# A FRAMEWORK FOR EXTENDED PERSISTENT IDENTIFICATION OF SCIENTIFIC ASSETS

***Tobias Weigel[1,2*], Michael Lautenschlager[1,3], Frank Toussaint[1,3], Stephan Kindermann[1]***

*[1]Deutsches Klimarechenzentrum, Bundesstrasse 45a, 20146 Hamburg, Germany*
**Email: weigel@dkrz.de*
*[2]Universität Hamburg, Bundesstrasse 45a, 20146 Hamburg, Germany*
*[3]World Data Center for Climate, Bundesstrasse 45a, 20146 Hamburg, Germany*

## *ABSTRACT*

*Several scientific communities relying on e-science infrastructures are in need of persistent identifiers for data and contextual information. In this article, we present a framework for persistent identification that fundamentally supports context information. It is installed as a number of low-level requirements and abstract data type descriptions, flexible enough to envelope context information while remaining compatible with existing definitions and infrastructures. The abstract data type definitions we draw from the requirements and exemplary use cases can act as an evaluation tool for existing implementations or as a blueprint for future persistent identification infrastructures. A prototypic implementation based on the Handle System is briefly introduced. We also lay the groundwork for establishing a graph of persistent entities that can act as a base layer for more sophisticated information schemas to preserve context information.*

**Keywords:** Data-intensive science, Persistent identifiers, Unique identifiers, Long-term archival, e-Science infrastructures, Scientific data management

## 1 INTRODUCTION

With the advent of data-intensive science (Hey, Tansley, & Tolle, 2009), the number and variety of individual digital entities is rapidly increasing, and scientific activities tend to shift from pure data generation to more sophisticated, often cross-discipline, data analysis and processing of resources. One of the challenges involved concerns dealing with not just the increasing volume of data but also the increasing number of published datasets and their thematic complexity. This puts up challenges from a perspective of both data management and scientific practice. In this article, we will discuss how some of these challenges could be met by consequently using persistent identifiers (PIDs) for the digital entities involved and present a framework to evaluate or build viable solutions.

Data management, as, for example, that done by the World Data Center for Climate (WDCC) (http://www.wdc-climate.de), will increasingly rely on the use of persistent identifiers to cope with the increasing number of data and metadata objects. Persistent identifiers are seen here as a means to maintain a stable data space across migration of storage and database systems when it comes to long-term archival in the time frame of 10 years or longer. Moreover, persistent identifiers can promote sharing and reuse of – often expensively acquired – scientific data, as stated in the report of the European Commission's High Level Expert Group on Scientific Data (2010). As an example from the Earth System modeling community, the Coupled Model Intercomparison Project, phase 5 (CMIP5) (http://cmip-pcmdi.llnl.gov/cmip5), has produced a volume of about 10 PB of data, using computational resources from computing centers around the globe. The experiment output is currently distributed through a federated system of gateway portals and data nodes (the WDCC data node can be found at http://ipcc-ar5.dkrz.de) powered by an open source software framework called the Earth System Grid Federation (ESGF) (http://www.esgf.org). Data management within this federation could benefit from assigning persistent identifiers to the low-level datasets, given the use cases of data replication, curation, and delivery.

Many other use cases also hint at the necessity to not just provide stable identifiers to digital entities but also to augment such identifiers with contextual information. This encompasses the broad variety of metadata found across disciplines, many differing in the used encodings, level of detail, and overall heterogeneity. Yet the unifying factor is that according to proper scientific conduct, this increasingly complex information space must be subjected to long-term preservation. This article will not present a framework for long-term data archival, such as the Open Archival Information System reference model (ISO 14721), but will rather focus on the particular functionality to persistently identify (or, more precisely, persistently locate) already managed entities. Stable, persistent identifiers are one known key ingredient for long-term preservation systems, yet the questions of how to deal with the very heterogeneous context information are unanswered across scientific communities.

This especially manifests in the ongoing discussions within the European FP7 project EUDAT (http://www.eudat.eu). EUDAT aims to establish a collaborative scientific data infrastructure that spans various scientific communities. The interest in persistent identification is high, yet the problem space is complex and largely unanswered given the many communities involved and the resulting heterogeneity and complexity of use cases. Participating communities cover the whole range of sciences and humanities, including, for example, the aforementioned Earth sciences ( European Network for Earth system modeling (ENES): http://www.eudat.eu) (European Plate Observatory System (EPOS): http://www.epos-eu.org), biodiversity (European Plate Observatory System (EPOS): http://www.epos-eu.org), linguistics (Common Language Resources and Technology Infrastructure (CLARIN): http://www.clarin.eu), and human physiology (Virtual Physiological Human: http://www.vph-noe.eu). As diverse as these initiatives and communities may seem, all of them deal with increasing amounts of data and context information, and the need for persistent identifiers is a unifying theme within the frame of EUDAT. Use cases currently under investigation include the safe replication of data, where PIDs may help to maintain links between all replicas, a common metadata catalog, and the staging of data for user-space processing activities.

Besides EUDAT, the emerging Research Data Alliance (Virtual Physiological Human: http://www.vph-noe.eu), currently involving European, US, Australian, and Canadian actors, has taken up issues of data management and persistent identification and will likely feed into, as well as be driven by, the EUDAT problem space among others.

