HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

HMC PAPER | 2 | FAIR Digital Objects

Helmholtz Kernel Information Profile

December 2022

HMC | HELMHOLTZ METADATA COLLABORATION

Keywords: FAIR Digital Object, HMC, Kernel information profile

DOI: 10.3289/HMC_publ_03

Citation: Helmholtz Metadata Collaboration; Helmholtz Kernel Information Profile, 2022.

Acknowledgement
This publication was supported by the Helmholtz Metadata Collaboration (HMC), an incubator-platform of the Helmholtz Association within the framework of the Information and Data Science strategic initiative.

Call for Review

You are all invited to comment on this version. Please send your feedback by email to info@helmholtz-metadaten.de.

Version: 1.0

This document was generated in a FAIR manner. All previous versions are available on request.

Authors (ORCID)**:** Constanze Curdt (0000-0002-9606-9883), Gerrit Günther (0000-0001-6243-1728) Thomas Jejkal (0000-0003-2804-688X), Christian Koch (0000-0002-4344-0837), Florian Krebs (0000-0001-6033-801X), Andreas Pfeil (0000-0001-6575-1022), Anton Pirogov (0000-0002-5077-7497), Jan Schweikert (0000-0003-4774-2717), Pedro Videgain Barranco (0000-0003-0000-4784), Martin Weinelt

HMC group: Cross-cutting Topic Working Group “From Development to Deployment”

Editors: Sören Lorenz, Ants Finke, Christian Langenbach, Klaus Maier-Hein, Stefan Sandfeld, Rainer Stotzka

Licence: Attribution 4.0 International (CC BY 4.0)

Contact: HMC Office
GEOMAR Helmholtz Centre for Ocean Research Kiel
Wischhofstr. 1-3
24148 Kiel, GERMANY
E-mail: info@helmholtz-metadaten.de

www.helmholtz-metadaten.de

https://creativecommons.org/licenses/by/4.0/

Content

Preface

In this document we present our proposal of basic properties that should be part of every PID Kernel Information Profile and PID Record created in the framework of the Helmholtz Metadata Collaboration (HMC). By following these suggestions, we aim to establish a top-level commonality across all research fields in the Helmholtz Association allowing to base cross-community services on top. However, the results presented herein are not limited to the Helmholtz Association, but can also be adopted outside the Helmholtz Association in order to connect contents of data infrastructures. Before reading this document, we recommend to familiarize with basic terms and concepts like Persistent Identifiers, PID Kernel Information Profiles and FAIR Digital Objects as we will touch them only briefly.

1 Introduction

The mission of the Helmholtz Metadata Collaboration (HMC) is to facilitate the discovery, access, machine-readability, and reuse of research data of the Helmholtz Association. This will be achieved by digitizing and qualitatively enriching research data with metadata and implementing this approach throughout the organization. Concerning the wide scope of Helmholtz research fields, there are a couple of major challenges to cope with, e.g.,

- existing research data management in different levels of maturity,
- numerous of applicable standards, existing services and proprietary interfaces,
- huge amounts of siloed data due to a lack of interoperability and machine-actionability.

One first step in addressing these challenges is to create a common ground on which further commonalities can be identified and implemented. A promising approach on how to create such a common ground for realizing machine-actionable and interoperable research data management also supporting data stored in legacy systems was intensively discussed in different groups of the Research Data Alliance (RDA) and beyond in the past years. The results of these discussions were summarized by the concept of FAIR Digital Objects (FAIR DOs). The main idea behind FAIR DOs is to introduce a thin layer of services, policies and machine-actionable, interoperable descriptions (i.e., data types) of digital resources on top of existing infrastructures. At the core of FAIR DOs are Persistent Identifiers (PIDs) with the purpose of identifying any kind of digital content in a globally unique and persistent way. This applies to data and metadata as well as to data types, software and even services - everything mentioned is identified via PIDs. However, PIDs are just alphanumeric character sequences not helping much with machine-actionability for interoperability, but they are typically accompanied by a PID record, which is basically a key-value-map for holding PID-related information directly at the PID resolver.

In the context of FAIR DOs, this PID Record is used to hold machine-readable information describing what is referenced by a certain PID allowing fast decision-making based on the PID Record before trying to resolve the referenced content. The contents of a PID Record, i.e., which keys are supported, their cardinality and value ranges for each key, are defined by a PID Kernel Information Profile (KIP). These PID KIPs are in the focus of this guidance document. In the following we describe PID KIPs in detail, their characteristics, and the motivation and process towards defining a Helmholtz KIP. At the end, we give a few examples and an outlook how to proceed based on what was presented.

PID Kernel Information Profiles

The idea of PID KIPs was initially drafted by the **RDA Working Group (WG) on PID Kernel Information** [1]. In their **final recommendations** [2] a list of 15 basic properties that should be part of every PID record was defined, together with a list of guiding principles describing the appropriate creation and usage of PID KIPs. These principles, which are also of particular importance for our work, are the following:

> **Principle 1:**
>
> The primary purpose of a PID KI record is to serve machine actionable services.

This means, you **should avoid** including attributes whose **main audience are humans**, e.g., titles, textual descriptions, and comments. This information should be stored in linked FAIR DOs referring to metadata documents which are made machine-readable by their data type.

> **Principle 2:**
>
> The PID KI record is a non-authoritative source for arbitrary metadata. If the information for an attribute duplicates metadata maintained elsewhere, the external source is the authority.

This principle is related to the use of PID kernel information. All information stored in the PID record are non-authoritative, which means, that they might be outdated or even invalid at access time. It's in the duty of the provider of such information to keep them up to date in the PID record. Thus, the only authoritative source of metadata is the original source of information stored in the PID record.

> **Principle 3:**
>
> PID Kernel Information is stored directly at the resolving service and not referenced.

This means, that values which are **required for interpreting** a PID record should **not be stored in an external system**, e.g., additional metadata necessary for understanding the referenced resource. The main reason is, that mandatory round trips using different interfaces should be avoided for performance, interoperability and reusability reasons. Where this is not possible, caching should be used very heavily.

> **Principle 4:**
>
> Change to a PID KI record can be [applied] only by a data object owner or owner delegate (e.g., PID record manager).

This principle implies, that there should be some kind of authentication and authorization mechanisms in place avoiding an unauthorized/uncontrolled modification of PID records.

Principle 5:

PID KI record values should change infrequently with update initiated only by an appropriate authority, avoiding human interaction on updates where possible.

The guiding principle here is that PID records are **made by machines for machines**. Human interaction would be time-consuming and error-prone, thus PID records should be filled and updated according to a KIP by tools taking care of translating information coming from, e.g., repositories into information following the KIP definition used.

Principle 6:

Attributes (items) in the profile are expressed as key-value pairs where the values are simple (indivisible).

In other words, you should **avoid using complex values**, e.g., JSON or XML structures, as a consumer might have difficulties interpreting them. Furthermore, it is not recommended to use frequently changing values, e.g., incomplete lists, as this would increase the frequency of changes of PID records making a proper caching and indexing of them hard.

Principle 7:

A profile should adhere to the following two requirements. Doing so may reduce migration issues in the event of profile revision:

a) Every attribute in a profile depends only on the identified object and no other objects.

b) Every attribute in a profile describes the object directly and does not describe another attribute in the same profile.

Explaining this principle works best with a counter example. Let us assume our PID refers to a file and we want to include file format information, e.g., format name and version, into the PID record. Including both information into the KIP of the referenced file would break with this principle, as the version number is related to the file format and not the referenced resource. Instead, file format and version should be stored in a separate PID's record and this PID should then be used in our profile to uniquely identify the file format in its specific version.

Following these principles, the members of the WG identified the aforementioned list of 15 properties defined as the Draft KIP, which are:

- **PID** - The global identifier for the object.
- **kernelInformationProfile** - The PID of the KIP itself used for validation of the PID record.
- **digitalObjectType** - A PID referring to a type definition for the referenced digital object.
- **digitalObjectLocation** - A URL pointing to the digital object associated with the PID belonging to this PID Record.
- **digitalObjectPolicy** - A PID pointing to policy information, e.g., access and modification policies and license information.

- **etag** - A checksum of the referenced digital object.
- **dateCreated, dateModified** - ISO 8601 date string indicating the modification/ creation date of the referenced digital object.
- **version** - A version string related to the referenced digital object.

Additionally, six properties are used for describing the provenance of the referenced digital object, e.g., to refer to its revisions, specializations or alternatives. This profile is also registered at an instance of the Data Type Registry hosted at Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen (GWDG) and is available under PID `21.T11148/0c5636e4d82b88f86132` [3].

Why an Extension for the Helmholtz Association?

