RDA Recommendation on PID Kernel Information FINAL

Authors: Tobias Weigel, Beth Plale, Mark Parsons, Gabriel Zhou, Yu Luo, Ulrich Schwardmann, Robert Quick, Margareta Hellström, Kei Kurakawa

Version: this version reflects comments from community review period and RDA Council feedback. Last update Nov 19, 2019.

Abstract

Global middleware infrastructure is insufficient for robust data identification, discovery, and use. While infrastructure is emerging within sub-ecosystems such as the DOI ecosystem of services purposed for data and literature objects (i.e., DataCite, CHORUS, CrossRef), in general the layers of abstraction that have made the Internet so easy to build on, is lacking for data especially for computer (machine) automated services. The goal of the PID Kernel Information recommendation is to advance a small change to middleware infrastructure by injecting a tiny amount of carefully selected metadata into a Persistent ID (PID) record. This carefully chosen and placed information has the potential to stimulate development of an entire ecosystem of third party services that can process the anticipated billions of PIDs and do so with more information at hand about a digital object (no need for costly link following) than just its globally persistent ID (PID).

The key challenge of the PID Kernel Information working group was to determine which from amongst thousands of relevant metadata elements are suitable to embed in the PID record. This recommendation lays out principles to guide in the identification of information suitable for inclusion in the PID record.

The information contained in a PID record is represented by a PID Kernel Information profile which must be publicly and globally available. For PID Kernel Information to be effective in stimulating an ecosystem of data services, the number of different profiles of PID Kernel Information must be small and their content stable. The recommendation includes a draft profile with illustrating examples and cases for adoption in practice.

1. Background and Scope

The primary purpose of the guiding principles is to help profile developers determine which information should (and should not) be included in a PID Kernel Information profile.

What is a PID Kernel Information profile?

PID Kernel Information is a set of attributes stored within the PID record, *i.e.,* information stored at a global or local PID registry and accessible by a resolver. The primary purpose of PID Kernel Information is in support of smart machine actionable decisions that can be accomplished through inspection of the PID record alone. PID Kernel Information profiles are registered schemas for interpreting PID KI records. PID records may be created according to specific profiles and checked for conformance against them. In other words, PID KI records are concrete instantiations of profiles, comparable to how we consider objects as instantiations of classes in object-oriented programming.

Scope and applicability

The principles apply to profiles for PIDs that reference (point to) data objects which have a single or canonical digital manifestation. The object itself can be digital data, code, metadata or the digital representation of a physical object, etc.

These principles are geared toward PID systems with the following attributes: 1) they register, store, and retrieve a small amount of metadata, and 2) there is a globally discoverable service available through which information about the PID Kernel Information profiles can be retrieved, which are referenced in the PID records. Current systems that potentially meet these requirements include the Archive Resource Key (ARK) and the Handle service (and Data Type Registry) and conceptually any URN-based service with a managed global resolver. We expect other systems do as well. The WG used the Handle service as a model or test case in the development of these principles and seeks feedback on how they apply to similar systems.

The WG also builds on the existing RDA Recommendation for a Data Type Registry[1] as a globally available (and distributed) type registry for both data types, such as data format definitions, and PID Kernel Information profiles. Whether the same registry should take both roles at the same time needs further exploration.

[1] **DOI**: dx.doi.org/10.15497/A5BCD108-ECC4-41BE-91A7-20112FF77458

Community development

This document is relevant to the multiple stakeholders who are interested in a global ecosystem of FAIR Digital Objects. The WG envisions global convergence around a relatively small number of PID Kernel Information profiles, which will require ongoing discussion, consensus, and evaluation. For example, there could be an “Internet of Things” profile for physical devices and another profile for the data objects and streams that the devices produce. Should these “IoT” data objects have a different profile than research data cited in the scholarly literature? This will require ongoing community discussion, but the principles herein can guide the discussion.

The WG behind this recommendation recognizes the importance of additional issues that were out of scope of this RDA working group (i.e., where multiple profiles may be applicable to the same object and profile management on a global scale) but which may result in additional guiding principles as the concept matures.

