

Persistent Identifiers

Introduction

This module is to familiarise researchers and administrators with persistent identifiers as they apply to research. It gives an overview of the various issues involved with ensuring identifiers provide ongoing access to research products. The issues are both technical and policy; this module focuses on policy issues. Technical details on identifier solutions, and the ANDS *Identify My Data* product in particular, are available separately from the ANDS web site at <http://ands.org.au/services/identify-my-data.html>.

After going through this module, you will be aware of data provider (researcher) or data publisher (institution) responsibilities for managing identifier persistence. You will be aware of the services ANDS provides and how they will support your use of persistent identifiers.

This module goes through the same issues as the introductory ANDS module on Persistent Identifiers (<http://ands.org.au/guides/persistent-identifiers-awareness.html>), but in more detail. The introductory module is not a prerequisite for this module.

Why persistent identifiers?

The current need for persistent identifiers came out of problems with the way URLs were used in the early days of the World Wide Web. Originally, URLs were understood to be strictly network locations for digital resources. The URL specified the particular location, on a particular server, from which the resource could be retrieved. The URL could then be given to others, so that they too could access the resource.

As long as nothing changes about the way the data is accessed, this works fine. In the long term however, this arrangement has proven to be fragile: URLs a year later, let alone five years later, often no longer work. (This phenomenon is known as 'link rot'.) Studies have found half the URLs in scholarly publications will fail after a period of seven to ten years (See: <http://doi.acm.org/10.1145/602421.602422>, <http://arxiv.org/abs/cs/0511077v1>.)

There are several reasons for link rot of data URLs in particular:

1. Data has a life cycle, which involves it going online, and eventually going off-line (For more on the data life cycle, see e.g. <http://resolver.net.au/hdl/102.100.272/6R22YGTRH>, Appendix B.) Links to data may fail simply because the data is no longer online. This can happen for a number of reasons, such as:
 - A dataset is published online. After ten years, the data no longer of interest, and is taken off-line to make room for more recent data.
 - A dataset is published online at an institution by a researcher. The researcher retires, dies, or changes institutions. The data eventually drops off-line, as no-one is there to advocate keeping it online.
 - Data is lost in a crash, and there are no backups.
2. Data needs to be maintained proactively if it is to stay online. However, maintaining the data can involve shifting it to a different server, or a different directory structure on the same server, as part of its normal life cycle. Any of the following changes will also mean the old URLs will fail:
 - Data is moved from a collaborative lab to an institutional repository.
 - The departmental website directory structure is rearranged: subdirectories are created for each research group.
 - The institutional repository is changed to a different platform, with different queries to retrieve documents.



3. URL domains themselves may become inaccessible, because the physical server is retired, or because the domain registration expires:
 - Conference publications are published on a URL domain specifically for that conference; the website registration expires after five years.
 - Content is published through a commercial web provider. Five years later, the provider goes out of business, and no longer hosts content.
 - The lab web server is upgraded to a new machine, with a new web address. The content is moved across to the new machine, and the old server address is retired.

You may already have experienced many of these situations. When research outputs are published online by an institution, users expect that the outputs will be well-managed, so some of the pitfalls given above should be avoided as a matter of good practice—for example, losing data without backups, or having the web domain expire (though institutions themselves can amalgamate or be disbanded). However, some of the situations listed above will still happen if the outputs are well-maintained; in fact some need to happen when upgrading systems, to make sure the outputs remain accessible.

This means that a URL in the traditional sense—a network address for a resource—cannot be relied on to provide ongoing access to that resource. Persistent identifiers are an attempt to deal with that problem, for the subset of data online that is well-managed. (If the data is not well-managed, there is no guarantee it will stay accessible anyway, so there is no point having a persistent identifier for it.) The definition of a URI (Uniform Resource Identifier, not to be confused with a Uniform Resource Locator, or URL) now acknowledges this issue, and is no longer bound to a specific network address for a resource (<http://www.ietf.org/rfc/rfc3986.txt>). (In this document, we distinguish between URLs used as locators, and URIs used as general identifiers.)

Managing data online includes managing the persistent identifier for the data, so that it continues to provide information about whatever it identifies—no matter where it is stored online, and no matter where it is along its life cycle. In fact, the persistent identifier can be used to obtain meaningful information about a resource, even if the resource is no longer online. So long as anything contains an identifier for a resource (including a citation in print), users should be able to find out what the resource was.

