Evaluating FAIR Digital Object as a distributed object system

This manuscript (permalink) was automatically generated from stain/2022-fdo-paper@25b1e52 on May 12, 2022.

Authors

- **Stian Soiland-Reyes**
  0000-0001-9842-9718 · stain · soilandreyes
  Department of Computer Science, The University of Manchester, UK; Informatics Institute, Faculty of Science, University of Amsterdam, NL · Funded by BioExcel-2 (European Commission H2020-INFRAEDI-2018-1 823830); BY-COVID (European Commission HORIZON-INFRA-2021-EMERGENCY-01 101046203)

- **Carole Goble**
  0000-0003-1219-2137 · carolegoble · CaroleAnneGoble
  Department of Computer Science, The University of Manchester, UK · Funded by BioExcel-2 (European Commission H2020-INFRAEDI-2018-1 823830); EOSC-Life (European Commission H2020-INFRAEOSC-2018-2 824087); BY-COVID (European Commission HORIZON-INFRA-2021-EMERGENCY-01 101046203)

- **Paul Groth**
  0000-0003-0183-6910 · pgroth · pgroth
  Informatics Institute, Faculty of Science, University of Amsterdam, NL

Abstract

FAIR Digital Object

The concept of **FAIR Digital Object** [1] has been introduced as way to expose research data as active objects that conform to the FAIR principles [2]. This builds on the *Digital Object* (DO) concept [3], first introduced in 1995 [4] as a system of *repositories* containing *digital objects* identified by *handles* and described by *metadata* which may have references to other handles. DO was the inspiration for the ITU X.1255 framework [5] which introduced an abstract *Digital Entity Interface Protocol* for managing such objects programmatically, first realized by the Digital Object Interface Protocol (DOIP) v1 [6].

In brief, the structure of a FAIR Digital Object (FDO) is to, given a *persistent identifier* (PID) such as a DOI, resolve to a *PID Record* that gives the object a *type* along with a mechanism to retrieve its *bit sequences*, *metadata* and references to further programmatic *operations*. The type of an FDO (itself an FDO) defines attributes to semantically describe and relate such FDOs to other concepts (typically other FDOs referenced by PIDs). The premise of systematically building an ecosystem of such digital objects is to give researchers a way to organize complex digital entities, associated with identifiers, metadata, and supporting automated processing [7].

Recently, FDOs have been recognized by the European Open Science Cloud (EOSC) as a suggested part of its Interoperability Framework [8], in particular for deploying active and interoperable FAIR resources that are *machine actionable*. Sevelopment of the FDO concept continued within Research Data Alliance (RDA) groups and EOSC projects like GO-FAIR, concluding with a set of guidelines for implementing FDO [9]. The FAIR Digital Objects Forum has since taken over the maturing of FDO through focused working groups which have currently drafted several more detailed specification documents (see section [**sec:next-step-fdo?**]).

FDO is an evolving concept. A set of FDO Demonstrators [10] highlight how current adapters are approaching implementations of FDO from different angles:

- Building on the Digital Object concept, using the simplified DOIP v2 specification [11], which detail how to exchange JSON objects through a text-based protocol [1] (usually TCP/IP over TLS). The main DOIP operations are retrieving, creating and updating digital objects. These are mostly realized using the reference implementation Cordra. FDO types are registered in the local Cordra instance, where they are specified using JSON Schema [12]) and PIDs are assigned using the Handle system. Several type registries have been established.
- Following the traditional Linked Data approach, but using the DOIP protocol, e.g. using JSON-LD and schema.org within DOIP (NIST for material science).
- Approaching the FDO principles from existing Linked Data practices on the Web (e.g. WorkflowHub use of RO-Crate and schema.org).

From this it becomes apparant that there is a potentially large overlap between the goals and approaches of FAIR Digital Objects and Linked Data, which we'll cover in the next section.

Linked Data

In order to describe *Linked Data* as it is used today, we'll start with an (opinionated) briefing of the evolution of its foundation, the *Semantic Web*.

A brief history of the Semantic Web

The **Semantic Web** was developed as a vision by Tim Berners-Lee [13], at a time the Web had been widely established for information exchange, as a global set of hypermedia documents that eare cross-related using universal links in the form of URLs. The foundations of the Web (e.g. URLs, HTTP, SSL/TLS, HTML, CSS, ECMAScript/JavaScript, media types) were standardized by W3C, Ecma, IETF and later WHATWG. The goal of Semantic Web was to further develop the machine-readable aspects of the Web, in particular adding *meaning* (or semantics) to not just the link relations, but also to the *resources* that the URLs identified, and for machines thus being able to meaningfully navigate across such resources, e.g. to answer a particular query.

Through W3C, the Semantic Web was realized with the Resource Description Framework (RDF) [14] that used *triples* of subject-predicate-object statements, with its initial serialization format [15] being RDF/XML (XML was at the time seen as a natural data-focused evolution from the document-centric SGML and HTML).

While triple-based knowledge representations were not new [doi:10.21954/ou.ro.0000f821], the main innovation of RDF was the use of global identifiers in the form of URIs[2] as the primary identifier of the *subject* (what the statement is about), *predicate* (relation/attribute of the subject) and *object* (what is pointed to). By using URIs not just for documents[^5], the Semantic Web builds a self-described system of types and properties, the meaning of a relation can be resolved by following its hyperlink to the definition within a *vocabulary*.