2. PID Kernel Information Guiding Principles

PID KI records are instances of PID Kernel Information profiles stored at a local or global registry. Where PID Kernel Information profiles consist of *attributes*, PID KI records consist of attributes and their relevant *values*. A core assumption behind the guiding principles is that a PID KI record is primarily meant to serve automation needs so the record is small and there are relatively few profiles in existence.

The guiding principles in defining a PID KI profile are as follows:

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

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.

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

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

Principle 5: PID KI record values should *change infrequently with update initiated only by an appropriate authority, avoiding human interaction on updates where possible*.

Principle 6: Attributes (items) in the profile are expressed as key-value pairs where the values are simple (indivisible).

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.

2.1. Examples and further information

This section is illustrative; it explains the guiding principles, provides examples, and when possible illustrates appropriate creation and use of PID Kernel Information profiles.

Principle 1

Principles 1 and 3 taken together define the primary intended purpose of PID Kernel Information. PID Kernel Information is intended as a small amount of cached information (metadata) stored at the resolving services, for instance, the PID KI is stored with the Handle record at a Local Handle Service. An ID resolving service is part of the global infrastructure for *machine actionable services*. For example, suppose an internet scale service processes a continuous stream of PIDs, attempting to single out research data. The service cannot afford to resolve links to a repository for every PID record. The service will consult, for instance, a type registry to obtain the profile type record to interpret the PID KI. The type registry may also contain further definitions for types which are not profiles, e.g. data formats or scientific units, which may also be referenced in PID KI records, but can also be cached.

Principle 2

Principle 2 resulted from considerable discussion in the WG who eventually determined that the PID KI is a *non-authoritative source of the information* (with an exception below). There are several reasons for this. First, the data management approach typical for PID record metadata is usually not designed for metadata management operations necessary to keep their quality at an acceptable level. Second, facilities such as searching and schema management are not at the same level of complexity at a PID resolver. Finally, resolution reliability and performance take precedence over level of detail and complexity of PID record information, and the underlying systems and processes are designed accordingly.

One exception to principle 2 are the attributes which are inherently related to the mechanics of persistent identification and Kernel Information profiles. These are the PIDs of an object, its locations and a reference to a profile the specific record conforms to. For these, the PID record will be the authoritative source.

Principle 3

As in principle 1, PID Kernel Information is targeted to internet scale services. Even at fast network speeds, distance still matters. So for the most efficient network scale decision making to occur, the PID Kernel Information is *stored and managed at a PID resolver* itself.

Principle 4

Principle 4 addresses the issue of *change ownership*, that is, controlled update of a PID KI record. Specifically, a concrete PID KI record (instance) can only be updated by the **data object owner** or their **owner delegate**. PID KI is cached metadata, but metadata nonetheless, so the issue of who can make changes to it has to follow rules. Whether the authoritative metadata is part of a digital object or maintained in a separate digital object is not relevant here.

Controlled record change rules have implications on the content of a PID KI profile. To illustrate, suppose a third party acts to derive a data object from an existing data object not under their control. Tracking this derivation in the PID record of the original data object cannot be done by the third party because they lack permission to edit the PID record of the data object from which they are deriving. Tracking derivation, however, is still an integral part of capturing provenance. To enable this even while original records cannot be changed by third parties, other compliant implementations exist, such as contacting the original owner, preferably via an API endpoint.

Principle 5

Principle 5 addresses the complex issue of *change frequency* to a PID KI record. The PID KI record contents should have their update initiated only by an appropriate authority. The appropriate authority is the data object owner (see principle 4) or owner delegate. The two roles differ in the amount of detailed knowledge and authoritative power that each has over digital objects and authoritative metadata. Owner delegates may not be able to affect the preservation of objects or be responsible for doing so, and they may have insufficient knowledge of the meaning of objects or metadata, thus rendering them unable to execute a mapping to a concrete PID KI record. Delegates may however still be responsible for keeping the record intact (available and readable).

