



# PILIN Ontology for Identifiers and Identifier Services

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.

## Version History

Version	Date	Status & changes	Expression identifiers
V1.0	2007-12-20	Release	PILIN/ <a href="http://resolver.net.au/hdl/102.100.272/RPLZ54PQH">RPLZ54PQH</a> hdl:102.100.272/ <a href="http://resolver.net.au/hdl/102.100.272/RPLZ54PQH">RPLZ54PQH</a>

To cite the *latest* version of this work use <http://resolver.net.au/hdl/102.100.272/G9JR4TLQH>

To cite *this* version of this work, use <http://resolver.net.au/hdl/102.100.272/RPLZ54PQH>

**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.**

<b>1</b>	<b>Introduction.....</b>	<b>5</b>
<b>2</b>	<b>Model Components.....</b>	<b>6</b>
2.1	Entity.....	7
2.2	Quality.....	7
2.3	Attribute.....	8
2.4	Relation.....	8
2.5	Association.....	8
2.6	Action.....	8
2.7	Identifier System Model.....	9
2.8	Identifier Systems Ontology.....	10
<b>3</b>	<b>Entities.....</b>	<b>10</b>
3.1	Party.....	10
3.2	Authority.....	10
3.3	Administrator.....	10
3.4	Policy.....	10
3.5	Label.....	10
3.6	Context.....	11
3.7	Name.....	12
3.8	Identifier.....	12
3.9	Medium.....	13
3.10	Encoding Scheme.....	13
3.11	Information Model.....	14
3.12	Thing.....	14
3.13	Data source.....	14
3.14	Service.....	14
3.15	Identifier Management System.....	14
<b>4</b>	<b>Instances.....</b>	<b>15</b>
4.1	Context: Known Naming Systems.....	15
<b>5</b>	<b>Relations.....</b>	<b>16</b>
5.1	Notation.....	16
5.2	Responsible.....	16
5.3	Equivalence.....	16
5.4	Synonymy.....	17
5.5	Alias.....	17
5.6	Preferred Identifier.....	17

5.7 Representation.....	18
5.8 Same .....	20
5.9 Manages.....	21
5.10 Contains.....	21
5.11 Realises.....	22
5.12 Homologue.....	22
5.13 Other Relations.....	22
<b>6 Entities (Defined Classes).....</b>	<b>22</b>
6.1 Authority.....	22
6.2 Concrete .....	22
6.3 Abstract .....	22
<b>7 Relations (Defined Classes).....</b>	<b>23</b>
7.1 Realises.....	23
7.2 Homologue.....	24
<b>8 Qualities.....</b>	<b>25</b>
8.1 Registered.....	26
8.2 Actionable.....	26
8.3 Resolvable.....	26
8.4 Unique.....	26
8.5 Universal.....	27
8.6 Persistent.....	27
8.7 Accountable.....	28
8.8 Trusted.....	28
8.9 Reserved.....	29
8.10 Published.....	29
8.11 Citable.....	29
8.12 Verified, Verifiable.....	29
8.13 Nameable.....	30
<b>9 Actions.....</b>	<b>30</b>
9.1 Cite.....	31
9.2 Create.....	32
9.3 Register.....	32
9.4 Deregister.....	33
9.5 Reserve.....	34
9.6 Publish.....	34
9.7 Identify.....	35
9.8 Act.....	35
9.9 Resolve.....	36
9.10 Retrieve.....	37
9.11 Query.....	37
9.12 Verify.....	38
<b>10 Appendix 1: Representations of the same name.....</b>	<b>38</b>
10.1 hdl:102.100.272/XYZZY and info:hdl:102.100.272/XYZZY are representations of the same name.....	38
10.2 hdl:102.100.272/XYZZY and (URL-encoded) hdl %2A102.100.272%2FXYZZY are representations of the same name.....	39
10.3 If I claim hdl:102.100.archer is an alias for hdl:102.100.821, then I am also claiming hdl:102.100.archer/XYZZY and hdl:102.100.821/XYZZY are representations of the same name.....	39

10.4 The concrete names <http://purl.kerry.org/XYZZY> and <hdl:100/XYZZY> cannot be the same concrete name.....40

**11 Appendix 2: Use of Homology.....40**

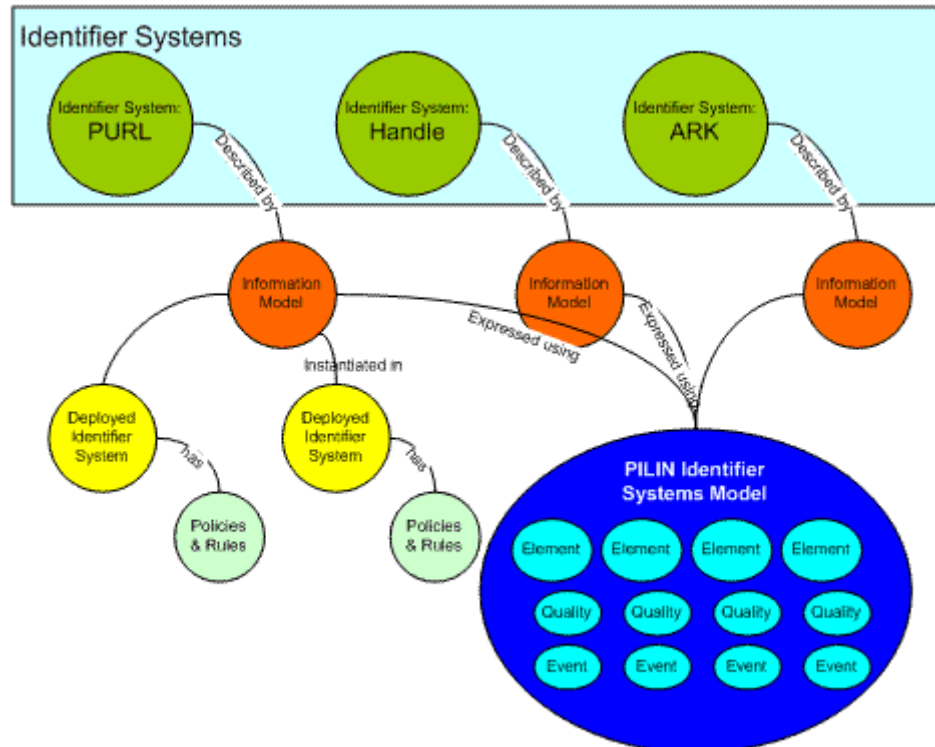
**12 References.....42**

# 1 Introduction

This document defines the overall PILIN *Identifier Systems ontology*. The Identifier Systems ontology is a formal, abstract Information Model for describing identifiers and actual (or theoretical) *identifier systems*. As an information model, it is specific to identifiers and identifier systems, though it may overlap with information models for other domains.

The purpose of the Identifier Systems ontology is to provide a mechanism for documenting and comparing the design decisions and characteristics that are part of an identifier system. The Identifier Systems ontology is meant to be applicable to any identifier system. This document is the formal, normative description of the Identifier Systems ontology. Other documents provide a more friendly description of the Identifier Systems ontology and its application.

Within the Identifier Systems ontology, an (actual) identifier system is described in terms of an *identifier system model*; such an identifier system model is expressed using the model defined herein. An identifier system model is created for each identifier system, and these identifier system models are the basis for the comparison of identifier systems. An identifier system model is used in defining and building an operational identifier system (an implementation); that implementation of the identifier system may be deployed in multiple instances, with each of these deployed instances of an identifier system having different operational and business policies. The overall relationships these components of the PILIN Identifier Systems ontology are illustrated below.



The words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [RFC 2119].

Note that the following elements of the ontology have not yet been finalised:

- “Trusted” may be eliminated from the ontology.
- A complete OWL model of the ontology has not been completed.

## 2 Model Components

The PILIN Identifier Systems ontology consists of:

- *Entities* – *key parts of an identifier system in the Identifier Systems Model.*
- *Instances* – *specific instances of entities which are relevant to any Identifier Systems Model. (There are only few of these.)*
- *Actions* – *operations that MAY be applied to any of the entities. Actions may have results, or else they may change the state of the entity. Results can be any thing, and need not be defined as a class of entities from this ontology.*
  - *e.g. an identifier may be resolved to a URL. The URL is the result of the action.*
- *Qualities* – *characteristics that MAY be applied to refine any of the components. Qualities usually apply only to entities, but some qualities apply to other qualities. Qualities may be either unary or binary predicates. Qualities MAY have subclasses. Unary qualities may have a Boolean range.*
  - *e.g., Resolvable(identifier X)*
  - *e.g., Actionable(identifier X, action Y)*
  - *e.g., Encoded(identifier X, encoding Y)*
  - *e.g., Verified(Resolvable(identifier X))*

*Qualities often have an associated action. In that case, the Identifier Systems ontology defines the quality in terms of the action: the quality is "has the action applied to it", or "can have the action applied to it" (e.g. Verify vs. Verified, Verifiable). In other words, all Actions are primitives in the ontology. Some Qualities are primitives, and do not depend on Actions. Other Qualities are not primitives, but are instead defined in terms of primitive Actions.*
- *Attributes* - *of entities.*
- *Relations* - *between entities.*
- *Associations* - *of entities with entities.*
- *Identifier System Model* – *the description of an (actual) identifier system. The identifier system model SHALL be expressed using the Identifier Systems ontology.*

The PILIN Identifier Systems ontology assumes the basic types of OWL ontology (see <http://www.w3.org/TR/owl-guide/> ), Classes, and Properties:

- **Classes** are what **Individuals** are members (**instances**) of, and are used to define basic concepts. Classes are **subclasses** of other Classes, in an is-a hierarchy.
- **Datatype Properties** are binary relations between instances of classes and literals or datatypes.
- **Object Properties** are binary relations between instances of two classes. The first class is the **domain** of the property, and the second is the **range**. N-ary

relations can only be represented in OWL through multiple binary relations, between instances of classes and "relation instances"; e.g. purchases(Fred, pen, \$3, shop) may be represented as purchases(Fred, purchase-instance1), purchase-item(pen, purchase-instance1), purchase-cost(\$3, purchase-instance1), purchase-vendor(shop, purchase-instance1).

