

Dissemination level:
Public

Table of contents

Disclaimer

This document describes the first version of the virtual platform, laying out architectural principles, identifying stakeholders, core components and standards. This document is an alpha version, meaning that components may change, new components may be added or removed in upcoming versions that extend or modify the core functionalities of the virtual platform.

Document history

Date	Author	Changes
05-03-2021	Marco Roos	Document review
24-03-2021	Karl Kreiner	Added document history table
07-04-2021	Karl Kreiner	Updated Introduction, Purpose of document and audience
10-04-2021	Anthony Brookes	Updates on Introduction and scope of document; document review
21-04-2021	Karl Kreiner	Incorporated changes from Anthony Brookes
06-05-2021	Karl Kreiner	Re-structured non-functional requirements section;
11-05-2021	Rajaram Kaliyaperumal	Text added for metadata section
02-06-2021	Karl Kreiner	Document clean up
11-06-2021	Günter Schreier	Updates on Introduction and scope of document
17-06-2021	Rajaram Kaliyaperumal, Karl Kreiner	Re-structured data standards section
18-06-2021	Mark Wilkinson	Added text to Resource FAIRness assessment service
18-06-2021	Rajaram Kaliyaperumal	Text added for SPARQL and Metadata provider service sections
18-06-2021	Ronald Cornet	Updated section on common data representations
23-06-2021	Nuria Queralt Rosinach	Updates on common data models
24-06-2021	Nirupama Benis	Added text to 7.5.7.2. Mappings
28-06-2021	Karl Kreiner	Updated overall architecture section
29-06-2021	Karl Kreiner	Shortened and restructured document
06-07-2021	David Reinert	Updates on VP API Index
08-08-2021	Günter Schreier	Review
11-07-2021	Anthony Brookes	Full review, corrections and additions
14-07-2021	Marco Roos	Document review and inclusion of semantic components in architecture
15-07-2021	Heimo Müller	Updates on Authentication and Authorization
16-07-2021	Rajaram Kaliyaperumal	Updates on CDM
16-07-2021	Karl Kreiner	Updates on architecture diagram
28-07-2021	Ana Rath	Review
26-08-2021	Franz Schäfer	Review
02-09-2021	Karl Kreiner	Final version
10-09-2021	Karl Kreiner	Added disclaimer

1. Introduction

1.1. What is the virtual platform?

The virtual platform (VP) is a service-oriented eco-system of inter-linked web-services designed to foster and facilitate research in the Rare Disease (RD) domain. The primary objective of the VP is to provide researchers with a unified way to access RD resources such as registries, biobanks, data repositories and catalogues. The VP is distributed and federated by design, meaning that most services are provided from multiple remote European locations rather than a central location. Central components to underpin the VP will be kept to a minimum. Services include but are not limited to:

1. Authentication and Authorization
2. Services for discovery and elaboration of patients, data, samples and information resources
3. Services to request and access data in a responsible manner
4. Services for dataset enhancement, such as pseudonymization, linkage, patient registration and allied consent management
5. Services for resources to be FAIR 'at source' for federated computational use, such as catalogues of ontological models representing rare disease data and metadata.

The VP itself does not comprise, provide or directly support the establishment of the various services, but simply specifies and offers technical principles, standards, requirements, software, services and tools that the community can adopt or base their own developments upon to inter-operate with the VP.

1.2. VP use case examples

One design principle of the virtual platform is use case driven design as described in chapter 3.2. Use cases are identified in collaboration with RD stakeholders to serve as paradigms for foreseen and unforeseen use cases that the VP will need to support. They are translated to requirements which then are mapped to components and services provided by the virtual platform. Use cases that have been identified can be categorized into various themes including the count of patients and epidemiology (e.g. what is the number of patients with a particular disease; how many patients are affected by *Osteogenesis imperfecta*; what are all patient registries for a particular country?), education and information (what are the guidelines/literature on and support for translation research? Which are the educational resources and trainings to standardize research data?), FAIRness (automatically find *and use* data and metadata across VP compliant catalogues and data sources, how do I design my data collection in a sustainable and reusable way that also computers understand), and complex research questions in the area of data analysis (e.g. clustering of patients by similarities in molecular subnetworks across distributed databases in Europe to uncover treatable phenotypes). The aforementioned services of the VP either directly support the use cases (e.g. services to request and access data in a responsible manner) or support a use case (e.g. authentication and authorization infrastructure when performing a record-level query), or support interoperability between resources.

1.3. Scope and audience of this document

The Virtual Platform (VP) Specification (VIPS) aims to provide RD stakeholders (see chapter 1.5) with resources relevant to RD research with guidance on possibilities of how their resource can become part of the VP. It depicts the overall structure of the VP (architecture, design and development principles), the use case design cycle supported by the VP, the components required to link resources to the VP and the agreed guidelines and standards. Its level of detail aims at the IT managerial level, not at the implementation level. For more granular details, links to relevant documents are provided.

1.4. Key abstractions of the virtual platform

Services mentioned in the introduction can be decomposed into a number of elements which constitute the foundation of the virtual platform. The following table provides an overview on key abstractions used in the virtual platform and this document:

Table 1: Key abstractions of the virtual platform

Term	Description	Example
Resource	A resource represents a patient registry, biobank, catalogue or data repositories. This list is non-taxative and continuously extended and refined.	A rare disease clinical registry.
Platform-wide service	A platform-wide VP service is a software element that facilitates the findability, accessibility, interoperability, and reusability of resources participating in the VP.	Authentication and authorization service provided to RD registry to enable a single-sign-on user experience.
Resource-level component	A resource-level component is a software element installed at the resource site to ensure findability, accessibility, interoperability and reusability from a virtual platform point of view.	A component translating record-level and resource-level queries to the data format understood by a given RD registry.
VP auxiliary component or service	A VP auxiliary component or service is a software element or process that supports the maintenance of quality, sustainability and FAIRification of both VP components and resources participating in the virtual platform.	A resource FAIRness assessment service as described in 5.3.1, online questionnaires to assess quality and sustainability of software tools following the guidelines described in appendix C.
VP Application Programming Interface (API)	To facilitate findability, accessibility, interoperability and reusability of resources, resource-level components and platform-wide services expose a standardized resource	API endpoint to retrieve the meta data description (expressed through the EJP-RD meta data model); API endpoint to retrieve access tokens through

Term	Description	Example
	application programming interface (API).	the authentication and authorization infrastructure

1.5. Stakeholders of the virtual platform

Chapter 1.3 introduced the notion of RD stakeholders. From a perspective of the virtual platform, the group of stakeholders can be broken down further as following table illustrates:

Table 2: Key abstractions of the Virtual Platform

Stakeholder	Description
Rare Disease Researcher (RD Researcher)	A RD researcher is the primary end-user of the virtual platform. RD researchers are interested in use cases as outlined in chapter VP use case examples.
Resource owner	A resource owner is responsible for the setup and maintenance of a RD resource, e.g. a registry, a biobank, a guideline or ontology.
Software provider	A software provider is either an EJP RD project partner, industry vendor or other organization that is providing open-source or commercial software that support resources or components of the virtual platform.
Infrastructure provider	An infrastructure provider is responsible for providing the technological infrastructure to run either software required by resources or VP resource-level components and platform-wide services itself.
Software developer	A software developer involved in the development of software for resources, resource-level components or platform-wide services of the virtual platform.
FAIR data steward	A FAIR data steward supports resource owners in the FAIRification process of their resources.