PID KI record values should change infrequently. By principle 2, PID KI record is a non-authoritative source of metadata. The metadata defined in the PID KI is determined by the PID KI profile; a profile represents a *class* of data/metadata objects that use the profile. The role of profile definer is to define PID KI profile by identifying which authoritative metadata are included in PID KI record entries. A profile definer represents a large community potentially the size of a discipline. The roles and their relationships are illustrated in Figure 1.

Taking these roles together, change frequency must be heavily considered by the profile definer during profile definition. The profile definer must also take into account record authority. The owner delegate (PID record manager) who has to make changes to the PID record cannot control the rate of change to the authoritative metadata, but must only try to reflect that change in the PID KI record. It is the responsibility of the profile definer at profile definition time to carefully consider change minimization in the selection and mapping of PID Kernel Information, and avoid introducing human interaction on updates. This is a critical aspect to consider at profile definition time.

In addition, though not a guiding principle, it is recommended that a PID KI profile itself, once defined, should undergo revision rather than be changed. Versioned profiles can be linked. An old version of a profile needs to remain accessible, as PID records may still exist, possibly without knowledge of the profile definer, which were created based on this profile. To summarize, the attributes in a profile should not change, the values in the PID KI records conforming to a profile can change (though slowly). As indicated in principle 5, this change should however be infrequent. The task of the profile definer is to preclude such foreseeable problems with changes at time of definition of the profile, which ultimately means to exclude such attributes from the profile and leave them in separate metadata only.

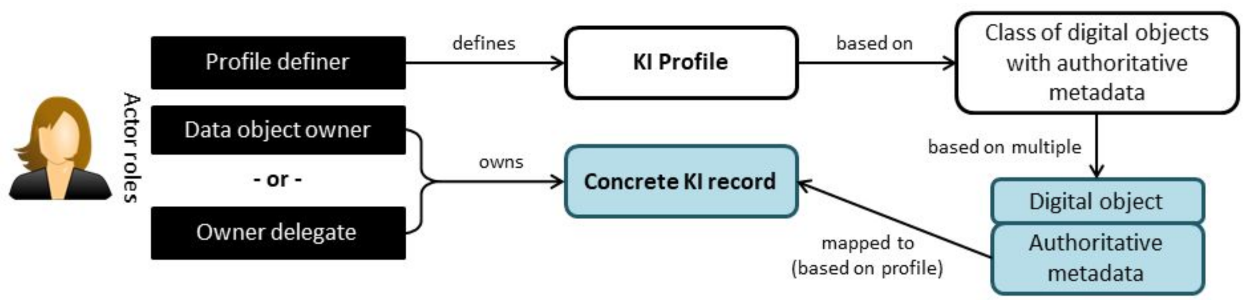

Figure 1. The roles in PID KI record management and change and their relationship.

Principle 6

Principle 6 addresses attribute makeup, in particular that an *individual attribute should be indivisible*. Complex attribute values increase the time a machine must spend parsing them, which is undesirable for the fast processing that the Kernel Information is designed for. Complex attribute values such as lists tend undergo higher frequency of change than indivisible attributes which raises the risk of error introduction under profile revision. Complex attributes may need heavy refactoring, up to the point where an attribute needs to be split into two or more attributes. Thus, it is a guiding principle to keep attribute values indivisible. A full list as an attribute may reasonably kept in a primary metadata storage system.

Principle 7

Principle 7 *constrains the subject of an attribute*. The subject of an attribute should be the digital object that the PID KI references. To resolve possible violations of principle 7 during profile design, it is recommended that the attributes in violation be moved to separate records, for instance maintained in a type registry.

The following examples violate principle 7. In addition to listing the attributes and value types for each attribute, they also give an example value as it could appear in an exemplary PID record; this is for further explanation.

