An Overview of Decentralized Web Technologies as a Foundation for Future IPFS-centric FDOs

Authors: Andrei Vukolov (Presenter), Erik Van Winkle, Erik Schultes, Line C. Pouchard, Sina Iman, Philipp Koellinger, Christopher Hill

Format: Presentation on Implementation-based work

Theme: Reports on implementations of FDOs and associated components such as PIDs, PID/FDO Profiles, PID Kernel Attributes, Types, Type Registries, DOIP, FDO compliant repositories, Signposting, etc.

Abstract

dPIDs are an emerging PID technology based on decentralized web architectures and self-sovereign identity movements [1]. dPIDs are PID containers that create persistent storage systems where each object is identified by a unique PID. dPIDs are inherently immune to content drift and resolves deterministically their mapped content, providing a reproducible binding between the content/metadata and identifier. As dPIDs take a decentralized network protocol approach to PIDs, the implementation of FDOF recommendations may have differences (similar to the implementation differences between DOIP FDOs and LD FDOs) [2]. The goal of this presentation is to be a primer on the decentralized web technologies behind dPID and their associated benefits, including a discussion of their potential usefulness for FDOs. dPIDs can form the fabric for a persistent, interoperable FDOs landscape.

Data replication via the peer-to-peer nature of the underlying network and content-based addressing facilitate the implementation of FDO-G2 [3], ensuring long-term persistence and mitigating the risk of data loss via implicit data replication and storage redundancy between network participants. Content addressing gives dPID the property of deterministic and verifiable resolution, exceeding the requirements of FDO-PIDR2. A subsequent benefit of this open protocol-based approach is that dPIDs prevent the formation of vendor-lock-in and data silos, facilitating FDO-PIDR1 and FDO-G1 principles implementation. Importantly, the provenance of data and data updates to dPIDs are registered by digital signatures based on W3C decentralized identifiers (DID), facilitating FDO-PIDR6. The data sovereignty is facilitated using a Directed Acyclic Graph (DAG) approach compliant with FDO-GR4, FDO-GR5 and FDO-GR6 requirement statements. These directed acyclic graphs also allow for granular machine actionability in compliance with FDO-GR1 and FDO-GR11. As PIDs are logged on Blockchain, tombstones for dPIDs are inherently permanent in line with FDO-GR12.

dPID's Underlying Technology

The main technologies underpinning dPID are: Content-addressed storage networks, DAG-based linked data, the Sidetree Protocol, Decentralized Identifiers (DIDs), and blockchain. All the algorithms and associated documentation are open, and publicly accessible, providing reusability (FDO-PIDR1). [4].

dPIDs are built on a content-addressable storage network called the InterPlanetary File System (IPFS) [5]. In IPFS, and content-addressed networks in general, payloads are identified by a hash computed from their content. This hash is called a Content Identifier (CID). The collection of all possible hashes used to locate the stored data blocks forms a global secure and decentralized namespace called the distributed hash table (DHT). An end user enters the CID as a key to specify the data that should be retrieved, then the DHT responds with the addresses of the nodes storing the data. The client then connects directly to one of these nodes to retrieve the data. Through CIDs and the DHT, IPFS completely eliminates content drift and significantly mitigates the risks of link-rot and vendor lock-in by providing built-in data replication capabilities between repositories (G1, G2). An additional benefit, the resolution of dPIDs based on CID resolution is deterministic (FDO-PIDR2).

The dPID technology can be viewed as a containerized file structure, using (Interplanetary Linked Data) IPLD to create DAGs, mapping CIDs with (meta)data in a fashion similar to JSON-LD, where URIs are replaced by CIDs (FDO-GR4, FDO-GR5 and FDO-GR6) [6]. dPIDs are currently constructed for the purpose of identifying research objects [7]. A simplified version of a dPID can be seen in Figure 1 [8].