- **Reifications** are **statements** about properties. Reifications allow properties to be treated as classes, since statements are classes. For instance, "XYZ identifies the buried treasure" is a property, relating "XYZ" to "buried treasure", which are both instances of classes. "Long John Silver said that 'XYZ does identify the buried treasure'" is a relation between Long John Silver, an instance of the pirate class, and 'XYZ does identify the buried treasure', an instance of the statement of the property relating "XYZ" to "buried treasure".
- **Defined Classes** are classes which are not primitives, but are defined as the domains of object properties. For example, given the object property "is-parent-of", the class "parent" is not a primitive class, but the class defined as the domain of that property: "All things which are-parent-of something".
- **Thing** is the class of which all classes are subclasses. **Nothing** is the empty class.
- The Identifier Systems Ontology defines the foregoing components in terms of those types. Definitions for each component follow.

## 2.1 Entity

Definition: A thing that is used to define a (core, representational) concept in an identifier system. Entity instances MAY be stored and managed in an identifier system, i.e., an identifier system MAY be realized through the storage and management of a collection of entities.

An Entity SHALL be a class.

An Entity MAY have subclasses.

Notes:

- *The set of entities are defined by the Identifier Systems ontology (Entities section).*
- *The representation and encoding for the entities are included in the definition of the Entity.*
- *Entities include statements. For instance, a rule is a reification of a property which should hold true: the rule is not a property, but a statement, and therefore an entity.*
- *Some entities in the Identifier Systems ontology are defined classes. They are listed separately.*

## 2.2 Quality

Definition: Qualities are used to describe how an entity SHOULD behave in an identifier system, i.e., an identifier system MAY be realized through the storage and management of qualities associated with entities.

Qualities are properties of entities OR qualities. Qualities MAY have subclasses. Qualities have Boolean values. To represent this in OWL, qualities shall be modelled as subclasses of the entities or qualities they describe. They may have further necessary conditions refining them, through object properties. E.g.

- *Actionable: subclass of: Identifier*
- *Protocol-in-Actionable: Domain: Actionable; Range: Identifier*
- *Web-Actionable(x) iff Actionable(x) and Protocol-in-Actionable(x, http)*

Notes:

- *The set of qualities are defined by the Identifier Systems ontology herein (Qualities section).*
- *Qualities are applied to the entities of an identifier system; the domain of qualities MAY be the Identifier Systems ontology or MAY be an individual identifier system model, i.e., some qualities apply to an entity in ALL identifier systems; others apply to an entity on a case-by-case basis for each identifier system.*

## 2.3 Attribute

Definition: Attributes are object properties, relating entity instances to instances of the allowed values for that attribute.

## 2.4 Relation

Definition: Relations are object properties, whose domain and range are both entities. Relations are symmetric (bidirectional). Relations have subclasses.

## 2.5 Association

Definition: Associations are object properties, whose domain and range are both entities. Association is not symmetric (i.e., unidirectional). Associations have subclasses.

## 2.6 Action

Definition: A process, triggered by a party through a system at some time, applied to (an instance of) one of the entities (input), producing a result. An action is therefore a four-way relation between an input entity, an output entity, a party, and a system.

The result (output) MAY be (an instance of) an entity, (the value of) an attribute of an entity, or a change in quality value to (an instance of) an entity, or a change in (the value of) an attribute within an instance of an identifier system. The result may be a representation (encoded).

An Action MAY have subclasses.

Actions are modelled as classes, which must have the following attributes as necessary rules:

- *Action-Input: Domain: Entity; Range: Action*

- *Action-System: Domain: Identifier Management System; Range: Action*
- *Action-User: Domain: Party; Range: Action*
- *Action-Result: Domain: Thing; Range: Action*
- *Action-Time: Domain: Time; Range: Action*

For some actions, inputs and results are inapplicable; these are modelled in OWL as having inputs/results of the class Nothing. Actions resulting in changes of state of a system are modelled as having result Nothing.

All actions on a system must be authorised, which means the following properties are also necessary rules:

- *Action-Authorisation: Domain: Authorisation-Instance; Range: Action*
- *User-Authorised: Domain: Party; Range: Authorisation-Instance*
- *System-Authorised: Domain: Identifier Management System; Range: Authorisation-Instance*
- *Input-Authorised: Domain: Entity; Range: Authorisation-Instance*

The user, input and system authorised must match the user, input and system actually carrying out the action.

Notes:

- *The set of actions are defined by the Identifier Systems ontology herein (Actions section).*
- *Actions MAY be combined into composite actions. Composite actions are not defined in the Identifier Systems ontology.*
- *The result of an action MAY be used as the inputs of another action.*
- *Actions MAY be performed by services. The design of the services and their association with actions is out of scope for the Identifier Systems ontology; their design and association is part of the design of an information system supporting an identifier system.*
- *In the e-Framework model for an identifier system, the actions map to functions in a Service Usage Model; the functions in turn map to services that are designed and implemented through service expressions and service implementations. The mapping of actions to services is out of scope for the Identifier Systems ontology.*

## 2.7 Identifier System Model

Definition: The PILIN Identifier Systems ontology representation of an identifier system.

Notes:

- *The identifier system model is used to define an individual identifier system.*
- *The representation of an identifier system model is a defined herein; it is the representation and encoding of an identifier system using the components of the Identifier Systems ontology.*
- *The definition of the identifier system model MAY be extended to include information that is not defined in the Identifier Systems ontology.*

- *The identifier system model MAY be encoded in a formal semantic model (e.g., OWL).*

## 2.8 Identifier Systems Ontology

Definition: The overall PILIN model of how to describe an (any) identifier system, using entities, qualities and actions.

Notes:

- *This document presents the Identifier Systems ontology.*
- *The Identifier Systems ontology is encoded in a formal semantic model (OWL).*

## 3 Entities

### 3.1 Party

A party is defined following ISO 2146: a person or a group which can participate in various processes, including managing or using identifiers.

### 3.2 Authority

An authority is a party responsible for a component (defined class). See Responsible.

### 3.3 Administrator

See Publish.

### 3.4 Policy

A policy is a set of rules (i.e., statements of properties), regarding the things (e.g., entities, actions and qualities) defined in the Identifier Systems ontology. Policies may be referred to by parties. A rule in a policy states a property whose domain is thing the policy is regarding. Policies have authorities responsible for them. Policies are exposed and on occasion enforced through identifier management systems: that means that the systems ensure that the properties stated in the policy are true of the entities, actions and qualities maintained through the system.

### 3.5 Label

A label is a signifier: a symbol which can potentially be used to refer to a thing. A label is a class and not an instance: any instance of a label is considered to be the same label.

## 3.6 Context

A context is an (OWL) thing. A context has a particular purpose for organising and managing (OWL) things. This discussion concentrates on the attributes of contexts relevant to their used in identifiers; there may be other things about contexts which are out of scope of this model.

- *A context may be identified by one or more identifiers.*
- *A context is associated with a set of labels. Each of those labels is in the context ("in" being the object property relating the two.) Each label can be in a given context only once, but a label can be in multiple contexts. (This constitutes a policy regarding labels in a context.)*
- *A context has additional policies regarding the labels in it.*
- *A context may have one or more authorities responsible for it.*
- *The primary authority responsible for a context is called the owner of the context; the owner may delegate authority over the context to other parties. **Delegation, ownership, and transfer of ownership (e.g. "delegate, allocate" actions) are not further modelled in this ontology.***
- *Within the Identifier Systems ontology, contexts are not decomposed.*

A context should have the following policies apply to its labels:

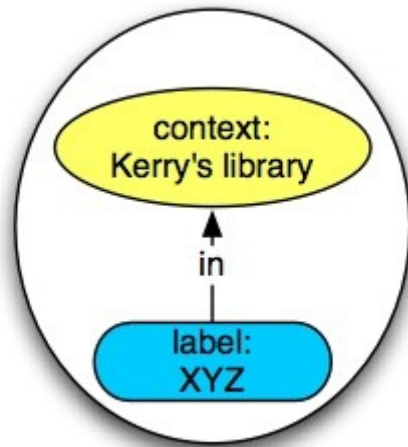
- All contexts have a policy on label formats, indicating which labels can be in the context. Even if a context allows any label whatsoever to be in the context, that itself counts as a label format policy.
- All contexts *managed* through a system have an access policy, indicating permissions for a party to act using the label in the context. Examples of such actions are permission to add label, to check if label is member of context, to remove label from context, to access historical ownership information on label.
- All contexts for *identifiers* have an association policy, on what things the labels in the context may identify, when used as identifiers. This association policy enables the context's purpose, and defines the use of labels in that context as identifiers.

A context should also have a label membership function: it should be knowable whether a label is in a context or not.

The context of "known naming systems" is an instance of Context used as a root in Identifier names. See definition of Identifiers below.

### 3.7 Name

#### Name



A name is a pairing of a label with a context the label is in (see above). The label must comply with the context's policy requirements (i.e., the policies regarding labels which that context has). The same label paired with a different context gives a different name.

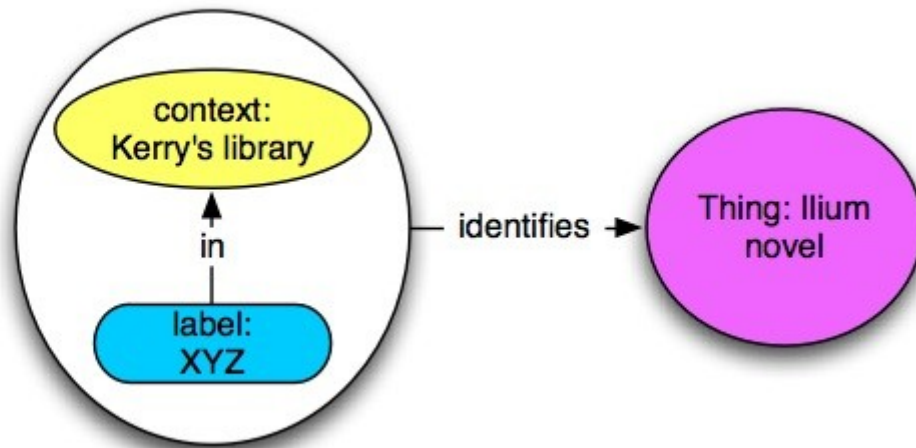
Notes:

- *Label and context MAY be represented through different encoding schemes.*
- *Any semantics of a label SHOULD be defined in the identifier system policy.*
- *Within the Identifier Systems ontology, the label is treated as decomposable. Actions & qualities may treat the name as decomposable, but only following procedures defined in the identifier system policy.*
- *Within the Identifier Systems ontology, the name SHALL not be decomposable except into a label and a context.*
- *The representation of a name (as defined below) is a signifier: it is a single thing, in contrast to a name which is a pair.*

### 3.8 Identifier

An identifier is an association of a name with a thing. (i.e., a mapping of a name to a thing). When a name is part of an identifier it shall only be associated with one thing in that context. This association is termed "identify".