Example PID KI profile snippet 1:

123xyz/file-xyz:

Attribute	Value type	Example value
LOCATION	URL	http://www.example.com/file-xyz
CREATED	DATE	2018-01-01
PART_OF_DATASET	PID	20.1000/100/dataset001
DATASET_CREATED	DATE	2018-01-31

This example shows the digital object pointed to by the PID KI record (see LOCATION), which is a single data file. This file is part of a dataset, referenced in PART_OF_DATASET. The timestamp for when the respective dataset was created is included in the profile, however this is a violation of principle 7(a): the attribute does not depend on the identified object (the file), but on a different object (the dataset).

A solution for example profile 1 is to remove DATASET_CREATED:

123xyz/file-xyz:

Attribute	Value type	Example value
LOCATION	URL	http://www.example.com/file-xyz
CREATED	DATE	2018-01-01
PART_OF_DATASET	PID	20.1000/100/dataset001

20.1000/100/dataset001:

Attribute	Value type	Example value
LOCATION	URL	http://www.example.com/dataset001
CREATED	DATE	2018-01-31

The solution is to create a second profile for datasets, with location and DATASET_CREATED attributes, and refer to records formed according to this profile in the PART_OF_DATASET entry. The DATASET_CREATED entry is then removed from the digital object profile for the single data file.

Example PID KI profile snippet 2:

123xyz/dataset002:

Attribute	Value type	Example value
LOCATION	URL	http://www.example.com/dataset002
DATA_FORMAT	STRING	netcdf
DATA_FORMAT_VERSION	INTEGER	4

In this example, DATA_FORMAT_VERSION is a refinement on another attribute, DATA_FORMAT. As such, it violates principle 7(b) and does not belong in this profile, but rather into a separate profile that further describes data formats, with multiple versions (4, 5, ...) of the "netcdf" format forming individual records. An even better solution would be to record the data format descriptions in a DTR, with separate entries for each netcdf version. This is described in the following solution; note that the second table shows a type registry entry, not a kernel information profile, and that the given values are not examples, but actual values for the concrete type record.

A solution for example profile 2:

123xyz/dataset002:

Attribute	Value type	Example value
LOCATION	URL	http://www.example.com/dataset002
DATA_FORMAT	TYPEDEF	typedef123/netcdf4

typedef123/netcdf4 (type registry entry):

Attribute	Value type	Value
DATA_FORMAT_FAMILY	TYPEDEF	typedef123/netcdf
DATA_FORMAT_VERSION	INTEGER	4

There should be another type definition entry "typedef123/netcdf" to describe the netcdf family or class of data formats, as well as other entries in these type definitions such as description, label etc.

3. Draft Kernel Information profile

The following is the schema for a recommended Kernel Information profile, describing which attributes must or may be included in a conforming Kernel Information record. The cardinality in the profile indicates whether attributes are mandatory or optional and whether they can be specified multiple times. This schema should be stored in a Kernel Information profile registry.

	Property identifier	Content format	Cardinality	Explanation
1	PID	Handle	1..n	Global identifier for the object; external to the PID Kernel Information
2	KernelInformationProfile	Handle	1	Handle to the Kernel Information type profile; serves as pointer to profile in DTR. Address of DTR federation expected to be global (common) knowledge.
3	digitalObjectType	Handle	1	Handle points to type definition in DTR for this type of object. Distinguishing metadata from data objects is a client decision within a particular usage context, which may to some extent rely on the digitalObjectType value provided.
4	digitalObjectLocation	URL	1..n	Pointer to the content object location (pointer to the DO). This may be in addition to a pointer to a human-readable landing page for the object.
5	digitalObjectPolicy	Handle	1	Pointer to a policy object which documents changes to the object or its Kernel Information record, including particularly object access and modification policies. 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.
6	etag	Hex string	1	Checksum of object contents. Checksum format determined via attribute type referenced in a Kernel Information record.
7	dateModified	ISO 8601 Date	0..1	Last date/time of object modification. Mandatory if applicable.
8	dateCreated	ISO 8601 Date	1	Date/time of object creation
9	version	String	0..1	If tracked, a version for the object, which must follow a total order. Mandatory for all objects with at least one predecessor version.
10	wasDerivedFrom	Handle	0..n	PROV-DM: 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.

