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

AD49

Virtual Platform Specification (VIPS) V3.0

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

Due date of deliverable: month 68

Dissemination level:
Public

Table of Contents

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

Date	Author	Changes
20-02-2022	Karl Kreiner	Re-structured document; added VP connection levels and introduced business functions
21-02-2022	Karl Kreiner	Aligned component descriptions with VP onboarding document
23-02-2022	Aylin Demir	Added section on EJP-RD Discovery portal
27-02-2022	Luiz Bonino da Silva Santos	Added VP Index
03-05-2022	Karl Kreiner	Compiled final version 2
21-08-2024	Karl Kreiner	Updated document structure

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. As such, the virtual platform relies on existing IT infrastructure and does not provide any infrastructure itself.

1.2. VP use case examples

One design principle of the virtual platform is use case driven design as described in chapter 2.3. 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 business functions[1] which then are mapped to components 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 components of the VP either

[1] https://pubs.opengroup.org/architecture/archimate31-doc/chap08.html

directly support the use cases (e.g. components 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) 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. RD Resource owners looking for connecting their resources to the VP should consult the VP onboarding document which can be found at **https://vp-onboarding-doc.readthedocs.io/en/latest/**.

1.4. Key abstractions of the virtual platform

Components 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 registry.
Platform-wide component	A platform-wide VP component 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 business function 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.
Business function	A function describes a specific functionality associated with the VP. Business functions manifest in components.	"Meta data provision" is a business function. A FAIR data point is implementing this business function
VP platform network	The collection of all connected RD resources	

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: Stakeholders 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. Design Principles

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.

2.1. 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 2.4 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.

2.2. 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 described in chapter 6 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 itself.

2.3. 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).

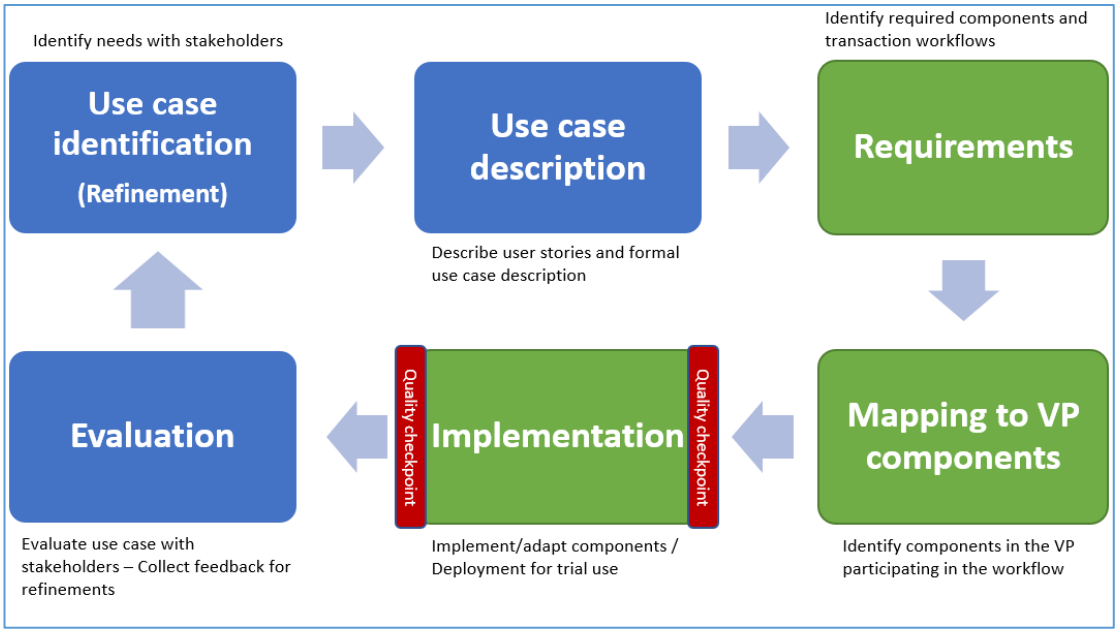

Figure 1: Iterative use case driven design of the VP

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[2] and more formally as a structured use case[3], 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:

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

Table 3: 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;
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 2.6.

2.4. Reference implementations

Components are not necessarily 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[4], 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.

2.4.1. 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.

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

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 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.

2.5. 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 4: 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 favour of a more advanced component.

2.6. 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.

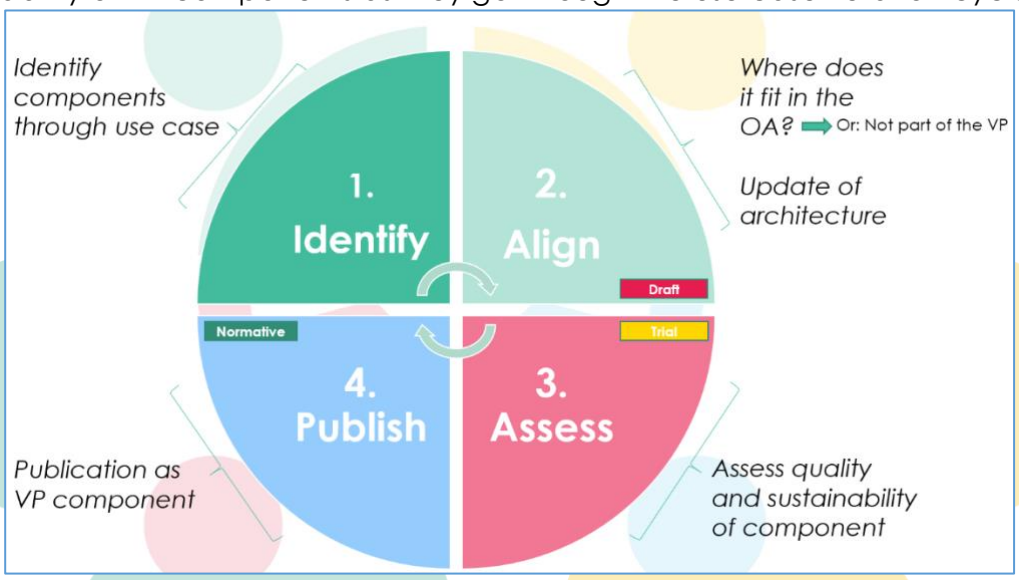

Figure 2: Component life cycle process

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.

