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	2007-09-19	Initial draft for public discussion.	PILIN/R4S8K6DQH hdl:102.100.272/R4S8K6DQH

PILIN Project Guidelines

Considerations for Managing Contexts

This document is a work in progress and may contain open questions not resolved during the timeline of the PILIN project. It represents the thinking of the PILIN team as at December 2007.

To cite the *latest* version of this work use http://resolver.net.au/hdl/102.100.272/N8R5K6DQH

To cite *this* version of this work, use http://resolver.net.au/hdl/102.100.272/R4S8K6DQH

1 Purpose/Issue

This document outlines considerations for parties on how best to manage the contexts for their identifiers. It presents the PILIN model of identifier contexts, and discusses context realisation, context naming, and context scope.

2 Background

This document presupposes the PILIN abstract identifier model:

- An *identifier* is an association of a *name* and a *thing*.
- A *name* in turn is a pairing of a *label*, and a *context* that the label is in. A label is unique in a context.
- Contexts are entities with particular purposes for organising and managing things.

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.

  - Contexts have *owners.*
  - Contexts have identifiers (i.e. a *context name* is associated with the context).
  - Contexts have *policies* applying to the labels in the context.
- Contexts are not differentiated by time, location, specific manager as distinct from owner, or broader domain. So the following are not considered discrete contexts.
  - Context: Names created by Jack for ARROW on 2003-05-21 through myRI protocol.
  - Context: Names created by Jack for ARROW through myRI protocol.
  - Context: Names created in Australian Education and Training through Handle.
- Two identifiers are *equivalent* if they are associated with the same thing.
- Identifiers are managed through *identifier management systems.*
- An identifier management system defines a *concrete context*. The system implements the concrete context's policy. The concrete context identifier differentiates it from other concrete contexts, so the identifier scheme must be included in the identification.
- Any context whose policies and labels are not managed through a concrete system is an *abstract context.* (Typically an abstract context is defined by owner and policy.)
  - E.g. The Monash University Library is an abstract context: it is defined by an owner (Monash University) and a purpose (identifying documents held at the university library). The Handle server hdl:1959.1 is a concrete context, with a Handle-specific identifier; it can be used to realise the Monash University Library context.
- A concrete context *realises* an abstract context if their identifiers are equivalent and their labels are identical. More than one concrete context can realise the same abstract context.
  - E.g. The abstract context "Monash University Library" can be realised through the concrete contexts hdl:1959.1 and http://purl.oclc.org/monash/
- Identifiers in an abstract context cannot be managed directly. They are managed through their counterparts in concrete contexts.
  - E.g. The abstract context identifier "label 93184 in the context Monash University Library" can be managed through the concrete context identifiers hdl:1959.1/93184 and http://purl.oclc.org/monash/93184

- Two identifiers are *homologous* if their concrete contexts realise the same abstract context, their labels are identical, and the identifiers are equivalent. (In other words, the same party registers the same label to do the same job in two different systems.)
- Names are *represented* through encodings of the label and optionally the context name, combined into a single representation.
    - E.g. the name "label 93184 in the context hdl:1959.1/93184" may be represented through the single string "hdl:1959.1/93184"

3 Scope

This document addresses managing contexts for names used in identifiers. The discussion of naming relates to concrete contexts; the discussion of scope relates to abstract identifiers.

4 Guideline: Context Realisation

4.1 Use concrete contexts

From the definitions above, clearly only concrete contexts allow management of the labels in those contexts. If there is no system to manage labels, the labels are not managed. Moreover, only names in concrete contexts are accountable (the system exposes the authority metadata), or actionable (an external service must consult a defined system to work out what to do with the name). So only concrete contexts are usable by computer systems.

4.2 Model abstract contexts

That said, the concrete identifier is by definition specific to a host and protocol, and will not survive changes in host and protocol. (It will survive changes in host ownership, if proper provision for transfer has been made.) Identifiers are usually replaced by equivalent identifiers with the same owner. We can uncouple the intent to identify a thing from the specific system used to identify it, by tying that intent to identify to an abstract identifier: the concrete identifiers that systems use are then realisations of the abstract identifier.

Abstract contexts underlie the use of names for things in human language: people usually assume shared knowledge rather than dictionary lookup when using words. This shared knowledge also drives the typical replacement of one identifier with another: the manager knows they point

to the same thing before registering the equivalence (if they register it at all).

Because abstract contexts constitute shared knowledge, it defeats the purpose to manage them explicitly: that turns them into concrete contexts by definition, and one more registry to keep track of. It also defeats the purpose to represent identifiers outside the system using abstract context names: such representations are neither actionable nor accountable, even if they are more persistent than concrete identifiers. And though abstract identifiers have persistence of association, they do not have persistence of actionability (which matters more to external systems), because they are not actionable. Distinct registries of identifier equivalence are the preferred approach to dealing with this issue.

But abstract contexts should be modelled by parties managing identifiers, so that they can work out what policies, services, and conventions apply to the specific identifier system in use, and what will apply to any identifier system they will deploy. The latter policies, services and conventions are not specific to a concrete context, so they are modelled as belonging to the abstract context.

