Placeholder Name of the conference

Placeholder Session title

https://doi.org/10.52825/xxxx..... DOI placeholder (WILL BE FILLED IN BY TIB Open Publishing)

Published: (WILL BE FILLED IN BY TIB Open Publishing)

Trusty PID

A Pattern for Immutable FDOs

Timm Fitschen[1,*] https://orcid.org/0000-0002-4022-432X, Tobias Kuhn[2] https://orcid.org/0000-0002-1267-0234

[1]IndiScale GmbH, Germany, [2]Knowledge Pixels AG, Switzerland

*Correspondence: Timm Fitschen, t.fitschen@indiscale.com

Abstract. Immutability, the property of data that prevents it from being modified after it is created, can significantly enhance the performance, reliability, and scalability of distributed systems. It ensures process isolation, prevents race conditions, enables efficient caching and replication.This contribution introduces “Trusty PID”, a pattern for immutability of FDO Records building on the Trusty URI specifications. While not all FDOs must be immutable there are use cases where immutability can enhance consistency of the global FDO database, validation of single FDOs and support versioning of FDO Records.

Keywords: FDO, Immutability, Handle System, Nanopublications, RO-Crate, RDF, Versioning, Validation

1. Immutability

Immutability, the property of data that prevents it from being modified after it is created, can significantly enhance the performance, reliability, and scalability of distributed systems. It ensures process isolation, prevents race conditions, enables efficient caching and replication.

This contribution introduces “Trusty PID”, a pattern for immutability of FDO Records building on the Trusty URI specifications. While not all FDOs must be immutable there are use cases where immutability can enhance consistency of the global FDO database, validation of single FDOs and support versioning of FDO Records.

2. Trusty URI

Cryptographic hashes are a natural and powerful tool to create identifiers for immutable pieces of digital data[1]. Creating such hash-based identifiers is trivial for data in the form of fixed bit sequences. Hashes can also be used to identify data at a more general level, such as an identifier for tables irrespective of their format (e.g. CSV or Excel). This can be achieved by providing a normalization function that deterministically transforms a concrete object into a bit sequence that represents its identity at the chosen level of generalization (e.g. tables in CSV and Excel would generate the same normalization if and only if they are equivalent at the chosen level).

Trusty URIs[2] are URI which end with a sequence of characters matching the pattern `{module-identifier}{data}`. The data contains a Base64-encoded cryptographic hash of the content of the resource. The `module-identifier` is a two-character identifier of the

normalization function and the hash function used for the Trusty URI. The specifications define two modules. „FA" is a SHA-256 hash sum of the binary content of a file. The modules „RA" (and „RB") calculate the SHA-256 hash sum of a multi-graph (single-graph) RDF data set which is being normalized and sorted before hashing, such that equivalent graphs result in identical hash sums.

3. Trusty PID

The general idea of Trusty URI also works for FDO. FDOs must be identified by globally unique, resolvable, persistent identifier (PID). To define Trusty PID, we must assume that PIDs allow to contain the hash of the content, i.e. the hash of the FDO Record.

By extending [2], a **Trusty PID** is a PID[FDO Specs] that is a sequence of components. The last component of every Trusty PID ends with at least 25 Base64 characters. The sequence of characters following the last non-Base64 character is called the artifact code. The first two characters of the artifact code are called the module identifier. The sequence of characters following the module identifier is called data part, which is identical to or contains a hash of a normalized FDO Record.

The most prominent examples of PID used in current FDO flavors are IRI-based (ARK, PURL, W3ID) and Handle-based PIDs (DOI, IGSN, plain Handle). Both systems use hierarchically structured, UTF8-encoded strings (IRI) or bit-sequences which have the form of UTF8-encoded strings. Trusty URI use the last IRI-`path` segment to store the hash. Handle PIDs are structured into `prefix`, `/`, and the `suffix` and the suffix may contain a Base64-encoded hash as well. Thus, both systems are equally capable of supporting identifiers for a Trusty PID.