11	specializationOf	Handle	0..n	PROV-DM: 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.
12	wasRevisionOf	Handle	0..n	PROV-DM: A derivation for which the resulting entity is a revised version of some original.
13	hadPrimarySource	Handle	0..n	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.
14	wasQuotedFrom	Handle	0..n	PROV-DM: 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.
15	alternateOf	Handle	0..n	PROV-DM: 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.

Explanatory notes:

- Even if a field is not mandatory, it is recommended to use it if corresponding information is to be added to a KI record. This will counteract evolution of competing fields across usage scenarios.
- The properties 10-15 are as defined in W3C PROV-DM[2] data model. The explanations given in this table are only brief summaries from the PROV-DM model. For more detailed explanations, please refer to the PROV-DM descriptions, which are binding for the meaning of the according properties.
- Relationships related to partitioning or constituency are not contained in the profile as they should be dynamically exposed following for instance the RDA Research Data Collections recommendation.
- Following the explanation for principle 2, the PID Kernel Information is the authoritative source for the following properties only: PID, KernelInformationProfile and digitalObjectLocation.

Digital Object Policy record structure

The "digitalObjectPolicy" attribute of the profile deserves additional explanation. This attribute points to a more detailed policy object. This policy object should inform any agent interacting with a digital object about the expected future changes to this object. These changes are the result of processes the object owner has agreed to. The processes should ideally be formally defined and available to any calling agent as well. However, the description of these processes goes far beyond the scope of Kernel Information; nonetheless, the resulting changes to an object and the interest of the caller are still within scope. Thus, the following is a suggested structure for such policy objects. Policy objects could exist in, for instance, a type registry as

[2] https://www.w3.org/TR/prov-dm/

they are usually the same for many objects and there should be a common interface to access them. An example policy document is given in Table 1 and explained in more detail below.

	Attribute	Type	Mandatory?	Possible values
1	objectLifeCycleType	String	Yes	static, dynamic_irregular, dynamic_regular
2	objectTombstoneInformation	String	No	(any, see below)
3	objectLicense	Handle or URL	No	(any, see below)

Table 1. Example digital object policy

1. **Object life cycle type.** Possible values are (exclusively):
    a. **Static:** Once the object has received an identifier, no future changes are expected. If a new revision of the object is generated in the future, it will become an independent object, but the link between these objects may be expressed with Kernel Information attributes ("revisionOf" and "wasDerivedFrom").
    b. **Dynamic (irregular):** The object has received an identifier and future changes are possible, but it is not known when or if they might happen. An example for this is data or source code, in particular, which may undergo versioning. It is a priori unknown whether or when a new version will be generated, but if it happens, it will replace the objects' contents.
    c. **Dynamic (regular):** The object has received an identifier and future changes are expected in regular intervals or according to a known plan, making them a standard scenario. A typical example for this are time series.
    In both the regular and irregular dynamic cases, the object can be marked as static in the future, for example, if the time series ends and no further elements are expected. Kernel Information attributes may change as part of object changes, such as "version", "etag" and "lastModified".
2. **Object tombstone information.** This is an optional attribute. It should be set if and only if the object's content are gone. The description in the attribute should specify the reason for this. Possible reasons include that the object was removed intentionally due to long-term storage policies or processes or due to legal reasons, or removed accidentally.
3. **Object license.** This should be of type PID or URL, pointing to a stable identifier for the object's license.

4. Exemplary high-level architecture

In general, the interplay of services relevant for interpretation of PID Kernel Information is as follows:

- **Persistent Identifier systems** store identifier metadata, part of which are Kernel Information records. Kernel Information records contain a pointer to a registered Kernel Information profile in a profile registry and refer to individual attribute definitions in type registries.
- **Profile registries** store the definition of profiles (schemas). The attribute and type definitions used in a registered profile are stored in a type registry. Profiles bear PIDs.
- **Type registries** store type definitions, for which attribute definitions and type definitions are particular examples. Types bear PIDs.