4.3 Register concrete contexts

Often in representing a name, the name context is left implicit. For example, filenames are usually given on their own, rather than with their directory path (let alone the network path), as long as the default assumption (current directory, local machine) is valid. But the context of an identifier must be explicit for an external service, which cannot make such default assumptions.

External services need access to an identifier management system to find out information about a given identifier; and they find out about identifier management systems through a registry of identifier management systems. It follows that for identifiers to be actionable through external systems, not only the identifier but the identifier system must be registered. Computer-actionable identifiers routinely register contexts: e.g. DNS for URL, Global Handle Registry for Handle, Name Mapping Authority for ARK, root authority for XRI.

This has the potential for infinite recursion, so the PILIN ontology provides for the predefined context “known identifier systems” as a root. In current web architecture, that context corresponds to the IANA-registered URI schemes. This means that in practical terms, an identifier actionable through the web must be representable with a URI, and that its context have a name covered by an IANA URI scheme. This requirement is problematic for newer identifier schemes (such as Handle), and there are several strategies of addressing it:

- Try to register the scheme with IANA (or failing that, with the info scheme within IANA): hdl:/x/y, info:hdl/x/y

- Present as actionable not the identifier itself, but service calls to the identifier (e.g. http://hdl.handle.net/x/y)
- Do not attempt to make the identifier actionable—which drops the need for the identifier to be represented as a URI (e.g. Handle: x/y)

4.4 Realise redundantly

More than one concrete context can realise an abstract context: a university library may use Handle, PURL, and normal URL to refer to the same things, and all three can be realisations of the same underlying abstract identifier. The advantage of realising an identifier in more than one identifier management system is that different systems may satisfy different requirements, particularly from external systems. If two requirements are not satisfied by the one system, the same identifier may need to be realised in two different systems.

> For instance, a party may maintain both a Handle and a PURL as equivalent identifiers. The PURL provides a more natural fit with Web URL infrastructure; the Handle supports richer metadata, and thus a wider range of identifier services. Alternatively, one identifier may be more persistent, and another more readily resolvable.

In addition, having two concrete contexts for the same abstract identifier adds robustness: if one identifier management system becomes unavailable, the other can still be used.

> Identifier management system deployments are already meant to make provision for reliability and redundant physical hosting; so this is not a compelling argument for identifier system redundancy. It is more appropriate if the two systems are available to discrete constituencies.

There are several disadvantages to realising an identifier redundantly. The two contexts are managed separately, and may become uncoupled; so there needs to be explicit provision for persistence of equivalence if the identifiers are used equivalently, which imposes added overhead. The equivalence of the two identifiers may not be obvious to external systems, which may treat them as if they are identifying distinct things. The identifier owner cannot force one identifier to be used over the other once they are released (even if it indicates one of the two is canonical); so any metadata or other information linked to one identifier may not be recoverable through the other identifier.

4.5 Completeness

A concrete context need not manage all the labels in the abstract context for the concrete context to realise the abstract. A concrete context realises an abstract context even if the concrete context manages only

one label from the abstract context. Often it is impractical to register all the abstract context identifiers in the concrete system. The concrete system may realise only a subcontext (see discussion below); there may not be the necessary data source or human resources available to complete the registration; or the identifier manager may decide that only some identifiers need to be realised in a particular way.

That said, the natural expectation is that a concrete context realises the complete corresponding abstract context. If I know that Monash University Library uses a Handle server with barcode numbers for labels, and I discover a barcode on a library book, I would expect the corresponding Handle to resolve. Since barcodes are themselves a concrete context, the expectation is really that, if two concrete contexts contain homologous identifiers, then all their identifiers should be homologous. If hdl:1959.1/93184 and http://purl.oclc.org/monash/93184 are equivalent, and hdl:1959.1/93185 exists, I expect http://purl.oclc.org/monash/93185 to exist and identify the same thing. The expectation is driven by users, and does not reflect a systemic requirement; but ensuring concrete contexts realise abstract contexts completely prevents any such confusion.

Conversely, a concrete context may realise several abstract contexts—that is, the various abstract context owners have decided to share the same system for their identifiers. Depending on how contexts are delineated, this is a commonplace occurrence; the subunits of an organisation for example usually share the same email identifier system. The larger the abstract context, however, the stronger the expectation that the concrete context is specific to that abstract context. One would not expect an individual staff member at RMIT to have their own email identifier system; but one would expect RMIT and UTS to have separate email identifier systems. Again, this expectation is driven by users and not systems; having semantically opaque names for contexts (as described below) can help temper the expectation.

5 Guideline: Naming Contexts

There is a longstanding expectation that contexts for names are semantically transparent labels. The use of DNS instead of IP numbers is emblematic of this expectation. Transparent context names avoids the overhead of a registry of contexts (such as DNS supports) in context discovery, which would be necessary if contexts were arbitrarily named but still needed to be discovered. While the registry of contexts exists, not all business processes need to orchestrate it with the registry of identifiers or of content. Web browser users for example do not need to do a Whois query whenever they access a blog.

Meaningful context names also have the known advantages of meaningful names in general: they allow the associated context to be determined by

inspection, they are easy to type and remember, and they allow branding. Since parties have less direct control over registries of contexts than registries of identifiers or content, meaningful context names are usual.

However, meaningful contexts names are just as fraught for persistence: contexts may also change in time (e.g. institutional merger, transfer of control of individual identifiers), and have their own intellectual property issues (e.g. cybersquatting). The use case of a thing moving from one party's control to another is less problematic if the context name is not meaningful: the old context name can be kept without a strong implication that the thing (or even the identifier) is still being managed by the original party.

6 Guideline: Context scope

6.1 Model multiple abstract contexts

An abstract context is defined through owners, purposes, and policies. An identifier manager documenting or planning the identifiers they are responsible for should model whether all the identifiers they own have the same owners, purposes, and policies. If they do not, they likely own several discrete abstract contexts. They should model what those abstract contexts are, in terms of their ownership, purposes, and policies. They should also model the relation between their abstract contexts.

Given the list of abstract contexts they own, managers can then plan how to realise them, using options such as those detailed in "Considerations for Ownership of Identifier Management Systems"—e.g. whether to realise them on the same identifier system or in separate systems, and whether management of the identifier systems will be delegated or federated. The relations between the abstract contexts may constrain how they are realised: two closely related contexts are likelier to be realised through the same system than two unrelated contexts.

6.2 Subcontexts

In modelling abstract contexts owned by the same party, the notion of a subcontext is useful. A context A1 is a subcontext of A2, and A2 is an enclosing context of A1, if:

- The labels belonging to A1 also belong to A2
- The policies applied at A2 also apply to A1

The subcontext may impose its own additional policy rules. A subcontext is distinct from its enclosing context; both contexts may be realised through the same concrete context, or in different concrete contexts.

A context can be a subcontext of another if the ownership, purposes, or policies of the former are more restrictive than the latter:

- Identifier system owners are usually organisations, in hierarchical relation with other organisations (e.g. company/unit, university/faculty). If one context is owned by the organisation, and the other context is owned by a subunit, then the subunit context can be a subcontext of the organisation context (so long as their policies overlap).
    - E.g. BHP needs identifiers for its widgets. The Singapore office will manage some of those identifiers, and the Perth office will manage others, although the company itself ultimately owns all identifiers, and the central office in Melbourne can override identifier management. The Singapore context and the Perth context are subcontexts of the company's identifier context—so long as the Singapore and Perth policies do not contradict the company's core identifier policies.
- Identifier context purposes are defined in terms of what things the identifiers will be associated with. If one context defines what may be identified in general terms, and another is more specific, then the more specific context is a subcontext of the more general.
    - E.g. BHP needs identifiers for its widgets. There is a business need to distinguish identifiers for blue widgets from identifiers for white widgets. The blue widget context and the white widget context are subcontexts of the company's widget context: they do not contradict the enclosing contexts general purpose of identifying widgets.
- Context policies maybe thought of as rules. If two contexts share the same rules, but one context imposes further policy constraints on its labels, then the more restrictive context is a subcontext of the less restrictive.
    - E.g. BHP needs six-letter identifiers for its widgets. The company can use either identifiers prefixed by Q, or identifiers prefixed by Z. The Q-identifiers and the Z-identifiers can be modelled as abstract contexts, with the additional policy constraints that identifier labels must start with Q or Z respectively. The Q-identifier context and the Z-identifier context are then subcontexts of the company's identifier context.

6.3 Realisation of subcontexts

An identifier manager can chose to have a concrete context realise either an abstract subcontext, or an abstract enclosing context.

- E.g. BHP needs identifiers for its widgets. The Singapore office will manage some of those identifiers, and the Perth office will manage others. The identifier manager could choose to deploy one concrete context realising the Singapore identifiers, and one concrete context realising the Perth identifiers. Alternatively, the identifier manager could choose to deploy one concrete context realising the entire company's identifiers. The Singapore and Perth offices would then share access to the concrete context.

A label in a subcontext is also in the enclosing context, by definition. So the identifier manager can choose either context to associate with the label in a name. The two contexts give two different names (the label is being associated with a different context).

- E.g. If BHP realises a single identifier context, the label XYZ gives the name BHP::XYZ. If the Singapore office of BHP realises its own identifier context, the label XYZ can give either the name BHP::XYZ, or the name SINGAPORE-BHP::XYZ, depending on whether the label XYZ should be managed centrally or locally.

If both the subcontext and the enclosing context are realised, then two equivalent identifiers are being maintained autonomously. (In the example above, BHP::XYZ and SINGAPORE-BHP::XYZ are presumably equivalent identifiers, but are managed separately, in Melbourne and Singapore respectively.) This can lead to confusion and lack of synchronisation, and is typically avoided: parties choose to realise either different subcontexts or the one enclosing context for a given purpose, but do not realise both.

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.