Figure 1 illustrates data management with PIDs and binary links between PIDs for the safe replication use case. An original data object is replicated at three physically separate sites. The original entity and all replicas refer to a common metadata description in a web portal catalog. If a user arranges the transfer of a copy of the data object to her workspace, the copy is linked to the specific replica. The illustrated linking scheme is only one possible alternative. Other solutions include, e.g., to link all replicas together or use a collection object to gather them.

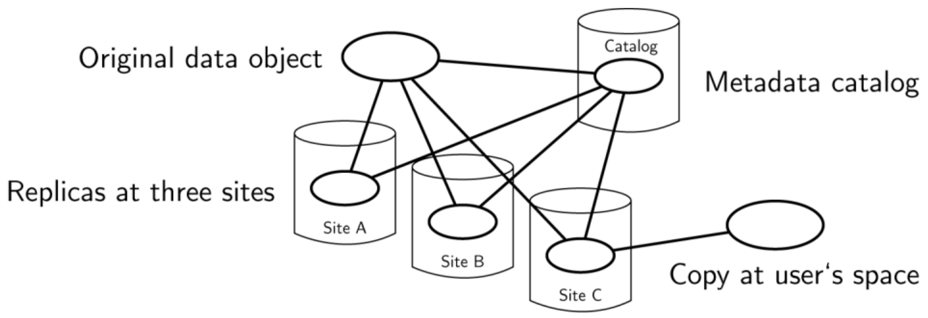

**Figure 1.** A possible linking schema for the safe replication use case

Persistent identifiers have so far largely been understood as some form of globally unique strings that introduce a stable layer of redirection on top of more unstable identifiers such as URLs. From a conceptual viewpoint, no context information is part of this relationship. To support the challenges we have presented so far, it is evident that this concept must be extended, but the specific details as to how this should happen are not, as yet, fully understood.

In this paper, we make a first attempt to clarify the understanding of what persistent identifiers (or the process of persistent identification per se) are in this new extended context. We will set out with a short review of reasons motivating the use of unique persistent identifiers and illustrate two use cases in Section 2 that highlight some of the unanswered problems. A more detailed discussion of the use cases in Section 3 leads us to establish the concept of inter-identifier links as a promising approach to empower the use cases. Our main course of action is then to define a suitable problem space and formulate a set of requirements in Section 4 that result from the use case analysis. In Section 5 we design a set of abstract data types (ADTs), which can act as a blueprint for potential implementations and accommodate the needs of long-term archival. The requirements and abstract data types together form the basic understanding of persistent identification, which, lacking a suitable widely accepted definition of an 'extended' persistent identifier, we propose to see as a well-fitting surrogate. Before we present areas of future work and draw final conclusions, Section 6 provides a perspective on how a practical solution based on this framework could be implemented using the Handle System (http://www.handle.net).

## 2 MOTIVATION AND USE CASES

Duerr, Downs, Tilmes, Barkstrom, Lenhardt, Glassy, et al. (2011) provide several reasons that motivate the use of unique identifiers for data objects. Users will benefit from being able to uniquely and unambiguously identify data of interest, an aspect that is also important for data citations as demonstrated by DataCite (http://www.datacite.org) (Brase, 2009). The barriers for finding and accessing data can also be lowered as large parts of the multitude of scientific data infrastructures – often specific to projects, organizations, or scientific communities – become transparent. Users will also not have to care about which infrastructure a particular data object is served from. Another benefit from establishing unique identifiers is to facilitate data management over time. Archival and storage technology consistently evolves, and frequent migration is inevitable. An identifier that is not only unique but persistent throughout such technological changes does not only work towards the benefit of data users but also eases the data management tasks of archive operators.

For more details on the general motivation, we refer the interested reader to the article by Duerr, Downs, Tilmes, et al. (2011), who also provide clear and demonstrative user scenarios. In the following, we will provide two exemplary use cases that work well along the same lines. Although they come from the background of Earth System modeling data, major principles should transfer well to other scientific communities involved in EUDAT or the Research Data Alliance.

The need for persistent identification of language resources has also led to the creation of ISO 24619, which provides a comprehensive vocabulary for persistent identification and requirements for PID frameworks in this particular domain. The requirements are more specialized than what we will develop here, as they are targeted at language resources and their intrinsic properties. Important requirements identified are the support for multiple resource locations per single identifier, association with XML-based metadata, and some means to identify parts of a language resource.

Each use case is written from an end-user's perspective, i.e., from someone who consumes digital material published by others.

### 2.1 Maintaining contextual information

**Use Case 1** *You were asked by a colleague about the details of a data analysis you did several years ago. You checked for the data, and it appears the input data is gone, but you still have its persistent identifiers written in your research diary. Using just these identifiers, you would now like to check if some form of metadata is still available to which you can point your colleague to help her better understand the original conditions, even though you are unable to reproduce the complete analysis.*

The motivation to keep the contextual information of scientific data intact even if the data is no longer available is also pointed out by Duerr, Downs, Tilmes, et al. (2011, p. 141): "There may be value to maintaining identifiers that resolve to metadata including provenance and context information for a data product used in scientific research even if the data is no longer available, although it may be that the scientific goal of being able to replicate original results is then impossible."

### 2.2 Tracing provenance

**Use Case 2** *You got an interesting dataset from a data portal, which appears to have been highly processed already but might just be what you need for your analysis. How do you ensure that no hidden assumptions that might contradict the assumptions for your own analysis went into these processing steps? You need to find out more about the processing history of this dataset as the information available in the portal does not cover these aspects. You would certainly prefer to know all details about the processing steps, but short of that even* *minimal information, such as what intermediate data products looked like or which data sources were combined at some point in the history of the dataset, may help you.*

The processing history is part of the *provenance* of the dataset. A general and broad definition that focuses on the view of provenance as a process is given by Moreau (2010, p. 23): "The provenance of a piece of data is the process that led to that piece of data". Moreau also presents different more specific definitions of provenance that each focus on different aspects. For the course of this paper, we take the more limited view of provenance as a directed acyclic graph (DAG) of derivative data objects.