[^5] URIs can also identify *non-information resources* for any kind of physical object (e.g. people), such identifiers can resolve with `303 See Other` redirections to a corresponding *information resources* [doi:10.22028/D291-25086].

The early days of the Semantic Web saw fairly lightweight approaches with the establishment of vocabularies such as FOAF (to describe people and their affiliations) and Dublin Core (for bibliographic data). Vocabularies themselves were formalized using RDFS or simply as human-readable HTML web pages defining each term. The main approach of this *Web of Data* was that a URI identified a *resource* (e.g. an author) had a HTML *representation* for human readers, along with a RDF representation for machine-readable data of the same resource. By using *content negotiation* in HTTP, the same identifier could be used in both views, avoiding `index.html` vs `index.rdf` exposure in the URLs. The concept of *namespaces* gave a way to give a group of RDF resources with the same URI base from a Semantic Web-aware service a common *prefix*, avoiding repeated long URLs.

The mid-2000s saw a large academic interest and growth of the Semantic Web, with the development of more formal representation system for ontologies, such as OWL, allowing complex class hierarchies and logic inference rules following *open world* paradigm (e.g. a *ex:Parent* is equivalent to a subclass of *foaf:Person* which must *ex:hasChild* at least one *foaf:Person*, then if we know *:Alice a ex:Parent* we can infer *:Alice ex:hasChild [a foaf:Person]* even if we don't know who that child is). More human-readable syntaxes of RDF such as Turtle (shown in this paragraph) evolved at this time, and conferences such as ISWC gained traction, with a large interest in knowledge representation and logic systems based on Semantic Web technologies evolving at the same time.

Established Semantic Web services and standards include SPARQL [16] (pattern-based triple queries), named graphs (triples expanded to *quads* to indicate statement source or represent conflicting views), triple/quad stores (graph databases such as OpenLink Virtuoso, GraphDB, 4Store), mature RDF libraries (including Redland RDF, Apache Jena, Eclipse RDF4J, RDFLib, RDF.rb, rdflib.js), and numerous graph visualization (many of which struggle with usability for more than 20 nodes).

The creation of RDF-based knowledge graphs grew particularly in fields like bioinformatics, e.g. for describing genomes and proteins. In theory, the use of RDF by the life sciences would enable interoperability between the many data repositories and support combined views of the many aspects of bio-entities – however in practice most institutions ended up making their own ontologies and identifiers, for what to the untrained eye would mean roughly the same. One can argue that the toll of adding the semantic logic system of rich ontologies meant that small, but fundamental, differences in opinion (e.g. should a *gene identifier* signify which protein a DNA sequence would make, or just the particular DNA sequence letters, or those letters as they appear in a particular position on a human chromosome?) lead to large differences in representational granularity, and thus needed different identifiers.

Facing these challenges, thanks to the use of universal identifiers in the form of URIs, *mappings* could retrospectively be developed not just between resources, but also across vocabularies. Such mappings can be expressed themselves using lightweight and flexible RDF vocabularies such as SKOS [17] (e.g. `dct:title skos:closeMatch schema:name` to indicate near equivalence of two properties). Automated ontology mappings have identified large potential overlaps (e.g. 372 definitions of `Person`) [18] .

The move towards *open science* data sharing practices from the late 2000s encouraged knowledge providers to distribute collections of RDF descriptions as downloadable *datasets* [3], so that their clients can avoid thousands of HTTP requests for individual resources. This enabled local processing, mapping and data integration across datasets (e.g. Open PHACTS [doi:10.1016/j.websem.2014.03.003]), rather than relying on the providers' RDF and SPARQL endpoints (which could become overloaded when handling many concurrent, complex queries).

With these trends, an emerging problem was that adapters of the Semantic Web primarily utillized it as a set of graph technologies, with little consideration to existing Web resources. This meant that links stayed mainly within a single information system. Another challenge facing potential adapters is the plethora of choices, not just to navigate, understand and select to reuse the many possible vocabularies and ontologies [doi:10.3233/SSW200033] , but also technological choices on RDF serialization (at least 7 formats), type system (RDFS [19], OWL [20], OBO [21], SKOS [17]), hash vs slash, HTTP status codes and PID redirection strategies [doi:10.22028/D291-25086].

Linked Data: Rebuilding the Web of Data

The **Linked Data** concept [22] was kickstarted as a counter-reaction to this development of the Semantic Web, as a set of best practices [23] to bring the Web aspect back into focus. Crucially to Linked Data is to *reuse existing URIs* where they exist, rather than always make new identifiers. This means a loosening of the semantic restrictions previously applied, and an emphasis on building navigatable data resources, rather than elaborate graph representations.

Vocabularies like schema.org evolved not long after, intended for lightweight semantic markup of existing Web pages, primarily to improve search engines' understanding of types and embedded data. In addition to several such embedded *microformats* (Open Graph [24], RDFa [25], Microdata [26]) we find JSON-LD [27] as a Web-focused RDF serialization that aims for improved programmatic generation and consumption, including from Web applications. JSON-LD is at time of writing used by 42.6% of the top 10 million websites [28] [4].

Recently there has also been a renewed emphasis to improve the *Linked Data Developer Experience* [29], for instance RDF Shapes (expressed in SHACL [30] or ShEx [31]) [doi:10.1007/978-3-030-21348-0_39] can be used to validate RDF Data [doi:https://doi.org/10.2200/S00786ED1V01Y201707WBE016] before consuming it programmatically or shaping it to other models.