## Identifier



### Note:

- Any model of bidirectional information about the thing (*thing > name*) is out of scope for the Identifier Systems ontology, but MAY be described in an identifier system.
- The representation of an identifier depends on an adequate representation of the context as well as the label. For more on this, see the Instance definition Context: Known Naming Systems.

## 3.9 Medium

A medium is the vehicle through which a message is transmitted from a sender to a receiver, as understood in Communication Theory.

## 3.10 Encoding Scheme

An encoding scheme defines a mapping of labels to labels. The mapping should be one-to-one. Since the mapping itself is an object relation between two labels, the encoding scheme is a statement of the object relation (and therefore an entity). The mapping relation is called **encode**.

- *Encoding schemes can have subclasses.*
- *The identifier system model SHOULD specify the defined encoding schemes.*
- *Because the encoding scheme mapping should be one-to-one, a label that was unique in its original context should be mapped to a label that is unique in its new context.*
- *Encoding is usually undertaken in order to represent a label to a specific audience for a specific purpose (see Representation). E.g., a label in typographic encoding may be re-encoded for a given audience to URL encoding, which in turn may be re-encoded to an Encryption.*
- *Whether an encoding scheme is appropriate to its purpose can depend on the medium through which the encoding is to be communicated. Different media may require different types of encoding scheme; e.g. Typographical Encoding*

*Schemes and Web-Encoding Schemes differ in the form of their respective results, print letters and octets, which are suited to their respective media.*

### 3.11 Information Model

A model for things being identified, their properties, and the relations between them e.g., FRBR, <http://dublincore.org/documents/abstract-model/> . The Identifier Systems ontology is an Information Model specific to identifiers and identifier systems.

### 3.12 Thing

A thing is what OWL says it is: namely, the root class. Any class modelled in an information model (including the Identifier Systems ontology) is a subclass of "thing".

This document takes an identifier perspective on things. A particular information model may refine or add new constraints on what may be considered a thing.

Notes:

- *The representation of the thing MAY be the same for all instances of the identifier system or each instance MAY have its own representation.*

### 3.13 Data source

A tool for the storage and management of data by a party. The data source exposes services allowing access to that data by other parties.

### 3.14 Service

A service is an action with a defined interface, that can be triggered by a human or a computer.

### 3.15 Identifier Management System

A Collection of definitions, information model, policies, services, and data sources to manage identifiers.

An Identifier Management System has an access boundary defined in terms of access to the data sources in the system. Things inside the access boundary are accessed only by the parties that own the system. Things outside the access boundary may be accessed by other parties.

Note:

- *The functionality of an identifier management system is exposed through services.*
- *A Registry is a trusted data source with a recognised authority. The data sources used in identifier management systems should be registries.*

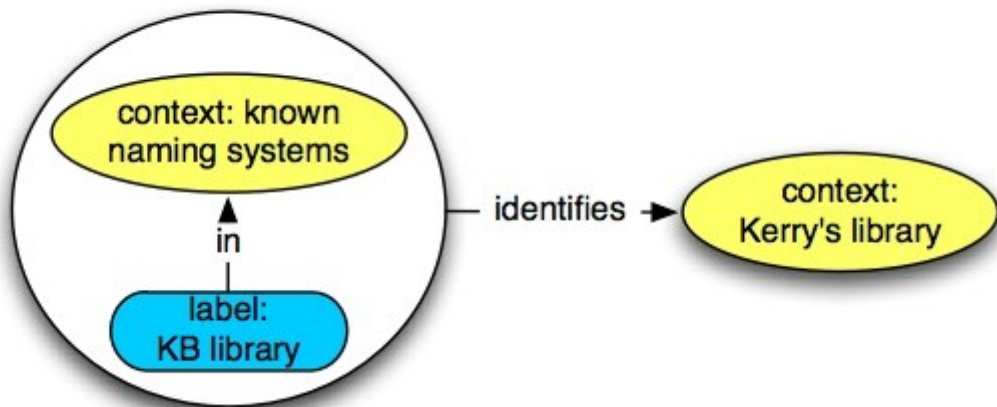
## 4 Instances

### 4.1 Context: Known Naming Systems

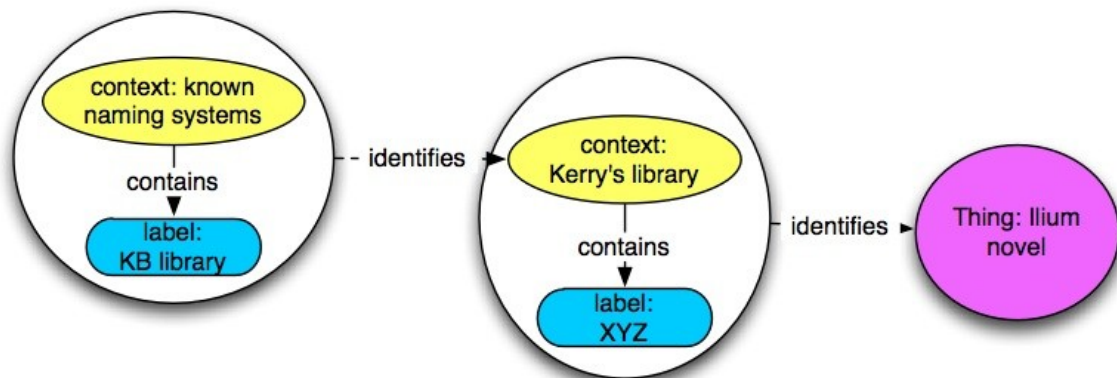
Contexts have identifiers; i.e. there are names that identify contexts, and those context names are used in the representation of contexts (see Representation): So in the foregoing example, a representation of the identifier should include not only the label, XYZ, but also an identifier for the context, Kerry's library. But that identifier in turn will involve a label—say, "Kerrylib", and a context for the label—say "Library names used in illustrating the PILIN Identifier systems ontology". And that context will have its own identifier, which will need its own context to be provided. This leads to an infinite regression of contexts: contexts of identifiers, contexts of contexts of identifiers, contexts of contexts of identifiers....

To prevent this infinite regression, we define a context instance of "known naming systems". All context names are assumed by default to be in the context of known naming systems. The context of known naming systems does not have any explicit policies or properties defined.

#### Identifying a context



This means that, to properly represent an identifier (see below), one must know the label, and the identifier for the context.



## 5 Relations

Further constraints on the applicability of any of these relations may be imposed by particular authorities in particular identifier system instances.

### 5.1 Notation

- $C::L$  is a name with context  $C$  and label  $L$
- $N \rightarrow T$  is an identifier with name  $N$  associated with thing  $T$

### 5.2 Responsible

*Domain: Party; Range: Component*

A party may be responsible for the maintenance and accuracy of the component. Such parties are called **authorities**.

An authority can be responsible for an entity, or (the reification of) a property with range over an entity (e.g., an attribute of an entity, or a relation between the entity and another component).

Note:

- *Many entities are either decomposable, or else related to other entities hierarchically through relations like "contain". The authority over the containing entity is often also responsible for the contained entities. In that case, the authorities of the contained entity may overlay, since the contained entities may have their own authorities. Precedence of authorities is determined through identifier system policy. Whether the properties of the containing entity are inherited by the contained entity is determined through identifier system policy. (That is to say, a "contains" relation is not necessarily an OWL "is-a" relation, which requires inheritance.)*
- *Authorities include managers of operational systems.*

### 5.3 Equivalence

*Domain: Identifiers; Range: Identifiers*

Two identifiers are equivalent if they are associated with the same thing at time  $t$

$$\text{equivalent}(N1 \rightarrow T1, N2 \rightarrow T2, t) \Leftrightarrow T1 = T2 \text{ at time } t$$

Note:

- *Equivalence is time-dependent (e.g., the identifiers may not be persistent in association).*
- *The equivalence of two identifiers is a fact independent of any authority.*
- *Determining whether the identifiers are associated with the same thing presupposes that the identifiers conform to a common information model, or at least that their respective information models are consistent as to that thing.*

- *Equivalence for other Domains is possible but is not defined in the Identifier Systems Ontology.*

## 5.4 Synonymy

*Domain: (Identifier, Identifier); Range: Authority*

Two identifiers are synonyms according to an authority if the authority asserts that they are equivalent, over a defined time span.

$$\text{synonymous}(N1 \rightarrow T1, N2 \rightarrow T2, \text{authority}, \text{time span}) \Leftrightarrow \\ \forall t \text{ in time span : claims}(\text{authority}, \text{equivalent}(N1 \rightarrow T1, N2 \rightarrow T2, t))$$

## 5.5 Alias

*Domain: <Identifier, Identifier>; Range: Authority*

*Modelled in OWL as:*

- *Is-Alias: Domain: Identifier; Range: Alias-Instance*
- *Is-Alias-Target: Domain: Identifier; Range: Alias-Instance*
- *Is-Alias-Authority: Domain: Authority; Range: Alias-Instance*
- An identifier is an alias of a target identifier if the identifiers are synonymous, but the alias identifier is managed to be dependent on the target identifier. That is to say, the authority enforces the synonymy asymmetrically: any change in what the target identifies will also change what the alias identifies (according to the authority)—but any change in what the alias identifies does not change what the target identifies.

## 5.6 Preferred Identifier

*Domain: Identifier; Range: Authority.*

An identifier is preferred according to an authority if the authority privileges that identifier over the set of all equivalent identifiers, and the authority guarantees its persistence (including persistence of reference)

- *Only concrete identifiers can have their persistence guaranteed through some management process.*
- *The authority may use the preferred identifier as a primary key in an identifier management system, to manage other identifiers that are equivalent to it.*
- *Preferred-identifier is not a relation between two identifiers, but between an identifier and an authority. It is included here because of its close relation to synonymy.*
- *This definition follows XRI's notion of a canonical identifier, which it defines as "the preferred synonym among all synonyms". This term refers to a policy choice, and does not necessarily imply a more basic or conventional representation of the identifier (the typical use of the adjective in mathematics)—although canonical representation can motivate the choice of*

a preferred synonym. To prevent confusion, we use "preferred" instead of "canonical".

## 5.7 Representation

*Domain: (Label, Name, Context, Identifier); Range: Label*

Definition: A representation of an entity is a signifier used to communicate the entity to an audience.