As explained by Moreau, such a graph consists of nodes representing data items and edges representing derivative operations on input data that produce output data. The provenance record of a dataset is established by querying the graph. Besides nodes for data, the graph may also be extended to contain contextual information, such as processing log files or additional metadata records. However, our discussion will focus on the core concept of derived data objects as these form the base layer to which additional information nodes must be attached through specialized edges.

We intentionally limit our solution in the level of detail that can and should be achieved. The derivative history may just be enough to provide some of the minimal insight requested by the use case. We acknowledge that more sophisticated provenance gathering and representation beyond the scope of our work is necessary to provide the means to answer more complex provenance queries that transcend the simplified DAG view. However, in view of data-intensive science, we reason that large coverage with low detail of information is a worthy goal nonetheless.

This solution favors a layer-based view, where higher-level layers can build upon the simple DAG and attach more detailed information or point to external systems. Provenance information that goes beyond the simple derivative view is seen as context information in terms of the first use case while the baseline provenance DAG provides the unifying element.

With DataCite promoting the citation of data through DOIs, there already is a powerful solution in place for the 'tail end' of the provenance graph. Yet DOIs are currently not meant to be assigned to intermediate or even temporary results. Due to the often unpredictable course of scientific data processing, it is nonetheless necessary to assign persistent identifiers to even those objects as early as possible, preferably at creation time, which seems quite out of scope for DataCite.

In general, the motivations behind DataCite fit very well with this nonetheless. The producer of an original dataset can get credit if processed data is cited and a provenance graph is available. It can act as another incentive for data authors to publish data with persistent identifiers and for authors of scientific articles to cite data that was used.

## 3 ANALYZING THE USE CASES

Before taking the discussion a full step towards a more abstract model, we will first have a closer look at some of the implications following out of the use cases and draw the need for inter-identifier links from them.

### 3.1 Contextual information network

In EUDAT, different kinds of entities will receive persistent identifiers. The term *persistent identifier* is used consistently throughout EUDAT and related activities, such as the Research Data Alliance, so we will not deviate from it here although the definition of a *locator* given by Duerr, Downs, Tilmes, et al. (2011) is a better match.

Besides scientific data objects, entities that receive identifiers include different kinds of metadata objects but may also go as far as information about related entities, such as person or organization records. For a scientific data object, all of these entities constitute its contextual information. The direct counterpart of the first use case is therefore the issue that if a part of this contextual information is lost, the value of the scientific data object diminishes significantly. It would be preferable to implement some form of graceful degradation, meaning that upon partial information loss, as much as possible of the remaining information is still meaningful. This concludes the general motivation to subject not only the scientific data object but also the contextual information to digital preservation and to assign persistent identifiers to all of them.

Next, we must ask how contextual information should be attached to data objects. With persistent identifiers being assigned to all entities, the obvious solution is to use the identifiers to link contextual information together, and indeed EUDAT is taking efforts to go in this direction. Since the links are not established between, e.g., data and metadata but rather between their respective identifiers, they should be called *inter-identifier links*, thereby forming an *identifier graph* that will consist of identifiers as nodes and inter-identifier links as edges. Loss of attached domain data will not impact the connectivity in the graph. But how and where should these links be encoded?

We suggest that such link information should be stored closely coupled with the persistent identifiers themselves to make sure that it is not easily lost. In practice, this is mostly a question of architecture design. Our goal is to provide a framework that supports this at the more conceptual level. In Sections 4 and 5 we will therefore show how such links are part of a metadata record that is strongly coupled with the actual identifiers through requirements and abstract data type definitions. So although the precise answer to the question is that this is a matter of practical implementation, we can nonetheless provide the boundary conditions for a robust architecture design.

## 3.2 Provenance

To support the second use case, the provenance graph must be encoded so that the user can query it later on. Given that all data involved receive identifiers, we basically have the same options as above. The data represent the nodes in the graph while the links represent the edges. The question of how to gather such provenance information is out of scope here. The interested reader may refer to, e.g., Moreau, Ludäscher, Altintas, Barga, Bowers, Callahan, et al. (2008).

To maintain the full provenance record of the processed dataset, the path through the graph that leads to it must not be interrupted, meaning that all edges that form paths to the dataset must be preserved. If one link in this graph breaks, the provenance record of the data is flawed. Thus, we would like to ensure that the edges enjoy a high level of preservation and that some form of graceful degradation is possible.

We therefore suggest that the provenance graph should become part of the overall identifier graph. A solution constructed around inter-identifier links and providing the possibility to transparently embed context information also works in line with an overarching view of persistent identifiers (and in this case, links between them) as a base layer, which higher-level services can extend. Vice versa, to achieve graceful degradation not just within the base layer but also across layers, the base layer must stay ignorant of the higher-level information, so that if, for example, more sophisticated provenance information at a higher layer is lost, the user can still rely on the base layer as a fallback for less detailed provenance and context information. Higher-level solutions that may build on top of the simple graph include Linked Data (Bizer, Heath, & Berners-Lee, 2009; Heath & Bizer, 2011), the emerging W3C provenance standards (the latest version of the data model should be available at: http://www.w3.org/TR/prov-dm), or solutions that are based on the Open Archives Initiative's Object Re-use and Exchange Protocol, such as the one presented by Duerr & Glassy (2012).

