International FAIR Digital Objects Implementation Summit 2024

Extended Abstracts

https://doi.org/10.52825/ocp.v5i.1180

Published: 18 Mar. 2025

An FDO-Based Exchange Interface for Research Data

A Conceptual Design of an FDO Based API

Roland Johannes[1,*], Fidan Limani[2], Yudong Zhang[3], and Daniel Schiffner[1]

[1]DIPF – Leibniz Institute for Research and Information in Education, Germany

[2]ZBW – Leibniz Information Centre for Economic, Germany

[3]GESIS – Leibniz Institute for the Social Sciences, Germany

*Correspondence: Roland Johannes, r.johannes@dipf.de

Abstract. Data exchange is a key activity that Research Data Centers (RDCs) provide. To do so, current interfaces typically cater to community-specific needs, leading to a fragmented approach when dealing with inter-RDC exchange scenarios. The interface we propose aims to facilitate data exchange and scientific workflows across RDCs in the German National Research Data Infrastructure (NFDI) initiative. We balance generic and domain-specific requirements by employing two key principles - a RESTful architecture style and the FAIR Digital Object (FDO) model - by mapping of the resources of the API to the layers of an FDO. The interface is designed to use essential HTTP methods to handle FDO instances, which not only includes the datasets, but also the HTTP requests and responses themselves. Additionally, our approach facilitates scenarios such as data enrichment: datasets that need to be combined or extended with other resources can be still represented as FDOs. Early deployments with two partners indicate that transitioning to an FDO-based approach presents certain challenges, but also a significant potential for improved data exchange and collaboration. In going forward, we aim to evaluate our approach within the FDO community, adapting to evolving standards while maintaining a minimally viable model for practical exploration. Ultimately, our interface aspires to streamline data-related services, promoting a cohesive ecosystem for research data management.

Keywords: Data Exchange, FDO, REST API

Introduction

As part of the German National Research Data Infrastructure (NFDI) initiative, one aspect for the KonsortSWD project is the support of data exchange and scientific workflows across the Research Data Centers (RDC). We therefore specify an interface that relies on standardized methods to enable and simplify such an exchange.

Currently, RDCs typically have an interface that supports community-specific data-related exchanges. If an exchange between RDCs is required, they must rely on individual agreements between the partners. However, as the number of RDCs increases, the need for finding a common interface - a common denominator of the operations - to support such needs rises. This can either be a complex, “catch all” interface to support all the involved RDCs, or a simple interface to provide only some of the use cases. Considering this situation, the challenge is balancing between universality and domain-specific requirements.

To address this challenge, we use the following two principles in our approach: (a) the RESTful architecture style, and (b) the FAIR Digital Object (FDO) model to represent the resources. The former is an established approach with a wide acceptance and easy adoption, whereas the latter enables us to encapsulate all the specifics and provide machine-actionability. With these principles in place, FDOs represent data in an environment of community developed services. Registries combine multiple aspects, such as registration of services, data profiles, FDO types, Persistent Identifiers (PID).

The core of our approach is to assure that the FDOs can be handled with the mechanisms of a RESTful API across all services, not to design an interface for the data-related services. Thus, we do not differentiate between the RDC services. Fig. 1 depicts the proposed interface, with example services shown at the top and the interface endpoints at the bottom. We use the GET, POST and PUT HTTP methods to read, create and modify elements of an FDO instance. Additionally, we provide the HTTP HEAD method to check certain elements, e.g. PID of an FDO instance. This enables efficient request handling. With this basic set of methods and a service providing FDO instances allows one to gain control over an FDO instance. In our understanding everything is an FDO instance, including requests and responses between a client and a server, as they should also be machine-actionable.

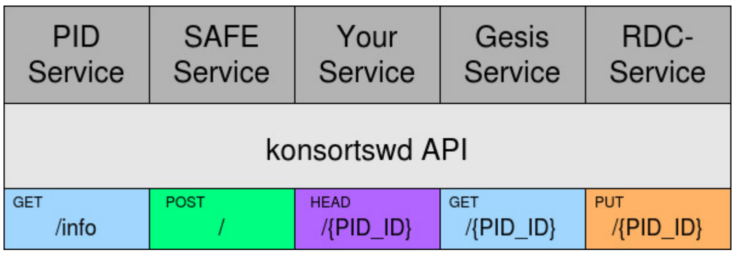

Figure 1. *Different services using a common API interface*

This enables the following scenario: A dataset located at a RDC may benefit from including data from other resources in the FDO ecosystem. The responsibility of other data types is defined by the dedicated provider of those FDOs. In order to get a description of a service, there are aliases for PIDs, such as the endpoint “info”. A GET request to https://researchservice.anywhere.world/info provides an FDO describing this service. This FDO contains a description of the available operations provided by the service, including available parameters, how to invoke them, and much more.

The “network effect” also applies to our API: the more the interface is used, the higher its value. Services, such as those used for registration, routing, distribution, storage, etc. should all use the same API specification (Fig. 1). A request can be forwarded between services as needed until it is resolved. If a consumer fetches an FDO using a PID, the request is forwarded to a network of PID services, which enables fetching FDOs regardless of their location. And this is a principle for every kind of service. In addition to its features, a service can also rely on (and delegate to) other available services. The FDO is the fundamental element of all services, and the API is solely based on it.

Let us consider a scenario of data enrichment and the support that our API specification would provide. A service could merge a few datasets and represent the resulting one as an FDO instance. The API can be used to get the necessary information to invoke such a merge operation at a service. For this, an FDO describing the service can be used by a consumer to inspect all functions available to enable the execution of the dataset merge. Our API specification enables all these possibilities by dealing with few HTTP methods on an FDO instance. The keyword here is “enables”: the API specification is defined to enable services to communicate b/w themselves. How this is implemented is independent from our API specification.

We already rolled out our interface with two partners and their existing infrastructures. Our first impression is that a migration to a FDO-based workflow can be a big challenge. In one case, the partner provides an FDO-based service using a protocol to work with objects (datasets, in this case, see DOIP[1] [https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf]). The first step towards adopting our API is to have FDOs that represent their data collection. This is done using an API gateway, that is part of our proof of concept. This enables the exchange of data (including metadata) between our partners while still retaining their existing APIs.

We want to evaluate our idea in the FDO community, including implementation scenarios or potential collaboration with existing FDO-based services. We defined our API extendable, potentially covering many data exchange situations. It layers the actual content, including the organization or structuring at both the data and the meta level. However, there is the need to verify these claims. In designing a data exchange interface for data-intensive communities, we assume that other fields would also benefit from this definition. Finally, while the discussions on the FDO standardization still continues, we chose to adopt a minimally viable model to allow an early exploration. As the FDO specification progresses, we will have to adjust to new features.

Data availability statement

The submission is not based on data. It describes an Application Progamming Interface (API) by design.

Underlying and related material

The design is based on the REST (REpresentational State Transfer) principles. It is an architectural style for distributed hypermedia systems. See: Fielding, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. Doctoral dissertation, University of California, Irvine, 2000.

Author contributions

Roland Johannes: Writing – original draft, Software

Fidan Limani: Writing – original draft, Writing – review and editing

Yudong Zhang: Writing – review and editing

Daniel Schiffner: Conceptualization, Supervision, Project administration

Competing interests

The authors declare that they have no competing interests.

References

[1] D. Foundation. “Digital object interface protocol specification,” Accessed: Mar. 1, 2024. [Online]. Available: https://www.dona.net/sites/default/files/2018-11/DOIPv2Spec_1.pdf.