The idea of a Helmholtz KIP has two reasons: on the one hand it is supposed to provide a top-level commonality of all FAIR DOs created within the Helmholtz Association. Based on the contained information, basic resolution, validation and retrieval should be enabled independent of the research field an FAIR DO originates from. On the other hand, these envisioned applications are also a reason for having a Helmholtz KIP, i.e. extending the RDA recommendations. Our evaluation of the RDA Draft KIP turned out its limitations towards a contextual assignment. Furthermore, several assumptions, e.g., that a URL is a sufficient description to access a digital resource, and a lack of required base services, e.g., a policy registry, led us to extend the recommended draft KIP. In the next chapter, services and workflows on how to realize and make use of KIPs are presented. Following that, we would like to present the Helmholtz KIP, which is supposed to be a minimal, interoperable set of properties which should be part of every KIP used in the Helmholtz Association, but can be extended if required. Finally, this document will be summarized with some real-world application examples of the proposed KIP and outlook on the future work of this topic.

2 Architecture

KIPs and Data Types play a central role in the FAIR DO concept. In this chapter, we will elaborate different components involved in storing, applying and using KIPs for validation. Finally, we will elaborate the processes which are enabled by those components.

PIDs, PID Records and PID Systems

PIDs are heavily used in scientific environments. Examples are **DOIs**, **ORCIDs**, or **Handles** [4-6], each managed in their own infrastructure (or system). They are globally unique strings for identifying resources, e.g., publications, persons or (meta-)data. PIDs can be resolved by their systems resolver(s), which means to retrieve additional information associated with the PID. This set of information is called the PID record and can usually be seen as a list of key-value-pairs. Among this information is usually administrative metadata and the location of an object (often a URL to a landing page). Some PID resolvers will directly redirect users to this location by default, hiding the PID record.

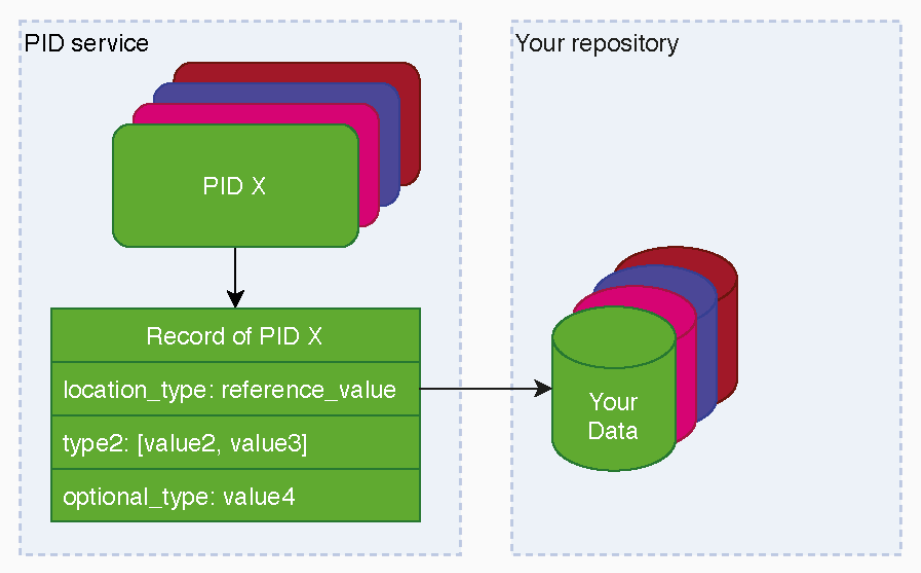


Figure 1: A PID service stores PIDs and their records. Records must be able to represent key-value relations. From this record, machines are able to act, e.g., are able to get the object (e.g., data in this image which could belong to a sensor's output or a file in a repository).

One important aspect of FAIR DOs is to have PIDs with rich metadata stored in the record. The amount and kind of stored information will depend on many aspects like disciplines, the kind of data, legacy aspects and more. For interoperability, it is important to have this information available in a structured and machine-actionable way. The Helmholtz KIP defines recommendations about which metadata should be available.

A number of PID systems do exist. A PID system stores the PIDs and their assigned records and offers to obtain (resolve) a record, given a valid PID. In case a system being in use is not able to represent records using the profile structure, they can still be used as before. To apply the profile in such a case, one can, for example, register an additional PID at a PID system which supports the structure, and use the PID of the incompatible system in the

digitalObjectLocation field in the record, which is described in the Helmholtz KIP. The FAIR DO should then be shared (or cited) using the compatible PID, but can internally still use the incompatible PID.

The Handle.net registry is a PID system which is also used in the data type registries of Gesellschaft für wissenschaftliche Datenverarbeitung mbh Göttingen (**GWDG**) [7]. It allows federation and flexible records. To register PIDs, one will need to obtain a PID prefix. There is a **global Handle PID resolver** [8], which works for all members in the federated Handle system, including DOIs, which are in fact branded Handle PIDs.

Data Types, Profiles and Data Type Registries

A Data Type is a schematic description of a data structure, which has a PID assigned to it. Similar to data types in programming languages, it defines which values it can hold and might be constructed or aggregated by other data types. It differs from these however, since it defines also semantic information about its possible values. Thus, a machine can recognize the meaning of the value. Within a PID record, Data Types, represented by PIDs and registered in a Data Type Registry (DTR), are used as keys. The values of these keys must follow the type's rules. Thanks to the machine-readable type definition, the value can be validated automatically.

A PID KIP defines how a record should be composed of Data Types, similar to a schema or policy. It defines an aggregation of mandatory and optional record fields and allows to validate whether the record actually fits the definition. This implies that a profile can act as a policy (e.g., for automated consistency checking) and as a type for PID records, e.g., for enhanced machine-actionability. To define a record-profile relation, a profile must be referenced by a record that claims to follow its definition. This means that, in a given record, there must be a type whose value is a PID pointing to the profile of that record. Due to their similarity to types, profiles might be defined, stored and accessed using a DTR. They do not necessarily need to be within the same instance as the Data Types, though. Like Data Types, the registry will assign a PID to each profile in the moment it is created.

The GWDG is offering multiple instances of Data Type registries. The **testing instance** [9] is for prototyping and testing profiles and Data Types, which means it will store both. A **production instance** [10] is also available, with plans to distinct between profiles and types.

Processes

In order to create, update, retrieve and validate PIDs according to a KIP, a client has to interact with the infrastructure mentioned before, namely the DTR, the profile registry (if not the same as the DTR) and the PID service. To simplify the adaption of clients to the FAIR DO concept, it is advised to use a service (or library) to hide this complexity.

We recommend a system in-between, abstracting the use of both the PID system and the registries. There is an RDA recommendation defining such services, calling them "**PID Information Types (PIT) Services**" [11]. Such services can be set up for a client tool which needs to work with PIDs and KIPs to offer all this functionality in one simple-to-use interface.

An **early adoption of the PIT Service** [12] concept is being developed by **FAIR Data Commons** [13]. It requires the PID record information to contain a type indicating the PID of a profile and will perform the validation when creating or updating PID records. It also allows to resolve a PID and to explicitly validate it, as well as some advanced interaction with types and profiles.

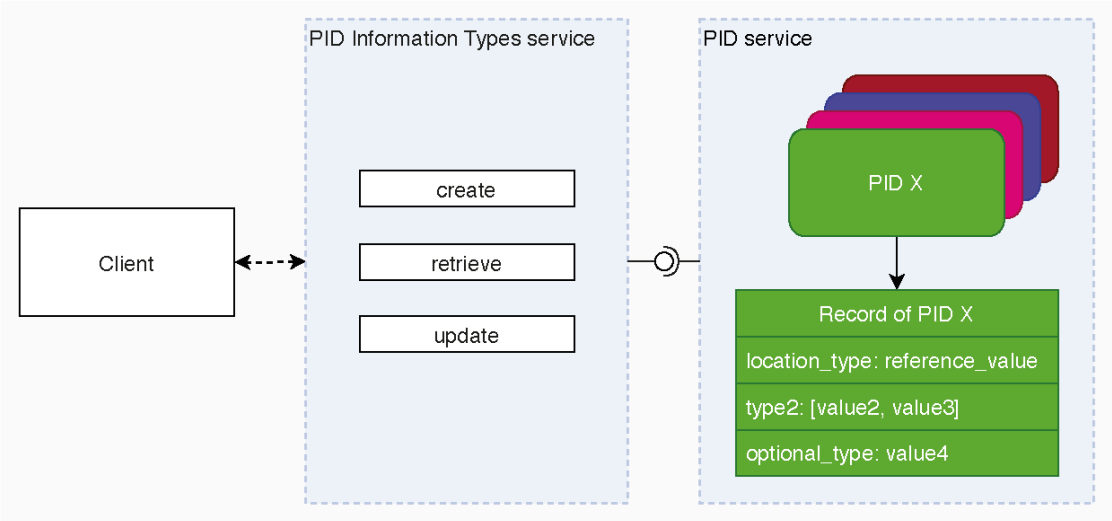


Figure 2: The PIT Service abstracts the PID service in the way shown in the image, unifying the API for different services. But it also offers implicit PID record validation, so the client does not have to do it.