Concepts

The motivation for persistent identifiers has been given in terms of broken URLs. A persistent identifier can be something distinct from a URL, but it can also be a particular kind of URL that is not strictly bound to a specific server or filename. Moreover, not all identifiers need be persistent. And not all identifiers need to act as hyperlinks which can download digital objects in a browser. In fact, not all identifiers need to identify online digital objects at all—which is why persistent identifiers can still be useful even if the object identified is no longer online, or never was online.

There is confusion about identifiers, when they are persistent, and what the difference is between identifiers and URLs used as locators. To clarify the underlying issues, we give definitions of the concepts involved as we will use them. Nothing prevents a URL from being used as a persistent identifier—but it needs to be managed specifically as a persistent identifier. We will see what that means below.

What is an Identifier?

In the strictest terms, any name associated with (*identifying*) a particular thing is an *identifier*. A *name* is merely a label that is understood in a particular *context*. A *label* is a string of characters (or more generally, a symbol). The same label in a different context is treated as a different name, and can be used to identify something else. For example:

- '12' is a label.
- '12' in the context of Arabic numerals is a name.
 - '12' in the context of the Internet Movie Database (IMDb) is a different name, disambiguated through the context of the search service using those names, <http://www.imdb.com/find?s=all&q=12&x=0&y=0>.
 - '12' in the context of ANDS PIDS is a third name, which can be disambiguated by naming the context



as well, as Handle '102.100.100/12'.

- Associating those names to different things gives different identifiers.
 - In the context of decimal numbers, '12' identifies the number 10+2.
 - In the context of octal numbers, '12' identifies the number 8+2.
 - In the context of movie titles, '12' identifies at least five movies released in the past decade. Identifiers are often required to be *unique* and unambiguous. To deal with this, the IMDb adds year numbers and ordinals to the label, to make it a unique identifier; e.g. '12 (2003/II)' (<http://www.imdb.com/title/tt0383166/>)
 - In the context of ANDS PIDs, '12' identifies a particular TARDIS crystallographic dataset: 'hdl:102.100.100/12'.

With this broad definition, 'Fred Johnston' or 'the book to the left of the red one' are identifiers, in the same way that 'http://www.example.com/a.pdf' or 'Record Number #8723837' are. They are all doing the same thing: they are using a name to identify a thing. There is no requirement that the thing is online, or that the identifier is usable in a browser. Identifying things does not depend on the identifier being a hyperlink; we can use a name to refer to Fred Johnston (and know who we mean depending on the context), whether or not there is a resource online corresponding to Fred—and without confusing Fred with that online resource: Fred is not his website.

For online identifiers, different identifier systems provide different contexts for the names they give access to. For example, the label '102.100.272.7/XYZZY' gives different identifiers within the Handle System and HTTP: the different systems associate the same string with different data. The presentation of an identifier normally indicates its context as well as its label, as we saw with ANDS PIDs; so the handle 'hdl:102.100.272.7/XYZZY' is a distinct identifier from the URL 'http://102.100.272.7/XYZZY'.

Data and Identifier life cycles

Data has a life cycle:

1. It is *created*.
2. It is edited and worked on for a period by the data providers, until it is ready for release.
3. It is made available (*published*) to users who cannot edit it (at its online location).
4. It is cited, accessed, and re-purposed by other users.
5. It is revised (*managed*) and updated (re-uploaded) by the data providers.

Note: other users may also publish the same data online; but the original data providers do not have control over those instances of the data (and they will not necessarily be updated in sync). So they are considered distinct data objects.

6. It is no longer in current use, and is taken off-line (*archived*); it may still be made available on request.
7. It may finally be *destroyed*.

This life cycle applies to any online digital object. Identifiers of the kind we are discussing are themselves digital objects; so they are subject to the same life cycle, and different stages of those life cycles are enacted through different identifier services: identifiers are *created* ('minted'), *updated* (managed), *published*, *used* online (mainly *resolved*), *archived*, and *deleted*.

What is Identifier Resolution?

An identifier can be used to name a thing, so that someone who sees the identifier knows who or what is being referred to. If they do not already know what the name refers to, they should be able to work this out. In the broadest sense, *resolving* an identifier is getting information about the thing identified, so that someone can work out what the thing actually is.