Having discovered two potential applications for binary inter-identifier links, we can also see that these links must be typed to distinguish provenance links from contextual information links on the same objects. Other use cases may evolve quickly and introduce even more types. In fact, the safe replication use case illustrated in Figure 1 hints at at least one more potential link type to connect replicas with an original data object.

We will revisit the topic of typed inter-identifier links when we formulate the requirements. An important characteristic of the links is that some links are immutable, e.g., provenance links, while others may change or be extended over time, such as links to contextual metadata. We will therefore try to accommodate both mutable and immutable links.

Figure 3 illustrates the two discussed link types in an exemplary identifier graph. Note how contextual information may refer to further entities. In fact, contextual information forms a subgraph with a central object at the root. The central object is linked through provenance links to other objects, creating the provenance graph

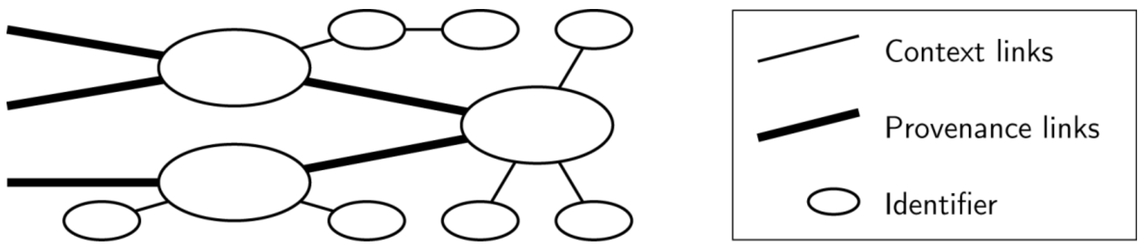

**Figure 2.** Typed inter-identifier links

of derivative data objects. These links may even be directed (one-way only or bidirectional), creating provenance link subtypes, such as 'predecessor' and 'successor'.

We must point out here that provenance and context links are by no means limited to a 1-to-1 relationship. The identifier graph is an abstract model, which only implies the existence of typed links and nodes. There may be any number of links of any type connecting one node with others. This of course directly leads to the question of how to provide a framework to organize a large number of links by, e.g., coalescing them into hierarchies and collections. A collection may, for example, be a node of a specific type that implicates collection semantics and refers to each of its respective collection members through individual links. The details of how such superstructures can be imprinted on top of the graph view will remain an important area of future work.

## 4 REQUIREMENTS

In this section, we will develop requirements for extended persistent identification, based on our use cases. We will start by defining core concepts and derive requirements in the following discussion. Similarities with the conceptual framework developed by Kahn and Wilensky (2006) exist; the major difference being that while Kahn and Wilensky define an overarching Digital Object, we separate the resource from other information, which helps us to draw the line between levels of preservation later on.

### 4.1 Definitions

First, we will provide some basic definitions for our core concepts. Note that we only look at the identifier resolution operations from a consumer's perspective. We will not try to further specify the processes of identifier acquisition and resource depositing here as is, for example, done by Kahn and Wilensky. Nonetheless, the definitions are largely compatible.

An *identifier* is a unique string of characters formed according to the *identifier naming policy*. This policy is unique and globally defined and may typically be the result of a community agreement or a standardization process.

A *resource* is data, arbitrary digital material in Kahn and Wilensky's sense, that is globally accessible through a URL. The URL is called the *resource link*. In contrast to ISO 24619, we require the use of a URL in particular as opposed to a URI to ensure that the resource can be easily located. Access may require authentication and may be unsuitable for nonhuman users (i.e., software agents). The URL is not necessarily stable: The same resource may be served through a different URL in the future, and the former URL may have become inaccessible. Resources may be held in databases, on web servers, or generally in any kind of data storage that provides an access interface that accepts URLs. We will in general call these the *resource repositories*. As we will see later on, a repository may be completely separated from the storage that holds identifier and persistent metadata information. One important condition however is that the repository is managed in some way, meaning that URL changes can be detected and propagated. This is conceptually slightly different from the common problematic phenomenon known as 'link rot' or 'URL decay', which persistent identification cannot address solely on its own. A data object is a particular type of resource that implies a less abstract and more domain-oriented view.

In contrast to ISO 24619, we take the view here that an identifier is used to generally identify one resource at most, requiring that each resource is served at an individual URL. For separate resources, including identical copies of the same resource, separate identifiers must be generated. We do however not preclude the possibility of having more than one identifier point to the same resource at this point. Pointing to multiple resources from a single identifier may be achieved in the future using collections that, e.g., aggregate replica identifiers.

*Persistent metadata* encompass additional information that is required by the use cases or operational services, e.g., contextual information (see use case 1), but most particularly inter-identifier link code. Metadata are further divided into immutable and mutable metadata. Similarly, there are also immutable and mutable links. Immutable metadata cannot be modified after initial registration of the identifier. Mutable metadata can always be changed. This reflects experiences made, for example, at the WDCC where parts of metadata may be subject to infrequent updates, e.g., contact information for responsible parties while, e.g., checksums of data objects are typically immutable metadata.