2. VP Architecture and overview of components

Using the VP services and following the introduced key abstractions of the virtual platform, resource-level components and platform-wide services can be organized into following logical architecture:

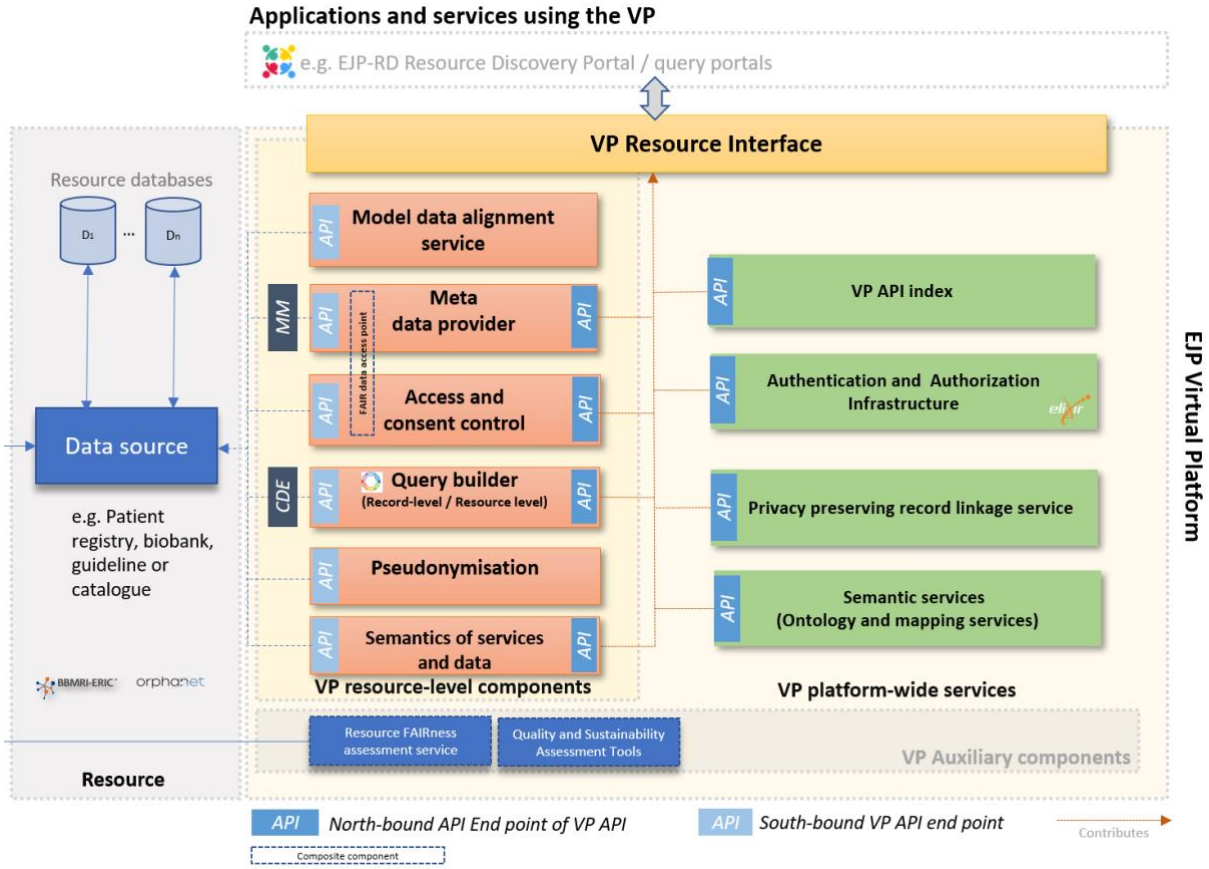

Figure 1: VP overall architecture

The virtual platform consists of resource-level components, platform-wide services and auxiliary services and components. VP Resource-level components expose an API to the outer world to allow interaction with the underlying resource, including components to expose meta data about the resource (depicted as meta data model - MM in figure 1), interfaces for access control, interfaces for querying data of the resource and interfaces for applying consent control rules on the resource. Resource-level components are operated at the resource's site. As indicated in figure 1, resource-level components may also be represented as composite components, components that orchestrate 2 or more components. A FAIR data point (FDP) is a composite component that contains, and gives access to, machine readable metadata about access restrictions, consent control, the types of data the resource contains, and the protocols by which data can be accessed. This metadata is used to orchestrate finding and using the resource, also in dynamically automated scenarios.

Platform-wide VP services include the VP API and fair data point index, providing a list of resources and API endpoints accessible through the VP, the authentication and authorization infrastructure (AAI), patient linkage services enabling resources and/or applications to perform privacy-preserving record and data linkage, and semantic services that provide the ontological reference models (e.g. the DCAT-based EJP RD metadata model, and ontological CDE model), and mappings to other representations and terminologies (e.g. CDSIK, FHIR, OMOP). Platform-wide VP services require a dedicated hosting infrastructure to be accessible.

Auxiliary VP services are technically not required to operate the VP but provide additional added-value services to resources, such as a resource FAIRness assessment service described in chapter 5.3.1
The VP resource interface is comprised of access to semantic services and the VP resource API as a collection of north and southbound API end points (see chapter Virtual platform API for more details) provided by resource-level components and platform-wide VP components. The VP resource API is exposed to applications and services - e.g., to a EJP RD resource discovery portal - that enables end-users to interact with the VP.
The concept of resource-level components and platform-wide services is an abstract concept in the sense, that there is no 1:1 mapping to a specific software tool, library or service. Instead, components are services that manifest themselves in reference implementations as detailed in chapter 3.3.

2.1. System boundaries

The virtual platform is comprised of resource-level components, platform-wide services and auxiliary components and services. Applications and outside services, such as the EJP RD resource discovery portal illustrated in figure 1 are technically consumers and therefore not part of the virtual platform. Resources are made accessible to the virtual platform through both, resource-level components and platform-wide services. As such, they are both consumers (by making use of VP functionality) and producers (by making data available in the ecosystem) but not part of the virtual platform.

2.2. Overview on components

The following table provides a short description of all components, their level of maturity (see chapter 3.4 for more details on the concept of maturity), a categorization with respect to where the component can be found (resource-level/platform-wide):

Table 3 : Overview on components provided by the virtual platform

Component	Short description	Chapter	Component/ Service type	Level of maturity
VP Index	A component providing a list of resources available through the virtual platform	5.2.1	Platform-wide	Trial-use
Authentication and authorization services	Authentication and authorization for VP components and RD resources	5.2.2	Platform-wide	Trial Use

Component	Short description	Chapter	Component/ Service type	Level of maturity
Record linkage services	Services that enable resources to link records (e.g., patients) and patient pseudonyms in a privacy-preserving way	5.2.3	Platform-wide	Draft
Access and consent control	A component supporting methods of authentication, authorization and expression of consent rules	5.1.3	Resource-level	Draft
Meta data provider	A component exposing standardised metadata about the resource at resource-level	5.1.1	Resource-level	Trial Use
Data model alignment service	A component to translate the representation of data from one data model to another	5.1.2	Resource-level	Draft
Resource (metadata) discovery service	A component enabling querying of resource-level metadata	5.1.4	Resource-level	Trial Use
Record (patient and sample data) discovery service	A component enabling safe querying of row-level data	5.1.4	Resource-level	Normative
Pseudonymisation	A component providing resources with the possibility to create patient pseudonyms	5.1.5	Resource-level	Trial Use
Semantics of data and services	Ontological models that describe data in a resource and metadata about a resource for machines and humans.	4.1.2, 4.2.1.2	Resource-level	Trial Use
Semantics services	Repositories of ontological models and mappings (Utility services)	4	Platform-wide	Normative

3. Design Principles and non-functional requirements

The VP will constitute the underpinning of a service-oriented, distributed and federated approach to creating more utility for the RD field following a use case driven design process, supported by a process to ensure sustainability and quality of VP components and services.

3.1. Federated and distributed approach

A design principle of the virtual platform is a distributed as well as a federated architecture approach which is reflected in the overall architecture through resource-level components outweighing the number of platform-wide components. This approach is taken with 2 goals in mind: first, to adhere with the requirements of the General Data Protection Regulation (GDPR) from a resource perspective (avoidance of moving data to centralized locations, where possible) and second, to foster sustainability of the individual VP components.

As a principle, data is not centralized in the virtual platform and stays at the site of the resource owner. Resource-level components support resource owners in making their data findable, accessible, interoperable, and reusable in a controlled manner within the RD research community. Resources that work with the VP are self-describing for humans (e.g., via a web site), for engineers and bioinformaticians (e.g., via a query API), and for machines (e.g., via ontological data and metadata). Some platform-wide services supplement resource-level components (e.g., reference ontologies contain additional semantic relations between data elements in different resources). When it comes to the discovery of data, the VP platform enforces a federated approach to both querying resource level and record level data.

The overall architecture foresees a limited number of platform-wide components though, such as components for authentication and authorization and components for record and data linkage. To increase the sustainability of platform-wide components the concept of reference implementations is introduced in chapter 3.3 following the idea that components of the VP constitute the structural design materializing in one or more concrete implementations. Another measure to increase the sustainability of platform-wide resources is to implement federations where applicable. For instance, an authentication and authorization infrastructure within the VP might federate the authentication process to another AAI. Thus, the authentication process can be orchestrated between multiple participating infrastructures.

3.2. Use case driven design

A core design principle of the virtual platform is use case driven design. Use cases are identified and described in a non-technical way in collaboration with stakeholders and end-users of the virtual platform (e.g., the European Reference Networks).