Creating and Updating PIDs

The main difference between creating a new PID and updating an already existing one is that, in the latter case, you will need to know the PID in order to update its record. Besides this, the process is the same. It should be noted that an update should not happen too often. The main reason being maintaining the FAIRness of the object, for example in case the location of the object changed. In both cases, for creating and updating a PID, it should be verified that the record is still valid, i.e., machine-actionability is given. Both processes can be summarized by the following steps:

1. Choose a profile that fits your use case.
2. Put together the PID record information. In the case of an update, you may do this by resolving the PID and modifying the record according to your needs. In the case of a new PID, you will need to create a completely new record.
3. Use profile validation to make sure that your record only contains machine-actionable types, that all values are valid and that all mandatory types are available in the record. This usually involves a lot of requests to a DTR and is accelerated by caching within the PIT Service.
4. When successful, use the API of the PID system you chose to create a new PID or update the existing PID's record.

A PIT Service implementation can simplify these processes by replacing Step 3 and 4 by a simple API call. It also offers a common resolution API for implemented PID services. In the case of an update, where you want to retrieve the current record (by resolving the PID) in Step 2, this might be useful.

Deleting PIDs

The idea of PIDs is not to be deleted, but to be maintained. There should be no need to delete an PID, as there are enough PIDs available. The reason is simple: once a PID has been published it is impossible to know if it has been used, for example, to cite an object. If the object is no longer available or has been registered by accident, the record should reflect this fact by a concept called a tombstone. A tombstone consists of one or more attributes, indicating the state of the object and explaining what happened to it. It is often imagined to be a human readable string, so this message can be displayed to the user. But as of Principle 1 in the introduction, it will be better to use a type with a machine-readable value that explains the problem, similar to HTTP status codes, which can be translated to human readable messages. Obviously, those attributes have to be accepted by the profile.

Resolving PIDs

The idea of PID resolution is simple: There is an API that takes a PID and returns its record, e.g., as JSON representation. This API, including the protocol, the input format, and the output format might look completely different depending on the PID system. To have a common API for such systems, abstraction services can be used to hide those differences. PIT services are

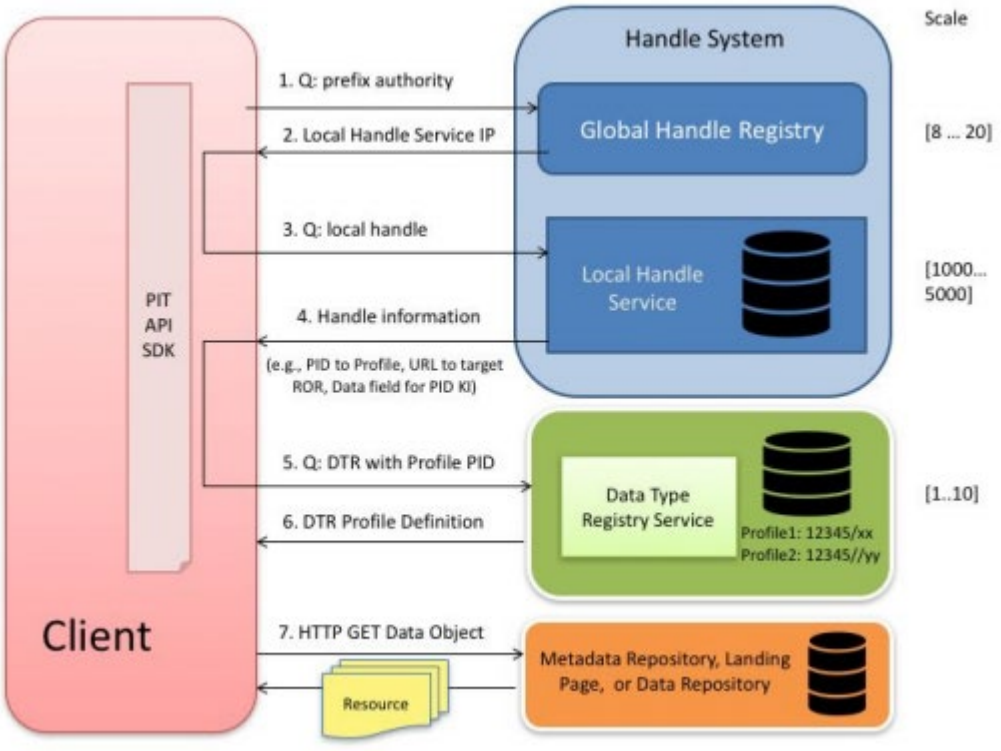


Figure 3: PIT architecture and data flow as illustrated in the RDA Recommendations on PID Kernel Information.

such abstractions, but also implement the more complicated process of validating the records (using the profile).

Figure 3 is taken from the **RDA Recommendations on PID Kernel Information** [2] and describes the process of resolving a PID from the Handle system, then retrieving the profile definition (for validation with in the PIT service) before the client receives the record from the service to finally accessing the data object.

Using the record information, a client can use the types and values to work with it, e.g., to get the actual object. A record usually also contains information spanning up a PID graph by linking to FAIR DOs with some kind of relationship to the records object, e.g., metadata, previous or newer versions and other data the client might be interested in.

3 Helmholtz Kernel Information Profile

The idea behind a custom Helmholtz Kernel Information Profile (KIP) is to define a Helmholtz-wide minimal KIP to be able to exchange FAIR DOs in a standardized way. Based on the Kernel Information Profile recommendations of the RDA, the Helmholtz KIP was extended by fields which are required to create a minimal FAIR DO in the shared view of the HMC Hubs. We expect that many discipline-specific KIPs will emerge, since every discipline has its own requirements about data and metadata. By basing all discipline-specific KIPs to a Helmholtz KIP we can guarantee minimal interdisciplinary interoperability. The following section will describe limitations of the RDA recommendations and explain our solutions. Followed by an overview of the fields of the Helmholtz KIP. In the last section we discuss limitations of the Helmholtz KIP and give recommendations on information that may be used in discipline-specific profiles.

Relation to RDA Recommendations

The Helmholtz KIP is based on an endorsed RDA recommendation, although some noticeable differences are present. This section introduces those differences and presents a detailed view of the proposed KIP for the Helmholtz Association.

The fields (RDA) *kernelInformationProfile*, (RDA) *digitalObjectType* and (RDA) *digitalObject-Location* are directly inherited from the RDA's recommendation and keep the same meaning and cardinality. The HMC recommendation adds the optional property *digitalObject-LocationAccessProtocol* to include information about how to access the data which is referenced, although this field may be included as part of a different object in the future. The (RDA) *dateCreated* and (RDA) *dateModified* properties are again taken from the RDA recommendation, but the HMC one adds *underEmbargoUntil*, comparable to properties like 'datePublished' in **other specifications** [14]. The (RDA) *digitalObjectPolicy* and *checksum* properties are also defined in the same way as in the RDA recommendation, as well as (RDA) *version*. There is some consideration about whether or not version information may overlap with *provenanceGraph.* It should be noted that *license* is defined on its own and not as part of the (RDA) *digitalObjectPolicy* as a temporary solution to store the license, while the (RDA) *digitalObjectPolicy* is being defined and a system to hold policy objects has been developed. The Helmholtz KIP also includes the properties *signature, topic, locationPreview/locationSample* and *contact* which are not part of the RDA recommendation.

Properties that allow to create relations to other FAIR DOs are also defined. The properties (RDA) *wasDerivedFrom,* (RDA) *specializationOf,* (RDA) *wasRevisionOf,* (RDA) *hadPrimarySource,* (RDA) *wasQuotedFrom* and (RDA) *alternateOf* are directly adopted from the RDA recommendation. Additionally, this recommendation includes the properties *isMetadataFor, hasMetadata, wasGeneratedBy* and *provenanceGraph.*

Description of Profile Properties

In this chapter, we describe the single properties of the Helmholtz KIP. Each property is described in the following way:

Human-readable name of the property

Semantic	*A short description providing information about the purpose of this property.*
Cardinality	*Information about cardinality and obligation of the property. Possible values are: 1 (single occurrence, mandatory), 0/1 (single occurrence, optional), 1+ (multiple occurrences, mandatory), 0+ (multiple occurrences, optional), 1r (single occurrence, optional but recommended)*
Format	*Information about the format of a value of this property. In the course of this document, human-readable format names are used. The implementation of a KIP will use PIDs pointing to a data type registered in a Data Type Registry for unambiguousness.*
Example	*Example(s) how values for the described property may look like.*

Room for further elaboration for extending the short description, for providing further descriptions, or to summarize rationales coming from discussions in Cross-Cutting Topic working group 4 - From development to deployment (CCT4).