When resolution is an online service, it returns metadata about the object named. That metadata can be used to distinguish the object being identified from any other objects. This means that digital identifiers are not restricted to identifying online content. For instance, 'http://person.example.com/johnston/fred' can be resolved to a page



listing Fred's contact details and description. The identifier is identifying Fred himself, and it resolves to metadata which distinguishes Fred from anyone else; this does not mean that 'http://person.example.com/johnston/fred' downloads a copy of Fred. (Again: Fred is not his website.)

The main way of working out what a URL identifies, nonetheless, is doing exactly that: clicking it, and downloading what it identifies. Getting access to the thing identified (or a representation of it), as opposed to merely metadata about it, is *retrieval*. A URL is normally used for retrieval, but the actions of resolution and retrieval can be separated. If the object goes off-line, clicking the URL can instead show information about what the object used to be: instead of getting a PDF, we get bibliographic details and an abstract. The URL is then providing resolution, but it is no longer providing retrieval. Repositories often make this distinction: the URL published for a resource takes you to a splash page describing the resource, including a separate link to download the resource itself. The splash page still counts as a resolution of the identifier: it tells you what you are about to download.

The distinction made here between resolution and retrieval is further elaborated at http://www.oasis-open.org/committees/document.php?document_id=29430&wg_abbrev=xri.

What is a Persistent Identifier?

We have already used a first cut notion of what a persistent identifier is: an identifier which can be resolved to an appropriate representation of the resource (including downloading the resource itself, if it is online), and which can be updated when the resource changes location or goes off-line, so it continues to resolve appropriately to a representation of the same resource. In other words, a persistent identifier continues to give intelligent information about a single object being identified, whatever happens to that object.

Two-tiered resolution

Several identifier schemes rely on a two-tiered system of indirection to achieve persistence: the identifier is resolved to the resource's current URL (as a 'locator'), which is then used to retrieve the resource. This is conceptually possible because the current URL of a resource is distinguishing metadata about the resource, so an identifier can resolve to a URL. Under this arrangement, the current URL may change as the object moves, but the identifier itself does not have to—so long as the URL resolution is kept up to date. Users can keep using the original identifier to access the resource, and the identifier is managed to keep its resolution up to date.

The two-tiered system introduces the possibility of using identifiers that are not URLs. A URL can be persistent by having it aliased to whatever the current URL of the resource is, and updating that alias; that is the model that HTTP redirection is based on. But a resolver—a service performing identifier resolution—can also take an identifier which is not a URL, and map it to the current URL of the resource. So long as the resolver is in a known place, the identifier itself does not need to be a URL. In fact, some communities prefer identifiers which are not URLs, to avoid confusion between the persistent identifier and the current URL.

This defines how technology can keep an identifier persistent; but it only tells part of the story. The real challenge is in the policy and the expectation of trust that surrounds a persistent identifier.

Identifier trustworthiness

Having an identifier whose resolution can be updated is little use, unless its resolution actually does get updated when it needs to be, and without disrupting outside users. This expectation means that the *identifier manager* makes an undertaking to the *end user* that they will do their job in keeping the identifier up to date.

Such an undertaking has to be meaningful, as external users will come to rely on the identifier. (If they cannot rely on it, there is no point in having a persistent identifier to begin with.) It is commonly assumed that identifiers will be maintained indefinitely, but such an open-ended undertaking is meaningless: few institutions can reasonably guarantee that their persistent identifiers will still work after a thousand years. Instead, the undertaking is like a contract, which means it makes more sense to guarantee the identifier will be maintained for a well-defined, fixed time span.

While the identifier manager can make such an undertaking, the end users of the identifier have to build trust: they should not have to accept the persistence undertaking without question and should have actions of recourse if the undertaking is not fulfilled. This means that accountability measures need to be built into the identifier system, before users can rely on the persistence of the identifiers. For example, if the undertaking is to persist identifiers for a fixed time span, users must be able to discover how long that time span is. Details on who is



responsible for the identifier (*authority data*) need to be accessible: that way, if the identifier does fail to be updated, users have a contact point to seek redress. The reliability of identifier services has to be established, as a kind of Service Level Agreement: users should know how much of the time they can expect identifiers to be inaccessible, given that identifiers are part of the digital infrastructure for accessing data.