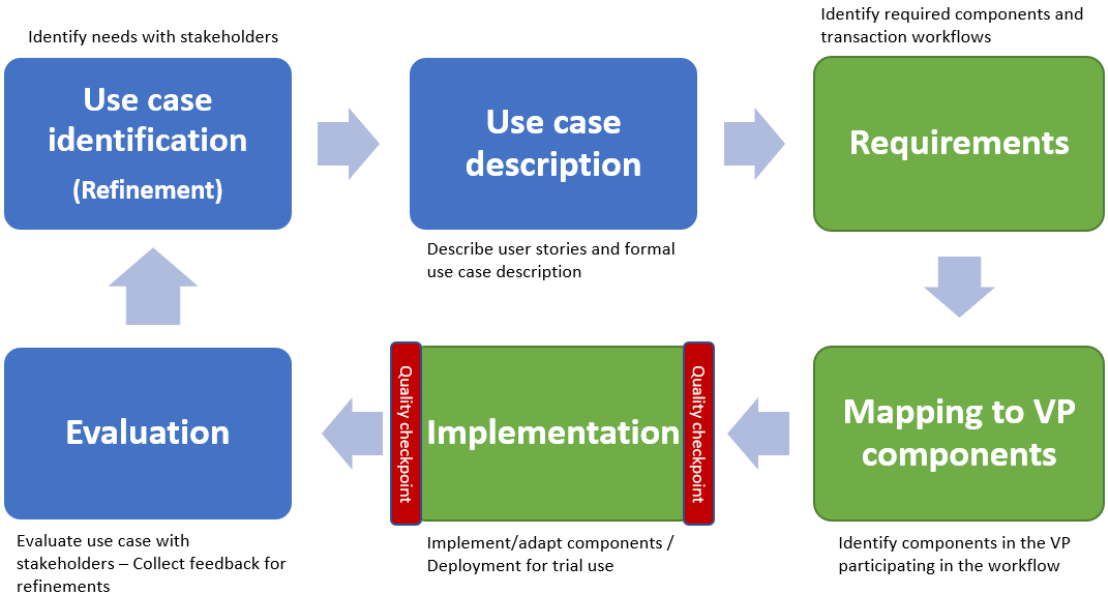

Figure 2: Iterative use case driven design of the VP

The design process is illustrated in figure 2 and is executed in an iterative manner. Blue boxes denote involvement of stakeholders and end-users. Green boxes denote technical tasks.

First, possible use cases are identified in collaboration with stakeholders and end-users. Use cases describe interactions with the virtual platform (and its resources) that provide added value to end-users. Once a use case is identified, it should be expressed as a user story[1] and more formally as a structured use case[2], containing the user roles involved, the flow of execution and pre- as well as post-conditions, if applicable (see Appendix C for a detailed description of elements to be described in the use case). Based on the use case description, requirements – from the VP perspective – are derived. Requirements map the use cases to components (both VP components and components that are not part of the VP) and formulate a transaction workflow (e.g., expressed through sequence diagrams) to illustrate the communication between the components. Next, the components identified in the requirements are mapped to the overall architecture of the VP. Table 4 shows the available mappings with respect to the overall architecture:

Table 4: Mapping of components to the virtual platform

Mapping	Description
Part of the virtual platform	The component identified in the requirement is readily available and already part of the VP;

[1] See https://en.wikipedia.org/wiki/User_story for more details
[2] https://en.wikipedia.org/wiki/Use_case

Possible candidate for inclusion	The component might be part of the VP, but usefulness and feasibility are subject of study in the use case evaluation. The component might also represent a stand-in for another component that is not available yet. For instance, a prototypic API to count patients at a resource, might be replaced by another, more sophisticated component at a later stage of the design cycle.
Not part of the virtual platform	The component identified is not part of the virtual platform, e.g., a user interface prototypically show-casing VP interactions for end-users.

Once the components and their interactions are described in the requirements and the mapping to VP components is done, the use case can be implemented, meaning that components are either newly developed or existing components are adapted to the needs of the use case. Implementation also includes the deployment of components, whereas resource-level components are installed at the resource site and platform-wide deployments are deployed to a hosting environment. Once the deployment is done, the use case can be evaluated together with stake holders and end-users. In this phase, feedback is collected which is then used to refine the use case. If refinement is needed, the cycle starts all over again whereas refinement can be done on the use case itself (involving stakeholders extending the scope of the use case) or on the component-level (the use case description does not change, but components involved may be exchanged or improved).

Example: The counting use case has been identified as use case where multiple registries use the virtual platform to gain a total count of patients present in all registries. The requirements have identified the need of a web-based user interface where end-users can initiate the count request and are presented with the results in tabular form showing the number of patients with an officially classified rare disease. As a result, the following components have been mapped from the architecture: a query builder – available at the resource site –, the ontological types for 'patient' and 'rare disease', and the VP index, where the URL endpoints of the query builder are listed. The web-based user interface is considered an application using the VP for this scenario.

Finally, the implementation phase of the iterative use case design cycle includes the notion of quality checkpoints. This relates to the management of quality and sustainability measures related to components and services of the virtual platform and is detailed in chapter 3.5.

3.3. Reference implementations

The architecture presented in chapter VP Architecture and overview of components represents a logical view of resource-level components and platform-wide services provided by the VP. It does not imply that a component or service is represented through exactly one software element. Instead, the component or platform-wide service provides the structural design (expressed through requirements) which is then

translated to one or more reference implementations. Especially at resource-level, components might have multiple reference implementations to address needs of the software system that is in place at the resource's site:

Example 1: 2 RD registries (Registry A and Registry B) would like to participate in the virtual platform and expose machine readable information in terms of the EJP RD meta data model through the meta data provider. Registry A uses the free of charge EDC system REDcap[3], while Registry B uses a commercial solution. From the perspective of the virtual platform, both make use of the resource-level components EJP RD metadata model and metadata provider. However, registry A uses a different reference implementation than B, with the first being specifically tailored to the requirements of REDCap and the second tailored to the needs of the commercial application.

The same principles are applied to platform-wide VP services as the following example illustrates:

Example 2: The authentication and authorization infrastructure is based on the principles of open authentication (OAuth) and Open ID connect principles. The Elixir Life Science authentication and authorization infrastructure can be considered as a reference implementation of the authentication and authorization infrastructure.

3.3.1. Types of reference implementations

Two types of reference implementations are used in the virtual platform. First, **software-vendor-specific implementations** are complete implementations resource-owners can use, that require little or no configuration to work out-of-the-box:

Example 1: a meta data provider component for REDcap-based patient registries providing information about a resource following the EJP-RD meta data model. The meta data provider can be installed at any REDcap-powered site and used out-of-the-box with little or no need for further customization.

In contrast, **generic reference implementations** provide a framework for the implementation of a component that need to be completed by resource owners and software developers:

Example 2: the generic implementation of a meta data provider might be used by a resource owner to connect a proprietary registry software to the virtual platform. The generic implementation contains interfaces that must be implemented by the source owner.

The scope of reference implementations is neither restricted to a specific language nor technology but may make use of a variety of tools and methods, including the provision of libraries, plug-ins or fully containerized solutions for the installation at the resource's site.

Resource owners and specifically infrastructure providers may have restrictions with respect to technology that can be used at a resource's site. Therefore, a resource

[3] https://projectredcap.org/software/

owner might decide to implement a component on their own rather than using a vendor-specific reference implementation or a generic reference implementation. In this case, resource owners and software developers would provide their own implementation adhering to the VP specifications.

3.4. Levels of component maturity

Components that are part of the virtual platform may be available in various levels of maturity, as they go through the use case design cycle. The level of maturity is expressed following loosely the recommendation of the Capability Maturity Model (CMM). The following table describes four labels to express the level of component maturity:

Table 5: Levels of component maturity

Level	Description
Draft	The component being described has not been implemented yet or is under active development but has not been shared or evaluated with stakeholders of the VP.
Trial use	The development of the component has been completed (with respect to one life cycle iteration) and ready for trial use during the use case evaluation phase. However, no guarantees are made that future versions remain compatible if older versions of the component. At this stage, reference implementations as described in the previous chapter should be available.
Normative	The component described is in a stable state, can be used by clients and guarantees a life cycle management process for future updates.
Deprecated	The component described is considered as being deprecated and might be removed from the document in the near future. For instance, a prototypic component developed in an iteration cycle might be dropped in favor of a more advanced component.

3.5. Component lifecycle management

Components of the VP need to be scrutinized for their sustainability. Sustainability is not only a question of continued financial support, but also has important "quality" aspects. Functioning of the VP is critically dependent on some of the components. The dependency on external software and services and use of standard protocols and data formats (as described in chapter VP data and meta-data standards) is also important. Consequently, the iterative use case cycle is supported by a component lifecycle management process. The goal of the process is to ensure quality and sustainability of VP components as they go through the use case iteration cycle.