Still more work to be done in adapting the Web [doi:10.3233/SW-190372]

Interoperability Framework for Fast Data

Considering FDO/Web as interoperability framework for Fast Data

Table 1: Considering FDO and Web according to the levels of interoperability [32]:

Quality	**FDO w/ DOIP**	**Web w/ Linked Data**
Symbiotic: *to what extent multiple applications can agree to interact/align/collaborate/cooperate*	Purpose of FDO is to enable federated machine actionable digital objects for scholarly purposes, in practice this also requires agreement of or compatibility between FDO types. FDO encourages research communities to develop common type registries to be shared across instances. In current DOIP practice, each service have their own types, attributes and operations. The wider symbiosis is consistent use of PIDs.	Web is loosely coupled and encourages collaboration and linking by URL. In practice, REST APIs end up being mandated centrally by dominant (often commercial) providers, which clients are required to use as-is with special code per service. Use of Linked Data enables common tooling and semantic mapping across differences.
Pragmatic: *using interaction contracts so processes can be choreographed in workflows*	FDO types and operations enable detailed choreography (see CWFP). `0.TYPE/DOIPOperation` has lightweight definition of operation, `0.DOIP/Request` or `0.DOIP/Response` may give FDO Type or any other kind of "specifics" (incl. human readable docs). Semantics/purpose of operations not formalized (similar operations can be grouped with `0.DOIP/OperationReference`).	"Follow your nose" crawler navigation, which may lead to frequent dead ends. Operational composition, typically within a single API provider, documented by OpenAPI 3 [33], schema.org Actions [[34]), WSDL/SOAP [35], but frequently just as human-readable developer documentation/examples.
Semantic: *ensuring consistent understanding of messages, interoperability of rules, knowledge and ontologies*	FDO semantic enable navigation and typing. Every FDO have a type. Types maintained in FDO Type registries, which may add additional semantics, e.g. the ePIC PID-InfoType for Model. No single type semantic, Type FDOs can link to existing vocabularies & ontologies. JSON-LD used within some FDO objects (e.g. DISSCO Digital Specimen, NIST Material Science schema) [36]	Lightweight HTTP semantics for authenticity/navigation. Semantic Type not commonly expressed on PID/header level, may be declared within Linked Data metadata. Semantic of type implied by Linked Data formats (e.g. OWL2, RDFS), although choice of type system may not be explicit.
Syntactic: *serializing messages for digital exchange, structure representation*	DOIP serialize FDOs as JSON, metadata commonly use JSON, typed with JSON Schema. Multiple byte stream attachments of any media type.	Textual HTTP headers (including any signposting), single byte stream of any media type, e.g. Linked Data formats (JSON-LD, Turtle, RDF/XML) or embedded in document (HTML with RDFa, JSON-LD or Microdata). XML previously main syntax used by APIs, JSON now dominant.
Connective: *transferring messages to another application, e.g. wrapping in other protocols*	DOIP [11] is transport-independent, commonly TLS TCP/IP port 9000), DOIP over HTTP	HTTP/1.1 (TCP/IP port 80), HTTP/1.1+TLS (TCP/IP 443), HTTP/2 (as HTTP/1* but binary), HTTP/3 (like HTTP/2+TLS but UDP)
Environmental: *how applications are deployed and affected by its environment, portability*	Main DOIP implementation is Cordra, which can be single-instance or distributed. Cordra storage backends include file system, S3, MongoDB (itself scalable). Unique DOIP protocol can be hard to add to existing Web application frameworks, although proxy services have been developed (e.g. B2SHARE adapter).	HTTP services widely deployed in a myriad of ways, ranging from single instance servers, horizontally & vertically scaled application servers, to (for static content) multi-cloud Content-Delivery Networks (CDN). Current scalable cloud technologies for Web hosting may not support HTTP features previously seen as important for Semantic Web, e.g. content negotiation and semantic HTTP status codes.

Mapping of Metamodel concepts:

Table 2: Mapping the Metamodel concepts from the Interoperability Framework for Fast Data [32] to equivalent concepts for FDO and Web:

Metamodel concept	FDO/DOIP concept	Web/LD concept
Resource	FDO/DO	Resource
Service	DOIP service	Server/endpoint
Transaction	(not supported)	Conditional requests, `409 Conflict`
Process	Extended operations	Primarily stateless, `100 Continue`, `202 Accepted`
Operation	DOIP Operation	Method, query parameters
Request	DOIP Request	Request
Response	DOIP Response	Response
Message	Segment, `requestId`	Message, Representation
Channel	DOIP Transport protocol (e.g. TCP/IP, TLS). JSWS signatures.	TCP/IP, TLS, UDP
Protocol	DOIP 2.0, ++	HTTP/1.1, HTTP/2, HTTP/3
Link	PID/Handle	URL

A comparison framework for middleware infrastructures

Comparing FDO and Web as middleware infrastructures

Table 3: Comparing FAIR Digital Object (with the DOIP 2.0 protocol [11]) and Web technologies (using Linked Data) as middleware infrastructures [37]

Quality	FDO w/ DOIP	Web w/ Linked Data
Openness: *framework enable extension of applications*	FDOs can be cross-linked using PIDs, pointing to multiple FDO endpoints. Custom DOIP operations can be exposed, although it is unclear if these can be outside the FDO server. PID minting requires Handle.net prefix subscription, or use of services like Datacite, B2Handle.	The Web is inheritedly open and made by cross-linked URLs. Participation requires DNS domain purchase (many free alternatives also exists). PID minting can be free using PURL/ARK services, or can use DOI/Handle with HTTP redirects.
Scalability: *application should be effective at many different scales*	No defined methods for caching or mirroring, although this could be handled by backend, depending on exposed FDO operations (e.g. Cordra can scale to multiple backend nodes)	Cache control headers reduce repeated transfer and assist explicit and transparent proxies for speed-up. HTTP `GET` can be scaled to world-population-wide with Content-Delivery Networks (CDNs), while write-access scalability is typically manage by backend.