Concrete PID Kernel Information records will refer to both the profile they adhere to and the individual attributes definitions they use from it. While conceptually distinct, the role of profile and type registries may be combined into a single registry system.

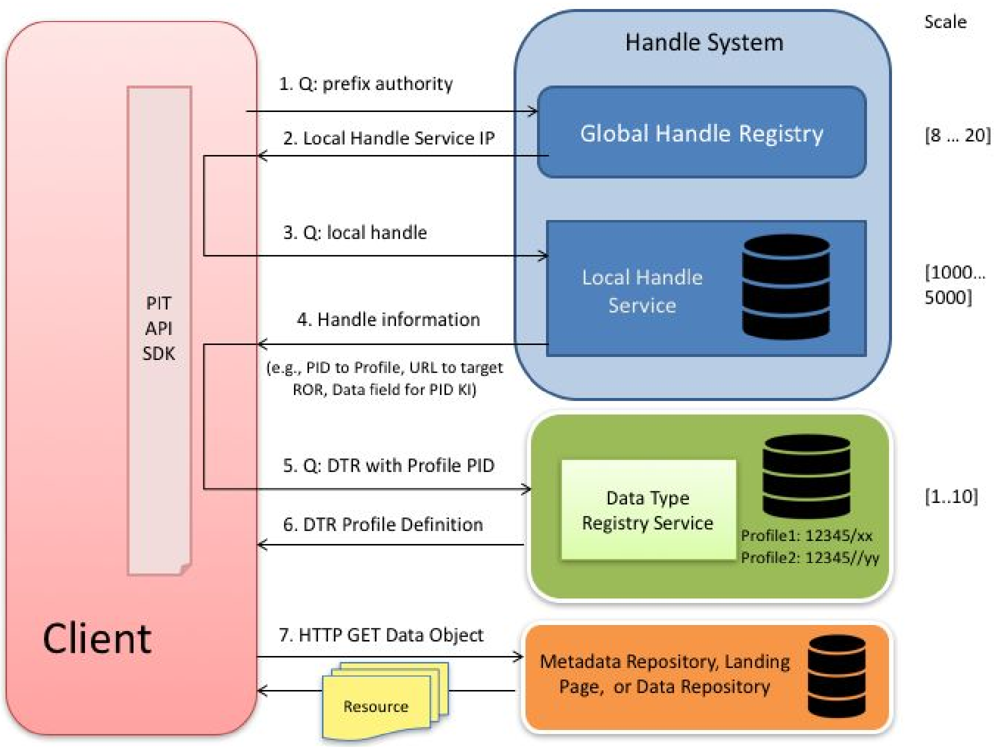

Figure 2. Resolving a PID to a digital object through a series of interactions with digital object services.

The guiding principles are designed architecture agnostic. However, to implement workflows that use Kernel Information, concrete instances of the systems and registries must be used. Thus, we become more specific in this section and explain, as an example, one possible architecture based on the Digital Object Architecture and the Handle System. The solution under discussion in the Data Type Registry Working Group is an exemplary model for a combined profile and type registry, possibly federated.

Based on these exemplary systems, the basic interaction process between a client and multiple registry servers is explained in the diagram below. Of course, other implementation approaches are possible and we very much welcome further implementations also based on different PID systems.

5. Use cases and community usage stories

Several stakeholder groups have evaluated the recommendation for application within their specific domains. The following cases are explained in this section or in the reference footnotes:

- ENES/ESGF data infrastructure (Earth System Modelling)[3]
- EUDAT B2HANDLE (Generic e-infrastructure service)[4]
- Deep Carbon Observatory
- Sustainable Environments Actionable Data Training project (SEAD-Train)[5]
- PRAGMA Rice Genome Project[6]

Adoption by the ENES/ESGF data infrastructure

The ENES community (European Network for Earth System modelling) has implemented data infrastructure services to assign Handles to files and datasets managed in the global Earth System Grid Federation (ESGF) for the CMIP6 series of experiments. As part of this, essential information about files and datasets are already being written into Handle records by automated procedures as part of data distribution in the data federation. These procedures also include cases of object versioning and retraction, which affect the information stored in Handle records.