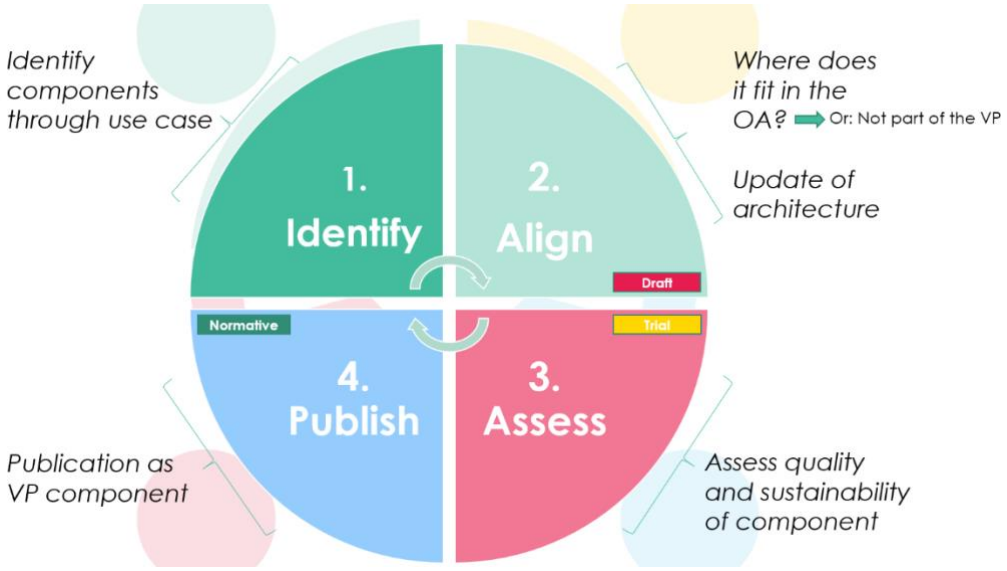

Figure 3: Component life cycle process

The life cycle management process consists of four steps that are done at various level of the use case design cycle as indicated in figure 3. First, potential new VP components are identified during the requirements phase of the design cycle that are required to fulfil the needs of the use case. The result can either be the introduction of a new component or the refinement of an existing one. In the "Align" step, the component is aligned with the overall architecture of the virtual platform. At this stage, it is decided if the component will be considered part of the VP or if it is an application or component that is interacting with the VP. If it is decided that a component becomes part of the VP, the architecture is updated accordingly. This corresponds to an update of the Virtual Platform Specification Document. The decision for inclusion of a component in the VP is part of the work of the work focus Overall Architecture whereas the decision is made with all involved EJP-RD partners.

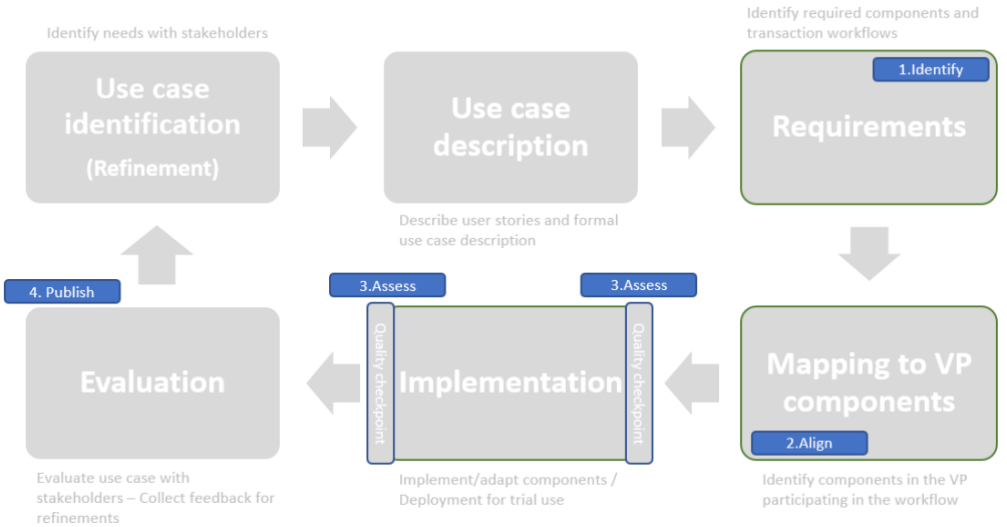

Figure 4: Relation between use case design and component life cycle

The assessment phase consists of two "quality checkpoints" where component owners can evaluate their component structural design and their reference implementations with respect to quality and sustainability. There are no strict rules for how quality and sustainability should be guaranteed, instead these considerations are the highlight of

open discussions by the maintainers/owners of VP components with quality and sustainability experts in Pillar 2 at the quality checkpoints. During these discussions ways to minimise continuity-risks for the components in the VP are discussed. As a guideline, to assess quality and sustainability of VP components,

Appendix B: VP Quality and sustainability Measurements lists measurements represented as a set of questionnaires in the following areas:

1. Component dependencies
2. Licensing and ownership
3. Contingency plans
4. Documentation with respect to potential users
5. Credentials
6. Technical information
7. Interoperability

These guidelines form the basis of moving components from one maturity level to the next. While primarily intended as a tool to support quality and sustainability in VP components and their reference implementations, the principles can be extended to non-VP components as well.

3.6. FAIR principles

The VP follows FAIR principles by design to improve the findability, accessibility, interoperability and reusability of its elements, in accordance with the required levels of service. This impacts a number of areas of the platform spreading through functional and non-functional requirements. Among the non-functional requirements we can cite:

- **Unique identification** - digital objects within the platform should be identified using globally unique and persistent identifiers;
- **Self-description** - the digital objects within the VP that are susceptible to external intervention, e.g., services and data, should be self-describing so that machines (computational agents), as well as humans, can discover these elements and "learn" what they are, if they can be reused and how to handle them.
- **Machine-actionability** – in order to improve and/or support machine actionability, well-structured information should be provided, including both machine-readable syntaxes as well as the semantics that allows computational agents to process these descriptions and infer (at least in part) what this information represents. Technically, this can be accomplished by using ontologies.
- **Well-defined access mechanisms** – mechanism(s) to access data and metadata for both read and write (e.g., query APIs or a download protocol) should be well defined (described) in a way that machines can understand. Access *restrictions* should be part of the machine-readable metadata.

3.7. VP Resource integration level

The VP resource integration level is an indicator how deep a resource is integrated into the VP eco-system.

Table 6: VP Resource integration levels

Integration level	Description and requirements
L1	The resource is identifiable to the VP and listed in the VP index.
L2	The resource is known to the VP, listed in the VP index, and exposes machine-readable metadata. (Self-description)
L3	The resource is known to the VP and metadata is exposed to the outside world. Well-defined access mechanisms are provided by invoking the VP authentication and authorization infrastructure and through consent control.
L4	The resource is fully integrated in the VP with all properties described in level 1-3. Level 4 adds the possibility to use record-level data such as in queries.

4. VP data and meta-data standards

This chapter outlines the data and meta data standards used by VP platform and resources linked to the virtual platform. They can roughly be divided into meta data standards (e.g., the EJP RD metadata model), data exchange mechanisms (including SPARQL or RESTful APIs, the foundation of the VP API), common data exchange mechanisms (data formats that are used to exchange data between resources and VP components and services) and data models (common data models to represent data stored in resources). There is a relation to deliverable 12.2 including FAIR tools and data standards. The standards listed here are the standards relevant for the first version of the virtual platform, however it is expected that this list will grow with future versions of this document. In this chapter standards are only briefly outlined and their relationship to components and services in the virtual platform is highlighted.

4.1. Meta-data standards

4.1.1. Common core metadata schemas

Resources must list metadata properties in a standard way for the VP to function efficiently. The requirement for VP compliance is to provide metadata at least in accordance with the EJP RD metadata model. This provides a common, predictable way by which metadata can be assessed, regardless of the metadata content. For instance, the metadata model contains the recommended predicate to link the applicable license to the resource.

4.1.2. EJP RD meta data model

To improve the findability of rare disease resources it is important to provide standardized and machine-readable descriptions of these rare disease resources. Within the EJP RD project a metadata model is being developed as an extension of the widely used Data Catalogue Vocabulary (a W3C standard)[4]. A stable version (version 1.0) of EJP RD metadata has been released (see Appendix A and deliverable

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

D11.2 Second Ontological model of resources metadata for more details). This version of the metadata model provides schemas to describe

(a) Patient Registries

(b) Biobanks and

(c) Guideline resources.

A VP compliant resource serves machine readable metadata about itself in terms of the EJP RD metadata model and provides an interface in accordance with the FAIR Data Point specification (which follows the DCAT standard). Optionally, EJP RD provided software (e.g., RD-NEXUS or the FAIR Data Point reference implementation) can be implemented to serve the required metadata conform these specifications.

