pilin persistent identifier linking infrastructure

web: http://resolver.net.au/hdl/102.100.272/0N8J991QH

email: policy@pilin.net.au

Version History

Version	Date	Status & changes	Expression identifiers
V1.0	2007-12-19	Release	PILIN/ 1B4YS1PQH hdl:102.100.272/ 1B4YS1PQH

PILIN Guideline

Identifier Association Guidelines

To cite the *latest* version of this work use http://resolver.net.au/hdl/102.100.272/WBNMH9DQH

To cite *this* version of this work, use http://resolver.net.au/hdl/102.100.272/1B4YS1PQH

1 Purpose/Issue

This document provides guidance on what "things" should have persistent identifiers assigned to them by an identifier manager, and at what time.

2 Background

In theory, it is possible to assign persistent identifiers to identify and manage every version, format and copy of every thing created or managed by an activity, and every component of every thing created. However this is clearly unworkable for most activities.

For that reason, this document contains guidance on what should be persistently identified within an activity. The PILIN project recommends that an activity wishing to assign persistent identifiers should

1. create an *information model* that defines the universe of things created or managed by the activity.

Copyright © Monash University

This work is licensed under the Creative Commons Attribution-Share Alike 2.5 Australia License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.5/au/

This work was created as part of the PILIN project. The PILIN project is funded by the Australian Commonwealth Department of Education, Science and Training, (DEST) under the Systemic Infrastructure Initiative (SII) as part of the Commonwealth Government's Backing Australia's Ability – An Innovation Action Plan for the Future (BAA) under the ARROW Project.

2. create an *identifier association policy* that specifies:
    - *what* things from the information model are assigned persistent identifiers, and
    - *when* identifiers will be assigned to things.

If the intent is to use identifiers to access copies of things, then the identifier association policy should also specify:

3. the *resolution behaviour* of the identifiers.

3 Scope

This document is restricted to persistent identifiers.

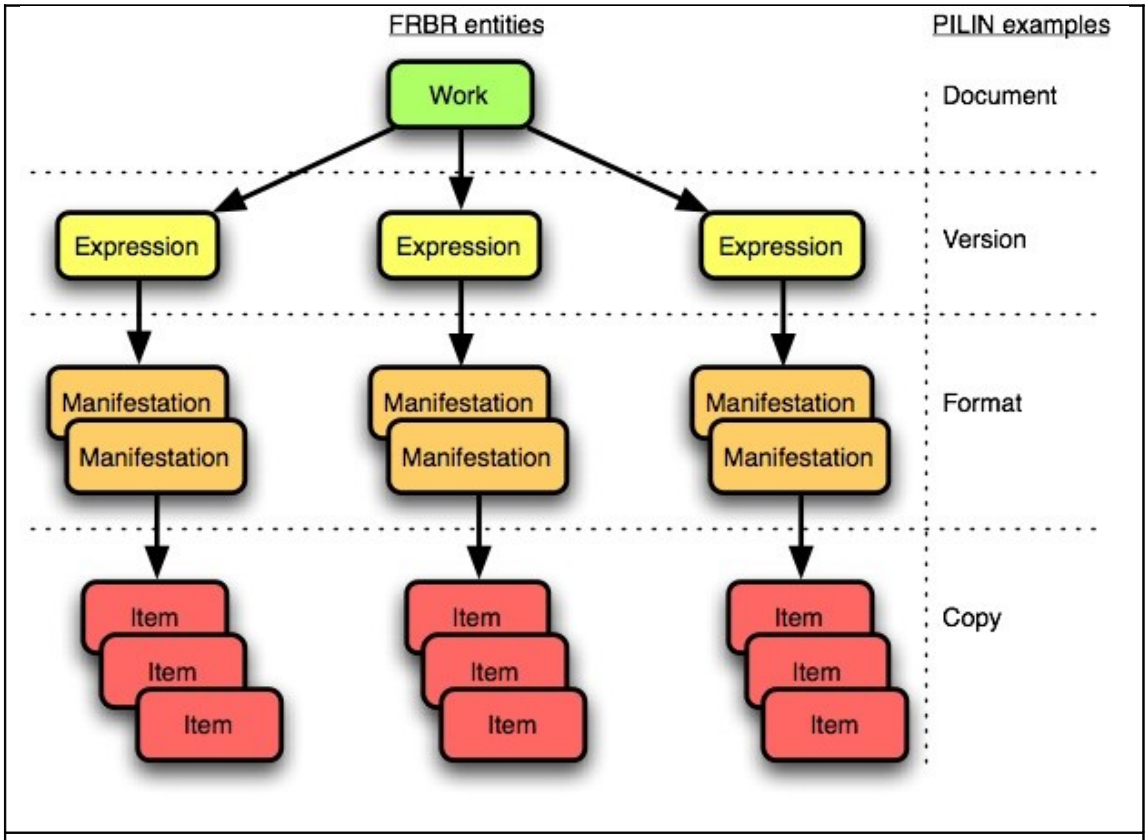
Note that resolution is used in this document as it is defined in the PILIN ontology [1]: an action that takes an identifier as input and returns information on how to access the thing being identified (the association data for the identifier). It does not include actually accessing or obtaining the thing based on that information, which is not the responsibility of an identifier management system. So this document discusses, for instance, what URLs a Handle might map to—but not how a browser should process that URL, once resolved.