- *Labels, names, contexts, identifiers can all have multiple representations*
- *Other things may have representations, but they are not defined in the Identifier Systems Ontology.*
- *The function  $\text{reprs}(X)$  returns the set of representations for  $X$*

**Labels** are represented through encoding schemes. There are many ways to encode a label, each defined by a distinct encoding scheme.

$$\text{reprs}(L) = \text{set of encodings of } L$$

Spelling this out:

$$\text{reprs}(L) = \{ C: \text{Encoding-Scheme} \mid \text{encode}(L, C) \}$$

**Contexts** are represented by representations of their identifiers

$$\text{reprs}(C) = \{ N: \text{Name} \mid N \rightarrow C \cdot \text{reprs}(N) \}$$

**Names** are represented through serialisations of context representations and label representations.

- Serialisations are combinations of two or more representations into a single representation. (*Domain: (Label, Label); Range: Label.*) There are many possible serialisations of two representations, defined by distinct serialisation schemes. Serialisations are themselves represented through encodings. Serialisations must be reversible: parties must be able to extract, from a serialisation of the context representation and the label representation, the two individual representations.

$$\text{reprs}(C::L) = \{ CR: \text{reprs}(C), LR: \text{reprs}(L) \cdot \text{combine}(CR, LR) \}$$

Making the notion of 'combine' explicit through serialization:

$$\text{serialisations}(R1, R2) = \{ S: \text{Serialisation-Scheme} \mid \text{serialise}(R1, R2, S) \}$$

$$\text{reprs}(C::L) = \{ CR: \text{reprs}(C), LR: \text{reprs}(L) \mid$$

$$S: \text{serialisations}(CR, LR) \cdot$$

$$C: \text{Encoding-Scheme} \cdot \text{encode}(S, C) \}$$

**Identifiers** are represented by their names

$$\text{reprs}(N \rightarrow T) = \text{reprs}(N)$$

Note:

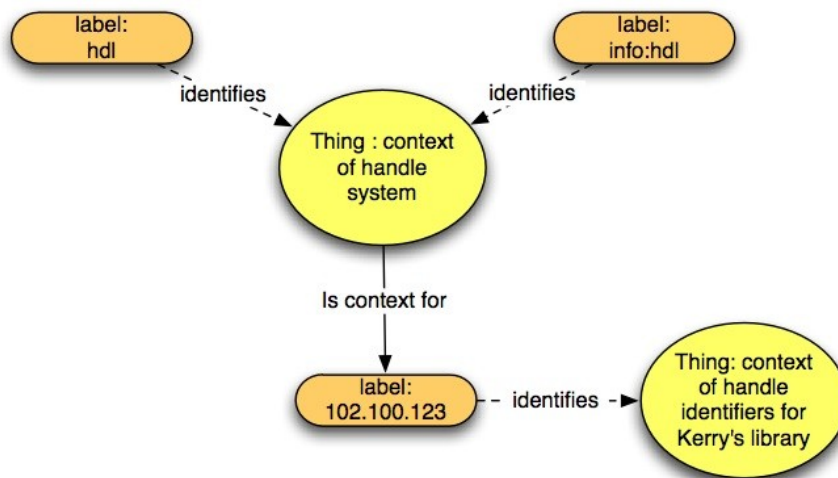
- *There is a constraint on serialisations of contexts and labels into name representations: the label must be explicit in the serialisation, but the context may be implicit (omitted). If the context is implicit, then it must be recovered in some manner specific to the audience and the purpose. This also applies to the recursive use of serialisations (of names of contexts). The context "known naming systems" is usually implicit in a serialisation of a name with that context.*

- *Formal serialisations are well-defined for concrete names. They are not typically well-defined for abstract names, which instead are usually represented through prose descriptions (e.g., "the word SHALL in the context of RFC 2119"). Contexts are frequently implicit in abstract names, because of the difficulty of naming abstract contexts. Contexts are also frequently implicit in concrete names, because the concrete context is assumed (or difficult to name).*

### 5.7.1 Illustration 1: Possible representations of an identifier.

- *Label: XY/ZZY*
- *Context: the Handle server 102.100.272*
- *Identifiers of the Handle server 102.100.272 =*  
 $\{ \text{"hdl:102.100.272"} \rightarrow \text{"the Handle server 102.100.272"},$   
 $\text{"info:hdl:102.100.272"} \rightarrow \text{"the Handle server 102.100.272"}, \dots \}$
- *reprs(XY/ZZY) = {XY/ZZY, XY%2FZZY, ...}*
- *reprs(the Handle server 102.100.272) =*  
 $\text{reprs(identifiers of the Handle server 102.100.272)} =$   
 $\text{reprs(names in identifiers of the Handle server 102.100.272)} =$   
 $\{ \text{"hdl:102.100.272"}, \text{"info:hdl:102.100.272"}, \dots \}$
- *reprs(the Handle server 102.100.272 :: XY/ZZY) =*  
 $\{ \text{CR: reprs(the Handle server 102.100.272), LR: reprs(XY/ZZY)} \mid$   
 $\text{S: serialisations(CR, LR) } \cdot \text{encodings(S)} \}$
- *= {CR: {"hdl:102.100.272", "info:hdl:102.100.272", ...},*  
 $\text{LR: } \{ \text{XY/ZZY, XY\%2FZZY, ...} \} \mid$   
 $\text{S: serialisations(CR, LR) } \cdot \text{encodings(S)} \}$
- *= {S: { serialisations("hdl:102.100.272", "XY/ZZY"),*  
 $\text{serialisations("hdl:102.100.272", "XY\%2FZZY"),}$   
 $\text{serialisations("info:hdl:102.100.272", "XY/ZZY"),}$   
 $\text{serialisations("info:hdl:102.100.272", "XY\%2FZZY"), ... } \cdot$   
 $\text{encodings(S)} \}$
- *= {S: {*  
 $\text{hdl:102.100.272/XY/ZZY,}$   
 $\text{hdl:102.100.272XY\%2FZZY,}$   
 $\text{XY/ZZYinfo:hdl:102.100.272,}$   
 $\text{context\ info:hdl:102.100.272\ label\ XY/ZZY, ... } \}$   
 $\cdot \text{encodings(S)} \}$
- *= {*  
 $\text{hdl:102.100.272/XY/ZZY,}$   
 $\text{hdl\%3A102.100.272\%2FXY\%2FZZY,}$   
 $\text{context\%25u22BAinfo\%3Ahdl\%3A102.100.272\%25u22BAlabel}$   
 $\text{\%25u22BAXY\%2FZZY,}$   
 $\dots \}$

## 5.7.2 Illustration 2: Explicit and implicit contexts



hdl:102.100.123 and info:hdl:102.100.123  
are the same name because  
hdl and info:hdl are equivalent identifiers for the same context

- *XYZZY : label, implicit context*
- *102.100.272/XYZZY : label, label of context, implicit context of context*
- *hdl:102.100.272/XYZZY : Local Handle Server view of context: label, label of context (102.100.272), label of context of context (hdl), implicit context of context of context*
- *hdl:102.100.272/XYZZY : Global Handle Server view of context: viewed differently: label, label of context (hdl:102.100.272), implicit context of context (known naming systems)*
- *info:hdl:102.100.272/XYZZY : Info-URI view of context: label, label of context (102.100.272), label of context of context (hdl), label of context of context of context (info-uri), implicit context of context of context of context (known naming systems)*

## 5.8 Same

*Domain: Thing; Range: Thing*

The relation of "sameness" is the expected relation of identity. It is used to establish identity between entities despite the fact that their representations may differ. In particular:

- Two **names are the same** if they have the same label in the same context. A label may be in more than one context; so two names are the same only with respect to a given context.

$$\text{same}(C1::L1, C2::L2) \Leftrightarrow L1 = L2 \ \& \ C1 = C2$$

- Two representations can **represent the same name**:

$$\begin{aligned} \text{same\_represented\_name}(NR1, NR2) \Leftrightarrow \\ \exists N: \text{Name} \mid \{NR1, NR2\} \subset \text{reprs}(N) \end{aligned}$$

Expanding this out ...

$$\text{same\_represented\_name}(NR1, NR2) \Leftrightarrow \\ \exists C: \text{Context}, \exists L: \text{Label} \mid \{NR1, NR2\} \subset \text{reprs}(C::L)$$

further ...

$$\text{same\_represented\_name}(NR1, NR2) \Leftrightarrow \\ \exists C: \text{Context}, \exists L: \text{Label} \mid \\ \{NR1, NR2\} \subset \{ CR: \text{reprs}(C), LR : \text{reprs}(L) \cdot \text{combine}(CR, LR) \}$$

i.e.

$$\text{same\_represented\_name}(NR1, NR2) \Leftrightarrow \\ \exists C: \text{Context}, \exists L: \text{Label} \mid \{NR1, NR2\} \subset \\ \{ CR: \text{reprs}(C), LR : \text{reprs}(L) \mid \\ S : \text{serialisations}(CR, LR) \cdot C: \text{Encoding-Scheme} \cdot \text{encode}(S, C) \}$$

This translates to: two name representations are the same when they represent the same context and they encode the same label, with possibly different serializations and encodings.

## 5.9 Manages

*Domain: Identifier Management System; Range: Thing*

An identifier management system manages an entity, if it is used to record and update representations of the entity and its attributes, which parties can consult.

## 5.10 Contains

*Domain: Context; Range: Context*

A context A contains another context B, if all labels contained in A are contained in B, and all policies of B apply to A. A is the enclosing context of B, and B a subcontext of A.

Note:

- *The contexts must be both concrete or both abstract.*
- *Contains is not an "is-a" relation, so it cannot be modelled through subclasses of contexts.*
- *Out of a set of contexts, the "smallest" context is the context that all other contexts in the set contain.*
- *The smallest abstract context that two concrete contexts realise is defined as the smallest out of the set of abstract contexts realised by both concrete contexts.*

$$\text{is\_realised\_by}(K) = \{A: \text{Abstract Context} \mid \text{realises}(A, K)\}$$

$$\text{smallest\_context}(C) = \{s: \text{Context} \mid \forall x \in C : \text{contains}(x, c)\}$$

$$\text{smallest\_abstract\_context\_realised}(C1, C2) = \\ \text{smallest\_context}(\text{is\_realised\_by}(C1) \cap \text{is\_realised\_by}(C2))$$

## 5.11 Realises

See Entities (Defined Classes)

## 5.12 Homologue

See Entities (Defined Classes)

## 5.13 Other Relations

Other object properties relating entities are discussed under Qualities. These are associations (non-symmetric properties). They include

- *the association of a name to a thing in an identifier ("identifies", inverse "is referent of")*
- *the association of a label to a context in a name ("is in")*

# 6 Entities (Defined Classes)

This section presents entities defined in terms of the foregoing relations.

## 6.1 Authority

See Responsible

## 6.2 Concrete

A thing is concrete if it is managed by some identifier management system.