4.2. Common data exchange mechanisms

The virtual platform relies on a set of commonly agreed APIs constituting the VP API, on top of commonly agreed data and metadata models to make resources 'self-describing' for efficient machine processing. As a core principle the VP API follows the principles of the Representational State Transfer (REST), a software architectural style used for web-services to define uniform interfaces, especially for the north-bound part of the VP API. Thus, platform-wide services including the meta-data provider, query builder, access and consent control as well as platform-wide services provide interfaces built on this architectural style.

The RESTful style defines interfaces on the concept of exchanging and manipulating resources (e.g., a patient record) through the exchange of resource representations (as described in chapter 4.3) through a set of standardized operations over the hypertext transfer protocol (HTTP).

Several standards built on the REST principles are incorporated in the VP API, namely the Beacon API used by the resource-level query builder component or the Open ID Connect interface used by the authentication and authorization infrastructure. The Health Level 7 (HL7) FHIR standard is used by resource-level component data model alignment, enabling the transformation of common data elements into and from FHIR-represented data.

Apart from RESTful interfaces, SPARQL interfaces are also provided through the VP API. SPARQL is a W3C recommended standard to both query RDF graphs. SPARQL, as well as being a protocol defines the request/response messages that pass these queries over HTTP. Many modern triple stores (special databases to store RDF graphs) also implement the SPARQL protocol which makes these triple stores queryable over the Web. When a resource is served via a triple store which implements the SPARQL protocol, it becomes possible to turn a SPARQL query into a REST API.

4.3. Common data representations

The RESTful style does not imply a particular data representation for the exchange of data over interfaces. That being said and to ensure the VP is sufficiently scalable and flexible, it will allow resources for exchange of data in a limited number of formats and representations. Currently, the predominant syntaxes (data formats) for exchange are

eXtensible Markup Language (XML) and JavaScript Object Notation (JSON). Both are highly generic, and both cater for the use of schemas to specify constraints.

Regarding data representation, a major distinction is between "regular" data and "linked" data. Regular data requires human interpretation of the schema to determine the semantics of data elements, whereas linked data provides a qualified reference to those semantics (i.e., the machine readability of ontologies). In general, representation of regular and linked data takes XML or JSON syntax.

As an example, taken from https://en.wikipedia.org/wiki/JSON-LD :
A JSON-representation of a person, with his name and homepage, could look like this.

```
{"name": "John Smith",  "homepage": "https://www.example.com/"}
```

With this representation, it is necessary to know what "name" or "homepage" exactly means. In the case of English, and with well-selected labels, this may in many cases not be a problem from the human perspective, but from a machine-readable perspective it is. This can be leveraged by using JSON-LD, in which a context can be added that makes the semantics of the labels explicit. A JSON-LD representation of the example above could be:

```
{
  "@context": {
    "name": "http://xmlns.com/foaf/0.1/name",
     "homepage": {
     "@id": "http://xmlns.com/foaf/0.1/workplaceHomepage",
     "@type": "@id"    },    "Person": "http://xmlns.com/foaf/0.1/Person"
}
```

In this example, the context explicitly links "name" and "homepage" to URIs. These URIs in turn provide a human-readable description of the meaning of the properties (http://xmlns.com/foaf/spec/#term_name and http://xmlns.com/foaf/spec/#term_homepage, respectively), as well as a machine-readable description. More specifically, they are identifiers of concepts represented using Resource Description Framework (RDF) and RDF Schema (specifically in http://xmlns.com/foaf/spec/index.rdf).

RDF can be represented in various forms, including XML, JSON, and Turtle, a syntax specific for RDF (for: Terse RDF Triple Language).

4.4. Common data models

Common data models are used at resource-level to represent data in a uniform way. Common data models are described in this section, as they are of particular interest for the query builder component as well as the model data alignment service.

4.4.1. Common Data Elements (CDE)

Common data elements (CDE) are a core data standard describing 16 common data elements essential for RD research which should be included in European RD registries.

This list comprises elements such as pseudonyms, personal information about a patient (date of birth and sex), patient status, care pathway, disease history, diagnosis, research and disability.

From a VP perspective, CDEs are a paramount building block for the record-level component query builder which is detailed in chapter 5.1.4.

4.4.1.2 Common Data Element Model (CDE- semantic model)

The proposed Common data elements (CDE) by JRC was a good start to improve the data quality in the RD registries. However, this set of CDEs lacks a semantic basis which is vital to improve the interoperability of the resources for both humans and machines. Within the EJP RD project a modelling group has been established to create a semantic model for JRC's common data elements. This group created models to represent each of the CDEs, using the SemanticScience Integrated Ontology as the core framework for representing the CDEs and their relationships. Within that framework, the group also mapped the concepts represented in the CDEs, and their possible values, into domain ontologies such as the Orphanet Rare Disease Ontology (ORDO), Human Phenotype Ontology (HPO) and National Cancer Institute Thesaurus (NCIT) (see Appendix A for more details).

4.4.2. OHDSI OMOP Common Data Model

The OMOP Common Data Model (CDM) [https://ohdsi.github.io/TheBookOfOhdsi/CommonDataModel.html] is a healthcare standard for medical observational data in (research) hospitals developed by the OHDSI community [https://www.ohdsi.org/]. OMOP CDM enables the capture of information such as encounters, patients, providers, diagnoses, drugs, measurements and procedures. Once a database has been converted to the OMOP CDM, evidence can be generated from disparate sources in a systematic way using standardized analytics tools. A library of standard open-source analytic routines is currently under development for data quality and characterization, medical product safety surveillance, comparative effectiveness, quality of care, and patient-level predictive modelling. CDM does not require a specific technology since it is a relational database model (all data is represented as records in tables that have fields), which means that the data will typically be stored in a relational database using a software platform like PostgreSQL, Oracle, or Microsoft SQL Server. See below an example of encounters table.

Table 7: OHDSI OMOP Common data model - Example

Encounter ID	Start date	Stop date	Type
70	2010-01-06	2010-01-06	outpatient
80	2011-01-06	2011-01-06	outpatient
90	2012-01-06	2012-01-06	outpatient
100	2013-01-07	2013-01-07	outpatient
101	2013-01-14	2013-01-14	ambulatory
102	2013-01-17	2013-01-24	inpatient

4.4.3. CDISC ODM

The Clinical Data Interchange Standards Consortium (CDISC) focuses on medical research data, such as clinical trial data. To store and interchange study data and study metadata associated with clinical trials, CDISC has developed the Operational Data Model (ODM), which is specified as an XML schema. A small fragment of ODM is shown below for illustrative purposes:

```
<ClinicalData StudyOID="P2006-101" MetadataVersionOID="101.01">
  <SubjectData SubjectKey="1000" TransactionType="Insert">
    <StudyEventData StudyEventOID="Screen">
      <FormData FormOID="DEMOG">
        <ItemGroupData ItemGroupOID="DM">
          <ItemDataString ItemOID="USUBJID">101-001-001</ItemDataString>
          <ItemDataString ItemOID="SEX">F</ItemDataString>
        </ItemGroupData>
      </FormData>
      <FormData FormOID="LABDATA">
        <ItemGroupData ItemGroupOID="LB">
          <ItemDataDatetime ItemOID="LBDTC">2006-07-
14T14:48</ItemDataDatetime>
          <ItemDataString ItemOID="LBTESTCD">ALT</ItemDataString>
          <ItemDataString ItemOID="LBORRES">245</ItemDataString>
        </ItemGroupData>
      </FormData>
    </StudyEventData>
  </SubjectData>
</ClinicalData>
```

As pointed out earlier, this representation requires interpretation of the meaning of the XML tags that are specified in the ODM schema, as well as of the String values and the elements, such as "Screen" for screening, "DEMOG" for Demography, F for female, or ALT for alanine aminotransferase.

5. Description of components

This chapter provides an overview of resource-level components and platform-wide services envisioned for the first version of the virtual platform.

5.1. Resource-level components

5.1.1. Metadata provider

A Metadata provider is a generic resource-level component providing the description of resources that enables discovery of these resources by the VP. The metadata provided by metadata provider service will enable dynamic integration of new resources into the VP. To be discovered by the EJP RD VP, the metadata provider service of the resources must provide their metadata according to EJP RD common

core metadata. The content provided by metadata service can be represented as JSON (schema provided by EJP RD), turtle or JSON+LD. The FAIR Data Point software can be used as a metadata provider service but, in that case, the FAIR Data Point should be configured with EJP RD common core metadata model and/or employ a proxy (e.g., RD-NEXUS) to also respond to Restful queries.

