




# HTTP-Based Implementation Strategies for the FAIR Digital Object Framework Identifier Resolution

Luiz Olavo Bonino da Silva Santos<sup>1,2,\*</sup> , Claudenir Morais Fonseca<sup>1</sup> , and Tiago Prince Sales<sup>1</sup> 

<sup>1</sup>Semantics, Cybersecurity and Services group, University of Twente, the Netherlands

<sup>2</sup>Biosemantics group, Leiden University Medical Center, the Netherlands

\*Correspondence: Luiz Olavo Bonino da Silva Santos, [l.o.boninodasilvasantos@utwente.nl](mailto:l.o.boninodasilvasantos@utwente.nl)

**Abstract.** Directly or indirectly, the FAIR principles define a number of requirements for the data and services ecosystem. Among them, there are requirements for the identifiers of digital objects, including the separation between metadata and the objects they describe, the need for identifier to be globally unique and persistent and that the metadata record includes the identifier of the object they describe. In order to pursue an increased level of automation, the FAIR Digital Object Framework defines a predictable identifier resolution behaviour that not only support the access to the target object but also allows the client application to request the reference to a minimal metadata record named Identifier Record containing basic information such as the object's location, its type and reference to its metadata records. In this paper we report and comment an experiment of implementing these identifier resolution behaviours using different HTTP-based approaches.

**Keywords:** FAIR Digital Object Framework, Identifier Resolution, FAIR Principles

## 1. Introduction

Identifiers play a critical role in the digital realm. Normally, an identification system is composed of identifiers, presented as arbitrary sequences of characters, and a resolution protocol. The resolution protocol is responsible for returning content whenever a given identifier is requested. In the context of machine-actionability, it is expected that identifiers' resolution protocols of digital objects behave predictably so that artificial agents can be programmed to consistently interact with these systems.

However, in practice, we can observe that the resolution behaviors of existing identifiers can be unpredictable, leading to significant challenges. One consequence of this unpredictability is that building applications that need to deal with a multitude of identifiers becomes more difficult as their resolution returns different types of results. For instance, the resolution of one identifier returns the actual target object, another identifier resolves to the target object's metadata, and a third identifier resolves to an HTML landing page containing diverse information about the object. These three resolution results, i.e., the object itself, its metadata record and an HTML landing page

are objects of different nature. Consequently, if a given application is handling multiple identifiers and each one resolves to objects of a different nature, the application either cannot properly function or should cater for this diversity.

The FAIR principles [1] introduced several requirements for identification systems and their identifier. The most obvious and direct ones are derived from principles F1 and A1, stating that “(meta)data are assigned a globally unique and persistent identifier” and “(meta)data are retrievable by their identifier using a standardized communications protocol”, respectively. The former, expected that we assign identifiers to both metadata and the objects they identify, and the latter requires that metadata and other types of digital objects can be retrieved by resolving their identifier.

Moreover, in the original FAIR principles paper [1], we have that humans should increasingly “rely on computational agents to undertake discovery and integration tasks on their behalf” and, for this to happen, we need “machines to be capable of autonomously and appropriately acting when faced with the wide range of types, formats, and access-mechanisms/protocols that will be encountered during their self-guided exploration of the global data ecosystem”. From these statements, we can derive that computational agents would benefit from a mechanism where given a discovered identifier, they can ask questions such as *What is this that I have discovered? How can I get more information about it? What can be done with it? and What am I allowed*

These aforementioned challenges in applying the FAIR principles in the current digital communication infrastructure are the main motivators of the FAIR Digital Object Framework (FDOF) [2]. In this paper, we present and discuss the implementation options for the FDOF’s identifier resolution. The paper is further organised as follows: Section 2 identifies and discusses related work, Section 3 introduces the FAIR Digital Object Framework and its identifier resolution behaviour, Section 4 discusses possible implementation strategies for the FDOF’s identifier resolution. Finally, Section 5 provides the final remarks and pointers to future work.