Decentralized Identity Foundation's (DIF) Sidetree is a higher order protocol which allows nodes in a network to collaborate and keep track of sequential updates to CID-based data objects made by end users, in a trustless fashion (FDO-PIDR6) [9]. Decentralized Identifiers are used to sign these data object commits and gate update rights on the sequential streams. The dPID protocol is built on top of a Sidetree implementation (Ceramic) and associated composable database (ComposeDB), which enables distributed indexing and querying capabilities. While Sidetree on its own doesn't provide a way to index, discover, organize, and query for existing streams (nodes) depending on which schema (entity) they implement, or track references made to other streams, ComposeDB is a type of graph database built on SideTree to achieve all of these properties. An example of a graph of independent authors, their publications, and the relations between the different types of entities can be seen in Figure 2, explaining how dPIDs can cooperate with other PIDs and DIDs in the space (ORCID, DOI, etc).

Finally, blockchain is used as the Anchoring point for Sidetree once nodes in the network have reached consensus on the global state (FDO-GR12). Furthermore, by registering the mutable root CID-based identifiers on a blockchain smart contract registry, dPIDs achieve strong consistency (as opposed to Sidetree's eventual consistency) and enable a namespace that square Zooko's triangle [10], i.e. decentralized, secure, and human readable. The latter property is leveraged to create compatibility with PID schemas which are not hash-based, such as providing a root identifier conformant to the existing PID schemas (.e.g, DOI, ARK, RAID). In this sense, blockchain-based registries provide aliases enabling cross-compatibility between existing PID namespaces and the dPID technology.

Benefits: PID containers as persistent file systems, Deterministic Resolution, Strong consistency and high throughput, FDO-PIDR2

Content-addressed storage (CAS) allow for natively deterministic resolution of CIDs to their mapped content. As such, inheriting the deterministic resolution properties of CAS networks is a design goal for dPIDs (REF). dPIDs can be construed as persistent file systems linking digital objects which allow for granular referencing and resolution to their mapped content (FDO-GR1, FDO-GR11). As such, dPIDs act as "PID containers". The current resolution anatomy of a dPID is as follow [8]:

$$HTTP\ URL\ =\ \{registry\}/\{version\ identifier\}/\{\}/\{Component\ suffix\}$$

Deterministic Resolution: To enable persistent resolution of nodes by address, we employ an algorithm for deterministic graph traversal. . This deterministic traversal simply requires the root dPID, exposing the IPLD DAGs and their associated CIDs. Currently planned traversal algorithms for deterministic resolution include the options listed in table 3 [8].

Strong consistency and high throughput. An inherent trade-off exists between achieving strongly consistent PIDs which satisfy the three desirable properties of a namespace and enabling high throughput. dPIDs solve this tension by anchoring the version-invariant identifier on a blockchain-based registry and leaving all subsequent updates of the mapped DAGs to SideTree's consensus mechanism. For applications that do not require backward compatibility with existing PID namespaces and/or conveniently short URLs for humans (as opposed to long CIDs), Sidetree's eventual consistency can be sufficient [9].

Conclusion

We present a novel PID technology based on content-address networks, decentralized identifiers, and the Sidetree protocol. While dPIDs natively satisfy many of the FDO Forum's FDO Requirement Specifications. dPID FDOs will come with different implementation specifications, just as DOIP FDOs and Linked Data FDOs have implementation differences. By laying a foundation of mutual understanding around the dPID technology, we aim to facilitate conversations around the possibilities offered by deterministic resolution for the emerging world of FDOs and the role of open, content-addressed networks in preserving scientific artifacts.