It must be emphasized at this point that *persistent* metadata in this sense are not to be confused with fully-fledged *domain* metadata that are currently held and curated in large databases of scientific data centers, such as the WDCC. Persistent metadata in our context are meant to be *minimal*, comparable to Kahn and Wilensky's *key-metadata*, and do not enjoy the same management facilities scientific data centers provide. Domain metadata in this respect are in fact a resource (which consequently may carry its own persistent metadata). Persistent metadata may well form a subset of domain metadata; however, the details are very much dependent on the particular use cases that work on the PID level and not on the domain level.

A *resolution operation* takes an identifier string as input and provides a response. We define two subtypes of resolution operations: metadata resolution and resource resolution. For all operations, the response may be that the identifier is unknown (or malformed if it does not match the naming policy).

A *resolver service* is a publicly accessible service that performs metadata resolution operations and also takes part in resource resolution operations.

If successful, a *metadata resolution operation* returns all persistent metadata, which include the resource link. A set of resolver services may be considered to be canonical, and the requirements discussed here may in fact hold only for such distinct sets – the point being that many requirements can be violated by deviant services not working in the way intended by the infrastructure's designers. We therefore exclude such services from our considerations.

A resolver service is not necessarily a web service at a defined URL. It may also be, for example, an API implemented in a publicly available software package that maintains and queries a distributed hash table. A resolver service may directly or indirectly perform services, e.g., it may redirect a request to a cascade of other services to produce a response.

A *resource resolution operation* returns the resource. Note that this operation can only be performed by a resolver service and some a priori unknown resource repository in combination: The resolver service can provide the resource link, and the particular repository serving the link target location returns the resource. Considering practical implementations, this operation is thus more of a conceptual nature while the metadata resolution operation may actually be implemented as a single self-contained core service or a set thereof.

A resolution operation is considered *unsuccessful* only if it is permanently unsuccessful because, e.g., the resolution service does not recognize the identifier or because the resource is gone from the repository. Temporary disturbances, such as server maintenance, network disruptions, and so on, are not considered here as the cause for unsuccessful resolution.

An identifier is *resolvable* by a resolver service if the service succeeds at least in a metadata resolution operation. An identifier is *fully resolvable* by a resolver service if the service succeeds in all resolution operations.

A *registered identifier* is an identifier that is resolvable by at least one resolver service, implying it to be classified more precisely as a *locator* rather than an identifier in terms of Duerr, Downs, Tilmes, et al. (2011). The registration process may involve, for example, a central registry service or database or a distributed solution such as a distributed hash table. Here we do not specify this any further, however.

## 4.2 Developing the requirements

Given the usage scenarios of EUDAT etc., persistent identifiers should fulfill the purpose of providing long-term stable resolution to resources using unique identifier strings. The uniqueness criterion is captured in the following requirement:

**Requirement 1** *A registered identifier must be globally unique: The result returned by a resolution operation of the same registered identifier must be identical for any two non-identical resolver services.*

This also mandates that there is only one namespace shared among all identifiers and if there is more than one namespace, the namespace reference must be a mandatory part of the identifier naming policy. Note that this does not imply that all resolution services must be able to successfully resolve all identifiers – that is required only by req. 3, see below.

For inter-identifier links to be navigable by a service, the service must be able to resolve the link target identifier in all cases. As it would be impractical and often impossible to maintain a complete list of all resolver services at all times, we require the existence of a central service.

**Requirement 2** *All registered identifiers must be globally discoverable: There must be at least one globally accessible resolver service that directly or indirectly resolves all registered identifiers.*

This requirement directly implies a number of more detailed requirements for issues such as that resolution must continue even if the central resolver service is continuously unreachable, the entity maintaining it goes out of business, and so on. A solution that aims to conform to this requirement will thus have to take care of these implicit requirements as well. At this point we must point out that such a solution can by no means be limited to a purely technical perspective as this would never be sufficient to deal with such organizational and social dimensions; it can only provide the mechanisms that take care of the technical workflow if, for example, the central server is relocated.

An important consequence for practical implementation of req. 2 is therefore to put policies in place that deal with these non-technical issues.

This still leaves the possibility for some resolver services not being able to resolve all identifiers, which can be sound for a highly distributed system, for example, but may be confusing for a human user who happens to know only one resolver service. This might, for example, be a project-specific service that the user has been using successfully in the past. If now given an identifier that is, however, unrelated to the project, the resolver service will be unable to resolve the identifier, and thus the expectation of the user will not be fulfilled. For the sake of improved usability, we can thus formulate a non-essential requirement, meaning that we will not enforce this requirement in our continuing discussion.

**Requirement 3 (non-essential)** *Any globally accessible resolution service that successfully resolves one identifier must be able to resolve any registered identifier.*

Note that this only concerns metadata resolution operations because resolver services cannot fully perform resource resolution operations on their own. This requirement implies that there is a single information space shared among all resolution services.

The repository that holds a resource may change over time as underlying technology may be replaced. Thus, the link to the resource must be regarded as being unstable on a long-term perspective. To ensure persistency of the link between a registered identifier and a resource, we therefore formulate the following requirement.

**Requirement 4** *The link to the resource that an identifier identifies must be changeable: If the resource is relocated, it must be possible to reflect this in the identifier.*

This implies that the resource link is part of the mutable metadata.

In general, long-term preservation of resources can be captured in the following requirement.