An RDA logo next to the human-readable property name states, that the particular property was adopted from the RDA recommendations without changes.

kernelInformationProfile

Semantic	A PID pointing to the KIP which provides the structure of this PID record.
Cardinality	1 (mandatory)
Format	PID
Example	`21.T11148/0c5636e4d82b88f86132`

The *kernelInformationProfile* property points to the KIP a PID record follows. It allows to validate the record and to obtain information about all contained properties in a machine-readable format. Thus, the KIP is a special kind of Data Type describing a PID record and is therefore available in a Data Type Registry.

digitalObjectType

Semantic	A PID pointing to a data type in a Data Type Registry describing the object referenced by *digitalObjectLocation*.
Cardinality	1 (mandatory)
Format	PID
Example	21.T11148/2037de437c80264ccbce

According to the RDA recommendations, the idea was to have data types uniquely identifying the type of the object referenced by *digitalObjectLocation* in order to allow automated decisions. Discussions in CCT4 brought up the potential need for more flexibility, e.g., providing additional information like *mimeType* or applicable *standards*. A solution in this direction requires further investigation in the future and in a broader context.

digitalObjectLocation

Semantic	A web-resolvable pointer to the actual object described by this PID record.
Cardinality	1+ (m and repeatable)
Format	URL
Example	https://reposito.ry/data.bin https://doi.org/10.5281/zenodo.5091604

Typically, *digitalObjectLocation* is a web-resolvable pointer accessible via HTTP. In this context, protocol-specific features like authentication or content negotiation may or may not be supported. The repeatability of *digitalObjectLocation* allows to provide alternative locations for the **identical** object. It is not meant to be used to provide different representations, e.g., a byte sequence and a landing page. In cases where *digitalObjectLocation* is not accessible via HTTP, *digitalObjectLocationAccessProtocol* can be used to provide more information to an access client if required.

digitalObjectLocationAccessProtocol

Semantic	Additional information which can be used by an access client for accessing *digitalObjectLocation*
Cardinality	0/1 (optional)
Format	String
Example	{“protocol”:“DOIP”, “version”:“2.0”, “type”:“0.TYPE/DO”}, {“protocol”:“HTTP”, “type”:“application/json”}, {“protocol”:“HTTP”,“type”:“text/html”}

The idea for a *digitalObjectLocationAccessProtocol* property came up while discussing about how to access *digitalObjectLocation* with protocols different from HTTP. But also, for HTTP endpoints not supporting proper content negotiation, having information about supported (mime)type(s) can be beneficial for an access client. In this first version of the Helmholtz KIP all members of CCT4 agreed to use a complex JSON structure as a value for *digitalObjectLocationAccessProtocol*, even if this breaks with Principle 2 described in the introduction chapter. However, in future versions this information should be resolvable via PID and stored either in its PID Record or in a dedicated service.

dateCreated

Semantic	The date in ISO 8601 format, when the object referenced by *digitalObjectLocation* was initially created.
Cardinality	1 (mandatory)
Format	ISO Date/Time
Example	2021-04-14T10:43:31Z, 2021-04-14T10:43:31.175+00:00, 2021-04-14

dateModified

Semantic	The date in ISO 8601 format, when the object referenced by *digitalObjectLocation* was modified.
Cardinality	0/1 (mandatory if applicable)
Format	ISO Date/Time
Example	2021-04-15T11:23:12Z, 2021-04-15T11:23:12.123+00:00, 2021-04-15

As there was some discussion in CCT4 about what the properties *dateCreated* and *dateModified* refer to, we want to explicitly point to Principle 5, presented in the introduction chapter. It says that *"Every attribute in a profile describes the object directly and does not describe another attribute in the same profile"* [2]. This also applies to the PID Record itself, such that these dates are only used to describe the object referenced by *digitalObjectLocation* and must not be used to e.g., describe the creation date of the PID Record.

underEmbargoUntil

Semantic	The date in ISO 8601 format, when the object referenced by *digitalObjectLocation* is planned to be publicly accessible or was made publicly accessible.
Cardinality	0/1 (optional)
Format	ISO Date/Time
Example	2024-04-14T10:43:31Z, 2024-04-14T10:43:31.175+00:00, 2024-04-14

The motivation for introducing this property was the common practice in some scientific domains, to put (raw) data under embargo for some years, before it is made publicly available. However, even if the primary data is under embargo there might be publicly accessible metadata available and citation might be desirable. Access restrictions are typically enforced by the systems in which the data is stored, e.g., via authentication and authorization mechanisms. From the KIP perspective, this property makes sense insofar, that before this date access restrictions can be expected for the public and dereferencing *digitalObjectLocation* will only make sense if appropriate credentials are presented to the access client.

digitalObjectPolicy

Semantic	A web-resolvable pointer to a policy object describing e.g., object access and modification policies.
Cardinality	0/1 (optional)
Format	PID
Example	-

According to the RDA recommendations, the policy object should allow that "*a caller should be able to determine the expected future changes to the object from the policy, which are based on managed processes the object owner maintains*." [2] In the course of the recommendations document, the RDA WG further states, that this policy object should contain for example the following three fields:

- **objectLifeCycleType**: A mandatory string with values like *static, dynamic_irregular,dynamic_regular*.
- **objectTombstoneInformation**: An optional string stating why the object referenced by *digitalObjectLocation* has gone.
- **objectLicense**: An optional PID or URL pointing to a license for the object referenced by *digitalObjectLocation*.

However, as there is currently no system available or planned for holding this kind of policy information, we decided to use for the moment only the license information stored in a separate property. As soon as an international consensus was found on how to manage policy information, the Helmholtz KIP will be updated accordingly.

version

Semantic	Version of the object referenced by *digitalObjectLocation*, either as single number or following semantic versioning conventions.
Cardinality	0/1 (optional, mandatory for all objects with at least one predecessor)
Format	Integer or String
Example	1, 1.0.0, 1.2.3

The *version* property allows to distinguish between two versions of the same object. It can be used either just as an indicator for changes, e.g., by counting up a single number, or to add semantic information about applied changes, e.g., by using semantic versioning. According to the RDA recommendations, the version property is mandatory for all objects with at least one predecessor version. In that case, the predecessor should point to the previous version using one of the PROV-related properties, e.g., *wasDerivedFrom*.

license

Semantic	URL referring to the license of the object referenced by *digitalObjectLocation*, its modifiers (if applicable), and its version.
Cardinality	1r (recommended)
Format	URL
Example	https://www.apache.org/licenses/LICENSE-2.0, https://creativecommons.org/licenses/by-nd/4.0/legalcode

As mentioned in the *digitalObjectPolicy* property, the members of CCT4 decided to pull out the *license* property as long as there is no clear way of storing policy information. Another open question, which has not yet been fully resolved, is how to represent a license in a PID record. There are three layers of licenses: the license contract itself, a human readable representation and a machine-readable representation. Unfortunately, especially for the latter layer there seems to be no common consensus. Only for **Creative Commons with CC_REL** [15] does a specification exists to represent a license in a machine-readable way. Thus, for the time being, we recommend to provide license information in the form of a URL pointing to the license document as long as no general way of representing licenses in a machine-readable form is available.

checksum

Semantic	Checksum of the object referenced by *digitalObjectLocation* in any supported checksum format.
Cardinality	1 (mandatory if applicable)
Format	Formatted String (there exists a data type for this)
Example	sha1:d6605ede08f4a56aab089f2b8a6447b56739761a

The main purpose of the *checksum* property is to validate the content received from *digitalObjectLocation* to detect transmission errors or subsequent changes. Typically, the checksum should not have to be calculated at the time of PID creation but should already be available in the system *digitalObjectLocation* is pointing at. In contrast to the RDA recommendation, CCT4 decided to use the widespread approach of combining checksum algorithm and value in a single string value, e.g., sha1:d6605ede08f4a56aab089f2b8a6447b56739761a, instead of creating a special data type for each checksum algorithm. By doing so, reusing existing checksums is much easier and validation can be performed by available tools without requiring the additional roundtrip of obtaining the checksum algorithm from a PID.

signature

Semantic	A cryptographic signature of this record in a specified format, including especially the checksum for advanced integrity checks and assumptions about reproducibility.
Cardinality	0+ (opt and repeatable)
Format	Probably like whatever mail standard is used most commonly.
Example	-

The idea of a *signature* property came up while discussing the role of the checksum property. As the checksum is only related to the object referenced by *digitalObjectLocation* it makes no assumptions on the PID record itself. In order to provide a certain level of trust in the PID record, the idea was to add a signature property verifiable via a public key infrastructure like the ones used for email, https certificates or vaccinations. In some PID systems, e.g., the ones based on the Handle.net Registry, signatures are already included and will be used in the future.