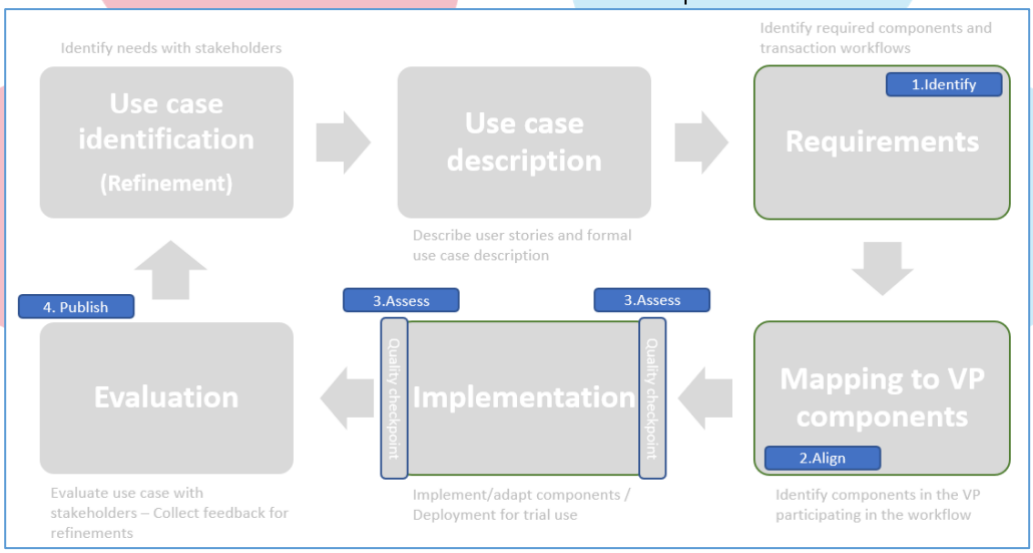

Figure 3: Relation between use case design and component life cycle

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, the 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.

2.7. 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. 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.

3.1. Meta-data standards

3.1.1. 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.

3.1.2. 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)[5]. A stable version (version 1.0) of EJP RD metadata has been released (see Appendix A and deliverable 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.

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

3.2. 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 3.3) 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.

3.3. 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).

3.4. 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.

3.4.1. 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.

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).

3.4.2. 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 5: 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

3.4.3. 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.

4. VP business functions

In the initial chapter we introduced following top-level functionality of the virtual platform:

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.

Based on identified use-cases – as described in chapter2.3– core business functions have been identified for the virtual platform that are described in this chapter. These business functions support resources to become part of the virtual platform. We have defined 3 distinct connections levels reflecting how deep a resource is linked to the virtual platform:

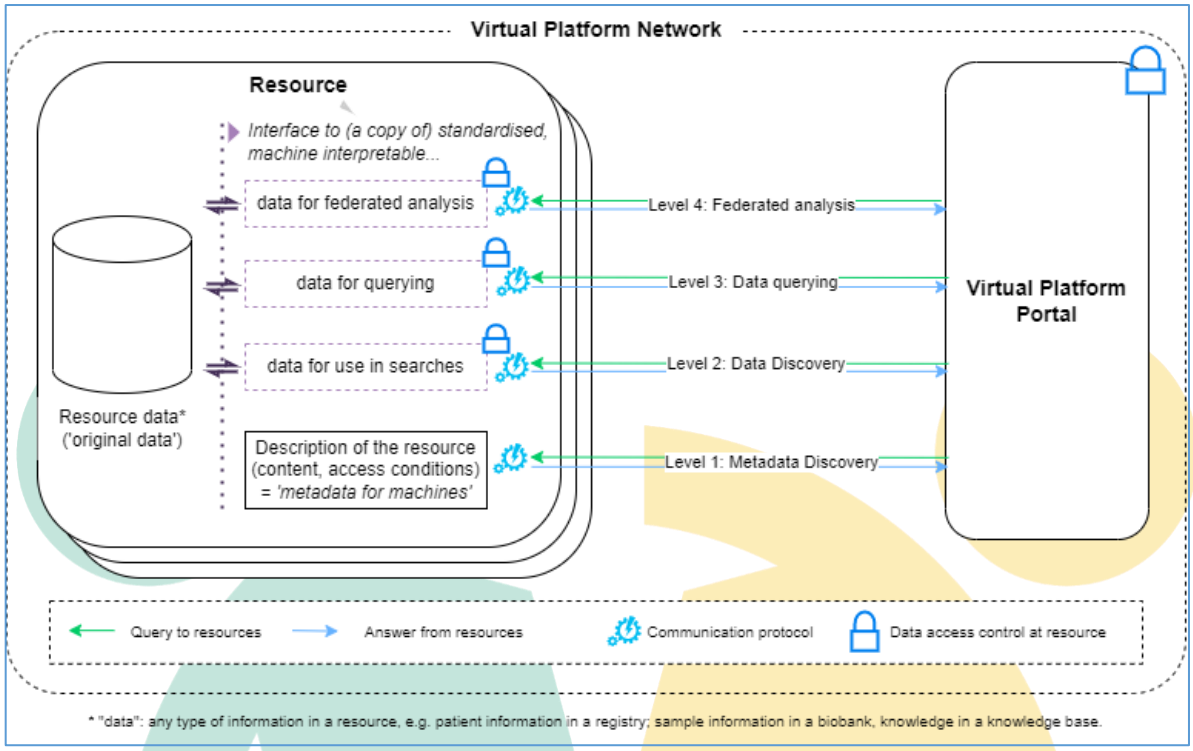

Figure 4: Illustration of the levels by which resources can contribute to and benefit from the Virtual Platform. Resources that apply the recommended extendible standards 'for machines' at source automatically increase the functionality of the Virtual Platform for the community

The following table provides an overview on the 3 levels with examples:

Table 6: VP Resource integration levels

Level	Name	Description	Query example
1 * (mandatory)	Metadata discovery	**At this level, the provider commits to openly publish online some standardised metadata about the offered resource, and hence make this available to the VP**	Keyword in your metadata (label theme in the FAIR Data Point) (e.g. Which resources are collecting data about Duchenne muscular dystrophy(ORPHA:98896)?)
2	Data discovery	**At this level, a resource is identified based on remote queries regarding its characteristics and content, responding with yes/no or approximate record count information. The questions are answered against summary metadata and safe content of each catalogue/resource.**	Counting data filtered by sex (e.g. How many female patients with Duchenne muscular dystrophy (ORPHA:98896)?)
3	Data reuse and analysis	**At this level, the provider commits to support**	Comparing multiple resources at patient level

Level	Name	Description	Query example
		interrogation and analysis on its resource's rich content.	(pseudonymised) data (e.g. Find records of patients with cardiomyopathy (HP:0001638) and Duchenne muscular dystrophy (ORPHA:98896) with a treatment protocol that matches my search criteria)

Based on these connection levels we derived business functions for the virtual platform that support resources in establishing the connection on various levels.
Business functions serve as specification whereas VP components are software products and services that implement one or more business functions.
The following table illustrates 3 different types of business functions:

Table 7: Types of business functions

	Description	Example
Resource-level	The function has to be realized directly at the resource.	Meta data provision has to be done through the resource itself
Platform-wide	The function is provided to multiple resources	Authentication or Authorization
Hybrid	The function can be provided directly at the resource or be provided to multiple resources	Code mapping services could be run at resource-level or shared by multiple resources

Business functions manifest themselves as VP components, concrete software products that either implement one or more functions on resource-level (e.g. a FAIR data point) or platform-wide (e.g. the Authentication and Authorization infrastructure). Chapter 5 provides an overview on available components.

4.1. Meta data provision

The business function meta data provision must be available at resource-level and is the first stage for a resource to contribute to the VP and prerequisite for basic functions of the VP that other levels build on.
It means that a resource provides minimal information about itself accessible in a standardised, machine interpretable way (i.e., 'metadata for machines'). The 'Data Catalogue Vocabulary' (DCAT), a widely used vocabulary, recommended by the World Wide Web Consortium (W3C) for describing resources for computers is used as the base model for the EJP-D metadata. Next to *having* metadata in terms of the DCAT-based EJP RD metadata model available, a standard mechanism is needed for resources to provide automatically (rather than manually) these metadata. The FAIR Data Point (FDP) specification describes the standard way by which the VP expects resources to provide access to their DCAT-based metadata (see https://specs.fairdatapoint.org/ and https://doi.org/10.1162/dint_a_00160).

4.2. Meta data discovery

Once resources expose their meta data, it must become discoverable through the virtual platform. The business function meta data discovery provides this through a resource-wide function that is currently provided through the VP index component.

4.3. Data Discovery

This business function enables discovery of anonymous/ aggregate/ count information captured within the data records themselves. A query request will typically return information as either yes/no, or counts, though it may return limited information from the data without exposing the records themselves (for example, a list of HPO codes contained in the dataset).
Level 2 queries allow researchers to determine if the resource is likely to contain relevant information to investigate their research question according to their respective study design and methodology. Level 2 queries are intentionally limited only to resource discovery – does the resource contain data of interest, and/or how much. Queries at the level of an individual are not within the scope of Level 2.
At level 2, resource discovery queries are sent from any query point in the VP network (e.g., the EJP RD VP portal) and carried out programmatically using the GA4GH Beacon 2 framework standard. The Beacon2 framework provides an API and a JSON exchange format for query and results.
The resource must expose its data and/or metadata via the GA4GH Beacon2 framework standard according to EJP RD filters and models (GitHub - ejp-rd-vp/vp-api-specs: API specification of VP).

4.4. Data querying

This business function provides complete data discoverability on record-level datasets, thus enabling resources to be level-3 compliant. The outcome of these queries is not limited to aggregated/counts responses as in Level 2, but any data table described by any query that follows the EJP RD CDE semantic model representation (for example, retrieve a table that contains phenotype and genotype information from all patients that are diagnosed with a certain rare disease). For this reason, level 3 doesn't require a set of allowed predefined queries as Level 2 does, so both sensitive and non-sensitive information can be retrieved from these queries.

For resources that want to be queryable at Level 3, the data must be represented and exposed in a structured, machine-readable format that follows the Clinical And Registry Entries (CARE) Semantics Model (https://github.com/ejp-rd-vp/CARE-Semantic-Modell.. The complete set of requirements for resources to engage in the EJP RD VP is still being worked out, including metadata requirements to expose the semantic data model, as well as secuirty and privacy aspects.

For FAIRified record-level data (that follows the EJP RD CDE Semantic Model), the CDE-in-a-box tool ([https://github.com/ejp-rd-vp/cde-in-box) allows the access through

complex queries (SPARQL) in a secure Triplestore repository controlled by credentials, namely GraphDB (https://graphdb.ontotext.com/).

4.5. Model data alignment

The CDE semantic model was built to represent, in 'ontologised' linked data form, the CDEs (see chapter 3.4.1) 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.

4.6. Semantic Mappings

Resources may express their data in a variety of terminologies. Depending on the use case, e.g. data discovery it might be necessary to map codes from one terminology to another. As a consequence, the VP provides the business function "semantic mappings" to offer services for applications using the VP to perform this mapping. The business function is hybrid, meaning that such a mapping could be done on resource-level or a service is provided shared by multiple resources. Currently following mapping services have been identified and RESTful APIs have been defined to perform the mapping.

Table 8: Available semantic mappings

Mapping	Description	API specification
Gene/Disease Mapping Service	This API focuses on Diseases and Genes related to them, Orphanet proposes a link between genes and diseases related to them and this API implements that.	https://github.com/ejp-rd-vp/Orphanet-GenesDisease-Mapper-API
Orphanet Mapping Service	Mapping between Orphanet codes and external terminologies	https://github.com/ejp-rd-vp/Orphanet-Mapping-API
Disease Hierarchy Query Expansion Service	API to support the navigation the levels of medical classifications.	https://github.com/ejp-rd-vp/orphanet_hierarchies_service

4.7. Authentication and authorization

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 9: 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.

4.8. Pseudonymisation

The platform-wide pseudonymization function 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.

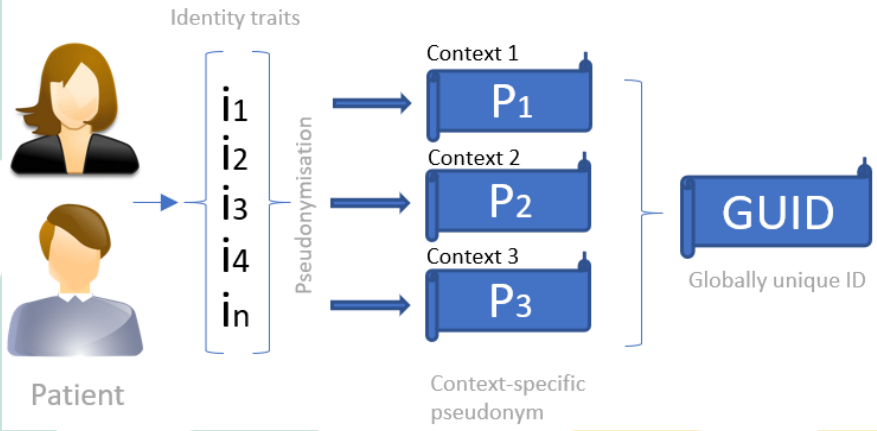

Figure 5: Pseudonymization and context-specific patient identifiers

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.

4.9. Record linkage

The record linkage function should provide 2 distinct elements built on top of the pseudonymization function:

(a) **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.

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

Figure 6 : Process of utilising different contexts to merge different data sources for secondary use 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 pseudonymized 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 pseudonymized and re-identification by a trusted third party might still be possible. For anonymisation the pseudonyms must be removed irrevocably.

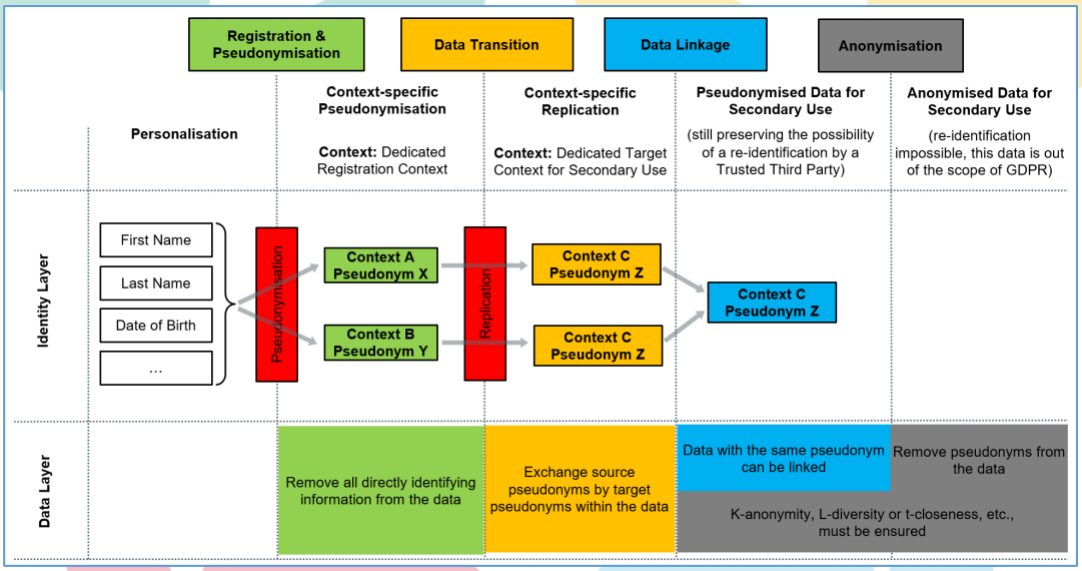

Figure 6 : Process of utilising different contexts to merge different data sources for secondary use

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

4.10. Access and consent control

The access and consent control function is a function that manages access to sensitive record-level data on a resource-level, specifically required by data queries. It manages the access to the data with respect to

(a) Authentication against the authentication and authorization infrastructure by means of the protocols provides by AAI
(b) Checking authorization to verify that a given user is authorized to access a particular data set.

The second aspect of this function 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.

4.11. Federated analysis

Federated analysis is a cross-level business function that allow resources to participate in a federated analysis task, e.g. performing a count on patients with a specific disease across multiple resources. The main requirements for enabling a resource to contribute to federated analysis on the VP are:

- the semantic models for resource metadata (level 1) and data elements (Levels 2 and 3),
- a mechanism to access (meta)data in terms of the semantic models (levels 1-3),
- and the ability to run an analysis method in a privacy-preserving manner.

Additional requirements allow for increasingly more powerful and automated forms of federated analysis.

The minimal use case is a meta-analysis on metadata retrieved from each resource via their FDPs (level 1).
The ultimate use case is fully automated analysis over multiple resources where all steps are automated: finding the data, accessing data via a specified protocol, semantically linking data elements across resources, checking access conditions against what is required for the analysis, authentication, authorisation, and attribution.

Federated analysis is thus already possible for levels 1 and 2 where data are considered not sensitive by definition. For level 3 the analysis is limited to what can be authorised by the resource being queried. Importantly, federated analysis on the virtual platform can be performed by 'visiting data'.

Beyond the federated data querying, analysis can be envisioned in two ways:
The analysis is carried out 'at source', the results from each source are aggregated. In most cases the aggregated results are not sensitive. Each source can keep a record of data use.

The analysis is carried out in one or more secure, transient locations on data that are retrieved from the sources by queries (e.g., as in Level 3) into a cache. The overall result is stored, but the temporary locations are destroyed after the analysis.

The first is potentially the most powerful. It requires that a resource provides compute capabilities 'at source'. In principle analysis can be carried out on sensitive data if the results are not sensitive. The VP specifications do not yet define minimal requirements for compute capabilities 'at source' beyond queries. Therefore, the second scenario is more directly available if levels 1-3 have been implemented.

5. Virtual Platform components

As described in the previous chapters, virtual platform components are software products and services that provide one or more business functions on either a

resource-level or being provided as a platform-wide service to be shared by multiple resources. This chapter provides an overview of available components.

5.1. FAIR Data point

This tool helps make your resource discoverable by making your metadata available, adding it to the VP index. This can be installed alone, or via the CDE-in-a-box or FAIR-in-a-box solutions. This makes your resource's metadata appear in the VP. With FDP you can present your metadata in connection level 1 to the VP. Using the FAIR-in-a-box solution also provides limited non-authenticated Level 2 access – independent of Beacon2 – via the "grlc" tool that is distributed with FAIR-in-a-box. Grlc provides anonymous/aggregate record level data from pre-approved queries intended to be exposed without authentication. The FDP reference implementation (alone or via CDE-in-a-box/FAIR-in-a-box) can be either hosted on your own infrastructure, through an agreement with a non-commercial provider, or by one of several FAIR-compliant commercial hosting providers, including Castor and MOLGENIS (described below) or FAIR Data Systems S.L., among others.

5.2. CDE-in-a-box / FAIR-in-a-box

These systems allow you to easily install the requirements to implement the CDE models in your registry. It will present your metadata as required by the DCAT standard used in EJP RD which is essential for connection level 1. It will also provide the ability to answer queries about your data in an aggregated fashion, helping you connect to level 2 and 3 to the VP.
CDE-in-a-box (CiaB)/ FAIR-in-a-box (FiaB) can be either hosted on your own infrastructure, through an agreement with a non-commercial provider, or by one of several FAIR-compliant commercial hosting providers (at their commercial consulting and/or hosting rates).
The primary difference between these two "boxes" are:

- FiaB has a fully automated installer;
- FiaB uses Docker volumes (less likelihood of accidental deletion of data);
- FiaB is fully standalone and does not remain dependent on installation folders or Docker networks;
- FiaB is ready to use your SSL certificate, to provide https access to your FAIR Data Point,
- FiaB has a variety of optional accessory tools, such as metadata updaters and a pre-configured grlc service to serve non-authenticated, aggregated Level 2 queries.

CDE-in-a-box is available at https://github.com/ejp-rd-vp/cde-in-box

5.3. Beacon-in-a-box

This system allows you to easily install the Beacon endpoint that can be queried by the VP Portal. It will present your metadata as required by the DCAT standard used in EJP

RD which is essential for connection level 1. With Beacon-in-a-box you can also expose your data in connection level 2, e.g. to query portals.
Beacon-in-a-box is available at https://github.com/ejp-rd-vp/EJP-RD-Beacon-in-a-Box

5.4. VP Index

The VP Index registers and indexes metadata of resources that are part of the Virtual Platform. In one sense we can say that a given resource is part of the Virtual Platform if its metadata is available at the VP Index.

The VP Index provides semantically rich metadata in RDF as defined in the source FAIR Data Points. These metadata are based on the W3C DCAT recommendation and have been extended to accommodate specific needs from the EJP-RD community (see section 3.1 for the EJP-RD metadata schemas).

The FAIR Data Point, and consequently the VP Index, uses a number of standards to structure, define the meaning and expose metadata. Among these standards we have HATEOAS REST, RDF, DCAT and Linked Data Platform.

To facilitate the use of the metadata by the VP Portal, a special interface has been developed with a subset of the available metadata items as required by the VP Portal.

URLs:

- VP Index: https://index.vp.ejprarediseases.org.
- API documentation: https://index.vp.ejprarediseases.org/swagger-ui/index.html
- VP Portal interface: https://qv.index.vp.ejprarediseases.org
- VP Portal interface API documentation: https://qv.index.vp.ejprarediseases.org/swagger-ui.html
- VP Portal interface test deployment: https://qb-index.ejprd.semlab-leiden.nl

5.5. RD-Nexus

RD-Nexus is a flexible web-based tool that enables data discovery. Thanks to the network-oriented approach of RD-Nexus, users can discover data from a Federation of Networks while retaining control over the amount of shared information. Admin users can both create new networks and request to join existing networks.

5.6. Molgenis

MOLGENIS is data platform for researchers to accelerate scientific collaborations and for bioinformaticians who want to make researchers happy. It is free, open source, simple to use and available via http://molgenis.org. The newest versions include the 'FAIR Data Hub' which offers integrated database template, ontologies and APIs for

FAIRification including CSV, Excel, GraphQL, JSON+LD, RDF, Beacon v2 and FAIR Data Point. Currently, MOLGENIS has more than 100 active running servers and is used by four ERNs.

5.7. Disease Hierarchy Query Expansion service

This service is dedicated to support implementation and usage of Orphanet's classifications. Rare disorders are often multi classified in several medical domains (group of disorders). Datasets from resources are usually annotated at the most accurate level (specific list of disorders or subtype of disorder). The service allows to retrieve such annotated datasets or records by querying at higher level (group of disorders).
This API focuses on Rare diseases classifications based on ORDO (Orphanet Rare Diseases Ontology). Therefore ORPHAcodes or ORPHAcodes IRI are used.
The service attend to return from any ORPHA code queried a list of ORPHA codes related to it within the different levels of its classifications. The returned list could be used to expand the query to a resource from a single searched ORPHAcode and send to the resource endpoints the full list of related ORPHAcodes. The service could also be used at resource level, allowing to directly search by a unique ORPHAcode but returning the list of datasets or records annotated with any ORPHAcodes related to the classifications.

API parameters:
{*ORPHAcode*} : an ORPHAcode of a clinical entity as a node to start with
(Mandatory) IRI or code are accepted.
ex: /traverse?code=http://www.orpha.net/ORDO/Orphanet_558
or
/traverse?code=558

{*way*} : from the node, path to levels. Expected 'up' (to "parents"), 'down' (to "childs")
(optional. Without, retrieve +1/-1 from node) /traverse?code=558&way=down will return for each level, all the descendants of the specified Code

{*level*} : level to reach from the ORPHAcode as starting point (optional. Without, return all available levels)

{*classif*} : specify 1 to n Orphanet classifications ID to traverse (optional. Without, return results from all classifications)

Returns: apiVersion, and a list of existing "**parents**" or "**child**" from the queried unique ORPHAcode with label and code for each. Usage of {way} parameter will return level

Endpoints:
/ClassifTraversal/hierarchies
Returns id, name, description, apiVersion, supported_classifications endpoints, sampleRequests of the service.

/hierarchies/list
Returns **classifications_list, classification_number, name** of available classifications

Full documentation, accessible endpoints and sample queries available on GitHub :
https://github.com/ejp-rd-vp/orphanet_hierarchies_service

5.8. Gene/Disease Mapping Service

This service focuses on Diseases and Genes related to them, Orphanet proposes a set of relations between genes and diseases related to them and this API implements that. Based on Orphanet knowledge base (and ORDO Orphanet Rare Diseases Ontology) content, this service could be used to search a list of disorders by gene or genes by disorder.
Annotated resources or datasets by disease concept could be retrieved by a gene search or reverse, as annotated resources or datasets by genes could be retrieved by a disorder query.

We based our query mechanism on gene by using HGNC (HUGO Gene Nomenclature https://www.genenames.org/) ids or gene symbol.

Gene to disease(s)
This part of the service allows query based on HGNC codes or gene symbol

API parameters
'gendis/find?by={hgnc code or Symbol}&input={the value of hgnc code or Symbol}
ex: [http://purl.org/gendis/find?by=hgnc&input=3603]
ex: [http://purl.org/gendis/find?by=symbol&input=FBN1]

You will obtain the list of diseases linked to the specific gene (by code or symbol)

Returns ORPHAcode and label:
"orphaCode":
"http://www.orpha.net/ORDO/Orphanet_284963", "label":"Marfan syndrome type 1"

"inferredResourceResponses":
In Orphanet Knowledge base, genes are linked to specific entities such as "subtype of disease". For instance, FBN1 gene will be linked to Marfan type 1 and TGFBR2 gene will be linked to Marfan type 2. However, resources are often annotated at the level of "disorder" clinical entities. (Marfan syndrome in this example)

The API will return a result expanded to this "upper" level (from subtype to disorder level) when relevant as "inferredResourceResponses" :

"inferredResourceResponses":
{"orphaCode":"http://www.orpha.net/ORDO/Orphanet_558", "label":"Marfan syndrome"}

"orphanetEndpoint"
We added a feature that allows to query the Orphanet Resource Endpoint (Biobanks and Registries) based on the returned list of Orphacodes
(Example:
"orphanetEndpoint":
"http://155.133.131.171:8080/Orphanet/resource/search?code=300382&code=91387&code=2623&code=2833&code=284963&code=284979&code=3449&code=969&code=1885&code=558")

Disease (ORPHAcode) to Genes
This part of the service allows query based on Orphacodes to retrieve genes linked, with HGCN code and Symbol

API parameters
'gendis/find?by=orphacode&input={orphacode or IRI}
ex:
[http://155.133.131.171:8080/GENES/gendis/find?by=orphacode&input=http://www.orpha.net/ORDO/Orphanet_284963]
ex: [http://155.133.131.171:8080/GENES/gendis/find?by=orphacode&input=284963]

You will obtain the list of genes linked to the clinical entity orphacode concept, with HGNC code, symbol and "relation" between orphacode concept and each gene.
directLinkedResponses hgnc "3603" symbol "FBN1" relation "Disease-causing germline mutation(s) in" inferredResponses [] apiVersion "v0.1"

directLinkedResponses : the clinical entity concept (orphacode) is annotated with the gene in the Orphanet knowledge base

inferredResponses : the clinical entity concept (orphacode) is not directly annotated with the gene but a subtype of the entity is.

Example: FBN1 gene is linked to Marfan type 1. TGFBR2 gene is linked to Marfan type 2. The Marfan syndrome (orpha:558) is not linked directly to any gene, but has "marfan type 1" and "marfan type 2" as subtypes. Therefore, in order to obtain genes related to "Marfan Syndrome" we propose an inferred response including subtypes.

ex: [http://155.133.131.171:8080/GENES/gendis/find?by=orphacode&input=558]
"inferredResponses"
"orphaCode": "284963"
"label": "Marfan syndrome type 1"

linkedResponses
"hgnc":"3603", "symbol":"FBN1", "relation": "Disease-causing germline mutation(s) in"
"orphaCode":"284973"
"label": "Marfan syndrome type 2"

linkedResponses

"hgnc": "11773", "symbol": "TGFBR2", "relation": "Disease-causing germline mutation(s) in"

Full documentation, accessible endpoints and sample queries available on GitHub: https://github.com/ejp-rd-vp/Orphanet-GenesDisease-Mapper-API

5.9. Orphanet mapping service

This service, based on ORDO (Orphanet Rare Diseases Ontology) allows to retrieve codes from different terminologies, using ORPHACode.

Orphanet maintains an alignment between clinical entities with external terminologies or resources (ICD-10, OMIM, UMLS, MeSH, MedDRA and GARD). The mapping includes a semantic link that specifies the relationship between an ORPHAcode and the external terminologies. The alignments specify the comparability between terminologies by defining if the concepts are perfectly equivalent (exact mapping) or not:
E (Exact mapping: the two concepts are equivalent).
NTBT (ORPHAcode's Narrower Term maps to a Broader Term).
BTNT (ORPHAcode's Narrower Term maps to a Broader Term).

Endpoint
http://purl.org/orphanetws/Mapping?

API parameters
/map?from={origin}&code={code}&to={destination}
will return results for a clinical entity (resources), from any terminology listed to another (orphanet, omim, umls, mesh, meddra, icd).

from= {origin}
Where {origin} is the terminology "source" as input.
The value of Parameter 'from' must be one of :{orphanet, omim, umls, mesh, meddra, icd}.

code={code}
Where {code} is the reference of code in the {origin} terminology
Ex:
from=orphanet&code=558
from=icd&code=Q87.4

to={destination}
Where destination is the terminology targeted
The value of Parameter 'destination' must be one of :{orphanet, omim, umls, mesh, meddra, icd}.

Please note that ORDO (Orphanet Rare Diseases Ontology) which contains mappings from Orphanet to the list of available terminologies is always use as "pivot".
Ex: map?from=icd&code=Q87.4&to=omim
Will use actually under the hood map?from=icd&code=Q87.4&to=orphanet then "from" Orphanet to "omim" based on the possible ORPHAcodes returned first

Several mappings could be obtained by using several "to" parameters
Ex: map?from=orphanet&code=558&to=omim&to=icd&to=meddra
Will return from the orphacode 558 mappings to omim, icd and meddra at once

Results returns IRI of each mapping in the different terminologies and are ordered by types of relations (exactMatch, nTBT, bTNT)

Queries examples:
ex: [http://purl.org/orphanetws/mapping?from=icd&code=Q87.4]
ex: [http://purl.org/orphanetws/mapping?from=icd&code=Q87.4&to=omim]
ex :
[http://purl.org/orphanetws/mapping?from=orphanet&code=558&to=omim&to=icd]

Full documentation, accessible endpoints and queries sample available in GitHub:
https://github.com/ejp-rd-vp/Orphanet-Mapping-API

6. The EJP-RD Discovery Portal

The EJP-RD Discovery Portal – available at https://vp.ejprarediseases.org/ has been implemented to illustrate the network of connected resources and showcase the different connection levels. Figure 7 provides an overview of how the query interface is presented to end-users:

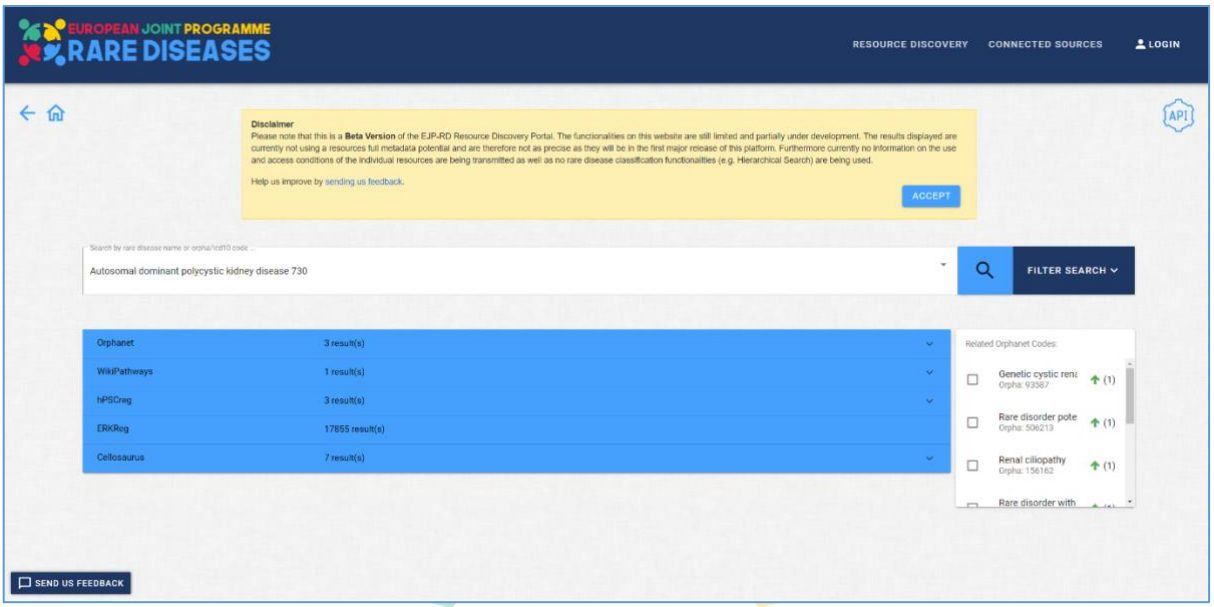

Figure 7. Screenshot of the EJP-RD resource discovery portal.

The portal is currently divided into three different pages: the landing page, the Resource Discovery page and the VP Network Resource page.
On the landing page, it is possible to access the EJP's own FAIRification page, the VP Index with tools and partners of EJP RD and the imprint. In order to give users an overview of the opportunities offered by the Virutal platform, exemplary Use cases are visualized.
Level 1 resources are displayed as a list via the VP Network Resources page. All resources are shown with their logo, a description and some additional information as figure 8 illustrates:

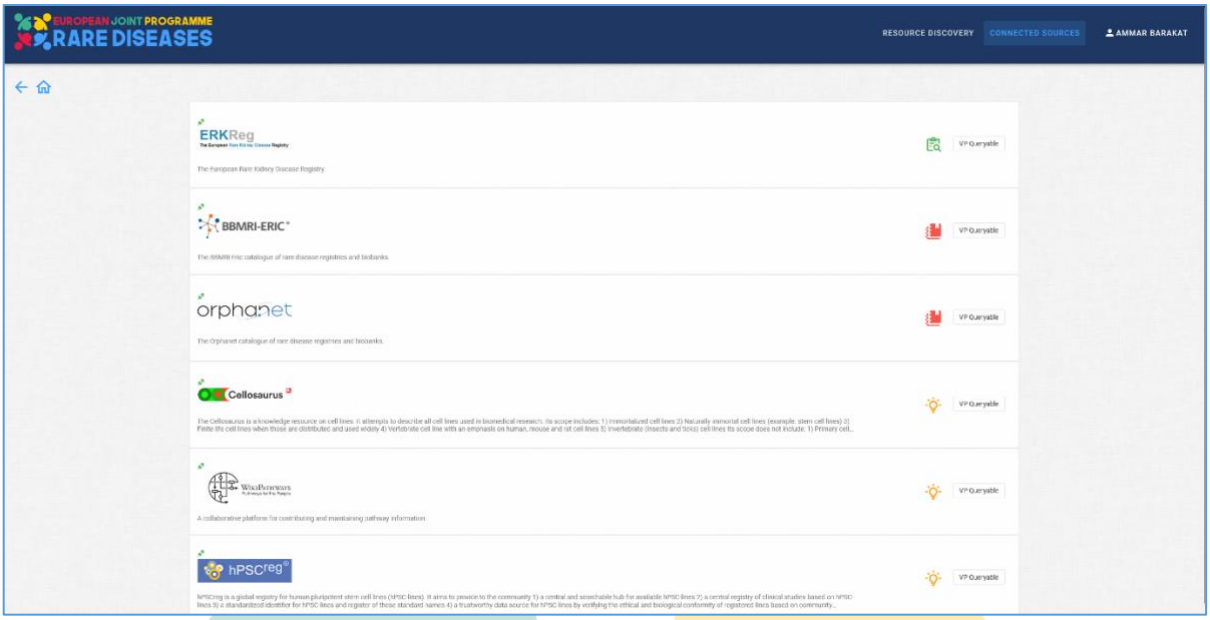

Figure 8. Connected resources as illustrated in the EJP-RD discovery portal.

The list of connected resources is retrieved through the VP index component.

Level 2 Querying is carried out through the search bar on the Resource Discovery page. For this, the ORPHA code, the ICD-10 code, the OMIM code or the name of the rare disease being searched for is entered into the search bar. The enteres information wikl always be mapped to an ORPHA code using the Orphanet Mapping service. The user can also specify the search query by adding filters. This search query, with the ORPHA code of the disease and the filters, is sent to the resources. The answers returned by the respective resources are displayed in the "Result" list. This result list thus represents the level 2 of the resources.
In addition to all this, the user can also login on the website to request some more resources and use additional filters, like gender, minimum age and maximum age without any restrictions.
For level 2 querying the portal supports the Beacon2 API for level-2 discovery of catalogue, dataset contents as well as level-2 discovery of individuals and samples.
As mapping components are available as described in the previous chapter, the portal also supports:

- Search for ORPHA codes with respect to hierarchy
- Extension of search to OMIM and ICD-10 codes
- Search for genes

The user interface was adapted and optimized after several workshops with endusers conducted by a dedicated working group.

7. Conclusion and summary

This deliverable describes the purpose and design principles of the EJP-RD virtual platform outlining key abstractions, functions identified for the VP and components implementing these functions. Quality and sustainability measures for components have been described.

The EJP-RD Discovery portal illustrates how resources linked to the virtual platform can be queried on multiple levels.

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://github.com/ejp-rd-vp/vp-api-specs/tree/v4.0_spec
EJP-RD query API	API specification	https://github.com/ejp-rd-vp/vp-api-specs
SAML 2.0	Authentication	SAML 2.0 - Wikipedia
Open ID Connect	Authentication	OpenID Connect – Wikipedia
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 recognised registry (e.g., Bio.tools), IRDiRC, ELIXIR CDR, RIR, BioContainers, 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 C: List of tables

Appendix E: List of figures