5.1.2. Data model alignment service

The CDE semantic model was built to represent, in 'ontologised' linked data form, the CDEs (see chapter 4.4.1) defined by JRC for RD registries. The aforementioned data standards are commonly used for health data and some of them are already adopted by ERN registries. Mapping of the CDE semantic model to/from the 3 data standards (CDISC, ODHSI OMOP and FHIR) is therefore needed. Therefore, a data model alignment service is envisioned comprising a mapping table for CDE terms and scripts for transforming data into the standards, and vice versa. The CDE model uses ontological terms from various ontologies to represent the CDEs. These ontological terms are to be mapped to the ontologies or terminologies used in the 3 standards, presented in a mapping table. The CDE Semantic model represents CDE data in RDF Turtle files, and similar mappings and support will be needed to/from the other 3 standards. Docker images will make supporting scripts more user friendly.

5.1.3. Access and consent control

The access and consent control component is a platform-wide component that manages 2 aspects:
A) authentication mechanisms following the principles of OpenID connect and/or SAML 2.0 provide resources with the possibility to (a) provide a single sign-on experience for end-users as illustrated in chapter 5.1.4 and
B) to provide resource-level component API endpoints with a token-based authentication.
Both ways of interaction are directed to the authentication and authorization infrastructure described in chapter 5.2.2.
The second aspect of this component is the provision of machine and human-readable consent information based on data models including Digital Use Condition (DUC) profiles that build on standards such as ADA-M, DUO, ICO, to enable applications outside the VP and other components inside the VP (e.g., query builders) to perform consent-based queries.

5.1.4. Query builder

The query builder is a resource-level component enabling data discovery on following levels:

a. **Resource-level** (e.g., represented as queries in catalogues such as Orphanet and BBMRI)
b. **Record-level** (represented as queries to RD registries supporting the Common Data Elements)

Query builders as resource-level components contribute API end points based on the GA4GH Beacon-2 plus standard (see Appendix A for the API reference) to the virtual platform API to enable federated queries as illustrated in following figure:

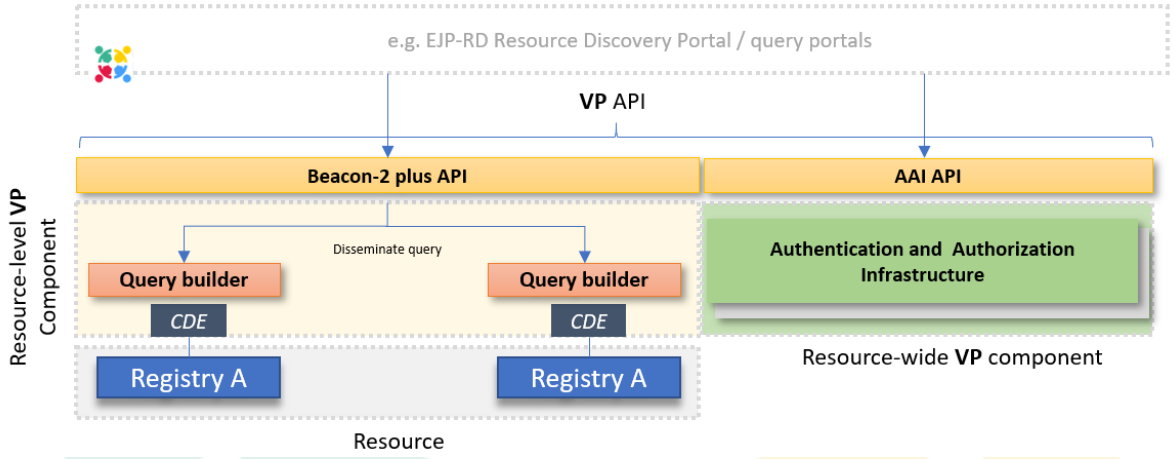

Figure 5: Federated approach of the query builder component

The figure illustrates the federated approach of executing record-level queries against RD registries (denoted by registry A and registry B) participating in the VP. An end-user of the EJP-RD resource discovery portal initiates the query within a web-based portal. The portal uses the VP authentication and authorization infrastructure to authenticate the user. The query itself is disseminated to the two participating registries with the portal presenting the combined results. The record-level query does not rely on platform-wide VP components (apart from the authentication and authorization and the discovery portal not being part of the VP) to execute the query. Instead, it relies on the beacon-2 plus API exposed by the resource-level VP component at the registry's side.

Instead of relying on a single discovery portal as illustrated in figure 5, a scenario with multiple independent portals is possible too, with RD registries forming private networks allowing record-level queries:

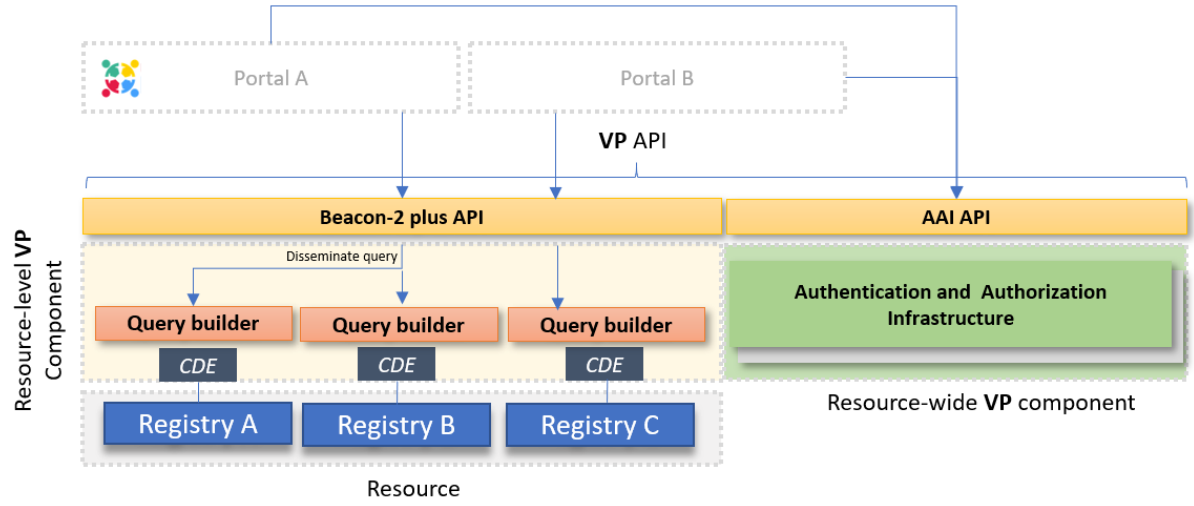

Figure 6: Federated approach of the query builder component – multiple portals

In this scenario, registry A and registry B form a private network whereas registry C can be queried through a separate portal B.

5.1.5. Pseudonymisation

The platform-wide pseudonymization component facilitates the generation and management of patient pseudonyms. Patient pseudonyms are part of the common data elements (CDEs). The idea is, that RD registries receive a unique identifier for a patient, only known to the RD registry itself and constructed in a way, that the identifier itself does not reveal any personal information about the patient.

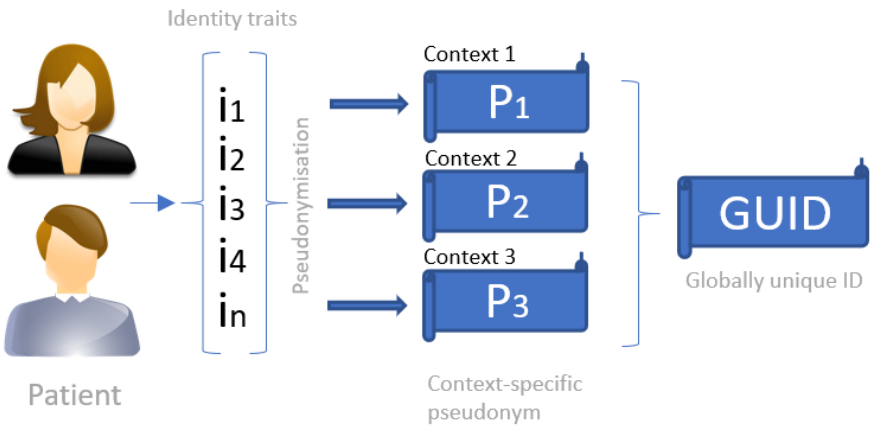

Figure 7: Pseudonymization and context-specific patient identifiers

As figure 7 illustrates, at resource-level patients are identified through a set of identity traits (e.g., first name, last name and date of birth), only known to the resource itself. The pseudonymization component de-identifies identity traits through hashing and encryption algorithms. Hashed and encrypted identity traits are exchanged with a platform-wide record linkage service that generates a context-specific pseudonym for the patient whereas the context corresponds to the RD registry. If the patient has already been registered in another context, the pseudonyms are being linked through an undisclosed globally unique id only known to the platform-wide record linkage service.