Quality	FDO w/ DOIP	Web w/ Linked Data
Performance: *efficient and predictable execution*	DOIP has been shown moderately scalable to 100 millions of objects, create operation at 900 requests/second [38]. DOIP protocol is reusable for many operations, multiple requests may be answered out of order (by `requestId`). Multiple connections possible. Setup is typically through TCP and TLS which adds latency.	HTTP traffic is about 10% of global Internet traffic, excluding video and social networks [{ 39}]. HTTP 1 connections are serial and reusable, and concurrent connections is common. HTTP/2 adds asynchronous responses and multiplexed streams [40] but still has TCP+TLS startup costs. For reduced latency [41], HTTP/3 [{ 42}] use QUIC [43]) rather than TCP, already adapted heavily (30% of EMEA traffic) of which Instagram & Facebook video is the majority of traffic.
Distribution transparency: *application perceived as a consistent whole rather than independent elements.*	Each FDO is accessed separately along with its components (typically from the same endpoint). FDOs should provide the mandatory kernel metadata fields. FDOs of the same declared type typically share additional attributes (although that schema may not be declared). DOIP does not enforce metadata typing constraints, this need to be established as FDO conventions.	Each URL accessed separately. Common HTTP headers provide basic metadata, although it is often not reliable. A multitude of schemas and serializations for metadata exists, conventions might be implied by a declared profile or certain media types. Metadata is not always machine findable, may need pre-agreed API URI Templates [44], content-negotiation [45] or FAIR Signposting [46].
Access transparency: *local/remote elements accessed similarly*	FDOs always accessed through PID indirection, but this means difficult to make private test setup.	Global HTTP protocol frequently used locally and behind firewalls, but at risk of non-global URIs (e.g. `http://localhost/object/1`) and SSL issues (e.g. self-signed certificates, local CAs)
Location transparency: *elements accessed without knowledge of physical location*	FDOs always accessed through PIDs. Multiple locations possible in Handle system, can expose geo-info.	PIDs and URL redirects. DNS aliases and IP routing can hide location. Geo-localized servers common for large cloud deployments.
Concurrency transparency: *concurrent processing without interference*	No explicit concurrency measures. FDO kernel metadata can include checksum and date.	HTTP operations are classified as being stateless/idempotent or not (e.g. `PUT` changes state, but can be repeated on failure), although these constraints are occasionally violated by Web applications. Cache control, `ETag` (~ checksum) and modification date in HTTP headers allows detection of concurrent changes on a single resource.
Failure transparency: *service provisioning resiliant to failures*	DOIP status codes, e.g. `0.DOIP/Status.104` , additional codes can be added as custom attributes	HTTP status codes e.g. `404 Not Found` , structured error documents in Open API (??)
Migration transparency: *allow relocating elements without interferring application*	Update of PID record URLs, indirection through `0.TYPE/DOIPServiceInfo` (not always used consistently). No redirection from DOIP service.	HTTP `30x` status codes provide temporary or permanent redirections, commonly used for PURLs but also by endpoints.