References

1. Hook, Daniel; Kramer, Bianca; Koellinger, Philipp; Quaderi, Nandita (2023): Innovation, Technology and Infrastructure - APE Conference 2023. figshare. Presentation. https://doi.org/10.6084/m9.figshare.21953864.v1
2. Soiland-Reyes, S., Goble, C., & Groth, P. (2023). Evaluating FAIR Digital Object and Linked Data as Distributed Object Systems. arXiv:2306.07436v2 [cs.DC]. https://doi.org/10.48550/arXiv.2306.07436
3. Ivonne, A., Christophe, B., Daan, B., Maggie, H., Sharif, I., Thomas, J., Larry, L., Karsten, P.-. von G., Robert, Q., Alexander, S., Ulrich, S., Stian, S.-R., Strawn, G., Dieter, . van U., Claus, W., Peter, W., & Carlo, Z. (2023). FDO Forum FDO Requirement Specifications. https://doi.org/10.5281/zenodo.7782262
4. Desci Labs. "nodes." GitHub. Accessed January 12, 2024. https://github.com/desci-labs/nodes.
5. Benet, J. (2014). IPFS - Content Addressed, Versioned, P2P File System. arXiv:1407.3561v1 [cs.NI]. https://doi.org/10.48550/arXiv.1407.3561
6. IPLD. "Docs." IPLD Documentation. Accessed 13 December, 2023. https://ipld.io/docs/.
7. Stian Soiland-Reyes, Peter Sefton, Mercè Crosas, Leyla Jael Castro, Frederik Coppens, José M. Fernández, Daniel Garijo, Björn Grüning, Marco La Rosa, Simone Leo, Eoghan Ó Carragáin, Marc Portier, Ana Trisovic, RO-Crate Community, Paul Groth, Carole Goble (2022): Packaging research artefacts with RO-Crate. Data Science 5(2) https://doi.org/10.3233/DS-210053
8. DeSci Labs & DeSci Foundation. "DeSci Codex: The Collaborative Data Exchange" Accessed 13 December, 2023. https://codex.desci.com/
9. Buchner, D., Steele, O., & Ronda, T. (Editors). "Sidetree Protocol Specification." Decentralized Identity Foundation. Accessed 13 December, 2023. https://identity.foundation/sidetree/spec/.
10. Swartz, A. (2011). "Squaring the Triangle: Secure, Decentralized, Human-Readable Names" Accessed 13 December, 2023. http://www.aaronsw.com/weblog/squarezooko.

Figures and Tables

dPID URL	**End State of Resolution**
Registry:	The registry used to map version-invariant PIDs to SideTree streams
PID:	The registry's unique and version-invariant PID
Version identifier or CID:	The version identifier (e.g. "v1" or "0") mapped to the CID of the manifest file or the manifest CID
Component index	The index of the component in the research object's data model, or the component ID
Component suffix	JSONPath to manifest components. Encoded with slashes and compact format for URL friendliness.

Table 1: Anatomical Description of dPIDs

dPID URL	**End State of Resolution**
https://beta.dpid.org/46/	Predetermined web interface of dPID:46
https://beta.dpid.org/46?raw	IPFS JSON-based manifest corresponding to dPID:46
https://beta.dpid.org/46/v1?raw	IPFS JSON-based manifest corresponding to dPID:46/v1
https://beta.dpid.org/46/3/root/exploring-lupus/casa/J15430131-3409153_cont.py	Predetermined web interface view of “J15430131-3409153_cont.py” in dPID:46/v3
https://beta.dpid.org/46/3/root/exploring-lupus/casa/J15430131-3409153_cont.py?raw	IPFS raw view of J15430131-3409153_cont.py” in dPID:46/v3

Table 2: Example Resolution behaviors for dPID

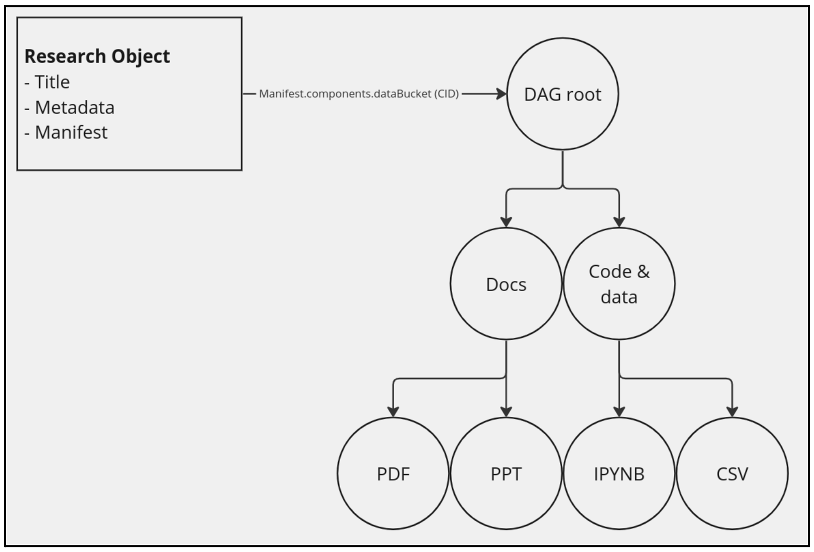

Figure 1: A research object entity and a visualization of its link to the data DAG