An identifier management system can manage one or more contexts, which makes those contexts concrete.

Deploying an identifier management system defines a single concrete context specific to that deployment (the identifier management system context: IMSC), as follows:

- The purpose of the IMSC is the purpose of the system.
- The labels of the IMSC are the labels managed in the system.
- The policies of the IMSC are the policies enforced by the system.
- The authorities over the IMSC are the authorities responsible for the system.

## 6.3 Abstract

A thing which is not concrete is abstract.

Abstract things are introduced into the Identifier Systems Ontology to model the relation between several concrete things. If two concrete identifiers are managed in different identifier systems, but are otherwise the same (e.g. same label, same

thing identified, same authority), we can describe what they have in common by constructing an abstract identifier, defined in terms of those commonalities.

- *An abstract context is not defined in terms of any one identifier management system, but it can still be defined in terms of the attributes that the concrete contexts described have in common: purposes, authorities, and policies.*
- *Example: Monash University has a Handle server and a PURL server. These two identifier management systems define their own concrete contexts. An abstract context can be defined to model the commonality between them: its authority is Monash University, its policies are the policies both servers enforce, and its purpose is the management of Monash University identifiers, regardless of specific system.*

## 7 Relations (Defined Classes)

This section presents relations with defined class entities as their domains or ranges.

### 7.1 Realises

*Domain: (Name, Label, Context, Identifier); Range: (Name, Label, Context, Identifier)*

The *realises* relation is the relation between a concrete thing and a corresponding abstract thing. In particular:

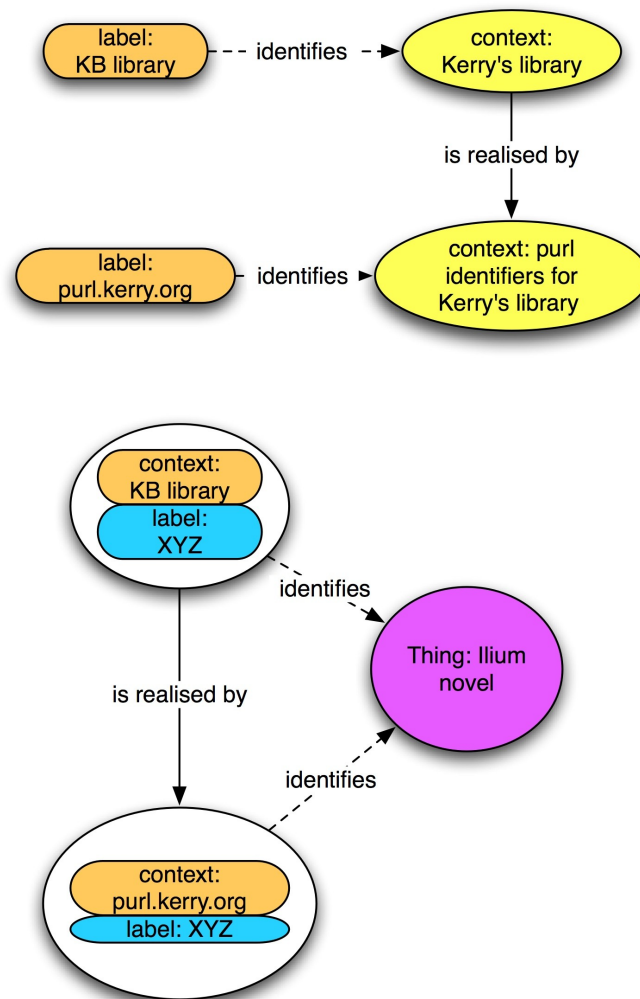
- *A concrete identifier realises an abstract identifier if the two identifiers are synonymous, and the concrete name realises the abstract name. The authority claiming synonymy is the authority the abstract and the concrete and the abstract contexts have in common.*
- *A concrete name realises an abstract name if the two labels are the same, and the concrete context realises the abstract context. E.g., because <http://purl.kerry.org> realises "Kerry's Library", the identifier "<http://purl.kerry.org/xyzy>" realises the identifier "KB's Library:xyzy"*
- *A concrete context realises an abstract context if the concrete context manages and supports all the policies of the abstract context, and every identifier in the concrete context realises a corresponding identifier in the abstract context. (That is, a concrete context must be contained in the abstract context it realises.)*

A concrete context is not identical to an abstract context. A concrete context may manage only a subset of the identifiers of the abstract context.

An abstract name is by definition not registered: only its concrete counterpart is. So abstract names and identifiers are managed by an identifier management system only indirectly: the system manages concrete entities as manifestations of abstract entities.

Note:

- *Contexts may have various relations with other contexts. These relations may not be hierarchical or isomorphisms. (For instance a context may be included in two different contexts.) These contexts are out of scope of this model, with the exception of a basic Contains relation and the Realises relation.*



## 7.2 Homologue

*Domain: Concrete Identifier; Range: Concrete Identifier*

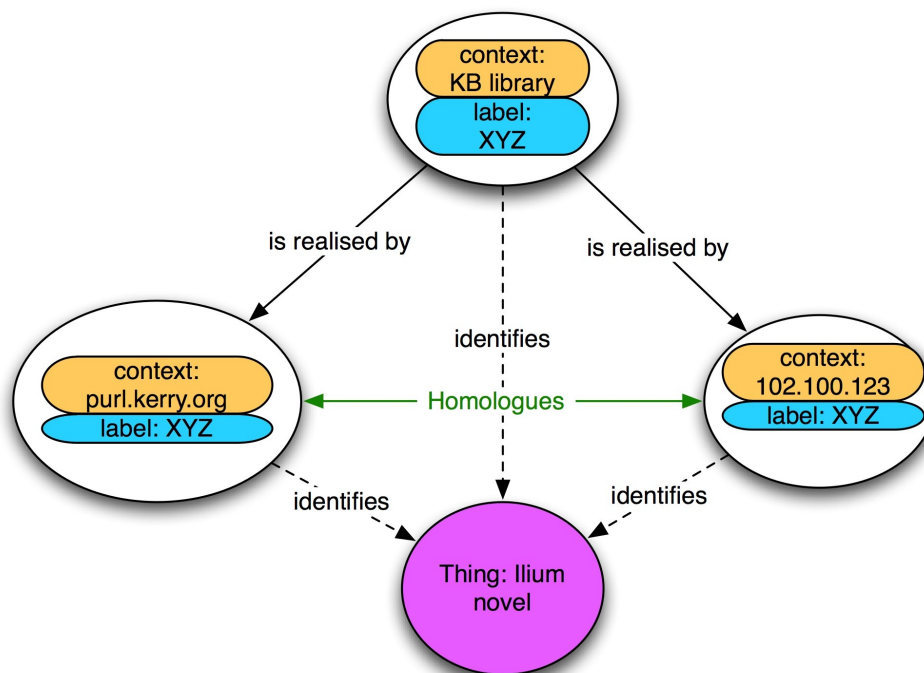
Two concrete identifiers realising the same abstract identifier are *homologues*. Homologues refer to the same thing (are equivalent), using the same label, and their contexts realise the same abstract context (as defined below).

The scope for homology is restricted to the smallest abstract context that both concrete contexts realise (as defined above). The scope for homology does not extend recursively to contexts which that common abstract context itself is related to.

Note:

- *Example: `http://purl.kerry.org/xyzzy` and `hdl:100/xyzzy` are synonyms only within the scope of "Kerry's Library", the abstract context both the PURL and the Handle system realise. That scope does not extend further to the abstract context "all libraries", or "all known naming systems". If it did, then an unrelated equivalent identifier "Nigel's Library:xyzzy" would also be a homologue --- and "homologue" would not be any different to "synonym".*

- The use of the term “homologue” reflects its use in evolutionary biology: homologues are structures with a common ancestry (here, shared parent abstract context), whereas analogues are structures with a common function (here equivalents).
- If two identifiers are homologues, they are equivalents. They are not necessarily synonyms, as there need not be an authority claiming that they are equivalent.
- The notion of homology (“how are Kerry’s PURL identifiers and Kerry’s Handle identifier doing the same thing”) motivates the introduction of abstract contexts and identifiers into the Identifier Systems Ontology.



## 8 Qualities

The entities within the PILIN Identifier Systems ontology are refined using a set of qualities.

- A quality is a subclass of an entity or a quality.
- Qualities MAY be assigned, via an identifier system model, to the entities of an identifier system.
- The Identifier Systems ontology places some restrictions on the domain of qualities and their relationship to entities. Further restrictions may be placed in Identifier Systems Models.
- Qualities MAY be further refined with quality subclasses. This may occur through additional necessary conditions on the quality.
- Qualities MAY be combined with (certain) other qualities.

- *From a user perspective, quality values are binary, i.e., an entity either has or does not have the stated quality. Internally, an identifier system may have a more complex state model with additional intermediate states.*
- *How quality values are managed and how qualities are associated with entities SHALL be defined in the identifier system model. In an identifier system with an underlying computational software system, some quality values MAY be managed by the software system and other MAY be specified by governance rules and policies.*
- *Some quality definitions are referred to their corresponding actions.*

## 8.1 Registered

*Subclass of: Thing*

See Register.

## 8.2 Actionable

*Subclass of: Identifier*

*Refinement to particular action:*

*Action-in-Actionable: Domain: Actionable; Range: Action*

*System-in-Actionable: Domain: Actionable; Range: Identifier Management System*

See Act.

## 8.3 Resolvable

*Subclass of: Identifier*

*Refinement to particular protocol:*

*Protocol-in-Resolvable: Domain: Resolvable; Range: Protocol*

See Resolve.

## 8.4 Unique

*Subclass of: Thing*

*Necessary Rule: Scope-of-Unique: Domain: Unique; Range: Thing*

Definition: A thing is unique if there exists one and only one of the thing within a defined scope (of the thing).

Scope: defined in the identifier system policy or Identifier Systems ontology. There may be more than one scope for uniqueness for an entity.

- *Example: an name is unique in its context*
- *Example: an identifier is an association of a name with a unique thing*
- *Example: a thing need not be identified by a unique identifier*

Notes:

- *The uniqueness of a label in a context is definitional to the concept of name.*
- *The uniqueness of a thing being identified by a name is definitional to the concept of identifier.*
- *An identifier may be resolvable in multiple ways (multiple resolution) while still being associated with a unique thing, so long as the thing is abstract enough to be represented by all the possible resolutions. For instance, the thing is an abstract document, and the resolutions are to particular instances of the document.*

## 8.5 Universal

*Subclass of: Identifier*

*Refinement to particular context:*

*Context-in-Universal: Domain: Universal; Range: Context*

Definition: An identifier is universal if it is the unique identifier associated with a thing in a given context. By default, this is the context of all known naming systems.