**Requirement 5** *A successful resource resolution operation or immutable metadata resolution operation must always return the same response given the same identifier.*

This concerns both immutable metadata and resource resolution operations and has several notable implications. First of all, it means that the resource and also the immutable metadata record must not be modified once an identifier has been registered. Secondly, it also means that if a registered identifier were deleted (unregistered), it must never be reassigned to a different resource (in fact, further below we will see that such an operation is impossible). It also implies that if a resource is versioned, i.e., if a new or revised version of the resource is published, it cannot reuse the old identifier.

This of course raises the question of how the difference between resources is defined, i.e., the *identity* of resources, which is however out of scope of this paper. Here we can only suggest that the view on identity may be community specific, yet from the infrastructural, technical perspective of software services, a notion of bitwise identity would be preferable.

Regarding mutable metadata, the resolution operation will return differing responses over time. In general, the decision about which information to put in which part of the persistent metadata – the immutable or the mutable part – is up to the user who registers an identifier and depends particularly on the use cases. Regarding inter-identifier links, we can however already see that links to contextual information should be part of the immutable metadata while the resource link or links to newer versions must be part of the mutable metadata.

To maintain the links, their code must be preserved. As the link code is part of the persistent metadata, we need to set up the following requirement.

**Requirement 6** *All metadata resolution operations must continue to succeed, even if a resource becomes unavailable.*

This requirement ensures that inter-identifier links remain resolvable even if both the source and target resource are unavailable. It is also reflected in the definition of a resolvable identifier: A resource resolution operation may fail, yet the corresponding metadata resolution operation will always succeed (as long as the persistent metadata record is not lost accidentally). Effectively, this means that identifiers cannot be purposefully unregistered (deleted). It does however not mean that resources cannot be deleted.

## 5 ABSTRACT DATA TYPE DEFINITION

According to our use cases, of all the data that we are discussing here, some data should preferably be stored as safely and persistently as possible. Roughly, this encompasses the identifier's persistent metadata record, which also includes inter-identifier links. We will call this the *primary level of preservation*. The secondary level of preservation contains the domain resource an identifier refers to.

How do we provide a vessel that can be used to set up an implementation or evaluate existing implementations, enveloping the primary level of preservation?

We will try to answer this question by providing definitions for two abstract data types (ADT) that offer the necessary operations and adhere to the requirements. These ADTs manifest the primary level, encapsulating operations that remain functional even if the secondary level is lost. Therefore the resource is not a part of this data type: There is only a 'get resource link' operation but no 'get resource' operation.

It is important to remember that the ADTs are not designed to ensure all requirements are satisfied in all cases. More precisely, the ADTs fulfill requirements 4 to 6. The first three requirements can only be satisfied by actual implementations conforming to them as several possible solutions exist. For example, the first requirement may be satisfied either by establishing a central naming authority or implementing a distributed hash table. Other solutions may exist as well, and at this point, we do not want to restrict the number of possible or existing conforming solutions.

In general, actual implementations also need to be highly scalable to deal with the potentially huge number of resources and links between them. We did not capture this in explicit requirements, but future work may well establish adequate performance requirements.

## 5.1 The persistent entity

The **persistent entity** ADT is defined by the following operations:

**Create instance:** Returns a new instance given mandatory parameters of identifier string, resource link, and immutable metadata, which includes immutable inter-identifier links. The operation must fail if the identifier violates req. 1, i.e., if the identifier is already resolvable. Immutable metadata may contain immutable inter-identifier links. Note that conceptually, this is an atomic operation while practical implementations may in fact divide this into several operations.

**Get resource link:** Returns the resource link, i.e., the parameter of the last 'set resource link' operation or in case no such operation was performed yet, the value of the corresponding parameter to the 'create instance' operation.

**Set resource link:** As mandated by req. 4, the resource link must be changeable. Calling this operation must only change the result of future calls to 'get resource link' but not the result of any other operation. To prevent violation of req. 5, the operation *should* confirm that the resource at the new location is identical to the one at the old location. As the resource at the old location may however already be unavailable at this point, this may be not be possible at all times. As already mentioned in Section 4.2, the exact definition of the identity of resources is out of scope for now. One possible solution for bitwise identity is to include a mandatory checksum in the immutable metadata and validate the new resource against it. Short of that, it may come down to a strictly enforced operation policy prohibiting the reassignment of a non-identical resource at the new location.

**Get immutable metadata:** Returns the immutable part of the persistent metadata record including immutable inter-identifier links, i.e., the same value as was given as the corresponding parameter to the 'create instance' operation.

**Set mutable metadata:** Overwrites the mutable part of the persistent metadata record. If inter-identifier links are included, the operation must validate the resolvability of the target identifiers using the dedicated service mandated by req. 2. The operation should also enable the typing of links as discussed in Section 3.2.

**Get mutable metadata:** Returns the mutable part of the persistent metadata record, i.e., the same value as was given to the last call of 'set mutable metadata'. If this operation optionally resolves existing inter-identifier links, req. 2 must be fulfilled.

The potential operation *delete instance* would remove the instance, resulting in the identifier becoming unresolvable. This would violate req. 6. We therefore require that an implementation of the ADT may offer such an operation only for purposes of debugging or testing but not for regular operations.

To distinguish purposeful data removal from accidental data loss, an operation called *mark resource as deleted* might be the best solution that does not violate any requirements. Such an operation could, e.g., affect the result of the 'get resource link' operation to return a special flag. We will not discuss this further at this point.

## 5.2 The resolver service

There is also the ADT of a **resolver service** with only one operation:

**Resolve identifier:** Given an identifier, this operation either returns an instance that bears the given identifier or fails (returns 'unknown identifier') if and only if no instance with given identifier exists (to the knowledge of this resolver service). If a call to the operation with identifier X does not fail but returns instance A, any future calls with the same instance X must return the same instance A, differing only in mutable metadata, as mandated by req. 5.