topic

Semantic	One or more topics the object referenced by *digitalObjectLocation* is related to.
Cardinality	0+ (opt and repeatable)
Format	Controlled Vocabulary
Example	http://vocabularies.unesco.org/thesaurus/concept3052 (Artificial Intelligence)

The property a *topic* can be used to preselect specific FAIR DOs based on their PID Record or to add contextual information on what the object referenced by *digitalObjectLocation* is about. This should also be done using values from one or more controlled vocabularies, e.g., **FAIRsharing Subject Ontology** [16] or the **UNESCO thesaurus** [17] or one of its subgroups.

locationPreview / locationSample

Semantic	A web-resolvable pointer to a preview, e.g., a low-resolution image, of the object referenced by *digitalObjectLocation*.
Cardinality	0+ (opt and repeatable)
Format	URL
Example	https://reposito.ry/sample.bin

The idea behind *locationPreview* was to have a dedicated property for providing the object referenced by *digitalObjectLocation* in another representation. There are many potential use cases, e.g., to provide a subset of a huge dataset for software testing, to provide a low-resolution preview of image data, or to offer a preview of an embargoed dataset to allow the development of processing tools to use the data as soon as it is public. The members of CCT4 are aware, that this property is contradicting with Principle 1 of Kernel Information Profiles. Nevertheless, all CCT4 members agreed to use this property because the expected benefit outweighs the cost at this point.

contact

Semantic	A web-resolvable pointer to an institution or a person responsible for the object referenced by *digitalObjectLocation*.
Cardinality	0+ (opt and repeatable)
Format	URL
Example	http://orcid.org/0000-0003-2804-688X, https://ror.org/04t3en479

Allowing to add contact information has several advantages, but also disadvantages under some conditions. A clear advantage is that it increases trust in the data, as there is someone who can be contacted if any questions about the data arise. On the other hand, scientists may often change their focus or affiliation and are no longer able to serve as contact for the data. Thus, in addition to a person identifier, also an identifier of an institution might be added serving as long-term contact.

hasMetadata

Semantic	One or more PID(s) referring to related FAIR DOs providing metadata for the object referenced by *digitalObjectLocation*. It is the inverse to *isMetadataFor.*
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/095c4361f3044b203171

isMetadataFor

Semantic	A PID pointing to another FAIR DO describing the object referenced by *digitalObjectLocation*. It is the inverse to *hasMetadata.*
Cardinality	0/1 (optional)
Format	PID
Example	21.T11148/095c4361f3044b203171

The motivation for introducing the properties *hasMetadata* and *isMetadataFor* is to offer the possibility to link related FAIR DOs to each other, e.g., to build up a FAIR DO graph easily. Furthermore, they allow to identify metadata describing a data-focused object (*hasMetadata*) and to find data belonging to a metadata-focused object (*isMetadataFor*). With regard to machine-actionability these properties allow to decide, whether a sufficient amount of metadata is available in order to interpret associated data for a certain use case.

wasGeneratedBy

Semantic	A PID pointing to an activity which generated the object referenced by *digitalObjectLocation*.
Cardinality	0/1 (optional)
Format	PID
Example	21.T11148/a12d5666e7648f164348

Like the following fields, *wasGeneratedBy* is based on an element of the **PROV-DM ontology** [18] to be used in the Helmholtz KIP. It is not part of the RDA recommendation and it was introduced to offer a place to put e.g., information on which software in which version was used

to generate certain data. For full machine-actionability, the PID used as value of this property should link to a record that follows a special profile, which still has to be discussed.

wasDerivedFrom

Semantic	A PID pointing to another FAIR DO from which the object referenced by *digitalObjectLocation* was derived from.
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/152e4565632ac45f2319

According to PROV-DM, derivation is characterized as "[...] transformation of an entity into another, an update of an entity resulting in a new one, or the construction of a new entity based on a pre-existing entity." [18] Examples could be format changes or performing a data analysis on raw data.

As *wasRevisionOf*, *hadPrimarySource* and *wasQuotedFrom* are more specific sub-properties of *wasDerivedFrom*, PROV-DM recommends to prefer these fields over *wasDerivedFrom* where applicable.

specializationOf

Semantic	A PID pointing to another FAIR DO which is a specialization of the object referenced by *digitalObjectLocation*.
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/152e4565632ac45f2319

According to PROV-DM, an "entity is a specialization of another that shares all aspects of the latter, and additionally presents more specific aspects of the same thing as the latter." [18] An example could a be an object enriched by additional, use-case or domain specific information, e.g., annotations.

wasRevisionOf

Semantic	A PID pointing to another FAIR DO which is a specialization of the object referenced by *digitalObjectLocation*.
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/8791ac4631ef810de284

According to PROV-DM, *wasRevisionOf* may point to "[...] a derivation for which the resulting entity is a revised version of some original." [18] An example could be the revision of a FAIR DO which was required to fix errors in its referenced content, e.g., data or metadata.

hadPrimarySource

Semantic	A PID pointing to another FAIR DO which is a specialization of the object referenced by *digitalObjectLocation*.
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/56ba981f09ec54532ea6

According to PROV-DM, "a primary source for a topic refers to something produced by some agent with direct experience and knowledge about the topic, at the time of the topic's study, without benefit from hindsight." [18] This means, that *hadPrimarySource* specifies *wasDerivedFrom* in a way, that it only points to unprocessed information captured directly after its generation without possible opinioned processing, e.g., raw data captured by an instrument.

wasQuotedFrom

Semantic	A PID pointing to another FAIR DO from which the object referenced by *digitalObjectLocation* was fully or partly quoted.
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/402e89cd01298ae23b61

According to PROV-DM, *wasQuotedFrom* should be used "[...] for the repeat of (some or all of) an entity, such as text or image, by someone who may or may not be its original author." [18] It specifies *wasDerivedFrom* in so far, that it refers to another FAIR DO which partly of fully refers to the same content, e.g., to a subset of a huge data set.

alternateOf

Semantic	A PID pointing to another FAIR DO which is an alternate of the object referenced by *digitalObjectLocation*.
Cardinality	0+ (opt and repeatable)
Format	PID
Example	21.T11148/402e89cd01298ae23b61

According to PROV-DM, "alternate entities present aspects of the same thing. These aspects may be the same or different, and the alternate entities may or may not overlap in time." [18] Examples could be images of the same probe with different modalities.

provenanceGraph

Semantic	A PID pointing to another FAIR DO which refers to the provenance graph of the object referenced by *digitalObjectLocation*
Cardinality	0/1 (optional)
Format	PID
Example	21.T11148/123eb0a89ac32563eb12

The purpose of *provenanceGraph* is similar to *hasMetadata* with the difference, that this property is dedicated to provenance. It is imaginable, that a PID which appears as *hasMetadata* value is also used as value of the *provenanceGraph* property to offer direct access to provenance information. There is also an overlap between *provenanceGraph* and all other provenance related properties, e.g., *wasGeneratedBy* or *wasDerivedFrom*. The reason for adding this property additionally was to allow providing a pointer to the entire provenance chain of a FAIR DO, whereas the other properties are only related to direct neighbours. This allows direct access to all retrospective as well as to prospective provenance of FAIR DOs offering a *provenanceGraph*.

Delimitation and Extension

The properties of the Helmholtz KIP presented in the previous chapter are meant as guidance and recommendation towards a basic, high-level harmonization of Helmholtz' (meta-)data products in the form of FAIR DOs. The final outcome fulfils two main guiding principles we were focussing at: On the one hand, we maintained full compatibility to the internationally endorsed RDA recommendation on PID Kernel Information in order to ensure interoperability on an international level. This was done by including all properties without modification and by committing ourselves to the agreed principles of defining Kernel Information properties. On the other hand, we introduced additional properties, most of them are optional, in order to satisfy Helmholtz-specific needs and to close identified gaps towards a general applicability in the Helmholtz Association.

At this point we also have to stress, that the Helmholtz KIP is not meant to be a one fits all solution. The additional properties are supposed to serve as a basis for further contextualization of FAIR DOs and for a high-level harmonization, but they are not introduced to satisfy a specific use case. Instead, all data experts within the Helmholtz Association are invited to derive additional KIPs from the Helmholtz KIP by adding use-case focussed properties and registering the resulting profile in a Data Type Registry, e.g., the one hosted at GWDG. However, when creating new profiles, one should always keep in mind to follow the main principles presented in chapter PID Kernel Information Profiles and to adopt all properties of the Helmholtz KIP without weakening their cardinality or changing their format.

4 Example

In this chapter we would like to present some examples on how to apply the Helmholtz KIP to real-world data sets of different kinds. The first example, the Iris data set, shows the common use case of publishing a simple data set. This use case is extended by explaining how a revised version of this data set can be published and also how metadata about the data set can be added as self-contained FDO. In the second example the type of the published data is a software project. The third example shows the publication of a whole PV system as an FDO, also fragment identifiers are shortly introduced to reference specific data objects behind FDOs. The content of the presented graphics is what will be part of an according PID Record. For reasons of readability we are using human-readable type names. In a PID record these would be PIDs of the according data types.