The record schema used by the ENES community was generated based on the practical needs of the data infrastructure developers and the services put in place for the users. As such, the attributes in the schema do not necessarily form a coherent system as might be expected when designing a schema according to the Kernel Information guiding principles and the recommended attributes of the profile. The system that is in place for the CMIP6 series of experiments was created during the early phases of this WG when the recommendation was not fully formulated yet and put in operations soon after, making changes to the running system difficult to achieve. For CMIP7, however, the ENES community will reevaluate and potentially

[3] Balaji et al.: Requirements for a global data infrastructure in support of CMIP6. Geoscientific Model Development (2018). doi:10.5194/gmd-11-3659-2018
[4] Weigel, Schwardmann, Klump, Bendoukha, Quick: Making Data and Workflows Findable for Machines. Data Intelligence (2020). doi:10.1162/dint_a_00026
[5] Plale and Kouper, SEADTrain Data Analysis, https://scholarworks.iu.edu/dspace/handle/2022/22312
[6] Juanillas et al., Rice Galaxy: an open resource for plant science, *GigaScience*, Volume 8, Issue 5, May 2019, giz028, https://doi.org/10.1093/gigascience/giz028

revise the schema along the principles, align existing attributes with the profile and decide on inclusion of new attributes as mandated by it.

Adoption by the EUDAT B2HANDLE service and user community

The EUDAT data infrastructure forms a central building block of the future European Open Science Cloud (EOSC). EUDAT offers multiple services on research data, including B2HANDLE as a PID management service. The B2HANDLE user community has already populated Handle records with various attributes. In 2016 and 2017, the B2HANDLE service developed a first EUDAT PID profile with multiple attributes, based on the most prominent usage scenarios by the user community. This profile was generated based on their needs and predates the discussions in the RDA community and the Kernel Information WG. After defining the profile, the B2HANDLE service also migrated existing records to the profile, which was a major effort that also generated tools and guidelines for other possible migrations for the future.

The EUDAT B2HANDLE service user community sees Kernel Information as critical element across multiple community use cases. As part of EOSC activities, the B2HANDLE service promotes usage of the recommendation - guiding principles, profile and value proposition - to new users. The focus is on integration with other middleware services, in line with the motivation for Kernel Information to be useful for machine actionable services. The evaluation led to a formulation of further requirements pertaining to the automation of workflows leveraging a solution based on PIDs and minimal metadata beyond just B2HANDLE as a service, but generally based on Kernel Information and other RDA recommendations (Data Type Registries, Collections and Data Fabric, in particular).

Adoption by the Deep Carbon Observatory Data Portal

Since 2011, the DCO Data Science Team at the Tetherless World Constellation of Rensselaer Polytechnic Institute has developed and maintained the DCO Data Portal, which provides access to the thousands of people, publications, data, and other resources available in DCO. The Portal makes extensive use of PIDs most notably something we call the DCO-ID. The DCO-ID is a Handle and is similar to the Digital Object Identifier (DOI) for publications, but it extends the scope to many more types of objects, including publications, people, organizations, instruments, datasets, sample collections, keywords, conferences, etc. Each DCO-ID can redirect to the Web profile (often a landing page) of an object, where more metadata can be found. In the DCO Data Portal each object is the instance of a class. The metadata items describing an instance are properties. All those classes and properties are organized by the DCO ontology. In previous work, the team extended the DCO ontology to incorporate the Data Type Registries and PID Types Recommendations from RDA. This also led to an extension to VIVO, the popular research discovery platform.

We are currently working toward toward adopting additional RDA Recommendations including Dynamic Data Citation, DDR, and Scholix in order to further increase the visibility, validity, and accessibility of DCO data. All of these Recommendations center around the use of PIDs. Therefore, the next question is how the PID KI fits into this broader use of PIDs. Will it facilitate the use and adoption of these multiple Recommendations?

The DCO Data Portal team is assessing how the PID KI helps or hinders the broader use of PID-related functionality within the DCO Data Portal.