Quality	FDO w/ DOIP	Web w/ Linked Data
Persistence transparency: *conceal deactivation/reactivation of elements from their users*	FDO requires use of PIDs for object persistence, including a thumbstone response for deleted objects. There is no guarantee that an FDO is immutable or will even stay the same type (note: CORDRA extends DOIP with version tracking).	URLs are not required to persist, although encouraged [47]. Persistence requires convention to use PIDs/PURLs and HTTP `410 Gone`. An URL may change its content, change in type may sometimes force new URLs if exposing extensions like `.json`. Memento [48] expose versioned snapshots. WebDAV `VERSION-CONTROL` method [49] (used by SVN).
Transaction transparency: *coordinate execution of atomic/isolated transactions*	No transaction capabilities declared by FDO or DOIP. Internal synchronization possible in backend for Extended operations.	Limited transaction capabilities (e.g. `If-Unmodified-Since`) on same resource. WebDAV locking mechanisms [50] with `LOCK` and `UNLOCK` methods.
Modularity: *application as collection of connected/distributed elements*	FDOs are inheritedly modular using global PID spaces and their cross-references. In practice, FDOs of a given type are exposed through a single server shared within a particular community/institution.	The Web is inheritently modular in that distributed objects are cross-referenced within a global URI space. In practice, an API's set of resources will be exposed through a single HTTP service, but modularity enables fine-grained scalability in backend.
Encapsulation: *separate interface from implementation. Specify interface as contract, multiple implementations possible*	Indirection by PID gives separation. FDO principles are protocol independent, although it may be unclear which protocol to use for which FDO (although `0.DOIP/Transport` can be specified after already contacting DOIP). Cordra supports native DOIP, DOIP over HTTP and Cordra REST API)	HTTP/1.1 semantics can seemlessly upgrade to HTTP/2 and HTTP/3. `http` vs `https` URIs exposes encryption detail [5]. Implementation details may leak into URIs (e.g. `search.aspx`), countered by deliberate design of URI patterns [52] and PIDs via Persistent URLs (PURL).
Inheritance: *Deriving specialized interface from another type*	DOIP types nested with parents, implying shared FDO structures (unclear if operations are inherited). FDO establishes need for multiple Data Type Registries (e.g. managed by a community for a particular domain). Semantics of type system currently undefined for FDO and DOIP, syntactic types can also piggyback of FDO type's schema (e.g. CORDRA `$ref` use of JSON Schema references [12])	Syntactically Media Type with multiple suffixes [53] (mainly used with `+json`), declaration of subtypes as profiles (RFC6906) [54]. In metadata, semantic type systems (RDFS [19], OWL2 [20], SKOS [17]). OpenAPI 3 [33] inheritance and Polymorphism. XML `xsd:schemaLocation` & `xsd:type` [55], JSON `$schema` [12]), JSON-LD `@context` [56]. Large number of domain-specific and general ontologies define semantic types, but finding and selecting remains a challenge.
Signal interfaces: *asynchronous handling of messages*	DOIP 2.0 is synchronous, in FDO async operations undefined. Could be handled as custom jobs/futures FDOs	HTTP/2 multiplexed streams [40], Web Sockets [57], Linked Data Notifications [58], AtomPub [59], SWORD [60], Micropub, more typically ad-hoc jobs/futures REST resources
Operation interfaces: *defining operations possible on an instance, interface of request/response messages*	CRUD predefined in DOIP, custom operations through `0.DOIP/Op.ListOperations` (can be FDOs of type `0.TYPE/DOIPOperation`, more typically local identifiers like `"getProvenance"`)	CRUD predefined in HTTP methods [61], (extended by registration), URI Templates [44], OpenAPI operations [33], HATEOAS incl. schema.org Actions [[34]), JSON HAL [62] & Link headers (RFC8288) [63]

Quality	FDO w/ DOIP	Web w/ Linked Data
Stream interfaces: *operations that can handle continuous information streams*	Undefined in FDO. DOIP can support multiple byte stream elements (need custom FDO type to determine stream semantics)	HTTP 1.1 [64] chunked transfer, HLS (RFC8216) [65], MPEG-DASH (ISO/IEC 23009-1:2019) [66]

...

As for the aspect of *Performance*, it is interesting to note that while the first version of DOIP [6] supported multiplexed channels similar to HTTP/2, allowing concurrent transfer of several digital objects. However multiplexing was removed for the much simplified DOIP 2.0 [11], which do support multiple asynchronous requests, but unlike DOIP 1.0 will require a DO response to be sent back completely, as a series of segments (which again can be split the bytes of each binary *element* into sized *chunks*), before transmission of another DO response can start on the transport channel. It is unclear what is the purpose of splitting a binary into chunks on a channel which no longer can be multiplexed and the only property of a chunk is its size [6].

Assessing DOIP against FDO

Table 4: Checking FDO guidelines [9] against its current implementations as DOIP [11] and Linked Data Platform (LDP) [67], with suggestions for required additions.

Guideline	DOIP 2.0	FDO suggestions	Linked Data Platform	LDP suggestion
G1: *invest for many decades*				
G2: *trustworthiness*				
G3: *FAIR principles*				
G4: *machine actionability*				
G5: *abstraction principle*				
G6: *stable binding*				
G7: *encapsulation*				
G8: *technology independence*				
G9: *standard compliance*				
FDOF1: *PID as basis*				
FDOF2: *PID record w/ type*				
FDOF3: *PID resolvable to bytestream & metadata*				
FDOF4: *Additional attributes*				
FDOF5: *Interface: operation by PID*				
FDOF6: *CRUD operations + extensions*				
FDOF7: *FDOF Types related to operations*				
FDOF8: *Metadata as FDO, semantic assertions*				

Guideline	DOIP 2.0	FDO suggestions	Linked Data Platform	LDP suggestion
FDOF9: *Different metadata levels*				
FDOF10: *Metadata schemas by community*				
FDOF11: *FDO collections w/ semantic relations*				
FDOF12: *Deleted FDO preserve PID w/ tombstone*				

The draft update specification *WD-RequirementSpec-1.0-20220317* (at time of writing in internal review by FAIR Digital Object Forum) clarifies these definitions with equivalent identifiers [7] and relates them to further FDO requiremes such as FDO Data Type Registries.

Assessing FDO against FAIR

Table 5: Assessing how FAIR principles is/can be fulfilled by FDOs [9] as DOIP [11], Linked Data Platform (LDP) [67], with examples øf existing Linked Data practices.