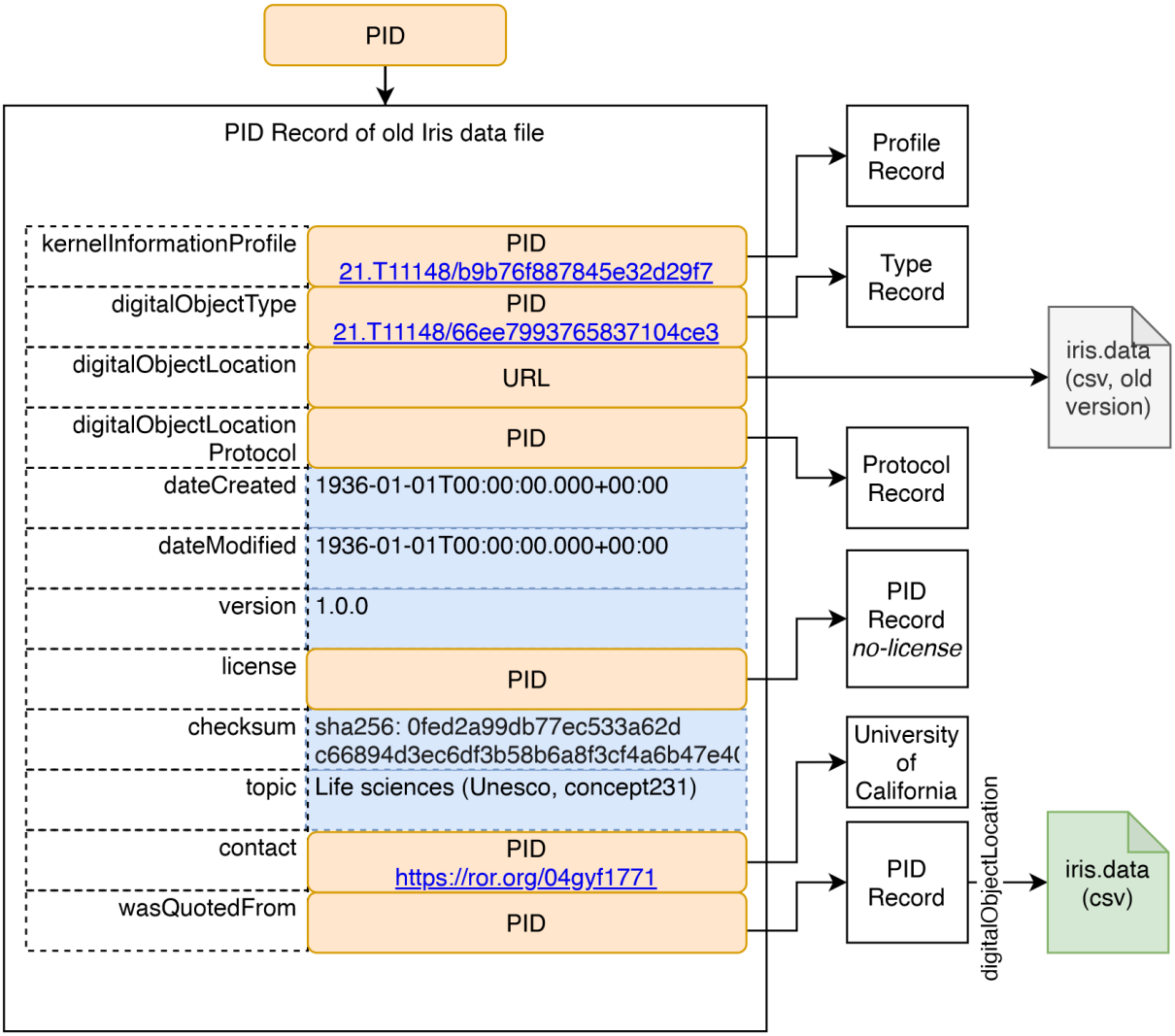
Iris Data Set

This famous data set is mostly used for machine learning and pattern recognition. It contains three classes, each with 50 instances of a type of iris plant. There are many versions of this data set available all over the internet. We selected the **version hosted at University of California** [19] to apply it to our KIP. The special thing about this data set is, that there are two versions available: the original one and an updated version in which some errors were fixed.

Together with one FAIR DO referring to the data set's metadata we ended up with three FAIR DOs, whose values are presented in the following figures.

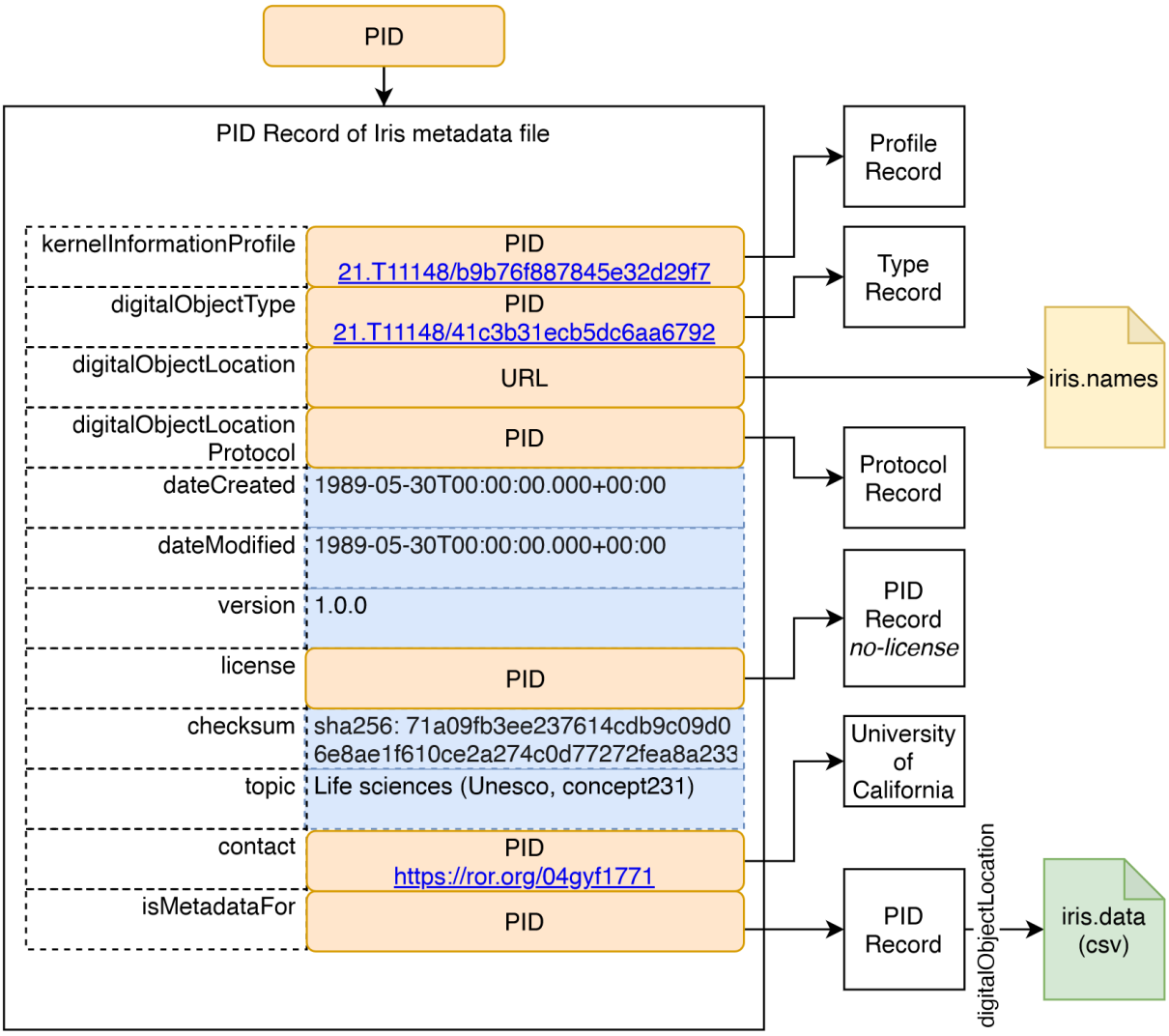
Iris Data - Original Version

The PID record to the original version describes the data object and its relation to the newer version, in this case via "wasQuotedFrom".