Earlier, we defined two abstract operations: resource resolution and metadata resolution. The metadata resolution operation is well covered by the operations of the persistent entity ADT. The resource resolution operation is a bit more complex as it must combine the resolver's 'resolve identifier' operation, the instance's 'get resource link' operation, and some additional operation in the secondary preservation level to actually retrieve the data. This transcends the scope of our ADT definitions, and thus it is up to actual implementations to fully enable this operation.

## 6 IMPLEMENTATION CONSIDERATIONS FOR THE HANDLE SYSTEM

Due to the conceptual similarities in the underlying framework, the Handle System is an obvious first candidate for implementing the ADTs. Based on the conceptual framework of Kahn and Wilensky, the Handle System is a distributed database, globally accessible and continuous, for referencing Digital Objects through persistent and unique identifiers. We will briefly outline the conformance of the Handle System with the requirements and hint at a possible implementation of the ADTs.

The Handle System offers a multitude of resolution services: Besides the central service hosted at the Corporation for National Research Initiatives (CNRI) (http://www.cnri.reston.va.us), other proxies exist and can be set up and join the global system after registration with CNRI. Given a set of these registered resolvers, req. 1 holds. At least the central service at CNRI can act as the pivotal resolver service required by req. 2. As mentioned in the discussion of req. 2, some important non-technical issues remain for the implicit requirements – there must be policies, for example, if CNRI shuts down its servers permanently. In general, all registered resolver services are able to resolve any registered identifier, thus also fulfilling req. 3. Handle data can be changed after initial creation, most importantly including the Handle target, thus fulfilling req. 4.

Req. 5 and req. 6 can be fulfilled using the Handle System in combination with a set of enforced policies. One possible way to achieve this is to employ a wrapper service around the Handle System, mandatory for all operations, which follows the ADT definitions. The Handle System offers a mechanism to store basic metadata along with identifier information, which can be exploited to store mutable and immutable metadata and establish the first level of preservation. A common policy stating how, e.g., inter-identifier links are to be encoded and a controlled vocabulary for link types must be set up as well.

A prototypic wrapper service targeting the Handle Service as a first base layer is currently under development (https://github.com/TobiasWeigel/lapis ).

## 7 OPEN ISSUES AND FUTURE WORK

We have discussed how the Handle System can be used as a framework for a conforming implementation of the ADTs, yet there are open issues. Full conformance can only be achieved by additional policies or services and can only be fully evaluated by a prototypical implementation.

Besides the Handle System, other candidates exist that may or may not provide a baseline for implementation. Among these are other solutions presented by Duerr, Downs, Tilmes, et al. (2011), for example, the ARK (https://confluence.ucop.edu/display/Curation/ARK) or PURL (http://www.purl.org) systems, which may be evaluated in terms of how they satisfy the requirements and how the ADTs could be implemented using them.

We have also deliberately left out important questions that are implicated in the EUDAT scope. For example, there may be a need for multiple resources being connected with a single identifier as is described in ISO 24619. Currently, the view here is strictly on a 1-to-1 basis: Each identifier identifies only one resource. How do the requirements and data type definitions change if we allow this?

In EUDAT and ISO 24619, another important question concerns collections of entities, possibly separated into policies with exclusive ownership and policies with non-exclusive ownership, so-called virtual collections. Collections can also facilitate proper structuring of context and provenance information. There are many reasons why collections should be able to change over time but, nonetheless, bear a persistent identifier. Introducing collections may, for example, give rise to another set of requirements.

The scalability of conforming solutions must be ensured as well, preferably through hard performance requirements. Defining such requirements based on practical reasoning should be addressed by future work and will be increasingly important in view of the Research Data Alliance activities that aim to spread out across communities and scientific domains.

The notion of the identity of resources and accordingly the exact criteria for being allowed to assign a new resource location is a topic that may not be solvable for all cases yet should be explored further. There may, for example, be a distinction between technical identity and conceptual identity, and in the framework of a PID infrastructure, only the technical identity may be essential to ADT operations. A definite solution is however missing.

There is also the question of how to deal with versioned entities. Because we have required that resolution results never change, versioning can only be implemented by issuing new identifiers and using links to connect them. How users or agents are able to retrieve the latest version or specific obsolete versions is so far unanswered. With the ability to type inter-identifier links, a ‘new version’ link type could be exploited by a resolver service to offer a special operation to automatically redirect an end-user to the latest version of a dataset. In our current framework, this will however violate req. 5, which on the other hand may be tolerated as this is precisely the intention of such an operation.

There may also be need for mutable resources, e.g., mutable domain metadata that change frequently or where for some other reason it seems impractical to strictly require immutability and thus issuing of new identifiers on change. There are also mutable resources in the EUDAT community that deserve persistent identification nonetheless, e.g., accumulating observational records of environmental data such as temperature or seismicity records. If we allow for mutable resources, what will happen to the requirements and ADTs? Clearly, this puts a whole new twist on the question of persistency.

The view of provenance as a directed acyclic graph is well suited for the low-level approach linked persistent entities provide. In this form of encoding, the provenance graph does however reside at the lower end of expressivity; it does not facilitate reproducibility of the respective process, nor does it mandate any form of documentation beyond registering input and output data. We thus promote a layer-based approach where higher layers extend the provenance information in particular while keeping the provenance DAG as a fallback option.