4 Guidelines: What to model

4.1 Have an information model

An information model is necessary for any party assigning identifiers to things. The model determines what possible things exist for the party to identify through identifiers. Once that model is in place, parties can decide what things are important enough to identify persistently. Modelling something as a discrete object does not mean that it must have a discrete persistent identifier; but it does mean that a decision is needed on whether it should have a discrete persistent identifier or not. The information model need not be a formal ontology; but it should be explicit enough that identifier managers can comply with it straightforwardly.

The things to be modelled in an information model are not restricted to tangible things, nor are they necessarily time-stable objects. The same object may be represented in an information model in multiple ways. We enumerate below the types of things a model may differentiate between. The enumeration is not intended to be exhaustive or prescriptive. It is influenced by the FRBR model [2], illustrated below



FRBR information model

4.2 Versions

A single thing may appear in different versions at different times. We say a new version of a thing comes into existence whenever there is any change in the content of the thing. These versions may include revisions, transformations, translations, annotations, and so forth. Versions are the type of thing encompassed by the FRBR information model's [2] notion of *Expression*, but we introduce it here as a general concept, independent of any specific information model.

If the difference between versions of the same thing is important to business processes, then the model should capture this by modelling different versions as different things, and representing the relationship between them. The relationship between versions is typically modelled as a dependence on an abstract thing that the versions are versions of (an abstract Work in the FRBR model). The model also needs to capture metadata distinguishing the different versions from each other; e.g. different draft/edition numbers, different creation dates; different contributors.

The model also needs to capture whether different versions of the same thing can exist at the same time, and whether they can be accessed

independently. If different versions are maintained separately, they can be accessed by systems separately, and this increases the requirement for distinct identifiers.

If on the other hand new versions overwrite old versions, and the old version becomes unavailable, then an identifier for the old version can no longer be used to access the old version. Such an identifier no longer has persistence of resolution; so there is less motivation in the business process for having the version identifier be persistent to begin with. But an unresolvable identifier can still be useful: its association may persist, e.g. through metadata. Whether it is useful depends on how the identifier will be used.

4.3 Presentations

A single thing may appear in different presentations. By presentations, we mean any change to the thing which does not affect its content. Presentations may include file formats, schemata, formatting, branding, and so forth. Presentations are the type of thing encompassed by the FRBR information model's [2] notion of *Manifestation*, but we introduce it here as a general concept, independent of any specific information model.

If the difference between presentations of the same thing is important to business processes, then the model should capture this by modelling different presentations as different things, and representing the relation between them. The relationship between presentations is typically modelled as a dependence on an abstract thing that the presentations are presentations of (e.g. a version). The model also needs to capture metadata distinguishing the different presentations from each other; e.g. different file formats, different schemas.

4.4 Copies

A single thing may appear in different locations. Things stored and retrieved from different locations are considered distinct copies of the same thing. Copies of the same thing have the same content and the same presentation. Copies are the type of thing encompassed by the FRBR information model's [2] notion of *Item*, but we introduce it here as a general concept, independent of any specific information model.