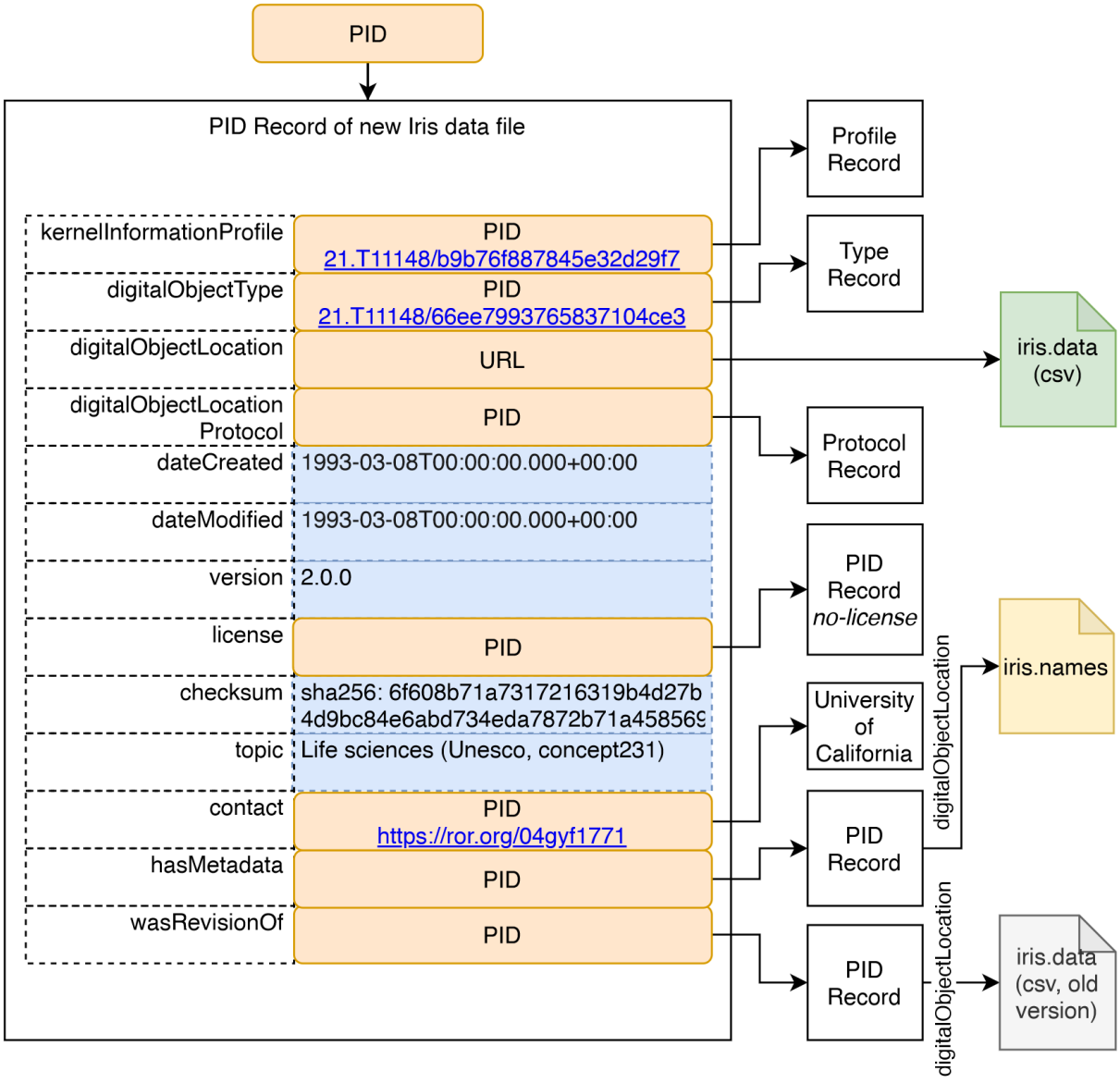
Iris Data - Metadata

When reconstructing the provenance of the iris dataset, we concluded that this metadata document likely belongs to the new version. It therefore refers to the new version via “isMetadataFor”. The metadata file in this case is a simple text document with various kinds of information about the data, similar to a readme file. It is not machine readable. The “digitalObjectType” should reflect this matter. Thus, by using this information a machine will not come to wrong conclusions and can notify the user about this fact, and e.g., ask for human intervention.



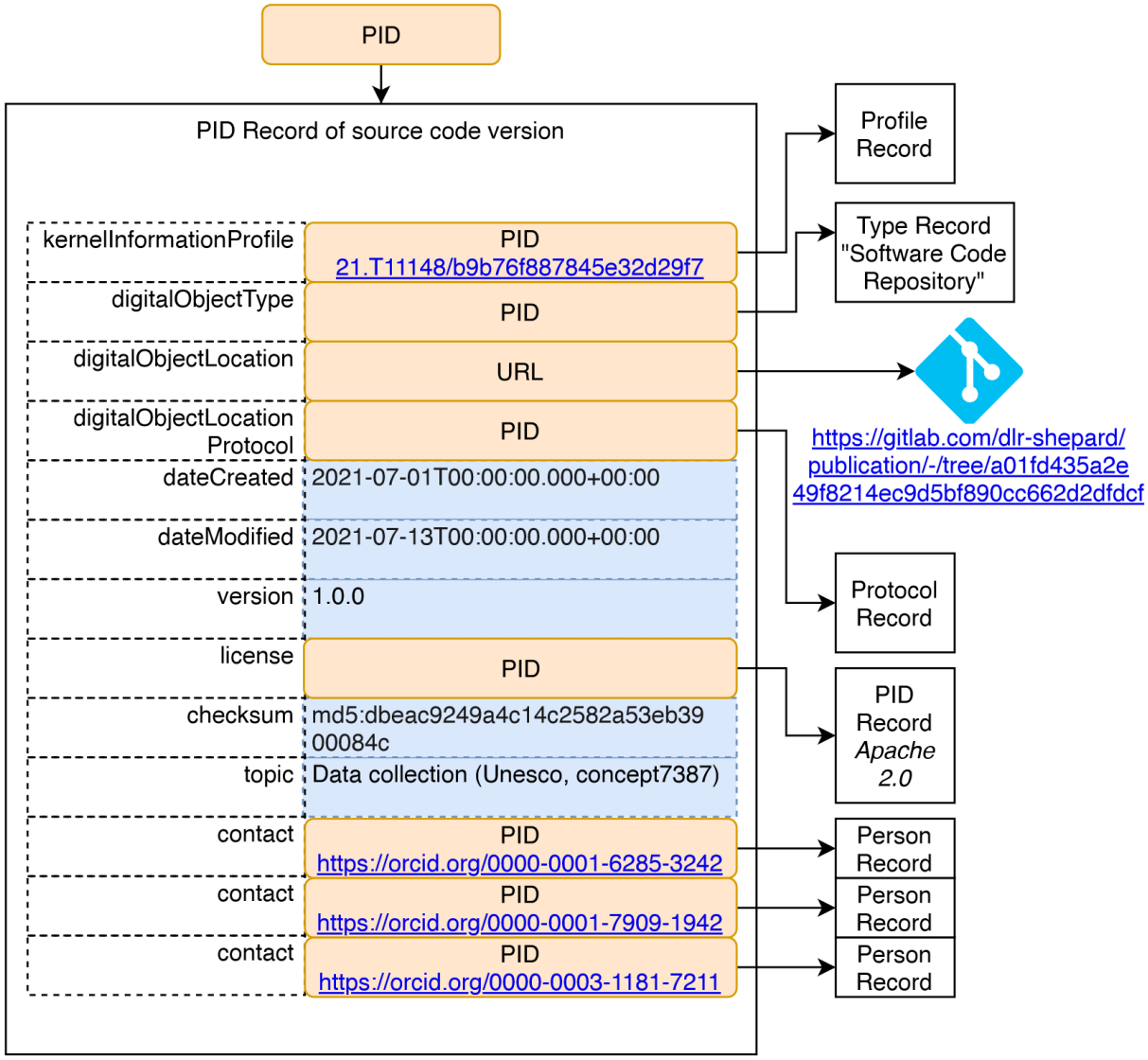
Iris Data - Revised Version

This describes the newest available version of the dataset. If refers to the old version via “wasRevisionOf‘, to its metadata document via “hasMetadata” and describes itself in a similar way as the old version. The version number is increased in relation to the old version. Note that the simplified PID records that belong to the metadata and the old data are the figures shown before. The result is a PID graph describing the relations between those three files as well as the files themselves.