Other examples of identifiers which do limit the range, e.g. ISBN, UUID, ORCID, or ROR, are also widely used as persistent identifiers for different purposes. It will not be possible to apply the Trusty PID pattern to those systems. However, as it was said, not every FDO necessarily needs to be immutable. Also, not every FDO flavor needs to support Trusty PID. It is already beneficial if some FDOs can be defined as immutable and that some clients and services can use their immutability for applications detailed below.

4. FDO Record Hash

None of the above-mentioned Trusty URI modules are out-of-box applicable to FDO in general. The `RA` module could be reused if FDOs of different flavors are first being serialized into RDF. Otherwise, a new module must be proposed and implemented which will transform any Record (and a wide range of serializations, such as the JSON-ish serialization used by the DOA and DOIP protocols, Sign-Posting's `link` tags, JSONLD as used in RO-Crate, and the RDF serialization used in Nanopublications) into a predictable, normalized serialization which can then be hashed using a well-known algorithm. In both cases, the main work that needs to be done, is defining and implementing the serialization of FDO Records of different flavors into a normalized form. However, to be as compatible as possible even with future flavors of FDO an approach is needed which allows to be flexible regarding the normalization function. The different FDO flavors have already found different expressions for similar things, yet it is still open which of these should be strictly considers equivalent. For example, we might use a predicate like ex:hasType in Nanopublications and a Handle-based PID like

21.T11966/FdoType, for which we decide at some point that they should be considered the same and therefore normalized to the same string “has-fdo-type”.

This flexible solution could be a normalization function (perhaps facilitating an FDO Operation) which is referenced via a PID in the Record of an immutable FDO. It would require any Record using a Trusty PID to have an Attribute like `(usesNormlization, FirstNormalization)`. In the future, a new normalization function like ‘ImprovedNormalization’ can be used for new Records in the same way. Old FDOs could be updated (getting a new PID, but keeping an immutable version history), or not. In either case they can be successfully normalized and validated in all system that recognize the Trusty PID and implement the given normalization function.

5. Application

5.1. Validity of FDO Records

Validity of FDO is based on Profile Definitions and Attribute Definitions. Profile Definitions are a list of mandatory and optional Attribute Definitions, referenced by the PID. Records reference the Profile Definition and an FDO is considered valid (among other things, such has having a PID) only if the Record contains at least all Attributes which are listed as mandatory by the Profile Definition. If an FDO Profile Definition changes, e.g., when a mandatory Attribute is being added or replaced, many, if not all Records relying on this Profile Definition will become invalid. The same problem occurs if the Attribute Definition changes.

5.2. Versioning

A solution could be versioning of Profile Definitions and Attribute Definitions. Records would then reference a specific version of an Profile Definitions and the Profile Definition would reference specific versions of the Attribute Definition it considers to be mandatory. However, versioning itself depends on the particular version being something that is immutable. While the Profile Definition X may be updated at some point, the version X(t1) of the FDO must never change, because versioning is essentially a historical fact about the state of the Profile Definition X at some (relative or absolute) point in time.

Data availability statement

No data collected.

Author contributions

TF: Conceptualization, funding acquisition, writing – original draft. TK: Conceptualization, writing – review and editing.

Competing interests

The authors declare that they have no competing interests.

Funding

This work was co-funded by the European Union's Horizon Europe research and innovation programme under grant agreement No 101137725.

References

[1] Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*. https://doi.org/10.48550/arXiv.1407.3561

[2] Kuhn, T., Dumontier, M. (2014). Trusty URIs: Verifiable, Immutable, and Permanent Digital Artifacts for Linked Data. In: Presutti, V., d'Amato, C., Gandon, F., d'Aquin, M., Staab, S., Tordai, A. (eds) The Semantic Web: Trends and Challenges. ESWC 2014. Lecture Notes in Computer Science, vol 8465. Springer, Cham. https://doi.org/10.1007/978-3-319-07443-6_27