When to use persistent identifiers

Persisting an identifier makes a commitment and entails costs in time and resources. (See discussion in the PILIN Persistence of Identifier Guidelines, <http://resolver.net.au/hdl/102.100.272/V89DC0DQH>). Normally it is self-defeating to promise to persist an identifier for every data artefact ever generated by a project. Objects already have identifiers that are not persistent (names, serial numbers, file locations); the extra effort to persist the identifier should only be made if there is a good reason to. In particular, it only makes sense to have a persistent identifier for a digital object if it will be available long enough that its default URL will likely change—either because the data object moves around, or because of periodic server upgrades. The former implies a lifespan of months rather than days, and the latter a lifespan of years. It is also more important for a publicly available object to have a persistent identifier than for an object available to a small number of users: those users can be informed individually if an object changes location, and the old link breaks; but that is much more difficult if an identifier has been included in a journal article.

Working out which data objects to prioritise for persistence needs input from the user community, as well as the parties responsible for putting the data online. (These stakeholders are discussed further below.)

Questions to be addressed include:

- What kinds of data are likely to stay online long enough?
- What kinds of data does the provider intend to keep online long enough?
- What kinds of data are likely to be linked to, with links that must remain functional?
- How long will data likely stay online?
- How will updates be provided on where the data moves to?
- What will happen after data goes off-line?

It often makes sense to keep persistent identifiers more abstract than concrete digital filenames; depending on how the identifier will be used, the one persistent identifier can encompass various drafts and revisions of the object, various formats and presentations of the object, and/or various file locations. Again, these decisions can only be made in consultation with the people creating and using the data.

Persistence can apply to any aspect of the identifier. What we have discussed so far is the persistence of the resolution of identifiers. But persistence also applies to the metadata associated with identifiers; to the services used to access the identifier (e.g. the distinct identifier resolvers); and even to the name used in the identifier. A hyperlink to a resolution call on a persistent identifier will still end up breaking if the persistent identifier is maintained, but the resolver goes off-line

Technologies

There are various technologies available to realise persistent identifiers. We briefly discuss a few major technologies.

HTTP URIs as persistent identifiers

As seen, the need for persistent identifiers arose because URLs kept breaking, as data moved or went off-line. This does not mean that identifiers using HTTP are inherently unable to persist: the failures of persistence we have gone through are much more issues of policy than technology. HTTP servers can alias URLs to redirect to other URLs: this is already enough to implement a two-tiered redirection strategy which underlies persistent identifiers, so that the original URL stays the same. Moreover, work on the Semantic Web (see for example http://en.wikipedia.org/wiki/Semantic_Web) has dispensed with the notion that URLs can only be used to address online resources: the Semantic Web uses URLs to refer to off-line abstractions as well. For that reason, URLs are now generalised to URIs: Universal Resource Identifiers, rather than mere Locators.



This distinction however is recent, and does not match common user expectations. Moreover, there is no built-in provision in HTTP for identifier services other than resolution (such as retrieving authority metadata, or policy guarantees). For that reason, alternative identifier schemes distance themselves from HTTP URIs in two ways, as described below: branding the HTTP URI as persistent, and using a URI scheme other than HTTP.

HTTP URIs containing persistent identifiers

Branding an HTTP URI as persistent preserves the advantage of using the HTTP protocol, which is now universal through the Web; but it makes the undertaking of persistence clearer to users. The most prominent instances of such schemes are PURL (<http://www.purl.org/>), which has been used by the National Library of Australia, and ARK (<http://tools.ietf.org/html/draft-kunze-ark-15>), at the California Digital Library.

To illustrate: an ARK looks like 'http://example.org/ark:/12025/654xz321'. This is clearly an HTTP URI, but it embeds a persistent identifier (ark:/12025/654xz321) inside a URI for an ARK resolver (here http://example.org). While 'ark:/12025/654xz321' is itself an ARK identifier (a name associated with a thing), the URI which includes a resolver, 'http://example.org/ark:/12025/654xz321', is considered fully equivalent. The structure of the URI, including the 'ark:/' component, indicates that this is intended to be an ARK, and not just any HTTP URI. PURL takes a similar approach, e.g. <http://purl.oclc.org/keith/home> is an HTTP URI which includes the persistent identifier 'keith/home'.

Distinguishing the persistent identifier from the HTTP service acting on it opens the door for other types of identifier services (such as policy guarantee). For example, appending '??' to an ARK HTTP URI, e.g. 'http://example.org/ark:/12025/654xz321??', is a request for the details of the persistence policy for the identifier.