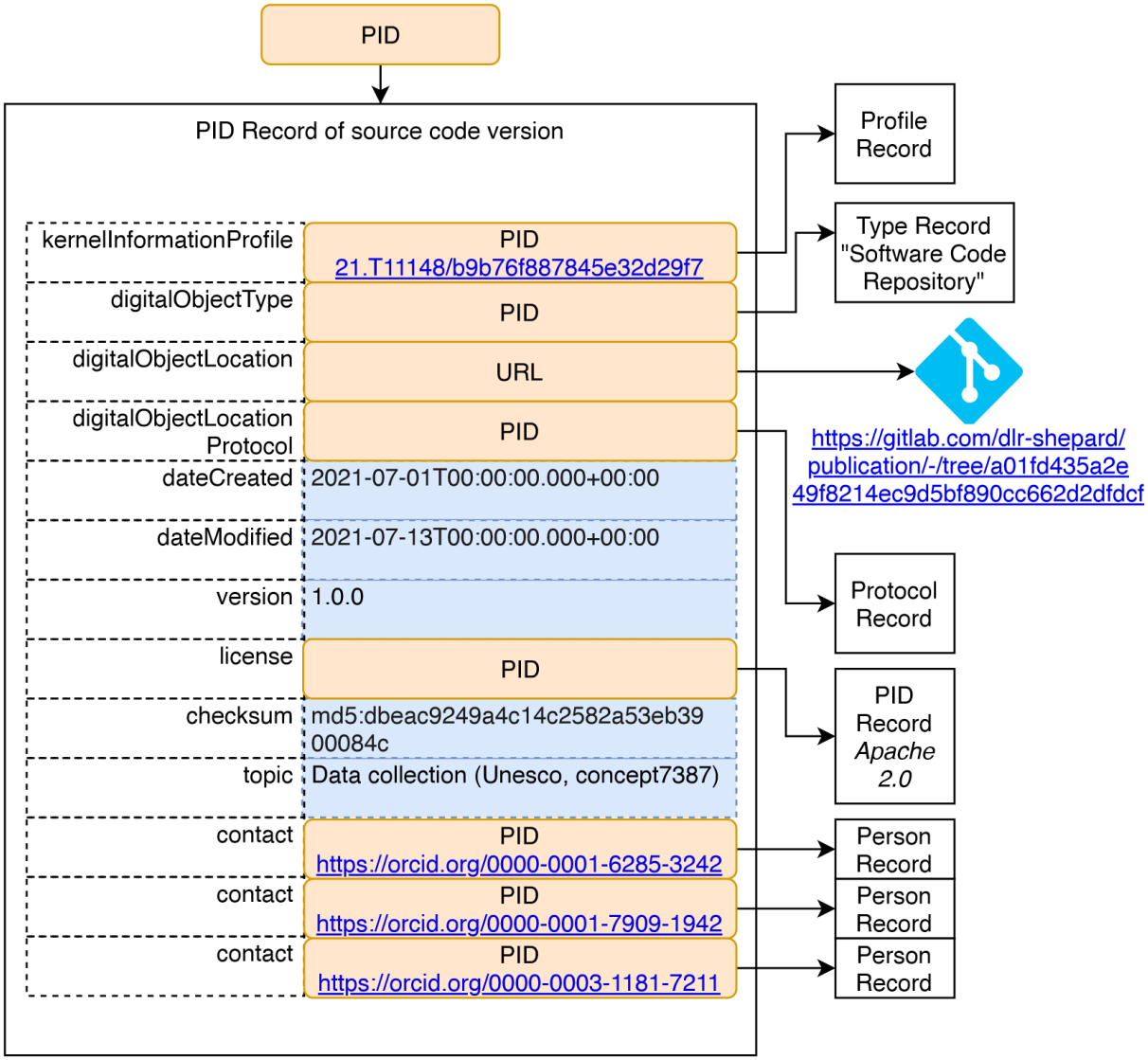
Software Publication

This example applies the Helmholtz KIP to a recent modern software publication. The example is also referenced in **Zenodo** [20] for comparison purposes.



Photovoltaic System

This example shows the PID record of a photovoltaic system (PV system). The record defines the existence of the PV system and allows to reference the whole system through its PID. The "digitalObjectLocation" is using a fragment identifier. The '@' character is a delimiter and divides the location in two parts. The leading part is pointing towards another PID record, the trailing part (everything behind the '@') is a fragment and identifies an entity in the data of the referenced PID record. In this case the PV system FDO points with the leading part of its location to another FDO containing the structural description of the PV system as RDF graph. The fragment part is referencing a specific node of the RDF graph, in this case the top-level node PV system, i.e., the PV system FDO references itself in another representation. Resolving the "hasMetadata" field of the FDO record leads to a collection with further information about the system e.g., master data and measurement data of the different entities of the PV system.



5 Outlook

In this document we presented an initial version of a Helmholtz KIP which can be applied to all FAIR DOs created in the context of the Helmholtz Association as well as to existing data products in order to make them visible as FAIR DOs within the Helmholtz Association and to increase their FAIRness. However, this KIP can be just a starting point offering a limited level of high-level harmonization. In the future, it should form the basis for many other, Hub- or use case-specific KIPs emerging in the Helmholtz Association and beyond. With an increasing number of applied KIPs there will also be the need of a governance model and a curation process, which both are already under discussion with HMC participation. Other aspects that are not yet included in the proposed KIP are the two properties *policy* and *signature.* According to the RDA recommendations, the policy property is supposed to be a pointer to a policy object. As there is currently no system available holding such policy objects in a standardized way, CCT4 decided to extract the license property to the current version of the KIP and postponed adding the other recommended properties, i.e., *objectLifeCycleType* and *objectTombstoneInformation*, after further discussions to a future version of this document. The same applies to the signature property. Some PID resolvers, e.g., the Handle.net Registry, already support signatures for PID records out of the box, but implementing this feature in scientific workflows requires some more information and defined policies. These will be worked out in the future and will then also be included in this document. However, for the time being we see the main task of HMC in the implementation of the Helmholtz KIP in order to gather experience, to identify possible gaps and to derive more specific profiles to fulfill certain needs in close cooperation with AP2, Hubs and domain scientists. This gives us the opportunity to identify additional commonalities which can be used to reveal potential for cross-community collaboration.

Abbreviations

CCT	Cross-Cutting Topic working group
HMC	Helmholtz Metadata Collaboration
FAIR DO	FAIR Digital Object
KIP	Kernel Information Profile
PID	Persistent Identifier
PIT	PID Information Type
RDA	Research Data Alliance
WG	Working Group

References

[1] PID Kernel Information Profile Management WG, Research Data Alliance. https://www.rd-alliance.org/groups/pid-kernel-information-wg

[2] Weigel, T., Plale, B., Parsons, M., Zhou, G., Luo, Y., Schwardmann, U., Quick, R., Hellström, M., Kurakawa, K. (2018). RDA Recommendation on PID Kernel Information (Version 1). doi:10.15497/rda00031.

[3] Eintrag recommendedKernelInformationProfile, ePIC Data Type Registry, Test instance. http://dtr-test.pidconsortium.eu/#objects/21.T11148/0c5636e4d82b88f86132.

[4] Digital Object Identifier (doi). www.doi.org

[5] ORCID. www.orcid.org

[6] Handle.Net Registry. www.handle.net

[7] Gesellschaft für wissenschaftliche Datenverarbeitung mbH Göttingen. www.gwdg.de

[8] Handle resolver, Handle.Net. https://hdl.handle.net/

[9] ePIC Data Type Registry, Testing instance. https://dtr-test.pidconsortium.net/

[10] ePIC Data Type Registry. https://dtr-pit.pidconsortium.net/

[11] Weigel, Tobias, DiLauro, Timothy, & Zastrow, Thomas. (2015). PID Information Types WG final deliverable. doi:10.15497/FDAA09D5-5ED0-403D-B97A-2675E1EBE786

[12] Typed PID Maker, GitHub. https://github.com/kit-data-manager/pit-service

[13] FAIR Data Commons: Technologies and Processes, Helmholtz Metadaten Collaboration: https://helmholtz-metadaten.de/en/fair-data-commons/overview

[14] ComputationalWorkflow Profile, 1.0-RELEASE (09 March 2021), Bioschemas. https://bioschemas.org/profiles/ComputationalWorkflow/1.0-RELEASE

[15] Creative Commons Rights Expression Language (CC REL), Creative Commons Wiki (last edited: 15 September 2021, 16:18). https://wiki.creativecommons.org/wiki/CC_REL

[16] FAIRsharing Subject Ontology (SRAO), Ontology Lookup Service https://www.ebi.ac.uk/ols/ontologies/srao

[17] UNESCO Thesaurus (last modified: 09 September 2022, 10:29) http://vocabularies.unesco.org/browser/thesaurus/en/

[18] PROV-DM: The PROV Data Model, W3C Recommendation 30 April 2013. https://www.w3.org/TR/prov-dm/

[19] Iris Data Set. in: Dua, D. and Graff, C. (2019). UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science. https://archive.ics.uci.edu/ml/datasets/Iris

[20] Haase, T., Dr. Glück, R., Kaufmann, P., & Willmeroth, M. (2021). shepard - storage for heterogeneous product and research data (1.0.0), Zenodo, doi:10.5281/zenodo.5091604