Notes:

- *A universal identifier allows all actions operating on a thing to interoperate using the same identifier.*
- *Universality in the context of all known naming systems is impossible in practice, but various strategies attempt to emulate it, including preferred identifiers, and services mapping between synonymous identifiers.*
- *Universality within an identifier's own context, on the other hand, is a common expectation for identifier management systems: namely, that any thing is identified by at most one identifier managed through the identifier management system. This allows the identifiers of that system to be used for deduplication, i.e. to ensure that only one instance of the thing identified is managed in the associated data management system.*

## 8.6 Persistent

*Subclass of: Identifier*

*Necessary Rule: Timespan-of-Persistent: Domain: Persistent; Range: Timespan*

*Refinement:*

*Type-of-Persistent: Domain: Persistent; Range: Quality*

Definition: A component is persistent if it is managed and maintained for a defined time span.

- *An entity is persistent if the element is managed and maintained.*
- *A quality is persistent if the quality value is managed and maintained.*
- *The time span is a property of the Persistent quality.*

Notes:

- *The time span is defined in the policies of an instance of an identifier system. The time span is not defined in the identifier system model unless persistence is the same for all instances of an identifier system.*
- *Persistent can be a second order quality, with other qualities and relations as its domain: persistence of actionability, of accountability, of association of an identifier.*
- *The timespan need not be open-ended or perpetuity. So long as the time span is defined and maintenance is consistent throughout the time span, the entity or quality is persistent even if the time span is a matter of days.*

## 8.7 Accountable

*Subclass of: Component*

*Necessary Rule: Accountable-to-Party: Domain: Accountable; Range: Party*

Definition: A component is accountable to a party if information on the provenance and custodianship of the component (i.e. the historical and current record of authorities over the component) exists, is maintained, is accessible by that party, and (where applicable) is resolvable by that party.

Notes:

- *Accountability applies to any component of an identifier management system.*
- *The main purpose of accountability is the ability to question the authority over a component about decisions taking in managing a component. This occurs in a general framework of establishing trustworthiness, and interacts with other policy and governance components. For instance, an authority may not be able to answer questions about their decisions in the absence of an audit trail (logs). These components are in scope of a general discussion of trustworthiness, but not in scope of accountability specifically.*
- *Accountability is realised through the maintenance of accountability metadata. Accountability metadata is data which allows parties to recover the provenance and custodianship of a component. It includes all provenance and authority metadata.*
- *The set of accountability metadata maintained for a component should be defined in the identifier system model. The definition should conform to the functional requirements on accountability data just given, but is otherwise at the discretion of the identifier system authority. Such definitions can include data schemas.*
- *Accountability does not model governance policy of an identifier system; accountability data MAY provide information needed for governance. Such governance is out of scope for the Identifier Systems ontology.*
- *The resolvability of custodianship required in the definition means that the party must be able to access (contact) the current owner, using data exposed through accountability data. If such access is not possible, then the component is no longer considered accountable. How such resolution is exposed is determined by the Identifier Systems Model.*

## 8.8 Trusted

*Subclass of: Entity*

*Necessary Rule: Trusted-by-Party: Domain: Trusted; Range: Party*

Definition: An entity (including identifiers and identifier systems) is trustworthy to a user if that user is confident that their use of the component will satisfy certain expectations. Trustworthiness is a subjective evaluation of an entity, but it can also be established through objective evaluation, which consists of verification actions with respect to the expected qualities.

Notes:

- *Trusted also applies to the information infrastructure of systems through which entities are published. This MAY include concerns over accessibility, performance, support, and backups. A cover term for these concerns is "reliability":, a subclass of Trusted applying to the user's interaction with the entity through the provided infrastructure, and subject to objective evaluation.*
- *Trusted is a combination of several other qualities, each of which is expected by the user. Different user communities have different expected qualities. An exhaustive definition of all such qualities is outside the scope of this Identifier Systems ontology.*
- *Accountability is one quality expectation typically included in trustworthiness. Reliability is another.*
- *Trustworthiness applies to the content of elements, as the outcome of curatorial actions. This MAY include concerns over accuracy, completeness, and updating.*
- *Trustworthiness is enabled through a governance structure and a policy structure. The scope and extent of those structures can be part of the evaluation of trustworthiness.*

## 8.9 Reserved

*Subclass of: Name*

See Reserve.

## 8.10 Published

*Subclass of: Entity*

See Publish.

## 8.11 Citable

*Subclass of: Entity*

See Cite.

## 8.12 Verified, Verifiable

*Subclass of: Entity*

See Verify.

## 8.13 Nameable

*Subclass of: Entity*

Definition: A component is nameable if it may be treated as a name.

Scope: defined in the identifier system policy or Identifier Systems ontology.

Notes:

- *The nameable quality provides for recursion in the model.*

## 9 Actions

The possible behaviours of an identifier system in the PILIN Identifier Systems ontology are defined using a set of actions.

- *Actions are non decomposable, atomic, ACID safe transactions.*
- *However, actions may be subclasses of other actions*
- *The result of applying an action MAY be:*
  - *(a [representation of an] instance of) an entity.*
  - *(the value of) an attribute of an entity.*
  - *a change in state to (an instance of) an entity, i.e., an update to the (binary) value of a quality of (an instance of) an entity.*
  - *a change in (the value of) an attribute.*
- *The Identifier Systems ontology places restrictions on the inputs and results of actions. Identifier system models may place further restrictions on the inputs and results of actions. Identifier system models may restrict the systems, users, and times of actions.*
- *Actions MAY be applied, via an identifier system model, to any of the elements in an identifier management system.*
- *Actions MAY be compounded to develop business processes and identifier system functionality through workflows and coordination processes; these are denoted composite actions. Composite actions are not defined in the Identifier Systems ontology.*
- *Actions are mapped into a business process workflow and functionality to define "service" requirements.*
- *Actions are triggered by actors. How to trigger actions is part of the design of an information system supporting an identifier system.*
- *When applied to an entity, the action operates on the entity as a non decomposable whole. While the entity MAY be composite, and the action MAY manipulate the parts of the entity, such manipulations SHALL be considered to be part of the internal working of the action.*
- *An identifier system MAY define events which manipulate the parts of an entity. The identifier system SHOULD define the transactional characteristics of the actions in the events workflow. Nothing in the event definition shall override the non decomposable, atomic, ACID safe transaction characteristics of the individual actions in the event workflow.*
- *Some action definitions are referred to their corresponding qualities.*

Two subclasses of actions are defined in order to model the action of Publish:

- *Curatorial actions are actions undertaken in order to realise or maintain desirable qualities of elements by changing their state. Curatorial actions occur by definition within the curation boundary of an identifier management system: if a new party become authorised to undertake a curatorial action, the curation boundary of the system expands to include them. Of the actions listed below, this includes Create, Register, Destroy, Reserve, Allocate, Publish, Identify, Assign.*
- *All other actions, which do not change the state of their inputs, are non-curatorial (Represent, Resolve, Cite, Query, Verify, Act). Non-curatorial actions may be triggered outside the curation boundary of an identifier management system (i.e. by external users). Curatorial actions may not—otherwise, this would by definition broaden the curatorial boundary to include the new actor.*

## 9.1 Cite

*Input: Thing; Output: Nothing*

Definition: Communicate a representation of a thing to an audience through some medium. An entity is citable if it can be cited.

Notes:

- *This action is distinct from the action of creating a representation for the thing to be communicated.*
- *This action is distinct from identifying a thing with a name as an identifier: the identifier still needs to be communicated to an audience.*
- *The Identifier Systems ontology presupposes that communication can only occur through signifiers: representation is the relation between the thing to be communicated and its chosen signifier.*
- *A recursive definition of the representation of identifiers is given under "Relations"*
- *The distinction between Identifying, Creating, and Citing may be illustrated as follows:*
  - *Identify: the phonetic word [tɹi:] is associated with the thing defined as "a woody plant with a trunk". The pair ([tɹi:], "a woody plant with a trunk") is an identifier.*
  - *Create Representation: Jack creates a representation of the identifier ([tɹi:], "a woody plant with a trunk") as the written word <tree>. This encodes the label of the identifier, in the encoding scheme of English spelling. (As usual for non-digital identifiers, the context is left implicit.)*
  - *Cite: Jack e-mails the word <tree> to Jill. Jack thereby represents the identifier ([tɹi:], "a woody plant with a trunk") to Jill as an audience.*
- *Identifiers may be citable. Identifier actions (e.g., service calls) may be citable.*
- *The typical use case is to cite a resolution action on an identifier, which allows the identifier to be resolved once accessed through the document. e.g.,*

*"<http://purl.org/1/321>". Cite and Resolve are logically separate: Cite provides the representation that Resolve is parameterised on.*

- *Any identifier may support one or more citation conventions. Citation conventions are defined through distinct serialisation and encoding schemes for entities.*
- *Making the entity citable SHALL result in the creation of a specific representation of the entity, e.g., a URL with specific syntax and semantics.*
- *Citation is meant to include the concept of "bookmark".*
- *Different values of the citable quality value MAY change the applicability of the policies that control the entity.*
- *Citable status MAY represent a composite of other qualities and attributes.*
- *Citable is explicitly defined as a separate quality, versus explicitly modelling it as attributes of an entity or as a type of status.*
- *The representation must be compatible with the medium. For instance, the representation must use Web-Encoding if the communication is to occur through the Web. This means that there are subclasses of "Cite" according to medium: an entity is "Web-Citable" if the representation can be embedded in a document on the web, "Print-Citable" if it can be embedded in a print document, and "Speech-Citable" if it can be embedded in spoken text (e.g. read out over the phone). An identifier in particular should be usable in non-digital environments.*

## 9.2 Create

*Input: Nothing; Output: Thing*

Bring a thing into being. Create is typically tightly coupled with Register in identifier management systems; but this is not always the case.

- *Labels and names may be created without being registered.*
- *Identifiers may be created without being registered, though this is problematic if the identifier is to be actioned by computer.*
- *Creation MAY be applied to:*
  - *any of the entities (those that are defined within an identifier system model).*
  - *an identifier system itself (its provisioning).*
- *Since Creation is a conceptual rather than a systems-dependent activity, its opposite Destroy is considered impossible in this ontology: a concept cannot be destroyed, though its representation can be deregistered.*

## 9.3 Register

*Input: Entity; Output: Nothing*

Definition: A party registers an entity if they make it be maintained (stored or represented) and managed in an identifier management system.