Non-HTTP URIs

The second way to distance persistent identifiers from HTTP URIs is to have a digital identifier scheme dissociated from the HTTP protocol. This forestalls the conflation of resolution and retrieval: the identifier can tell you about what is being identified, without downloading it from a particular location. One such identifier scheme is URN (<http://tools.ietf.org/html/rfc1737>), which was kept distinct from URL in the '90s, but failed to get wide adoption. We already saw that ARK identifiers could be treated as just names (ark:/), without any explicit association to a resolver. The most prominent system is Handle (<http://www.handle.net>), with DOI (<http://www.doi.org>) as a major profile; e.g. 'hdl:102.100.100/12', 'doi:10.3998/3336451.0003.204'. Other widely used identifier schemes are LSID (<http://www.omg.org/cgi-bin/doc?dtd/04-10-08>) in the Life Sciences, and XRI (<http://www.oasis-open.org/committees/xri>) used mainly for identity and data exchange.

Handle is the system underlying the ANDS *Identify My Data* product.

Handle has its own protocol and resolution infrastructure, independent of HTTP. Handle resolvers retrieve a metadata record about the thing identified, rather than just a URL for the thing identified. This metadata record consists of a number of fields, possibly including a URL field. This means the metadata record can be used for retrieval, but is not limited to that role. Because the URL is only one type of metadata field, and Handles themselves do not look like the URLs they redirect to, the Handle is distanced from the URL.

That said, an HTTP resolver still has to be provided for such identifiers to be usable on the Web, and all recent schemes have such resolvers, in ways similar to ARK. Handles for example become HTTP URIs by prefixing a Handle resolver, as in <http://hdl.handle.net/102.100.100/12>, or <http://dx.doi.org/10.3998/3336451.0003.204>. But the identifier is not tightly bound to that resolver, and can be used with any number of other resolvers, or even other services. For example, <http://resolver.net.au/hdl/102.100.100/12>.

Responsibilities

Keeping an identifier persistent is not something that an identifier provider like ANDS can do on its own. The identifier provider has to enable multiple types of user to work together, to put together a plan on how best to ensure persistence, and to make persistence happen. We list the various parties involved, and their responsibilities; some of this discussion can be seen in the PILIN Persistence of Identifiers Guidelines, <http://resolver.net.au/hdl/102.100.272/v89DC0DQH>. The notion of publishing an identifier, like publishing data, involves a change in the type of user who gains access to the identifier, and how they gain access to it; we provide a general model for publishing identifiers, in terms of the Curation Boundary (see below).



Roles

We have seen that the *identifier manager* undertakes to the *end user* to maintain the persistent identifier. The identifier manager publishes the identifier, and so has institutional responsibility for it. The identifier manager is often the same as the *identifier provider*, who provides the services for managing and accessing the identifier—so the identifier manager sets up the identifier management system, and also issues updates to the system. In ANDS' case, they are distinct: for the ANDS Persistent Identifier Service, ANDS takes on the identifier provider role and the service's consumers assume the identifier manager role.

To maintain the identifier, the identifier manager has to coordinate with the *data manager*, who is responsible for keeping the resource identified online. The data manager in turn is publishing data on behalf of the *data provider*, who is typically the researcher.

If the data manager moves the resource to a new address, or takes the object off-line, the identifier manager has to be aware of this and update the identifier accordingly. The identifier manager and the data manager are also not necessarily the same person: the identifier may be managed by a different party from the data. For example, a department has one contact point for issuing updates to ANDS, but the updates originate in several separate labs in the same department: the labs have their own data managers, who are coordinating with the department's identifier manager, to communicate all the needed updates to the identifier provider.

Ensuring that the identifier is updated smoothly requires coordination between the identifier manager and the data manager. The more separated the identifier manager is from the data manager—especially if they belong to different institutional structures—the harder such coordination is to realise. This has been called the Our Stuff vs. Their Stuff problem: it is harder to persist identifiers for data that is under some other institution's control ('Their Stuff'), whereas managers working under a single authority ('Our Stuff') can more easily co-ordinate with colleagues and put the necessary procedures in place.

The data provider is also involved in working out the best policy structure for identifiers. The data provider has the best notion of how the identifier will be used by the user community, of how long the data will be useful, and which parts of the data identifiers should point to. (This involves information modelling: see the section on policies below.) The identifier manager is responsible to the data provider to keep their data accessible, as much as they are to the end users.

The responsibilities of the identifier manager mesh with those of the data manager, and it is important to plan them out to avoid clashes.