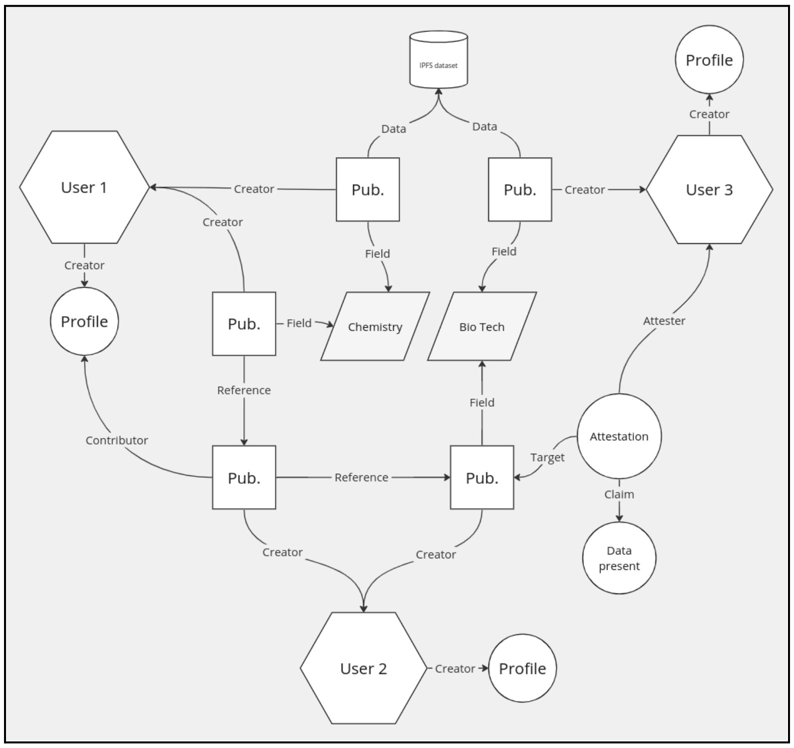

Figure 2: An example of a graph of independent authors, their publications, and the relations between the different types of entities.

Resolution Case	**Overview**	**Traversal Algorithm**
Root node	Addressing the latest state of some node N:	1. Query network for status of node N
Particular version	Addressing a particular commit C of a node N:	1. Query network for status of node N at commit C
Particular time	Addressing the state of a node N as of time T:	1. Query network for update history of node N 2. Find the newest commit C that was anchored before or at time T 3. Resolve node N at commit C
Particular version index	Addressing a particular version k of a node N:	1. Query network for update history of node N 2. Select commit C at index k in update history 3. Resolve node N at commit C
Outgoing edge	Addressing of an outgoing edge from a node N made against some other node:	1. Resolve N 2. Get value R from reference field 3. Resolve node R
Versioned outgoing edge	Addressing a versioned outgoing edge from a node N made against some other node:	1. Query network for status of node N 2. Get value of reference field R and version field C 3. Resolve node R at commit C
Incoming edges	Addressing of an incoming edge to node N, from some node N2 of entity type T:	1. Query network for all nodes of type T 2. Find node N2 with a. Reference field set to N1 3. Resolve N2
Versioned incoming edges	Addressing of an incoming edge to node N as of version C, from some node N2 of entity type T:	1. Query network for all nodes of type T 2. Query network for update history U of node N 3. Find versions V of node N2 with a. Reference field set to N b. Version field value in U before version C 4. Resolve N2, considering V the update history while targeting N at C
Versioned data DAG paths	Addressing a DAG node through UnixFS path P, in research object N as of version C:	1. Resolve N at version C 2. Get value DAG in manifest CID field 3. While P not empty a. Pop first segment S from P b. Set DAG' = lookup(S, DAG) c. Loop with DAG' as DAG 4. DAG is now the addressed node or leaf

Table 3: Traversal Algorithms for deterministic resolution of dPIDs in ComposeDB