Future research should thus be conducted regarding how to enhance the graph of inter-identifier links to increase its expressiveness, e.g., through Linked Data (Bizer, Heath, & Berners-Lee, 2009; Heath & Bizer, 2011) building on top of the linked identifiers. This could ultimately take the vast amount of low-level research data into the Linked Open Data cloud, including basic provenance information. Bechhofer, Ainsworth, Bhagat, Buchan, Couch, Cruickshank, et al. (2010) advocate the use of *Research Objects* as rich aggregations sitting on top of Linked Data. Providing essential elements of provenance at the lower level, we can help to sustain these kinds of higher-level objects.

## 8 CONCLUSIONS

We have presented a framework for persistent entities that offer features going beyond the mere redirection layer based view of persistent identification. We have defined the core concepts and constructed some building blocks from which more complex applications may serve many use cases not only the two exemplary ones presented in the beginning. Although the use cases may be specific to the Earth system modeling community, the framework can be applied by other communities as well if the requirements match. In such cases, the building blocks will and should be used in community-specific ways.

The requirements and data type definitions developed here also allow for evaluating existing or future solutions and, where possible, extend them. Vice versa, future use cases may introduce more requirements and extend the described abstract data types.

The main building blocks include mutable and immutable persistent metadata and the ability to construct general purpose information networks of associated items using typed inter-identifier links. The ability to link entities together is particularly well suited for a number of use cases whose importance cannot be underestimated in view of data-intensive science, including, for instance, safe replication and virtual collections. Moreover, the presented framework acknowledges the problem area of long-term archival of scientific data by establishing a dedicated level of preservation that can act as a fundamental fallback.

## 9 ACKNOWLEDGEMENTS

Acknowledgements go to Daan Broeder (Max-Planck-Institute for Psycholinguistics), particularly for pointing out the ISO 24619 reference.

## 10 REFERENCES

Bechhofer, S., Ainsworth, J., Bhagat, J., Buchan, I., Couch, P., Cruickshank, D., Roure, D. D., Delderfield, M., Dunlop, I., Gamble, M., Goble, C., Michaelides, D., Missier, P., Owen, S., Newman, D., & Sufi, S., (2010) Why linked data is not enough for scientists. *e-Science 2010, Sixth International Conference on e-Science*, Brisbane, Australia.

Bizer, C., Heath, T., & Berners-Lee, T. (2009) Linked data – the story so far. *Special Issue on Linked Data, International Journal on Semantic Web and Information Systems 5(3),* pp 1–22.

Brase, J. (2009). Datacite – a global registration agency for research data. *Fourth International Conference on Cooperation and Promotion of Information Resources in Science and Technology,* pp 257-261. Beijing, China.

Duerr, R. E., Downs, R. R., Tilmes, C., Barkstrom, B., Lenhardt, W. C., Glassy, J., Bermudez, L. E., & Slaughter, P. (2011) On the utility of identification schemes for digital earth science data: an assessment and recommendations. *Earth Science Informatics 4(3)*, pp 139–160.

Duerr R. E. & Glassy, J. R. (2012) Demonstrating preservation connections using OAI-ORE. *Winter Meeting 2012, ESIP Commons*, February 2012. Retrieved December 6, 2012 from World Wide Web: http://commons.esipfed.org/content/demonstrating-preservation-connections-using-oai-ore

Heath, T. & Bizer, C. (2011) *Linked data: Evolving the web into a global data space (Synthesis Lectures on the Semantic Web: Theory and Technology).*

Hey, T., Tansley, S., & Tolle, K. (Eds.) (2009) *The fourth paradigm: data-intensive scientific discovery*, Redmond, Washington.

High Level Expert Group on Scientific Data (2010) Riding the wave – How Europe can gain from the rising tide of scientific data. Retrieved October 10, 2012 from the World Wide Web: http://ec.europa.eu/information_society/newsroom/cf/itemlongdetail.cfm?item_id=6204

ISO 14721:2012, International Standards Organization.

ISO 24619:2011, International Standards Organization.

Kahn, R. & Wilensky, R. (2006) A framework for distributed digital object services. *International Journal on Digital Libraries 6(2),* pp 115–123.

Moreau, L. (2010) The foundations for provenance on the web. *Foundations and Trends® in Web Science* 2(2-3), pp 99–241, Delft.

Moreau, L., Ludäscher, B., Altintas, I., Barga, R. S., Bowers, S., Callahan, S., Chin, G., Clifford, B., Cohen, S., Cohen-Boulakia, S., Davidson, S., Deelman, E., Digiampietri, L., Foster, I., Freire, J., Frew, J., Futrelle, J., Gibson, T., Gil, Y., Goble, C., Golbeck, J., Groth, P., Holland, D. A., Jiang, S., Kim, J., Koop, D., Krenek, A., McPhillips, T., Mehta, G., Miles, S., Metzger, D., Munroe, S., Myers, J., Plale, B., Podhorszki, N., Ratnakar, V., Santos, E., Scheidegger, C., Schuchardt, K., Seltzer, M., Simmhan, Y. L., Silva, C., Slaughter, P., Stephan, E., Stevens, R., Turi, D., Vo, H., Wilde, M., Zhao, J., & Zhao, Y. (2008) Special issue: The first provenance challenge. *Concurrency and Computation: Practice and Experience 20(5)*, pp 409–418.

(Article history: Received 12 October 2012, Accepted 19 February 2013, Available online 4 March 2013)