Curation boundaries

The notion of a Curation Boundary (http://www.valaconf.org.au/vala2008/papers2008/111_Treloar_Final.pdf) is useful to understanding how identifiers interact with end users: it defines what it means to *publish* data. The curation boundary model defines publication to be when the data is exposed to people not involved in managing the data, i.e. when it crosses a *curation boundary*. For example, a research collaboration can have geographically widespread access to a resource, and edit it frequently as part of their work. Once a copy of the resource is available outside that group, to users with read-only access, it is expected to be reasonably stable, and straightforward to locate. As part of its stability and to establish trust, any modifications to the resource should be documented, as change metadata. The resource is then published. (Compare the distinction between alpha releases of software and official software distributions.)

A persistent identifier makes more sense for data which has crossed the curation boundary (is publicly cited by a large number of people, is stable, and will change network location only as a well-defined object). It is less urgent for data still in flux and only accessed by a small number of active users. For that reason, the responsibilities of the identifier manager are greater once the object crosses the curation boundary.

Policies

A range of policies need to be put in place to establish identifier persistence. These policies are described at some length by the PILIN project at http://linkaffiliates.net.au/pilin2/outputs/outputs_guidelines.html and <http://dx.doi.org/10.1045/january2009-nicholas>, and are also expanded on in the ANDS Expert Module on persistent identifiers (available soon). Here we summarise the kinds of policy considerations introduced.

The undertaking of identifier persistence needs policy infrastructure to be trustworthy. In particular, there need to



be policies on what *authority metadata* to make available for identifiers—that is, metadata on who created the identifier, and who is responsible for maintaining it, so that the claims of identifier persistence can be held accountable, and there is someone to query if things go wrong. Moreover, there should be a policy on what *time span* to guarantee the identifier will persist for: users can plan how they will use identifiers more realistically if they know they are guaranteed for a fixed time span, than if they are vaguely assured the identifier will be maintained indefinitely. There should be a policy on the *escalation pathway* for queries about out of date identifiers: if a persistent identifier fails, users must know who to ask to get the problem fixed (and who else to ask if they get no response); without that ability, users cannot trust that unforeseen identifier disruptions will be dealt with, and the identifier will remain persistent. For example:

- The (hypothetical) PID ‘123.456.789/7373’ was created and is maintained by the authority ‘123.456.789/8383’, and last updated on 2009–03–01 (authority metadata). The persistent identifier for the identifier authority, ‘123.456.789/8383’, resolves to the current contact details for the ANU Ectoplasm Lab, who are responsible for keeping the PID current.
- The ANU Ectoplasm Lab undertakes to keep the PID persistent (regardless of whether what it points to is online), for at least the next ten years (time span).
- If the PID is out of date, you can query, in order of priority, the ANU Ectoplasm Lab manager; the ANU Metaphysics Department manager; the Secretariat of the Dean of Non-Humanities; the ANU Vice-Chancellor’s Office.

Finally, there should be a policy on what *migration plans* should be in place if the identifier authority is disbanded: users should have the assurance that, if the current authority can no longer maintain the identifier, someone else will, so that the identifier will remain persistent for the time span originally promised. (That is another reason why such time spans should be fixed rather than indefinite: the longer they are, the harder it is to plan such arrangements.)

- If the Ectoplasm Lab cannot maintain its PIDs over the next ten years, it will transfer their management to the Metaphysics Department, who may delegate some other party to maintain them.

For the guarantee to be meaningful, there also needs to be a policy on *what things should be identified* by persistent identifiers. As discussed in the section ‘When to use persistent identifiers’ above, identifier managers should only commit to persistent identifiers for things they can sensibly keep identifying. Working this out requires *information modelling* of the domain over which the identifiers will be applied, to work out what possible things can be identified, and which things should be prioritised for persistence. The modelling is discipline-specific, and crucially depends on information from the data provider, about what resources they are providing and what their dependencies are.

- The Ectoplasm Lab will publish PIDs for any published research outputs of the Lab since 2006, and any datasets they generate which are cited in research. They will publish distinct identifiers for different versions of outputs and datasets, and an identifier for the latest available version, but not for different formats or local copies.

On the technical side, there also need to be policies and workflows on how to *integrate* persistent identifiers into the day-to-day management of data resources. This needs to be done to minimise the risk of identifier resolution being disrupted, or becoming out of sync with the data.