## 2. Related work

Current identification systems (Handle system, URIs, etc.) rely on the user’s discipline and best practices. For instance, on the Web, one could use a URI to identify a particular PDF file [3]. The URI may not resolve to anything or to something other than the identified object, for instance, an HTML page. Similarly, some Digital Object Identifiers (DOIs)[4], a particular implementation of the Handle system, resolve the identifier to the actual identified object, some to its metadata and most to a landing page. For artificial agents, landing pages are particularly challenging because it is not always (if ever) clear which of the potentially many links present on the page corresponds to the object identified by the DOI. The DOI example (DOI:10.47366/sabia.v5n1a3) on its homepage (www.doi.org) is the identifier of an article named “Saberes emergentes de las artes urbanas y cultura de paz. Un estudio de caso en San Salvador”. From a browser or a command line request (e.g., using curl), this identifier resolves, after some redirections, to the landing page of the paper. In the HTML code of this landing page, we can find more than one hundred URLs. Only one of these URLs refers to the actual PDF file of the article. An artificial client would have difficulties in identifying which of these many URLs points to the object of interest.

In the context of the Digital Objects Architecture (DOA) [5], two protocols are defined, the Identifier/Resolution Protocol (IRP) [6] and the Digital Object Interface Protocol (DOIP)[7]. The IRP is used for creating, updating, deleting and resolving digital

object identifiers, and these identifiers are associated with identifier records. The DOIP is used by repositories and registries of digital objects to create, delete, update and get information about these objects.

### 3. FAIR Digital Object Framework

The FAIR Digital Object Framework (FDOF) is a framework which defines a model to represent objects in a digital environment and a mechanism to create, maintain and (re)use these objects. The framework consists of a conceptual model expressed in terms of an ontology [8], a predictable identifier resolution behaviour, a mechanism to retrieve the object's metadata and an object typing system. The main goal of the FDOF is to define a set of features to provide foundational support for the FAIR principles. As the name suggests, the FDOF is inspired by the notion of Digital Objects introduced in [5] and extended to comply with the additional requirements derived from the FAIR principles.

In a scenario where computational agents explore the global data and services ecosystem, they encounter identifiers of different objects. The framework aims at providing features to allow answering the following questions about these encountered identifiers, in a way that can be interpreted by both machines and humans:

- What is the object that is identified by this identifier?
- How can I get more information (e.g., How to handle it? Who can handle it? What is it allowed to do with it?) about this object?
- What can be done with the object?
- What am I allowed to do with this particular object?

Similar to the DOA, the FDOF defines an entity named FDOF's Identifier Record (FDOF-IR), which is a specific type of metadata, containing basic information about the object such as its type, reference to its metadata record(s) and its location. In the upcoming extension of the FDOF, the FDOF-IR will also provide information about operations available for the given object.

The FDOF-IR is defined as a particular type of metadata. In FDOF we opted to differentiate the three pieces of metadata information contained in the FDOF-IR from other types of metadata information (e.g., provenance, keys, serialization format, size, etc.). The reason behind this differentiation is to separate the minimal information required by the framework's resolution protocol from the additional information that can be used by other applications, agreed upon by communities, etc. In this way, we can guarantee that any FDOF-enabled application can identify the type of the object (through the reference to the object's type), directly operate on the object (through the object's location reference) or get more information about the object through reference(s) to the object's metadata record(s). Other information about the object should be placed in the regular metadata record(s).

The FDOF is not intended to replace the current digital communication infrastructure but to complement it providing extra features supporting better machine-actionability to deal with different types of digital objects. Therefore, the FDOF should coexist with these existing infrastructures and, in some cases, leverage their features. If we compare the DOA and the FDOF in terms of the FDOF-IR (or just identifier record or kernel record in DOA's terminology), the FDOF identifier resolution behaviour combines some functionality of both DOA's IRP and DOIP. To allow a minimum impact on the current communication infrastructure, the FDOF identifier resolution behaviour, as default, re-

solves directly to the target object. However, the client has the option to request the identifier to resolve to the FDOF-IR instead. Since the FDOF-IR provides information about the object's type and metadata record(s), to facilitate the interactions, the FDOF resolution behaviour also offers methods to directly request these types of information.

## 4. Implementation strategies for the FDOF's identifier resolution

The FDOF's identifier resolution can be implemented in different ways to provide the functionality mentioned in the previous section. Currently, we are evaluating four possibilities, (1) using HTTP accept headers, (2) as an extension of HTTP with new methods, (3) using HTTP on specific FDOF paths, and (4) using HTTP Signposting [9]. These implementation options are discussed in the following subsections. For simplicity, in these discussions, we mention only the retrieval of objects and, therefore, use the HTTP GET verb as the basis. Other operations such as delete, update, add, etc., are also considered using their appropriate verbs in the same patterns as the GET. Moreover, in the examples below, we used the notation of  $\{ObjID\}$  as a placeholder of the main target object's resolvable Web link. The resolvable Web link represent the object's identifier in a format that can be directly resolved. If the object's identifier is an URI, the resolvable Web link is the URI, whereas if the object's identifier is of different types such as a DOI, the resolvable Web link is the URL of the DOI hosting service plus the actual DOI.

For each of these four implementation strategy options, we have create server-side code in Python<sup>1</sup> and used Postman<sup>2</sup> as the client application.

### 4.1 HTTP with accept headers

In this implementation option, we utilise only the standard HTTP verbs. Table 1 shows the default operation of retrieving the object using the HTTP GET verb. The other behaviours of the FDOF identifier resolution, namely accessing the identifier record, the metadata record, and the object type are done by using an accept header directive in the request corresponding to the expected media type. For each behaviour a different media type is used: *fdof-ir+trig* for the identifier record, *fdof-metadata+trig* for the metadata record(s) and *fdof-type+ttr* for the object's type declaration.

This implementation approach has the advantage of completely using the existing Web infrastructure and protocols. From the infrastructure developer perspective, the only required action is the registration of the media types with the Internet Assigned Numbers Authority (IANA). Digital object registration applications can be programmed to automatically create the artefacts with the proper file extensions so that the server can return to correct object representation when requested. And client application only need to send the proper *Accept* directive with their requests to the server.

The approach taken here uses the same mechanism of the HTTP content negotiation. And this makes this approach conceptually inconsistent because it subverts the regular HTTP content negotiation, which intends to return the same object/resource represented in different formats. For instance, if the object is a given article, the client can request the article in its HTML or PDF representation. Differently, in this FDOF identifier resolution implementation option, we are not providing different representations of a given object, but offering a mechanism for the identifier of an object to resolve to different objects, i.e., its identifier record, its metadata record or its type. Therefore,

<sup>1</sup><https://github.com/FAIR-Digital-Object-Framework/fdof-servers>

<sup>2</sup><https://www.postman.com>

we consider this implementation option to be conceptually inconsistent although it is technically feasible.

**Table 1.** HTTP with accept headers

Retrieve the object	Retrieve the Identifier Record	Retrieve Metadata Record(s)	Retrieve the Object Type
GET {ObjID}	Accept: fdof-ir+trig	Accept: fdof-metadata+trig	Accept: fdof-type+ttl

## 4.2 Custom HTTP verbs

In Table 2, we present how to implement the FDOF’s identifier behaviours using a custom HTTP-based protocol. Essentially, we defined new verbs on top of the standard HTTP verbs such as GET, POST, PUT, HEAD, DELETE, etc. The new verbs are: *GETIR* to retrieve the FDOF’s Identifier Record, *GETMETADATA* to retrieve the object’s metadata record(s) and *GETTYPE* to retrieve the object’s type according to the FDOF’s typing system. Naturally, verbs to add and edit these objects follow the same pattern of using the base HTTP verb plus the type of affected object, e.g., *PUTIR* and *POSTIR*.

Although this implementation choice is conceptually sound as every verb refer to a particular type of object, for a widespread adoption, it would require a significant effort. Every client and server in the world would need to be updated to support the new verbs.

**Table 2.** Custom HTTP verbs

Retrieve the object	Retrieve the Identifier Record	Retrieve Metadata Record(s)	Retrieve the Object Type
GET {ObjID}	GETIR {ObjID}	GETMETADATA {ObjID}	GETTYPE {ObjID}

## 4.3 HTTP with FDOF paths

Table 3 shows how to request the main target object, its identifier record, its metadata record and its type using a regular REST API approach. Here, for every object available, the server is configured to respond not only to requests using the object URI but also to related sub paths: */identifierRecord* for the object’s identifier record, */metadataRecord* for the object’s metadata record, and */type* for the object’s type.

This implementation option seems to be the most straightforward and developer-friendly one because it uses current infrastructure, programming strategies and architecture. It basically constitutes of a specific REST API. Therefore, like any other REST API, it is just a matter of having client applications that can make use of it. Even in the case of client applications that are not yet aware of the FDOF’s identifier behaviours, they would still have available the API endpoints of the main target objects and would only miss the extra paths to access the FDOF-IR, metadata records and type information. Client libraries implementing the sub paths can be made available facilitating even further the adoption of this approach by developers.

## 4.4 Signposting

Finally, in Table 4 we present the procedures of operate on the main target object, its identifier record, metadata records and type information using the Signposting ap-

**Table 3.** HTTP with FDOF paths

Retrieve the object	Retrieve the Identifier Record	Retrieve Metadata Record(s)	Retrieve the Object Type
GET {ObjID}	GET {ObjID}/ identifierRecord	GET {ObjID}/ meta-dataRecord	GET {ObjID}/type

proach. Signposting relies on patterns of typed relation links embedded in the header section of HTTP responses. Primarily, the Signposting approach has been designed to enrich responses of HTML landing pages in scholar repositories with machine-readable information such as author, versions of the document, among others.

In our implementation demonstration, we used typed relation links based on the different types of objects contemplated in the FDOF’s identifier behaviors. The approach added the relation types *fdof-ir*, *fdof-metadata* and *fdof-type* to the response header of the main target object’s identifier. In this manner, the client application can just send a HEAD request using the object’s identifier and parse the header for the values of these relation types.

Recently, a FAIR-specific Signposting profile has been defined<sup>3</sup>. In this profile, the authors use the relation types *describedby* and *type* to provide the reference to the object’s metadata record and type, respectively. Since the FDOF has two types of metadata, the identifier record and the regular metadata records, it would be ambiguous to use Signposting FAIR profile. We could replace our relation type *fdof-metadata* with the profile’s *describedby*, but we would still require another relation type for the identifier record. Similarly, we can replace the relation type *fdof-type* with the profile’s *type* as it allows to use concepts in vocabularies or ontologies other than Schema.org.

The Signposting approach, similarly to the HTTP with accept headers approach described in subsection 4.1, is based on existing infrastructure and standards. To more clearly express the specific properties of the FDOF, we would require to register at IANA a couple of new relation types. Naturally, FAIR Digital Object providers would need to use FDOF-compliant server to be able to define the relation types and their values, and for the server to provide the proper information in the objects’ headers. Also similar to other implementation approaches, client applications that implement this approach benefits from the extra information provided in the header while other non-compliant applications continue to work as before, just do not take advantage of the extra information.

**Table 4.** Signposting

Retrieve the object	Retrieve the Identifier Record	Retrieve Metadata Record(s)	Retrieve the Object Type
GET {ObjID}	HEAD {ObjID} + retrieve URL with <i>rel="fdof-ir"</i>	HEAD {ObjID} + retrieve URL with <i>rel="fdof-metadata"</i>	HEAD {ObjID} + retrieve URL with <i>rel="fdof-type"</i>

## 5. Conclusions

In this paper we reported and commented on an experiment in implementing the FDOF’s identifier behaviours using 4 different HTTP-based approaches. We considered conceptual consistency, adoption barriers, developer friendliness and reuse of

<sup>3</sup><https://signposting.org/FAIR/>

current infrastructure and standards as aspects in our reflections on the implementations. As previously discussed, we do not recommend the approach using HTTP with FDOF-specific accepted media types as it is conceptually inconsistent and deviates from the intentions of HTTP content negotiation. The approach of defining a new protocol, although based on HTTP but with new verbs is technically and conceptually sound but we believe would have a high adoption costs as all servers would need to be extended.

This leaves us with the approaches using FDOF REST API paths and Signposting. Both use off-the-shelf infrastructure, tools and standards, we are more inclined to the FDOF REST API approach. The reason is that it provides more flexibility for the needs of the framework without having to require registration of new link types at IANA nor access to server configuration for the inclusion of the relation and values in the response headers. An FDO creation pipeline application can automatically create the object's identifier record, its metadata record and type artefacts and deploy them in a Web-accessible file system using the proper sub paths as defined in sub section 4.3.

The immediate next steps in the evolution of the FAIR Digital Object Framework is the extension of the FDOF Conceptual Model with operations, improvements in the FDOF server implementation and development of an FDO creation/registration pipeline.

## Data availability statement

This paper reports on software implementation options of the FDOF's identifier resolution. Therefore, the main supporting artefact is software code and not data.

## Underlying and related material

The test FDOF server implementations reported in this paper can be found at this GitHub repository <https://github.com/FAIR-Digital-Object-Framework/fdof-servers>.

## Author contributions

Luiz Olavo Bonino da Silva Santos - Conceptualization, investigation, project administration, software (testing), validation, writing - original draft. Claudenir Morais Fonseca - Conceptualization, investigation, validation, writing - review and editing. Tiago Prince Sales - Conceptualization, investigation, software (development), validation, writing - review and editing.

## Competing interests

The authors declare that they have no competing interests.

## References

- [1] M. D. Wilkinson et al., "The fair guiding principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, p. 160 018, 2016, ISSN: 2052-4463.
- [2] *Fair digital object framework documentation*, Oct. 2022. [Online]. Available: <https://fairdigitalobjectframework.org/>.

- [3] I. Jacobs and N. Walsh, Eds., *Architecture of the world wide web*, Volume One, World Wide Web Consortium, Dec. 2004.
- [4] *Digital object identifier*, <https://www.doi.org/>, Accessed: 2024-07-07.
- [5] R. Kahn and R. Wilensky, "A framework for distributed digital object services," en, *International Journal on Digital Libraries*, vol. 6, no. 2, pp. 115–123, 2006, ISSN: 1432-5012, 1432-1300. Accessed: Aug. 25, 2021.
- [6] R. E. Kahn et al., "Digital Object Identifier Resolution Protocol Specification - Version 3," English, DONA Foundation, Tech. Rep., Jun. 2022. [Online]. Available: <https://www.dona.net/sites/default/files/2022-06/DO-IRPV3.0--2022-06-30.pdf>.
- [7] R. E. Kahn et al., "Digital Object Interface Protocol Specification, version 2.0," DONA Foundation, Tech. Rep., Nov. 2018.
- [8] L. O. Bonino da Silva Santos, T. P. Sales, C. M. Fonseca, and G. Guizzardi, "Towards a conceptual model for the fair digital object framework," in *Formal Ontology in Information Systems*. IOS Press, Dec. 2023, ISBN: 9781643684697. DOI: [10.3233/faia231131](https://doi.org/10.3233/faia231131). [Online]. Available: <http://dx.doi.org/10.3233/FAIA231131>.
- [9] H. Van de Sompel and M. L. Nelson, "Reminiscing about 15 years of interoperability efforts," *DLib Mag.*, vol. 21, no. 11/12, Nov. 2015.