If the difference between copies of the same thing is important to business processes, then the model should capture this by modelling different copies as different things, and representing the relation between them. The relation between copies is typically modelled as a dependence on an abstract thing that the copies are presentations of (e.g. a presentation). The model also needs to capture metadata distinguishing the different copies from each other. (The typical instance of this is the

location where the copy is stored—as copies are otherwise indistinguishable.)

4.5 Aggregates

An identifier is not restricted to identifying only a single, non-decomposable thing. A set, bag, or list of things can also be treated as a thing in itself, identified with an identifier. Conversely, a thing identified with an identifier may be decomposable in particular contexts.

Example: A digital library is a collection of documents. Each document is a thing which may be identified. But the set of all documents in the library is itself a thing we may need to identify, as is the ordered list of documents written by a specific author.

On the other hand, a document in the digital library need not be modelled as an atomic entity: it can be decomposed in different ways which may be relevant to library business processes (e.g. chapters, pages, diagrams, sentences, words, letters, pixels). Each of these decompositions may be separately identified. Those decompositions need not be consistent hierarchies (e.g. a sentence may span across a page boundary).

The aggregation and disaggregation of things within an information model is open-ended; so the information model must identify which aggregations and disaggregations are relevant to business processes. (For example, an aggregation of all documents with the same byte length would not normally be relevant to business processes.) More to the point, the model should identify which aggregations and disaggregations the actors triggering the business processes have in mind.

Example: Continuing our digital library example, if we have decided that documents are basic things to identify, the information model should address disaggregation questions such as the following:

- Do any business processes depend on the division of documents into pages? (e.g. document delivery, print citation)
- Do any business processes depend on the division of documents into words? (e.g. tokenisation for textual search)
- Do any business processes depend on the division of documents into diagrams? (e.g. retrieval of diagrams outside the context of the document)
- Do any business processes depend on the division of documents into letters? (note: computer encoding of texts assumes characters and therefore letters; but business-level

processes assume encoding as given)

The information model should also address aggregation questions such as:

- Do any business processes (other than search) depend on the presentation of all documents by an author as a group? (often yes, particularly in browsing)
- Do any business processes (other than search) depend on the presentation of all documents in the library as a group? (e.g. the library itself, harvested by a federation of libraries)

Note: Search is ignored as a motivation for persistent identification of aggregation: search generates an open-ended, dynamic aggregation of objects, by its nature not persistent. A search query result set *can* be assigned its own identifier. But if search is available to external users through a protocol, there is not enough motivation to provide those same users with persistent identifiers for search results.

4.6 Service Transformations

Various transformation services may operate on a thing to give different results. These transformations may take one version of a thing and return a different version or presentation of the thing. They may also be related to the thing more indirectly; e.g. services to retrieve abstracts, thumbnails, metadata, related datasets. Crucially, they are returned dynamically through a defined service, and are not necessarily stored and managed separately: they can be generated dynamically.

In an object oriented view of the world, these transformations are versions of the object, and may be managed through the same data source as the object itself. (e.g. in the Fedora Repository architecture [6], such transformations-as-versions are known as *disseminations*.) Being managed as separate objects, they can have their own identifiers.

In a service oriented view of the world, on the other hand, these transformations are dependent on the object they are transformations of, and need not identified separately; they can be identified through a combination of a service identifier and the identifier for the original object.

Example: In Fedora an object has the Fedora PID (persistent identifier) *example:9876*.

The PDF dissemination of the same object, which is a static object, is identified as *example:9876/pdf*.

An abstracting transformation service has a Fedora PID for its behaviour definition, *example:77*, and a method identifier (name) specifying the operation, *abstract*.

These identifiers can be combined to identify an abstract of the object over PDF: *example:9876/example:77/abstract/pdf* .

In an object oriented view, this is treated as a single identifier for the new object.

In a service oriented view, this is treated as a combination of two identifiers: *example:77/abstract/pdf* is the service identifier, and *example:9876* is the identifier for the original object.

Example: The Knowledge Tree CMS has the identifier 107851 on Sourceforge.

The RSS feed for updates to Knowledge Tree uses the identifier in a transformation service to generate the RSS: http://sourceforge.net/export/rss2_project.php?group_id=107851 .

The RSS feed is identified through a combination of a service identifier (the URL service request) and an object identifier—rather than through a single static identifier.