Principle	FDO/DOIP	FDO/LDP	Linked Data examples
F1: *PID for data/metadata*	PIDs required (FDOF1). Handle, DOI.	FDOF-IR (Identifier Record). PID can be any URI?	"Cool" URIs [47], PID using PURL services.
F2: *data has metadata*	FDO has key-value metadata. Unclear how to link to additional metadata.	FDOF-IR links to multiple metadata records	RDF-based metadata by content negotiation or FAIR Signposting. Embedded in landing page (RDFa).
F3: *metadata include PID*	`id` and `type` are required metadata elements PIDs, also implicit as requests must use PID	PID only required in FDOF-IR record.	PID inclusion typical, but often inconsistent (e.g. `www.example.com` vs `example.com`) or missing (use of `<>` as *this* subject)
F4: *searchable registration*	No, registries not required (except Data Type Registries). Handle registry only searchable by PID.		
A1: *retrieve by standard protocol*	Retrievable from PID (FDOF3). Informal DOIP standard maintanded by DONA Foundation	Formal HTTP standards maintained by IETF	

Principle	FDO/DOIP	FDO/LDP	Linked Data examples
A1.1: *protocol open/free/universal*	Required by G1. Partially realized, although Handle system is open protocol [68] it was covered by software patent US6135646A (expired in 2013), and only implementation of `Handle.net` software currently only available by public license] (not OSI Open Source). CORDRA free to use under BSD-like license, although not recognized by OSI as Open Source.	DNS, HTTP, TLS, RDF standards are open, free and universal, multiple open source clients/servers exist.	
A1.2: *protocol can do auth&auth*	TLS certificates, `authentication` field (details unspecified)	HTTP authentication, TLS certificates	
A2: *metadata even if data gone*	FDO thumbstone required (FDOF12)	Unspecified, however FDOF-IR links to separate metadata records	`410 Gone` status infrequently used, without metadata. Possible with signposting
I1: *formal knowledge representation*	Required by FDOF8	Unspecified	Always implied by use of RDF syntaxes.
I2: *use FAIR vocabularies*	Informally required by G3, formally by FDOF10 (but not in FDOR10)	Unspecified, implied by use of RDF?	FAIR practices for LD vocabularies increasingly common, sometimes inconsistent (e.g. PURLs that don't resolve) or incomplete (e.g. unknown license)
I3: *qualified references*	Implied by attributes to PIDs of other FDO	Unspecified	By definition (Linked Data is relating to pre-existing URIs [69]). Link relations
R1: *relevant attributes*	Required (FDOF4)	Unspecified. Multiple metadata records can allow multiple semantic profiles.	Usually, however a plethora
R1.1: *clear data license*	Unspecified (but will be in PID Kernal metadata?)	Unspecified	Dublin Core Terms `dct:license` frequently recommended, but not required, e.g. by DCAT 2 [70]
R1.2: *detailed provenance*	Unspecified (some CORDRA types add `getProvenance` methods). PID Kernel attributes?	Unspecified	W3C PROV-O, PAV
R1.3: *follows community standards*	Recommended (FDOF4, FDOF10)	Unspecified	Common practice, specially in bioinformatics, e.g. BioSchemas [71], BioPortal [72]

Next steps for FDO

Documents currently undergoing internal review:

WD-DocProcessStd-1.1-20220129 WD-MachineActionDef-1.1-20220301 WD-RequirementSpec-1.0-20220317 WD-ConfigurationTypes-1.0-20220317 WD-Granularity-1.0-20220317 WD-PIDProfileAttributes-1.0-20220317 WD-FDO-Upload-0.1-20220320 PED-DOIPEndorsement-0.1-20220326 WD-TypingFDOs-1.0-20220310

FDO Requirement Specifications 1.1 FDO Machine Actionability 1.1 FDO Configuration Types 1.0 FDO0 PID Profiles and Attributes 1.0 FDO FDO Granularity 1.0

References

1. **FAIR principles and digital objects: Accelerating convergence on a data infrastructure**
Erik Schultes, Peter Wittenburg
Data analytics and management in data intensive domains: 20th international conference, DAMDID/RCDL 2018, moscow, russia, october 9–12, 2018, revised selected papers (2019)
DOI: {10.1007/978-3-030-23584-0_1} · ISBN: 978-3-030-23583-3

2. **The FAIR Guiding Principles for scientific data management and stewardship**
Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, … Barend Mons
Scientific Data (2016-03-15) https://doi.org/bdd4
DOI: 10.1038/sdata.2016.18 · PMID: 26978244 · PMCID: PMC4792175

3. **A framework for distributed digital object services**
Robert Kahn, Robert Wilensky
International Journal on Digital Libraries (2006-04)
https://www.doi.org/topics/2006_05_02_Kahn_Framework.pdf
DOI: 10.1007/s00799-005-0128-x

4. **A framework for distributed digital object services**
Robert Kahn, Robert Wilensky
CNRI (1995-05-13) http://www.cnri.reston.va.us/k-w.html

5. **X.1255 : Framework for discovery of identity management information**
https://www.itu.int/rec/T-REC-X.1255-201309-I

6. **Digital Object Interface Protocol Version 1.0 | DONA Foundation**
https://www.dona.net/doipv1doc

7. **Digital objects as drivers towards convergence in data infrastructures**
Peter Wittenburg, George Strawn, Barend Mons, Luiz Bonino, Erik Schultes
https://b2share.eudat.eu (2019-01-06)
DOI: 10.23728/b2share.b605d85809ca45679b110719b6c6cb11

8. **EOSC interoperability framework**
Oscar Corcho, Magnus Eriksson, Krzysztof Kurowski, Milan Ojsteršek, Christine Choirat, Mark van de Sanden, Frederik Coppens
Publications Office of the EU (2021-02-05) https://doi.org/10.2777/620649
DOI: 10.2777/620649

9. **FAIR digital object framework**
Luiz Bonino, Oeter Wittenburg, Bonnie Carroll, Alex Hardisty, Mark Leggott, Carlo Zwölf
FDOF technical implementation guideline (2019-11-22) {https://github.com/GEDE-RDA-Europe/GEDE/blob/master/FAIR%20Digital%20Objects/FDOF/FAIR%20Digital%20Object%20Framework-v1-02.docx}