Notes:

- *Registration allows such desirable actions and qualities as resolve, verify, accountable, unique to be policed.*
- *By definition, registering an identifier means registering the association of a name with a thing.*
- *By implication, maintaining the identifier (as an association of a name with a thing) MAY require that the thing is also maintained.*
- *If it does, it MAY (but need not) require that the thing is maintained through the same system. Some entities, e.g., things that are identified in an identifier, are maintained external to the identifier system.*
- *Only concrete components can be registered. For instance, registering an identifier means that the association of a name with a thing is represented through association data stored and managed through a system. (See discussion, under Identify.)*
- *The set of quality values subject to registration (e.g., persistent, unique [single resolution], actionable) are defined and scoped within the identifier system model.*
- *Once an identifier is registered, qualities of the identifier reflecting this fact are assigned appropriate values by the identifier management system, following the system's policies.*
- *Populating (and updating) the (attribute) values of the qualities MAY be independent of registering the entity.*
- *Registration is modelled as distinct from creation in the Identifier Systems ontology; the identifier system policy MAY differ.*
- *Populating (and updating) of the (attribute) values of the qualities (for an instance of an entity) MAY be independent of creating the entity or its qualities.*
- *Attributes to be registered SHALL be defined in the identifier system model.*

## 9.4 Deregister

*Input: Entity; Output: Nothing*

Definition: Delete a registered entity from an identifier management system.

Notes:

- *Once an entity is deregistered, no other actions SHALL be applied to the entity.*
- *Deregistering an entity MAY NOT deregister (the reifications of) its associations registered in the identifier system.*
- *How an entity is deregistered, i.e., the deletion of data, is deferred to the design of an identifier system.*
- *Deregistering an identifier destroys the association, not the name or the thing.*
- *Deregister is the opposite of Register, not Create. It is meaningless to speak of un-creating (destroying) entities.*
- *The association between the name and the thing in a deregistered identifier may still be recovered via archival operation, e.g., in logs; but deregistering*

*the identifier ensures that it is no longer maintained, and no longer has the authority of the identifier system claiming it.*

## 9.5 Reserve

*Input: Entity; Output: Nothing*

Definition: Assign to a registered entity a “temporary” status or “in use” state.

Notes:

- *The reserved quality typically relates to incompletely populated identifier artefacts. This is a notion specific to only some identifier systems, and differs from system to system. It should not be considered a primitive quality, but modelled with other, existing qualities from the Identifier Systems ontology (e.g., not yet resolvable).*
- *An entity cannot remain reserved once it is published.*
- *Different values of the reserved quality value MAY change the policies that control the entity.*
- *Reserve is typically a batch process that generates a pool of entities.*
- *Reserve MAY be used to speed processing as a batch process.*
- *Reserve MAY be used to control generation, e.g., create a set of names that have ordered semantics.*

## 9.6 Publish

*Input: Component; Output: Nothing*

Definition: Informally, publishing a component is enabling access to that component by an end-user. A more formal definition of the action is as follows:

- An **administrator** of an identifier management system is a party authorised to trigger a curatorial action on some input through the system.
- To publish a component through a system is to authorise a party who is *not* an administrator of the system to trigger an action on the component through the system.
- The action, by definition, is non-curatorial. The party, by definition, is outside the curation boundary of the system.
- Publishing depends on the authorisation profile of the user. If an administrator is authorised to verify an identifier, this does not in itself change the state of the identifier; so the action of verifying is not curatorial. But because the administrator is also authorised to carry out curatorial actions, letting them verify the identifier does not count as publishing the identifier. Conversely, if we allow a non-administrator to start citing identifiers, this does count as publishing the identifier, even if the non-administrator cannot yet resolve the identifier.

An entity has the quality **Published** if it has had the Publish action applied to it.

Notes:

- *Enabling resolution of an identifier by non-administrators is the usual scenario for publishing an identifier. It is not the only possible scenario.*

- *Publish MAY be performed by changing one or more of the entity (attribute) values or quality values.*
- *When published, an entity becomes accountable to the parties it is published to. (For instance, changes in the entity now affect those parties, whereas changes in an unpublished entity do not.)*
- *The actual semantics of "published" are defined by an identifier system, which MAY represent a composite of other qualities and attributes.*
- *Whether an entity has been published MAY change the applicability of the policies that control the entity.*
- *Publishing MAY result in the dissemination of a specific representation of the entity, e.g., a URL with specific syntax and semantics. This dissemination is exemplified by citation.*
- *Cite and Publish are nevertheless independent actions. Citing an entity informs an audience that the entity exists; publish grants access to the entity through an access-controlled system. An entity may be cited but not published (in which case attempts to access the cited entity will fail), or published but not cited (in which case discovery of the accessible entity is unlikely).*

## 9.7 Identify

*Input: (Name, Thing); Output: Identifier*

Definition: Link a name with a thing in an identifier.

Notes:

- *The association "identify" has a set of quality values (e.g., persistent, actionable) that are defined by the information model for an identifier system.*
- *Registration is modelled as distinct from identifying in the Identifier Systems ontology.*
- *The identifier quality values are distinct from those of the name or the thing, e.g., association and name MAY have different persistence. That is because an identifier is a thing in its own right.*
- *The association of a name with a thing in an identifier must be somehow reified in order to be managed by an identifier management system, in accordance with system policy.*
- *A locator is an example of a typed attribute (e.g., a URI) that is part of the association between the name of the identifier and the thing.*
- *The reification of the association (such as a locator, or a dictionary definition) is termed "association data".*
- *The reification is typically either an alternate identifier for the same thing, or the result of a Resolve action ("information on how to access the thing"). A locator may be viewed as either. In both cases, they are information about the thing, rather than representations of the thing itself.*

## 9.8 Act

*Input: Entity; Output: (Thing, Nothing)*

Definition: Perform a non-curatorial action on an entity. Act is a superclass for those actions which an external user may perform through the identifier management system. An entity is actionable if the (named) action may be applied to that entity to perform some process.

Notes:

- *Not all entities are necessarily actionable through a given action. The set of entities that the action may be applied to SHALL be defined in the identifier system model.*
- *Act does not include curatorial actions. For instance, an identifier may be updated (curatorial) without being actionable (non-curatorial)*
- *The behaviour of the action is defined in terms of the entity, i.e., the instance of the entity is used by the action. Its action may be further defined in terms of the things associated with the entity (e.g., retrieval of associated thing). Some actions MAY be defined independently of an instance of the entity, i.e., they are defined for the entity itself.*
- *The set of actions (i.e., subclasses of the Act class) supported by an identifier management system is defined by the identifier system model.*
- *When an identifier is referred to as actionable, this is understood to mean actionable in terms of the association between name and thing. If the action involved only the name, the reference should be to an actionable name rather than an actionable identifier. Therefore Cite and Represent are not considered as subclasses of Acting on an identifier, and an entity is not actionable just because it is citable.*
- *On the other hand, Query and Verify do count as subclasses of Act under this definition. This means that actionability of an identifier must be related to a particular user profile to be meaningful: identifiers may be actionable by their administrators but not end users.*
- *An entity is extensibly actionable if the range of actions that can operate on it is not fixed, but can be expanded. In particular, users outside the curation boundary of the identifier management system should be able to set up their own actions on identifiers, using data accessible from the identifier management system.*

## 9.9 Resolve

*Input: Name; Output: Information*

Definition: Get information on how to access the thing identified by the name in an identifier. An entity is resolvable if the action Resolve may be applied to that entity.

Notes:

- *Resolve is a subclass of Act: it is a non-curatorial action, and is dependent on the association between name and thing.*
- *Resolve is explicitly defined as a separate quality (versus just modelling it as instance of the actionable quality), as the attributes and other definitions needed by the action are required to be stated to define the identifier system model.*
- *The definition of the Resolve action in an identifier system model defines how resolution is realised, in terms of the attributes that are part of the*

*association, e.g., returning a locator attribute. It also describes how attributes and qualities affect the resolution process.*

- *The definition of the Resolve action in an identifier system model describes what to do when there are multiple ways of resolving the entity, i.e., the meaning of "multiple-resolution".*
- *The item(s) returned from the resolution process is an entity, and thus it has qualities, e.g., it MAY be actionable.*
- *Resolve is distinct from Retrieve, although the two are often conflated.*
- *Revealing the association between the item and the thing, without providing any information on how to access the thing, does not count as resolution—e.g., mapping the identifier to a non-actionable other identifier, providing a dictionary definition.*
- *There may be subclasses of "Resolve" according to how the action is transacted. An identifier is "Internet-Resolvable" if the information on how to access the thing identified can be requested and consumed through a well-defined Internet application protocol. An identifier is "Web-Resolvable" if that protocol is a defined web application layer protocol. An example of the latter is a Web service query on a Handle, with its request in HTTP GET and returning a URL.*

## 9.10 Retrieve

*Input: Name; Output: Thing*

Definition: Obtain or access the thing identified by the name in an identifier. An entity is retrievable if the action Retrieve may be applied to that entity.

Notes:

- *Retrieve is dependent on the information on how to access the thing identified by an identifier, which is returned by Resolve; so the Retrieve action on an identifier presupposes a Resolve action.*
- *The Retrieve action is compound, and is made possible through a combination of identifier management system functionality (Resolve to a retrieval key) and data management system functionality (Obtain through the retrieval key). It is not the responsibility of the identifier management system alone; that responsibility ends with the Resolve action.*

## 9.11 Query

*Input: Entity; Output: Information*

Definition: Obtain selected information (state, qualities, and attributes) for (an instance of) an entity.

Notes:

- *The information to obtain is specified in the query. It may include all available information.*

## 9.12 Verify

*Input: Entity; Output: Boolean*

Definition: Confirm that a quality value for an (instance of an) entity managed in a system reflects the true value of the quality in the world, according to a defined set of identifier system policies. If a quality can be verified for an entity, that quality is **verifiable** for the entity. If a quality has been successfully verified for an entity, then that quality is **verified** for the entity.