The European Patient Identity Services (EUPID) are provided as reference implementation for this workflow. First, a web service (part of the platform-wide component record linkage services) provides VP API endpoints for patient registration and pseudonym generation. Second, a set of libraries is provided as resource-level component, supporting resources in interacting with the API, as well as hashing and encrypting identity traits on the local site.

5.2. Platform-wide services

The first version of the virtual platform includes 3 platform-wide services that are described in this chapter.

5.2.1. VP API index

The VP API index is a platform-wide component that holds a directory of accessible resources and their VP API endpoints as well as VP API endpoints of other platform-wide components. It is meant as a supporting service for applications (e.g., a resource discovery portal) to primarily know what resources and components the virtual platform contains. The index contains the following end points:

a. VP API endpoints for retrieving meta data about a given resource.
b. VP API endpoints for invoking record-level queries
c. VP API endpoints for invoking resource-level queries
d. VP API endpoints for authentication and authorization infrastructure (e.g., endpoints for retrieving tokens)
e. VP API endpoints for record linkage services

Following the design principle laid out in chapter 3.1 the VP index is federated by nature, thus allowing multiple instances to be run in parallel.
A reference implementation includes the *query builder catalogue directory* providing directory services for biobanks and registries (see Appendix A for more details on the API specification).

5.2.2. Authentication and Authorization Infrastructure

The authentication and authorization infrastructure (AAI) provides resources, resource-level components, platform-wide services and applications as well as services outside of the virtual platform with the means to provide end-users with a single-sign-on experience as well as API endpoints with means of token-based authentication based on the Life Science AAI. Life Science AAI is a common authentication and authorisation service for the 13 European life science research infrastructures. The service is managed by the life sciences community and operated by the e-infrastructures, including GEANT and EGI.

The Life Science AAI issues a new identifier called a Life Science ID to a user who registers to the Life Science AAI and accepts its usage policy. Users authenticate using authentication providers, such as their home universities or the Life Science AAI's Hostel IdP which can be linked to their Life Science ID. It's also possible to use an ORCID, LinkedIn, or Google account. Multiple accounts can be linked to a single LS AAI account, and will be recognized as the same person at the end service. To cater services with specific assurance needs, Life Science AAI supports an assurance framework and will provide a step-up authentication service in the near future.

Life Science AAI decorates the user IDs with **extra attributes**, such as the user's home organisation, home research infrastructure and researcher status. The Life Science AAI can also manage user's group memberships and permissions to access controlled access datasets. These attributes can then be consumed and used for access control enforcement by the services relying on the LS AAI.
The AAI supports three standards for performing these tasks:

Table 8: Supported authentication and authorization protocols

Standard	Description
SAML 2.0	A standard for exchanging authentication and authorization identities. It uses security tokens to exchange information between a SAML authority (AAI) and a SAML consumer (e.g. a resource-level component)
OAuth2	including support to encoding attributes to access tokens as signed JWT
Open ID Connect	Open ID connect is an authentication layer on top of the OAuth2.0 authorization framework based on a RESTful API using JSON as data exchange format.

There is a close relation to the resource-level component "access and consent control" where client-side functionality for interacting with the AAI (e.g. requesting and validating access tokens) is integrated.

Elixir Life Science AAI is considered a reference implementation from a VP perspective by providing a deployed AAI infrastructure along with recommendations for integration from the access and control component. Table 10: Applied standards list more material and resources on the AAI infrastructure.
.

5.2.3. Record linkage services

Record linkage services are services comprised of three individual platform-wide services:

(a) **Pseudonymization** of patient information as outlined in chapter 5.1.5 – in this step resources register patients and receive a context-specific pseudonym being part of the CDEs.

(b) **Data transition** – resources willing to pool record-level data for research purposes (e.g., a data analysis task) are re-registering patients in a 3rd context.

(c) **Data linkage** – based on the data transition, data can be pooled using de-identification techniques in this 3rd context.

Figure 8 illustrates the process of using context-specific pseudonyms to merge data from different data sources for secondary use. The basic requirement is that the patients of the data source have been registered and pseudonymised in contexts

associated to the data sources. This allows a transition of data from different data sources to a target context, where the patient identifiers (pseudonyms) are replaced by corresponding pseudonyms in the target context. Having all necessary record-level data transferred to the target context allows subsequent merging of data (data linkage) for secondary use. Furthermore, K-anonymity, L-diversity or t-closeness etc. could be ensured. In this state the data is still pseudonymised and re-identification by a trusted third party might still be possible. For anonymisation the pseudonyms must be removed irrevocably.

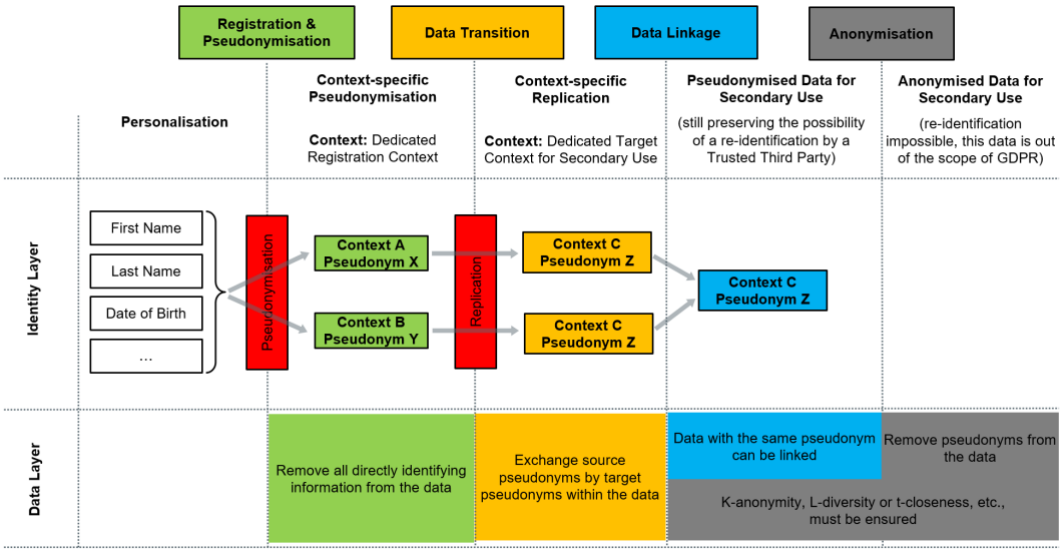

Figure 8 : Process of utilising different contexts to merge different data sources for secondary use

As reference implementation, EUPID Services can be used to support the data transition and context-specific pseudonymization process step.

5.3. Auxiliary components and services

This chapter describes auxiliary components and services of the VP supporting sustainability and maintainability aspects as well as the FAIRification process.

5.3.1. Resource FAIRness assessment service

A service to assess the FAIRness of the digital objects within the platform. As specifications become clear regarding the assessment of software and interfaces, the components of the VP components itself will become subject to FAIRness assessments.

In the initial year of the EJP-RD, we built on prior efforts related to the definition of FAIR Metrics to create a fully automated FAIR assessment tool. Each of the FAIR Metrics is represented by one or more FAIR Metric Tests, where the test examines the (meta)data of a digital object to determine if it conforms to the expectations of FAIR. FAIR Metric Tests are grouped together as "collections", where a given collection is considered relevant for a specific type of digital object (for example, the collection used to test a database record is distinct from the collection used to test a piece of software). Collections of tests are applied to the appropriate digital object to obtain an objective and reproducible FAIRness score.

The mechanism for testing is fully automated and proceeds as follows: The user supplies the Globally Unique Identifier for the metadata record of the digital object of interest. That identifier is then examined to determine if it meets the expectations for FAIR (i.e., that is persistent and resolvable using a widely accepted standard), and the GUID is then resolved to obtain the metadata record itself. A variety of paths are then followed in an attempt to retrieve a comprehensive metadata record - for example, in the case of a DOI, some metadata can be obtained from CrossRef or DataCite, while additional metadata may be available from the data host itself (e.g., Zenodo or Dataverse). Once all paths have been exhaustively followed, the metadata record is examined against a variety of metrics based on the expectations of FAIR (e.g., that the metadata uses FAIR vocabularies, that the metadata explicitly refers to the data, that the metadata contains a link to the license, and so on). Where necessary, the system also attempts to discover the data record described by the metadata and examine its properties.

This workflow is made available via a piece of software called the FAIR Evaluator, which has a public interface at: https://w3id.org/AmIFAIR. To date, >5500 evaluations have been executed using as many as 22 different metric tests. The highest Evaluation score to date is 20/22, which was achieved by the EJP-RD Orphanet project partner's FAIR Data Point.