10. **FAIR Digital Object Demonstrators 2021**
Peter Wittenburg, Ivonne Anders, Christophe Blanchi, Merret Buurman, Carole Goble, Jonas Grieb, Alex Hardisty, Sharif Islam, Thomas Jejkal, Tibor Kálmán, … Philipp Wieder
Zenodo (2022-01-18) https://doi.org/gp4wm4
DOI: 10.5281/zenodo.5872645

11. **Digital object interface protocol specification, version 2.0**
DONA Foundation
DONA foundation (2018-11-12) https://hdl.handle.net/0.DOIP/DOIPV2.0

12. **draft-bhutton-json-schema-00** https://datatracker.ietf.org/doc/html/draft-bhutton-json-schema-00

13. **Weaving the Web: the original design and ultimate destiny of the World Wide Web by its inventor**
Tim Berners-Lee, Mark Fischetti
HarperSanFrancisco (1999)
ISBN: 9780062515865

14. **RDF 1.1 Primer** http://www.w3.org/TR/rdf11-primer/

15. **Resource Description Framework (RDF) Model and Syntax Specification**
https://www.w3.org/TR/1999/REC-rdf-syntax-19990222/

16. **SPARQL 1.1 Overview** http://www.w3.org/TR/sparql11-overview/

17. **SKOS Simple Knowledge Organization System Primer** http://www.w3.org/TR/skos-primer/

18. **How Matchable Are Four Thousand Ontologies on the Semantic Web**
Wei Hu, Jianfeng Chen, Hang Zhang, Yuzhong Qu
Lecture Notes in Computer Science (2011) https://doi.org/bwdp52
DOI: 10.1007/978-3-642-21034-1_20

19. **RDF Schema 1.1** http://www.w3.org/TR/rdf-schema/

20. **OWL 2 Web Ontology Language Document Overview (Second Edition)**
http://www.w3.org/TR/owl2-overview/

21. **Mapping between the OBO and OWL ontology languages**
Syed Tirmizi, Stuart Aitken, Dilvan A Moreira, Chris Mungall, Juan Sequeda, Nigam H Shah, Daniel P Miranker
Journal of Biomedical Semantics (2011) https://doi.org/bn3fsc
DOI: 10.1186/2041-1480-2-s1-s3 · PMID: 21388572 · PMCID: PMC3105495

22. **Linked Data - The Story So Far**
Christian Bizer, Tom Heath, Tim Berners-Lee
International Journal on Semantic Web and Information Systems (2009-07) https://doi.org/fc8zjt
DOI: 10.4018/jswis.2009081901

23. **Linked Data - Design Issues** https://www.w3.org/DesignIssues/LinkedData.html

24. **The Open Graph protocol** https://ogp.me/

25. **RDFa 1.1 Primer - Third Edition** http://www.w3.org/TR/rdfa-primer/

26. **HTML Standard** https://html.spec.whatwg.org/multipage/microdata.html

27. **JSON-LD 1.1** https://www.w3.org/TR/json-ld/

28. **Usage Statistics of JSON-LD for Websites, May 2022**
https://w3techs.com/technologies/details/da-jsonld

29. **Designing a Linked Data developer experience** (2018-12-28) https://ruben.verborgh.org/blog/2018/12/28/designing-a-linked-data-developer-experience/

30. **Shapes Constraint Language (SHACL)** https://www.w3.org/TR/shacl/

31. **Shape Expressions (ShEx) 2.1 Primer** http://shex.io/shex-primer/

32. **An Interoperability Framework and Distributed Platform for Fast Data Applications**
José Carlos Martins Delgado
Data Science and Big Data Computing (2016) https://doi.org/gp3rds
DOI: 10.1007/978-3-319-31861-5_1

33. **OpenAPI Specification v3.1.0 | Introduction, Definitions, & More**
https://spec.openapis.org/oas/v3.1.0.html

34. **Schema.org Actions - schema.org** https://schema.org/docs/actions.html

35. **Web Services Description Language (WSDL) Version 2.0 Part 0: Primer**
http://www.w3.org/TR/wsdl20-primer/

36. **FAIR digital object demonstrators 2021**
Peter Wittenburg, Ivonne Anders, Christophe Blanchi, Merret Buurman, Carole Goble, Jonas Grieb, Alex Hardisty, Sharif Islam, Thomas Jejkal, Tibor Kálmán, … Philipp Wieder
Zenodo (2022) https://zenodo.org/record/5872645
DOI: 10.5281/zenodo.5872645

37. **A Comparison Framework for Middleware Infrastructures.**
Apostolos Zarras
The Journal of Object Technology (2004) https://doi.org/cj5q8r
DOI: 10.5381/jot.2004.3.5.a2

38. https://www.rd-alliance.org/sites/default/files/Cordra.2022.pdf

39. **Global Internet Phenomena Report 2022**
Sandvine
https://www.sandvine.com/global-internet-phenomena-report-2022

40. **Hypertext Transfer Protocol Version 2 (HTTP/2)**
M Belshe, R Peon
RFC Editor (2015-05) https://doi.org/gp32q9
DOI: 10.17487/rfc7540

41. https://blog.cloudflare.com/http-3-vs-http-2/

42. **draft-ietf-quic-http-34** https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34