The decision on whether to assign persistent identifiers to transformations depends both on user expectations and on the technologies that the identifiers will interact with. The OAI-PMH Harvest service, for instance, harvests only objects with identifiers, and not transformations of objects; so if a dissemination is to be harvested separately, it will need its own identifier. (However, an HTTP URL, including a URL query, is a valid unique identifier for OAI-PMH, so long as it is persistent.)

5 Guidelines: What to identify

5.1 Why do we identify?

Once the information model has described what could possibly be identified, the next step is to decide what definitely will be persistently identified.

The answer depends very much on the use that the identifiers will be put to. This becomes difficult to anticipate once identifiers are released by an

identifier manager to other parties. The availability of an identifier outside the identifier management system is also critical to notions of identifier persistence: identifiers are much more difficult to modify once they are released outside the immediate control of the manager [3].

The same holds for the things the identifier identifies. If the thing is not available outside a small group, then there will be little reference to the thing outside the group, and correspondingly less motivation for any such reference to be persistent. If the URL for an object changes, and only one institution ever uses the URL, changing the occurrences of the URL may be tractable. If on the other hand the thing is broadly accessible, then any change in how to refer to the thing is very disruptive. (This is the "patching identifiers" problem, discussed elsewhere [3]). So there is a much greater requirement for the identifier for the thing to be persistent.

For that reason, we introduce the notion of a curation boundary, and motivate the choice of what to identify through a persistent identifier in terms of the curation boundary.