Notes:

- *As defined, the verify action is restricted to quality values (and by extension attribute values) of an entity managed in a system. If an entity is not managed, verification is impossible. If the target of verification is not a quality or attribute value, verification is not well-defined.*
- *Although entities themselves are not verified, existence can be counted as a verifiable quality.*
- *Entity qualities MAY be verified independently of each other, e.g., a link checker verifies the resolvable quality, without verifying accountability.*
- *The identifier system policy used in verification may depend on the quality values of an entity, e.g., "publishing" an entity may change the verification requirements for other qualities of that entity.*
- *The meaning of verify for an entity is defined by the identifier system.*
- *Verified MAY be compounded with other qualities.*
- *Different values of the verifiable or the verified quality MAY change the policies that control the entity.*

## 10 Appendix 1: Representations of the same name

### 10.1 hdl:102.100.272/XYZZY and info:hdl:102.100.272/XYZZY are representations of the same name.

If two representations of names have the same label, and different identifiers are used in the two representations to represent their context, but the context identified in the two representations is also the same (e.g., hdl:/ vs. info:hdl/), then the two representations are of the same name by definition.

$$\begin{aligned} & \exists C1: \text{Context}, \exists L1: \text{Label}, \exists N1: \text{Name} \mid \{NR1\} \subset \\ & \{ CR1: \{ N1: \text{Name} \mid N1 \rightarrow C1 \cdot \text{reprs}(N1) \}, LR1 : \text{reprs}(L1) \mid \\ & S1 : \text{serialisations}(CR1, LR1) \cdot C1 : \text{Encoding-Schemes: encode}(S1, \\ & C1) \} \& \\ & \exists C2: \text{Context}, \exists L2: \text{Label}, \exists N2: \text{Name} \mid \{NR2\} \subset \\ & \{ CR2: \{ N2: \text{Name} \mid N2 \rightarrow C2 \cdot \text{reprs}(N2) \}, LR2 : \text{reprs}(L2) \mid \\ & S2 : \text{serialisations}(CR2, LR2) \cdot C2 : \text{Encoding-Schemes: encode}(S2, C2) \} \\ & \& \end{aligned}$$

$$\begin{aligned}
 &L1 = L2 \ \& \\
 &N1 \neq N2 \ \& \\
 &C1 = C2 \Rightarrow \\
 &\quad \text{same\_represented\_name}(NR1, NR2)
 \end{aligned}$$

The name representations `hdl:102.100.272/XYZZY` and `info:hdl:102.100.272/XYZZY` use different identifiers for the same context (i.e., different contexts of contexts): one is a hdl identifier, one is an info-uri identifier. Because the same context is identified in the two representations, and the labels are the same, the two representations are themselves representations of the same name, "XYZZY in the 102.100.272 Handle server".

## 10.2 `hdl:102.100.272/XYZZY` and (URL-encoded) `hdl:%2A102.100.272%2FXYZZY` are representations of the same name.

If the identifier for a context is represented in two different encodings, those different encodings also identify the same context. Two representations with the same label and different encodings of the same context are therefore representations of the same name.

$$\begin{aligned}
 &\exists C1: \text{Context}, \exists L1: \text{Label}, \exists CR1: \text{Representations} \mid \{NR1\} \subset \\
 &\quad \{ CR1 \in \text{reprs}(C1), LR1 : \text{reprs}(L1) \} \mid \\
 &S1 : \text{serialisations}(CR1, LR1) \cdot C1 : \text{Encoding-Schemes: encode}(S1, C1) \} \\
 &\& \\
 &\exists C2: \text{Context}, \exists L2: \text{Label}, \exists CR2: \text{Representations} \mid \{NR2\} \subset \\
 &\quad \{ CR2 \in \text{reprs}(C2), LR2 : \text{reprs}(L2) \} \mid \\
 &S2 : \text{serialisations}(CR2, LR2) \cdot C2 : \text{Encoding-Schemes: encode}(S2, C2) \} \\
 &\& \\
 &L1 = L2 \ \& \\
 &CR1 \neq CR2 \ \& \\
 &C1 = C2 \Rightarrow \\
 &\quad \text{same\_represented\_name}(NR1, NR2)
 \end{aligned}$$

## 10.3 If I claim `hdl:102.100.archer` is an alias for `hdl:102.100.821`, then I am also claiming `hdl:102.100.archer/XYZZY` and `hdl:102.100.821/XYZZY` are representations of the same name

If two identifiers for a context are synonymous according to an authority, then the same label represented with those two identifiers for the context gives representations of the same name, according to that authority. e.g.

- *An authority defines the Handle context label `102.100.archer` to be synonymous with `102.100.821` (an aliasing operation)*
- *therefore the names `hdl:102.100.archer/XYZZY` and `102.100.821/XYZZY` are representations of the same name, according to that authority*

$$\begin{aligned}
 &\exists C1: \text{Context}, \exists L1: \text{Label}, \exists N1: \text{Name} \mid \{NR1\} \subset \\
 &\quad \{ CR1: \{ N1: \text{Name} \mid N1 \rightarrow C1 \cdot \text{reprs}(N1) \}, LR1 : \text{reprs}(L1) \} \mid
 \end{aligned}$$

$$\begin{aligned}
& S1 : \text{serialisations}(CR1, LR1) \cdot C1 : \text{Encoding-Schemes: encode}(S1, C1) \} \\
& \& \\
& \quad \exists C2 : \text{Context}, \exists L2 : \text{Label}, \exists N2 : \text{Name} \mid \{NR2\} \subset \\
& \quad \{ CR2 : \{ N2 : \text{Name} \mid N2 \rightarrow C2 \cdot \text{reprs}(N2) \}, LR2 : \text{reprs}(L2) \mid \\
& \quad S2 : \text{serialisations}(CR2, LR2) \cdot C2 : \text{Encoding-Schemes: encode}(S2, C2) \} \\
& \quad \& \\
& \quad L1 = L2 \ \& \\
& \quad \text{synonym}(N1, N2, \text{authority}) \Rightarrow \\
& \quad L1 = L2 \ \& \ \text{claims}(\text{authority}, \text{equivalent}(N1, N2, t)) \Rightarrow \\
& \quad L1 = L2 \ \& \ \text{claims}(\text{authority}, C1 = C2) \Rightarrow \\
& \quad \text{claims}(\text{authority}, L1 = L2 \ \& \ C1 = C2) \Rightarrow \\
& \quad \text{claims}(\text{authority}, \text{same\_represented\_name}(NR1, NR2))
\end{aligned}$$

However, authorities are fallible, so the authority for a claim of synonymy between contexts must establish trust.

## 10.4 The concrete names <http://purl.kerry.org/XYZZY> and <hdl:/100/XYZZY> cannot be the same concrete name

Two different concrete systems define two different concrete contexts, and those contexts are not the same thing. Therefore the identifiers *for those concrete contexts* are not synonyms. Therefore the names in those two concrete contexts cannot be the same name.

- An abstract context "Kerry's Library" is realised by the concrete contexts <hdl:/100> and <http://purl.kerry.org>.
- The concrete names <hdl:/100/XYZZY> and <http://purl.kerry.org/XYZZY> realise the abstract name `Kerry's Library::XYZZY`.
- <hdl:/100> as a context is distinct from <http://purl.kerry.org>, even though both are realising the same abstract context.
- Therefore the context identifiers <hdl:/100> and <http://purl.kerry.org> are not synonyms: they identify distinct contexts.
- Therefore <hdl:/100/XYZZY> and <http://purl.kerry.org/XYZZY> are not the same name. They are homologous names, as defined above.

## 11 Appendix 2: Use of Homology

An abstract identifier might be equivalent only to a concrete identifier with a different label (e.g., the abstract identifier's label was not available for registration in the concrete context). In that case, we do not regard the concrete identifier as a realisation of the abstract.

Similarly, an abstract identifier might be equivalent to two concrete identifiers under the same authority, but the identifiers do not all have the same label (e.g., because the abstract identifier's label was not available for registration in one of the concrete contexts). In that case, we do not regard the two concrete identifiers as homologues, but only as synonyms.

- *This is an expedient choice. The alternative would be to allow a relation between labels within scope of an abstract context --- but such a relation is already covered in the Identifier Systems ontology by synonymy.*
- *e.g., The abstract identifier "Kerry's Library:elite" is to be realised in the concrete contexts <http://purl.kerry.org> and `hdl:100`. However, the Handle server is not under the exclusive policy control of "Kerry's Library", and imposes its own label policy --- that labels must not contain letters. As a result, the concrete identifiers registered are <http://purl.kerry.org/elite> and `hdl:100/7334`. The concrete label 7334 has an idiosyncratic relation with the abstract label "elite", which we have elected not to represent in the Identifier Systems ontology as a relation distinct from synonymy. So we choose not to model the relation between `hdl:100/7334` and "Kerry's Library:elite" the same way we model the relation between <http://purl.kerry.org/elite> and "Kerry's Library:elite":*
- *<http://purl.kerry.org/elite> realises "Kerry's Library:elite".*
- *`hdl:100/7334` is a synonym of <http://purl.kerry.org/elite>, according to the authority owning Kerry's Library.*
- *`hdl:100/7334` is not a homologue of <http://purl.kerry.org/elite> (they are not the same label registered in two different contexts).*
- *`hdl:100/7334` does not realise "Kerry's Library:elite", because it does not have the same label (even if it has the same referent) .*

However, given that the claim of synonymy has the authority of Kerry's Library behind it, the distinction between homology and synonymy is not important to external systems.

- *External systems will only treat two identifiers as synonyms if an authority exposes that information.*
- *External systems will only treat two identifiers as homologues only if an authority exposes that information; the identity of labels between two contexts may be a coincidence.*
- *In the example above, an external system cannot assume that <http://purl.kerry.org/elite> and `hdl:100/elite` are homologues, pointing to the same thing --- even if it knows that both <http://purl.kerry.org> and `hdl:100` are realisations of the same library. That is because, in the general case, the same label may not be available for registration in both systems.*
- *An external system can only treat these two identifiers as equivalent if an authority tells it that they are equivalent --- even if that authority is the Kerry Library authority stating "all <http://purl.kerry.org> and `hdl:100` labels are equivalent".*
- *But the minute that claim of equivalence is made by an authority, the relation between the identifiers becomes one of synonymy. And it would be the same relation of synonymy even if the labels were different.*
- *The only distinction from normal synonymy is that the authority claiming equivalence overlaps with the authority over the shared abstract context. Otherwise, an external system does not make any distinction between the two.*
- *So external systems depend on explicit exposure of equivalence information, which privileges synonymy over homology.*

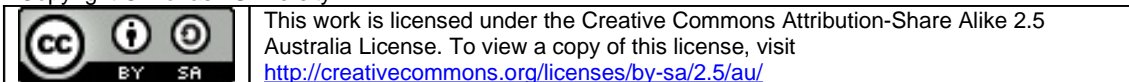
## 12 References

ISO 2146: *Information and Documentation - Registry Services for Libraries and Related Organisations* (ISO TC46 SC4 Working Draft, 13 December 2005). Word document. