

Enabling FAIR Discovery of Rare Disease Digital Resources

Rajaram KALIYAPERUMAL^{a,1}, Núria QUERALT ROSINACH^a, Kees BURGER^a,
Luiz Olavo BONINO DA SILVA SANTOS^{a,b}, Marc HANAUER^c and Marco ROOS^a

^a Leiden University Medical Center, Leiden, Netherlands

^b University of Twente, Enschede, Netherlands

^c Inserm, Paris, France

Abstract Background: Integration of heterogenous resources is key for Rare Disease research. Within the EJP RD, common Application Programming Interface specifications are proposed for discovery of resources and data records. This is not sufficient for automated processing between RD resources and meeting the FAIR principles. Objective: To design a solution to improve FAIR for machines for the EJP RD API specification. Methods: A FAIR Data Point is used to expose machine-actionable metadata of digital resources and it is configured to store its content to a semantic database to be FAIR at the source. Results: A solution was designed based on *grlc* server as middleware to implement the EJP RD API specification on top of the FDP. Conclusion: *grlc* reduces potential API implementation overhead faced by maintainers who use FAIR at the source.

Keywords. rare disease, metadata, data management, information storage and retrieval, FAIR, semantic web

1. Introduction

Integrating sensitive, sparse and heterogenous datasets is key for accelerating research and patient care in the Rare Disease (RD) field. RD resources such as biobanks, patient registries and omics datasets are often dispersed and hard to find in order to integrate. The European Joint Programme on Rare Diseases (EJP RD) [1] brings together researchers across the RD field to address data integration problems by implementing the FAIR (Findable, Accessible, Interoperable and Reusable) data principles [2]. One of the goals of the EJP RD project is to build a so-called ‘virtual platform’ (VP) via which centralized and federated RD resources can be found and interrogated. Importantly, the underlying infrastructure needs to enable automated processing between the resources. A common API specification has been proposed that programmers can use in their code to discover resources and data records [3]. However, documented APIs are not sufficient for a dynamic, AI ready VP that is also interoperable outside of the EJP RD project: resources and their content need to be self-describing with global, machine readable models, in line with the FAIR principles. This can be achieved by using global ontologies defined in the Resource Description Framework (RDF). For many system designs, including the EJP RD VP, it will be a requirement that common APIs can work

¹ Corresponding Author: Rajaram Kaliyaperumal, Leiden University Medical Center, Leiden, Netherlands, E-Mail: r.kaliyaperumal@lumc.nl

seamlessly with this technology. In this paper, we propose a design to implement common APIs as a complement to resources that apply RDF to implement FAIR principles. The Orphanet data catalogue was used as an example.

2. Methods

The FAIR Data Point (FDP) is general-purpose software that helps users expose metadata of their resources in a FAIR manner, particularly to meet the FAIR requirement of machine readability [4]. An FDP therefore relies on Semantic Web technologies by design, including the Resource Description Framework (RDF), triple store (an RDF database) and the SPARQL Protocol and RDF Query Language (SPARQL) a W3C recommended standard to manipulate and query RDF [5]. FAIR for machines in the FDP is achieved in two ways. First, the FDP applies widely used RDF-based vocabularies such as the Data Catalog Vocabulary and Dublin Core to describe the resource in machine readable terms. Secondly, the FDP uses triple stores to store and serve the metadata. The REST API of the FDP also supports the Linked Data Platform (a W3C recommended specification to manipulate RDF documents via HTTP) GET call specification [6].

Within the EJP RD project, a set of common API specifications is also being defined for programmers, including an API for discovery of resources and data records. Version 0.2 of this API aims to address discovery of resources at a metadata level. The proposed APIs are based on the OpenAPI specification to implement REST API calls that responds with JSON documents. A requirement for the EJP RD Virtual Platform is that the APIs and RDF-based FAIR data points can work together. This is achieved by *grlc*, a server application that converts SPARQL queries into API calls and lets users transform the API call's response to JSON documents [7].

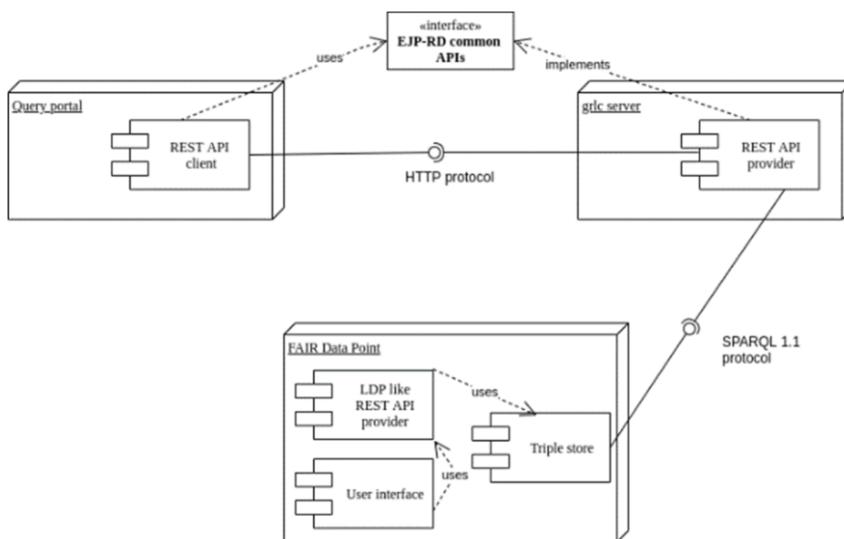


Figure 1. Design to implement EJP-RD API specification on top of a FAIR Data Point.

3. Results

We implemented an FDP that exposes metadata of biobank and patient registries captured by the Orphanet catalog² and configured it to store its RDF documents in a triple store. We propose a design where the EJP RD common APIs specification is implemented on top of the FDP (Figure 1). The *grlc* server is used as middleware that converts EJP RD specific common API calls into SPARQL queries and the response of the SPARQL queries into JSON documents. We created a reference implementation for the “Search resource” API call³. This API call searches biobanks or patient registries for a given disease code from the Orphanet catalog’s FDP. Figure 2 shows the request URL and response for the disease code ‘Orphanet_586’ for our implementation.

```

Request
http://grlc.io/api-git/rajaram5/SWAT4HCLS-hackathon-2021-ejprd-grlc/search-
resources?orphaCode=http%3A%2F%2Fwww.orpha.net%2FORDO%2Forphanet_586

Response
{
  "description": "Cystic fibrosis biobank",
  "homepage": "http://www.orpha.net/consor/cgi-bin/OC_Exp.php?lng=en&Expert=321039",
  "id": "https://ejprd.fair-dts.surf-hosted.nl/orphanet-catalog-tdp/biobank/Ra2b2175-a449-46d4-afde-784ada6136bd",
  "name": "Cystic fibrosis biobank",
  "type": "http://purl.org/biosemantics-lumc/ontologies/dcat-extension/Biobank"
},

```

Figure 2. *grlc* REST API’s request URL and response to find resources for disease code ‘Orphanet_586’

4. Discussion

In our work we showed how generic FAIR software solutions such as the FAIR Data Point and triple stores can be used to achieve specific findability and interoperability goals. We have presented a straightforward solution to accommodate common APIs for programmers without compromising FAIR principles. For implementors who maintain data sources FAIR at the source, implementing a common API might be a substantial overhead. With our solution to use middleware components such as *grlc* we show how this overhead can be reduced. A drawback in our reference implementation is that we only converted successful API call responses (http 200). Converting other API call responses such as ‘not found’ (http 404) and ‘unauthorized’ (http 401) might require further extension to the *grlc* server.

References

- [1] “EJP RD.” [Online]. Available: <https://www.ejprarediseases.org>. [Accessed: 10-Feb-2021].
- [2] M. D. Wilkinson et al., “Comment: The FAIR Guiding Principles for scientific data management and stewardship,” *Sci. Data*, vol. 3, p. 160018, Mar. 2016, doi: 10.1038/sdata.2016.18.
- [3] “EJP RD Query Builder API specs.” [Online]. Available: https://github.com/ejp-rd-vp/query_builder_api. [Accessed: 10-Feb-2021].
- [4] L. O. Bonino da Silva Santos et al., “FAIR Data Points Supporting Big Data Interoperability,” in *Enterprise Interoperability in the Digitized and Networked Factory of the Future*, M. Zelm, G. Doumeingts, and J. P. Mendonça, Eds. ISTE Press, 2016.
- [5] “SPARQL 1.1 Protocol.” [Online]. Available: <https://www.w3.org/TR/sparql11-overview/>. [Accessed: 10-Feb-2021].
- [6] “Linked Data Platform 1.0.” [Online]. Available: <https://www.w3.org/TR/ldp/>. [Accessed: 10-Feb-2021].
- [7] A. Meroño-Peñuela and R. Hoekstra, “Grlc makes github taste like linked data APIs,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9989 LNCS, pp. 342–353, doi: 10.1007/978-3-319-47602-5_48.

² Orphanet: catalog <https://www.orpha.net/consor/cgi-bin/index.php>

³ Reference implementation: <http://grlc.io/api-git/rajaram5/SWAT4HCLS-hackathon-2021-ejprd-grlc>