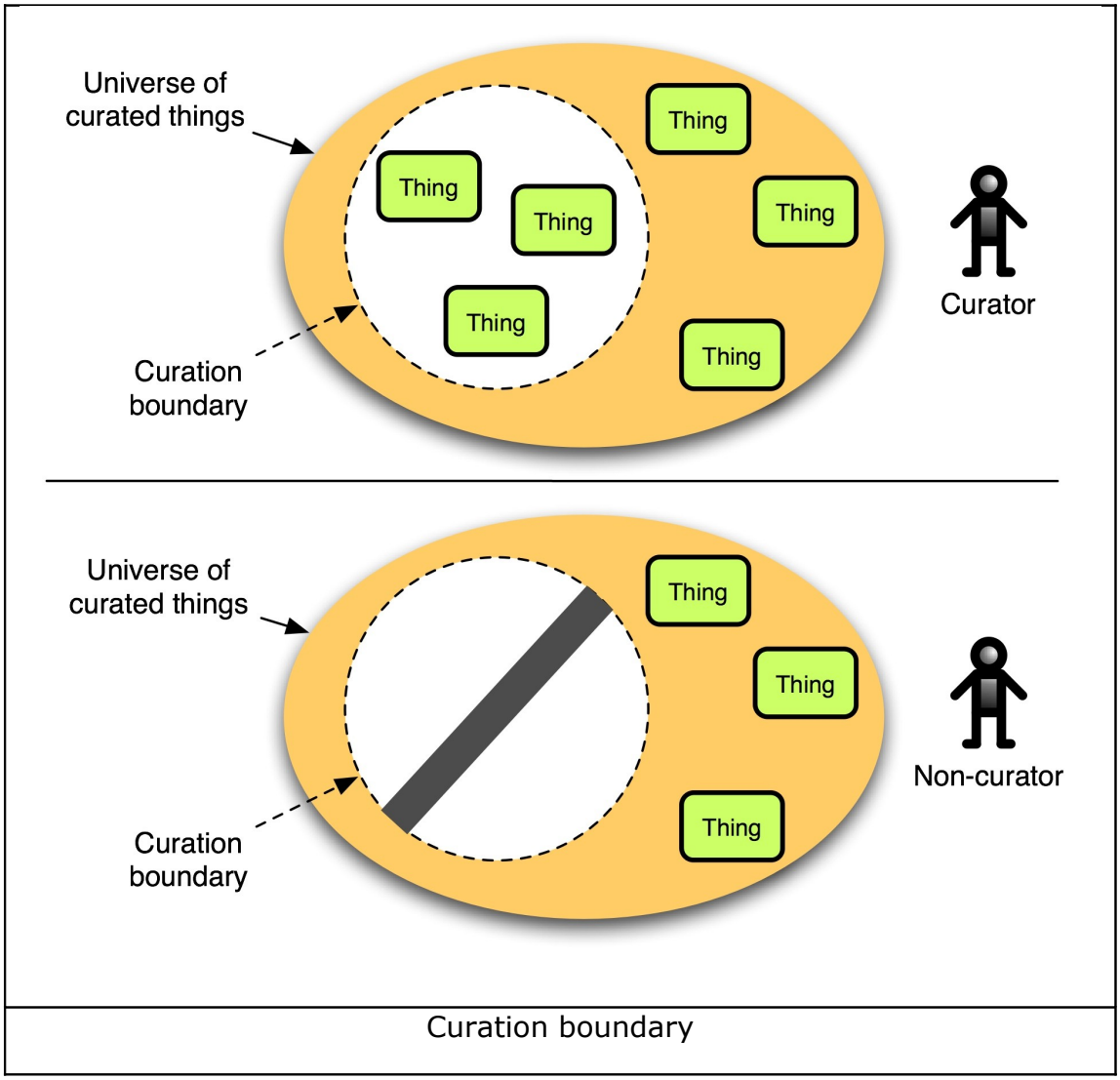
5.2 Curation boundary

The curation boundary is a concept drawn from work within the ARROW, DART and ARCHER repository projects [4] [5], though its application in an identifier context is slightly modified.

Access to a thing through a computer system can be modelled as mediated through a data source. The data source is where the thing is stored and managed, and it enforces access constraints on the thing according to user profile.

The curation boundary for a thing is defined by who has access to the thing through its data source. Things cross the curation boundary by being published:

- Things inside the curation boundary are accessible through the data source, but only by the parties curating things on the data source (i.e. creating and updating things). (The notion of curatorial and non-curatorial actions is defined in more detail elsewhere [1].) Those parties are called the *administrators* of the data source. Things inside the curation boundary are not (yet) published, and are considered too much in flux to publish. Any changes to the thing can occur without informing outside users, and without any need of accountability to outside users (who can't access the changes anyway): this allows the thing to remain in flux as long as necessary.
- Things outside the curation boundary have become accessible by parties who are not already administrators of the data source. Things outside the curation boundary are intended to be stable, and the administrators of the object are accountable to external users if the thing does change.



Identifiers are digital objects, which are themselves accessed through a data source (an identifier management system).. This means that identifiers have their own curation boundary, defined through access to the identifier management system, and are published when the identifier is accessible to external users. And this is quite independent of whether the thing being identified is outside its curation boundary: the thing identified and the identifier are accessed through distinct data sources, and curated by distinct parties.

So there are two curation boundaries involved in identifiers: one for the identifier (the *identifier curation boundary*), and one for the thing identified (the *data curation boundary*). There are also two publication events, one for the identifier and one for the thing being identified. The publication events are not necessarily synchronised, and mean different things:

- The thing is published, its identifier is not: the outside world has access to the thing, but not via the identifier.

- The thing is not published, but its identifier is: the outside world may not have access to the thing, but it does know it exists, because it knows the published identifier is identifying something (the thing is nameable to external users).

So when the data curation boundary is crossed, outsiders gain access to the thing. When the identifier curation boundary is crossed, outsiders become aware of the thing, and have a stable way of referring to the thing.

5.3 What do we identify?

The primary consumers of persistent identifiers are users outside the identifier curation boundary. Both those external users, and the administrators publishing the persistent identifiers, have certain expectations which persistent identifiers should meet. We can use those expectations to suggest what kinds of things are appropriately identified through persistent identifiers.

- Persistent identifiers refer to *published* things: if they are not published, the identifier is not resolvable. (It may still be useful to know something exists without being able to access it—non-resolvable identifiers are used; but the common end user expectation is for resolvability.) So persistent identifiers should only be set up for things crossing the data curation boundary.
    - As discussed below, the persistent identifier may be set up before the thing crosses the curation boundary.
    - What constitutes publishing depends on the information model used: it need not mean providing direct access to a digital presentation of the thing. Rather, it means providing external access to the chosen association data (e.g. resolution) of the thing. For instance, repositories often resolve identifiers to a metadata description of a thing. These descriptions can be hyperlinked to direct presentations of the thing; but releasing the metadata in itself counts as publishing the thing: *representing* it to an audience, rather than presenting it. This makes it possible for persistent identifiers to refer to things other than digital objects.
    - *Published* things include things which had been published but are no longer accessible. Once a persistent identifier is released, it must continue to refer to the same thing, whatever the status of that thing is. If the thing is no longer accessible (it has been archived or destroyed), the identifier should allow the user access to useful metadata on the thing.

This includes how to arrange for access, if the object is archived.

- Persistent identifiers refer to *stable* things. This does not mean that the thing itself does not change; but it does mean that any changes in the thing should be accountable, and within the range of what a user would reasonably expect, given their understanding of the information model. The concern with getting the thing stable enough to be published, and persistently identified, is the motivation for using the term "curation", and not just "access": the thing needs to be prepared to be publishable.
- Persistent identifiers refer to *conceptually meaningful* things. This expectation is contingent on the information model for the domain, which is why information models are necessary to establish persistence, as already discussed. If for instance we have an identifier for a new object based on aggregation or disaggregation of some things in the domain, the new object should be something that makes sense as a conceptual unit to the user, and to the processes they will use to interact with it. The information model should capture what in the domain makes sense to a user. Similarly, a set of pages from a document, or a collection of documents by an author, are (dis)aggregations that make sense to at least some users as a single conceptual unit; an aggregation of every third word in a document will probably not.
- Persistent identifiers refer to *citable* things. This is related to the notion of conceptually meaningful things: if the thing is something a user might want to cite a reference to (e.g. by having a single hyperlink to a representation of the thing in a document), then the thing is worth having a persistent identifier, which will be used for that citation. If there is no business motivation for citing the thing, there is much less reason to assign it a persistent identifier. For instance, presentations are cited in digital documents less often than versions; so there is less motivation to assign presentations distinct persistent identifiers, as opposed to treating them as disseminations of the same document (as is the norm in Fedora), and access them through services.
- As an administrator rather than user expectation: persistent identifiers should refer to things *under the control* of the identifier manager. (Obviously control is defined at a corporate rather than individual level.) If they refer to things outside the control of the identifier manager, then there is only a weak guarantee that the identifier manager can keep the identifier persistent: they can only react and not anticipate changes in how the thing is accessed. This expectation is more applicable if the thing identified is a digital object than if it is not.

- Persistent identifiers refer to things *describable through metadata*. (This is also primarily an administrator expectation, although other systems may also choose to attach metadata to a persistently identified thing, e.g. as annotations.) Metadata describing an identified new object is a common mechanism for aggregation: the new digital object usually has some value added to it in terms of the metadata describing why it was put together, and that metadata is attached to an identifier for the aggregate object. However, if it is difficult to conceive of metadata specific to that thing in particular, then the thing is probably not distinct at a conceptually meaningful level, and should not have its own identifier. For example, it is more difficult to come up with metadata about specific copies of a document than for the document in general. Users are usually not interested in the differences between copies, and those differences are mostly predictable given the location the copy is retrieved from.

5.4 When do we identify?

Since the identifier and the thing it identifies are managed separately, there are discrete steps to be coordinated:

- The thing is created (A)
- The thing is curated (behind its curation boundary) (B)
- The identifier is created (C)
- The identifier is associated with the thing (D, must follow B)
- The identifier is published (E)
- The thing is published (F)

Assume the identifier is persistent, and is used in preference to other mechanisms for accessing the thing (e.g. non-persistent locators). The thing should not be published before its identifier is published: if the thing is published before its persistent identifier, then users will start using dispreferred mechanisms (such as non-persistent identifiers) to get to the thing. Once this happens, users will be reluctant to switch to the preferred persistent identifier, when it becomes available.

On the other hand, the identifier can be published before the thing identified is published. This exposes the manager for the thing to a commitment to publish the thing, but that commitment can be delayed.

Example: I write a paper on an experiment. The paper cites a data object currently behind my data curation boundary: the data object is not yet ready to publish. I anticipate that by the time the paper is published, the data object will also be published, and

readers will be able to access it. I cite the data object using a persistent identifier; the identifier may be a placeholder, or it may point to a closed-access copy of the data object, or a web page describing the data object. When I cite the identifier, the identifier crosses its own identifier curation boundary: it has been published. At the moment, it will not resolve successfully to the unpublished data object; but I have made a commitment that it will eventually resolve successfully.

We have just determined that we persistent identify things that cross their curation boundary. To minimise disruption, we associate an identifier with a thing (D):

- After we have decided to move the thing across its curation boundary (B)
- Before we move the thing across its curation boundary (E)
- Before the identifier crosses its own curation boundary (F)

So we link identifier to referent before we publish either; but we can publish the identifier before the referent. Ideally they are both published at the same time.

6 Identifier resolution

If an identifier is associated with a concrete digital object (e.g. a copy of a file at a specific location), then resolution should return information on how to access the file (e.g. a URL for a file on the web).

If an identifier identifies a more abstract entity, such as a work, version, presentation, or aggregation, then the identifier manager has a choice as to whether to resolve to information on a concrete digital object (a *concrete resolution*), or to a presentation corresponding more closely to the abstract entity (an *abstract resolution*). For example, an identifier for a work may resolve:

- to a listing of all available copies of the work (in its various versions and manifestations)—leaving it to the user to decide which concrete digital object to access;
- directly to a particular copy of the work (e.g. a URL for a PDF manifestation of the latest version of the work);
- to a bibliographical citation of the work (which corresponds to the abstract concept more closely than a listing of copies does);
- to a listing of the identifiers of the available versions of the work—leaving it to the user to resolve the identifiers and navigate until they find an appropriate concrete digital object.

There are motivations for both concrete and abstract resolutions of identifiers for abstract entities. Concrete resolutions correspond to non-specialist users' expectation of hyperlink resolution in general. Abstract resolutions represent the abstract referent more accurately, and do not allow misinterpretation of the identifier scope; but they make access to concrete objects more cumbersome, particularly for machine-to-machine operations. The decision on how to resolve identifiers for abstract entities depends on the likely usage scenarios for the identifiers. In cases where a number of resolution behaviours are required, an identifier system could provide different resolution services for different purposes.

Because different resolutions present users with different digital objects (e.g. metadata objects, abstracts, different presentations of content), identifier managers should take care to differentiate these resolutions of the identifier from the thing actually identified by the identifier.

Example: An identifier resolves to a service call to an abstracting service on a document. The identifier manager needs to establish, and communicate to the user, what is being identified.

1. Identifier identifies the service transformation of a digital object (the abstract of the document). Any hyperlinks in the presented abstract to the source document must make clear that the source document has a distinct identifier from the abstract being viewed.
2. Identifier identifies the source digital object (the document itself). The abstract is presented to the user only as a preview of the thing identified. Access to the source document itself should also be provided, if possible. The presentation must make clear that what the user is seeing is only a preview, and not the actual document identified.

7 References

[1] PILIN Ontology for Identifiers and Identifier Services. Forthcoming.

[2] International Federation of Library Associations and Institutions 1998, Functional Requirements for Bibliographic Records (FRBR), http://www.ifla.org/VII/s13/frbr/frbr.htm

[3] Persistence of Identifiers Guidelines: Association with Things under one's Control, hdl:102.100.272/V89DC0DQH http://resolver.net.au/hdl/102.100.272/V89DC0DQH

[4] Treloar, A., Groenewegen, D. 2007, ARROW, DART and ARCHER: A Quiver Full of Research Repository and Related Projects, *Ariadne* Issue 51,
http://www.ariadne.ac.uk/issue51/treloar-groenewegen/

[5] Treloar, A., Groenewegen, D., Harboe-Lee, C. 2007, The Data Curation Continuum, *D-Lib Magazine* 13(9/10), September, http://www.dlib.org/dlib/september07/treloar/09treloar.html, doi:10.1045/september2007-treloar

[6] Staples, T., Wayland, R. & Payette, S. 2003, The Fedora Project: An Open-source Digital Object Repository Management System, *D-Lib Magazine* 9(4), April,
http://www.dlib.org/dlib/april03/staples/04staples.html, doi:10.1045/april2007-staples

Copyright © Monash University

This work is licensed under the Creative Commons Attribution-Share Alike 2.5 Australia License. To view a copy of this license, visit http://creativecommons.org/licenses/by-sa/2.5/au/

This work was created as part of the PILIN project. The PILIN project is funded by the Australian Commonwealth Department of Education, Science and Training, (DEST) under the Systemic Infrastructure Initiative (SII) as part of the Commonwealth Government's Backing Australia's Ability – An Innovation Action Plan for the Future (BAA) under the ARROW Project.