- Any requests to retrieve published resources locally must be done through the PID, and the repository interface will not allow direct access via the local URL. Any workflow to move published datasets between servers must be invoked against the PID, and must update the PID resolution in the process of moving the resource.

Identifiers need a policy for what *labels* they can legitimately use. Meaningful labels are much easier for humans to remember, but they pose a risk to persistence because their meaning can become obsolete: identifier providers and/or managers need to weigh that risk up.

- The Ectoplasm Lab will use sequence numbers for its labels, ordered by time of assignment.

Finally, there needs to be a policy on how to *cite* identifiers. URIs may already require normalisation when used by applications, because of Unicode characters, casing whitespaces, or placement of slashes (see <http://tools.ietf.org/html/rfc3986#section-6>). A similar policy may be needed for identifiers published by an organisation.



How to cite identifiers becomes more complicated still when the identifier is not inherently an HTTP URI. Should such identifiers be presented as raw names, such as 'hdl:102.100.100/12'? Should they be presented embedded instead in clickable HTTP URIs, as in <http://resolver.net.au/hdl/102.100.100/12>? If they are, which resolver service should be used, and can it be trusted to persist? A persistent identifier URL is not much use if the handle '102.100.100/12' persists, but the resolver <http://resolver.net.au/hdl/> does not. On the other hand, if the identifier is not cited in a resolvable form, the identifier is not presented as dependent on a particular resolver, which removes a difficulty for persistence—but makes the citation much less useful online.

- The Ectoplasm Lab will cite all its PIDs in the HTTP URI form 'http://resolver.net.au/hdl/x/y'.

ANDS Persistent Identifier Web Service Policies

As a utility service to be used across disciplines and institutions, ANDS does not impose a large number of policies on its identifiers: to a large extent these need to be implemented by the projects and institutions who create and maintain the identifiers. For example ANDS undertakes to persist the infrastructure for keeping its identifiers online. But it can hardly undertake to update your resource URLs for you: since you control the resource URLs, it's your responsibility to update them if they change. Likewise, ANDS cannot decide for you what things should be identified persistently, or what your identifier workflows should be. But through its simple HTTP-based interface, it ensures that identifier services can be integrated easily into existing data management workflows.

On the other hand, ANDS does fix certain identifier policies by having a running service for creating and publishing identifiers. In particular:

- ANDS PIDs are Handles, created currently on a single identifier management system common to all service users.
- ANDS PID labels are arbitrary rather than meaningful: they are generated as sequence numbers, and users cannot pick their own labels.
- ANDS maintains some authority metadata for its identifiers: each identifier has an owner named through the AGENTID field, and details of that owner are given in a Handle pointed to by that field.
- ANDS currently allows only textual descriptions and URLs to be added by users to PIDs.

The ANDS Identify My Data Service

ANDS provides a persistent identifier product called *Identify My Data*. It uses the established Handle technology to maintain persistent identifiers, but adds an extra access layer to simplify authentication and improve access.

Technology

Handle is a lightweight system, which maps identifiers to an arbitrary number of metadata fields. This allows two-tiered resolution to the current URL. Identifier managers can also store any number of other fields such as authority metadata, resource descriptions, and policy information. The ability to store multiple URLs for a Handle can also be used to support richer resolution services, such as Appropriate Copy resolution.

The Handle system is very robust, with extensive provision for data mirroring and reliability; it is widely used among repositories, including DSpace (where it is included in the repository distribution) and Fedora, as well as in publishing through DOI.

Services provided

ANDS Persistent Identifier Web Service provides the following administrative services as web services:

- **mint**: Creates a new handle, containing either a URL, a text description, or both.
- **addValue**: Adds (another) value to an existing handle, of either URL or text description type.
- **modifyValueByIndex**: Changes an existing value associated with a handle.
- **deleteValueByIndex**: Deletes a value from an existing handle.
- **listHandles**: Provides a list of handles owned by the person asking.



It provides the following non-administrative services:

- **getHandle:** Provides all metadata fields associated with a handle.

These web services are provided by a Tomcat server. Applications can call these web services to create and manage Handles. Further documentation is available at: <http://ands.org.au/resource/techdocs.html>

Documentation for ANDS *Identify My Data* is available at <http://ands.org.au/services/identify-my-data.html>.



This work is licensed under a [Creative Commons Attribution-NonCommercial-Share Alike 2.5 Australia License](http://creativecommons.org/licenses/by-nc-sa/2.5/au/)