Between now and the end of EJP-RD, we plan to execute a series of "before/after" evaluations, such that we can obtain a quantitative measure of the degree to which EJP RD partners have increased their "FAIRness".

5.3.2. Quality and sustainability tools

The component life cycle management described in chapter 3.5 may be supported by additional tools to capture the quality and sustainability measures described in appendix C. Thus, the questionnaires may be represented in online tools for data collection or self-assessment of components and reference implementations through their respective owners.

6. Virtual platform API

The VP API is a set of standardized, distributed endpoints that are either provided by platform-wide services or by resource-level components. The concept distinguishes two types of API endpoints: First, north-bound API interfaces are directed towards applications and services that interact with the VP. Second, south-bound API interfaces are directed towards the resources itself. The following figure illustrates the difference between the two concepts:

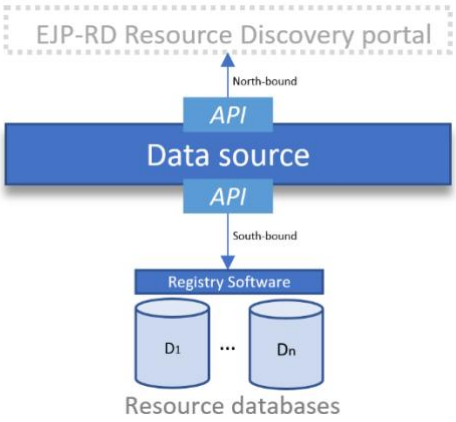

Figure 9: The concept of north- and sound-bound APIs illustrated through a RD registry

Following the example of the counting use case, an EJP-RD discovery portal provides end-users the possibility with counting the number of RD patients in various registries. These registries provide a north-bound API towards the discovery portal. The registry software itself makes use of the south-bound API translating the incoming request to a count query based on its internal data structures. The north-bound part of the VP API is based on RESTful principles, thus exposing a web-based interface to applications, whereas the API endpoints are described in the OpenAPI specification. The south-bound API may be exposed through a RESTful interface (e.g. a proxy service sitting on top of the registry software) but can also be described through standardized function and library calls or even through scripts directly interacting with the resource databases.

The following table provides an overview on API endpoints envisioned for the VP:

Table 9: List of north- and southbound APIs envisioned for the VP

Component / Service	Direction	Description
Meta data provider	North-bound	API endpoint to expose a resource description expressed through the EJP-RD meta data model
Meta data provider	South-bound	Library functions to create the resource description expressed through the EJP-RD meta data model
Data model alignment	South-bound	Library functions, scripts and tools to translate data at the resource site from one format to another.
Access and consent control	South-bound	Library functions provided to the resource software to enable authentication and authorization (e.g. single sign on) with the authentication and authorization infrastructure
Authentication and authorization infrastructure	North-bound	API endpoints enabling resources and applications to request/validate/refresh access tokens as well as an OpenID connection "well-known" configuration
Query builder	North-bound	API endpoints allowing applications such as a discovery portal to perform record and resource-based queries

Component / Service	Direction	Description
Query builder	South-bound	Library functions provided to the resource software to translate incoming queries to queries to the local databases.
Pseudonymisation	South-bound	Library functions supporting resources in creating patient pseudonyms in interaction with the record linkage services.
VP Index	North-bound	A list of north-bound API endpoints available to resources and applications including authentication and authorization, query resolvers, meta data providers and catalogues.
Record linkage service	North-bound	API endpoints to perform privacy-preserving record linkage, data transition and linkage.

7. Conclusion and summary

This deliverable describes the purpose and design principles of the EJP-RD virtual platform outlining key abstractions, an initial version of components envisioned for the first release as well as standards, quality and sustainability measures.

Within the current work plan the counting use case – involving several RD registries to perform a federated patient count – will serve as initial proof-of-concept of the architecture described in this document. As part of this proof-of-concept, the description of components will be refined and extended to further reference implementations applying the quality and sustainability measures described in this document.

Appendix A: VP Applied Standards

Following tables contains a list of standards, data models, data representation standards, meta-data models publicly available and with some being developed as part of the EJP-RD project. The list is not exhaustive but reflects elements that are used by components envisioned for the first version of the virtual platform.

Table 10: Applied standards

Standard	Type of standard	Material
EJP RD meta data model	Meta data model	https://github.com/ejp-rd-vp/resource-metadata-schema/tree/v1.0
Common data elements (CDE)	Data model	https://eu-rd-platform.jrc.ec.europa.eu/
CDE semantic model	Data model	https://github.com/ejp-rd-vp/CDE-semantic-mode l
OMOP common data model	Data model	https://www.ohdsi.org/
FHIR	Data exchange model	Index - FHIR v4.0.1 (hl7.org)
CDISC ODM	Data model	https://www.cdisc.org/
JSON	Data representation	https://en.wikipedia.org/wiki/JSON
JSON-LD	Data representation	https://en.wikipedia.org/wiki/JSON-LD
SPARQL	Query language	https://en.wikipedia.org/wiki/SPARQL
XML	Data representation	https://en.wikipedia.org/wiki/XML
Open API Specification	API specification	https://en.wikipedia.org/wiki/OpenAPI_Specification
Beacon API	API specification	https://beacon-project.io/
SAML 2.0	Authentication	SAML 2.0 - Wikipedia
Open ID Connect	Authentication	OpenID Connect – Wikipedia
Query builder API	VP API specification	GitHub - ejp-rd-vp/query_builder_api: This project focusses on the query builder API.
VP index API	VP API specification	GitHub - ejp-rd-vp/query-builder-catalogue-directory: The EJP-RD - 'Central catalogue directory' component.
Data Catalogue Vocabulary	RDF vocabulary	Data Catalog Vocabulary - Wikipedia

Table 11: Authentication and authorization material

Life Science AAI website	https://lifescience-ri.eu
Access and User Management System for Life Science	https://doi.org/10.5281/zenodo.4633191
Prototype implementation of distributed automated data access request, review and authorization and delivery systems	https://doi.org/10.5281/zenodo.3238496
AARC2 project. Blue-print architecture	https://aarc-project.eu/architecture/

Appendix B: VP Quality and sustainability Measurements

This section describes all sustainability related measurements. The list of measurements and associated questions is an initial draft evaluated in collaboration with EJP-RD project partners.

Table 11: Sustainability measurements

Category	Measurement / associated questions
Internal dependencies	Describe on this slide which other (technical) components of the EJP RD are depending on this - Describe the interface to those components
Internal dependencies	Describe on this slide which other (technical) components of the EJP RD this is depending on
External dependencies	Describe which external (technical) components (e.g., libraries) the component is depending on – describe the interface to each of these components
License and ownership	What is the license for the component? Who is owner? Could other institutes or people become (equal) partners to sustain the resource? Is there an institute that currently maintain for this (financially, documentation, location)?
Contingency	What would happen if the component would no longer be sustained? Would the EJP RD be able to use a drop-in replacement?
Other sustainability aspects	Is there a sustainability plan in place for the component? Is there a scientific advisory board?
Potential users	Is documentation available (e.g., READMEs, implementation guide, release notes) Training materials (e.g., webinar, course or materials in TeSS – ELIXIR's training portal, videos, training or demo environment, … etc.)
Credentials	Included in regonised registry (e.g., Bio.tools), IRDiRC, ELIXIR CDR, RIR, BioContainters, etc.) What kind of publication (published or planned) describe or use the tool (Peer reviewed or standalone) Seen as a default / standard /widely used by the community?
Technical information	Has the component a persistent global, unique identifier (for each release version)? Is the source code available (e.g. GitHub, GitBucket, SourceForge) Is there a published release cycle (inc. development plans / wish list)? Is performance data available? (e.g., average uptime, error files & logs)

An accompanying standard that supports these measures is the FAIR data maturity model as described in https://www.rd-alliance.org/groups/fair-data-maturity-model-wg.

Appendix C: Use case description elements

The following table describes elements that form the basis of a use case description as used in the use cycle design cycle.

Table 12: Use case description elements

Element	Description
Use case name	A concise, unique name for the use case (e.g., Counting use case)
Brief description	
Flow of events	The flow of events illustrates user interactions steps-per-step using the notion of *persona*. A *persona* is a stake holder of the VP, having a defined name (e.g., Marina, Alice, Bob).
Basic flow	The basic flow of events during interaction
Alternative flows	Alternative flow of interaction when certain preconditions are met.
Special requirements	Non-functional requirements associated with the use-case.
Preconditions	Preconditions that have to be met before the flow of events can start. (e.g., the persona needs to be authenticated)
Postconditions	A postcondition of a use case is a list of possible states that the system is in, once the flow of events has been finished.
Extension points	An extension point identifies a point within a use case, where another use case is referenced.

Appendix D: List of tables

Appendix E: List of figures