43. **QUIC: A UDP-Based Multiplexed and Secure Transport**
J Iyengar, M Thomson (editors)
RFC Editor (2021-05) https://doi.org/gkctrr
DOI: 10.17487/rfc9000

44. **URI Template**
J Gregorio, R Fielding, M Hadley, M Nottingham, D Orchard
RFC Editor (2012-03) https://doi.org/gp33dw
DOI: 10.17487/rfc6570

45. **Content negotiation - HTTP | MDN** https://developer.mozilla.org/en-US/docs/Web/HTTP/Content_negotiation

46. **FAIR Signposting Profile - Signposting the Scholarly Web** https://signposting.org/FAIR/

47. **Hypertext Style: Cool URIs don't change.** https://www.w3.org/Provider/Style/URI

48. **HTTP Framework for Time-Based Access to Resource States -- Memento**
H Van de Sompel, M Nelson, R Sanderson
RFC Editor (2013-12) https://doi.org/ggqvps
DOI: 10.17487/rfc7089

49. **Versioning Extensions to WebDAV (Web Distributed Authoring and Versioning)**
G Clemm, J Amsden, T Ellison, C Kaler, J Whitehead
RFC Editor (2002-03) https://doi.org/gp37bd
DOI: 10.17487/rfc3253

50. **HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)**
L Dusseault (editor)
RFC Editor (2007-06) https://doi.org/gp37bf
DOI: 10.17487/rfc4918

51. **Upgrading to TLS Within HTTP/1.1**
R Khare, S Lawrence
RFC Editor (2000-05) https://doi.org/gp33dv
DOI: 10.17487/rfc2817

52. **Hypertext Style: Cool URIs don't change.** https://www.w3.org/Provider/Style/URI.html

53. **draft-ietf-mediaman-suffixes-00 - Media Types with Multiple Suffixes** https://datatracker.ietf.org/doc/draft-ietf-mediaman-suffixes/00/

54. **The 'profile' Link Relation Type**
E Wilde
RFC Editor (2013-03) https://doi.org/gp32q7
DOI: 10.17487/rfc6906

55. **W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures** http://www.w3.org/TR/xmlschema11-1/

56. **JSON-LD 1.1** http://www.w3.org/TR/json-ld/

57. **WebSockets Standard** https://websockets.spec.whatwg.org/

58. **Linked Data Notifications** https://www.w3.org/TR/ldn/

59. **The Atom Publishing Protocol**
J Gregorio, B de hOra (editors)
RFC Editor (2007-10) https://doi.org/gp4p2c
DOI: 10.17487/rfc5023

60. **SWORD 3.0 Specification** https://swordapp.github.io/swordv3/swordv3.html

61. **Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content**
R Fielding, J Reschke (editors)
RFC Editor (2014-06) https://doi.org/gh4jxc
DOI: 10.17487/rfc7231

62. **draft-kelly-json-hal-08** https://datatracker.ietf.org/doc/html/draft-kelly-json-hal-08

63. **Web Linking**
M Nottingham
RFC Editor (2017-10) https://doi.org/gf8jcd
DOI: 10.17487/rfc8288

64. **Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing**
R Fielding, J Reschke (editors)
RFC Editor (2014-06) https://doi.org/gp32q8
DOI: 10.17487/rfc7230

65. **HTTP Live Streaming**
W May
RFC Editor (2017-08) https://doi.org/gp32rc
DOI: 10.17487/rfc8216

66. **ISO/IEC 23009-1:2019**
14:00-17:00
ISO
https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/07/93/79329.html

67. **FAIR Digital Object Framework Documentation** https://fairdigitalobjectframework.org/

68. **Handle System Protocol (ver 2.1) Specification**
S Sun, S Reilly, L Lannom, J Petrone
RFC Editor (2003-11) https://doi.org/ggn83x
DOI: 10.17487/rfc3652

69. **Data - W3C** https://www.w3.org/standards/semanticweb/data

70. **Data Catalog Vocabulary (DCAT) - Version 2** https://www.w3.org/TR/vocab-dcat-2/

71. **Bioschemas - Bioschemas** http://bioschemas.org/

72. **NCBO BioPortal** https://bioportal.bioontology.org/ontologies

---

1. For a brief introduction to DOIP 2.0 [11], see $[{https://www.cordra.org/documentation/api/doip.html}].↩

2. Although it is possible with `0.DOIP/Op.Retrieve` to request only particular individual elements of an DO (e.g. one file), unlike HTTP's `Range` request, it is not possible to select individual chunks of an element's bytestream.↩

3. *Datasets* that distribute RDF graphs should not be confused with RDF Datasets used for partioning *named graphs*.↩

4. Presumably this large uptake of JSON-LD is mainly for the purpose of search engine optimization (SEO), with typically small amounts of metadata which may not constitute Linked Data as introduced above, however this nevertheless constitute machine-actionable structured data.↩

5. The `http` protocol (port 80) can in theory also upgrade [51] to TLS encryption, as commonly used by Internet Printing Protocol for `ipp` URIs, but on the Web, best practice is explicit `https` (port 443) URLs to ensure following links stay secure.↩

6. Although it is possible with `0.DOIP/Op.Retrieve` to request only particular individual elements of an DO (e.g. one file), unlike HTTP’s `Range` request, it is not possible to select individual chunks of an element’s bytestream.↩

7. FDOF* renamed to FDOR*. FDOF3/FDOF4 are swapped to FDOR4/FDOR3 in *WD-RequirementSpec-1.0-20220317*.↩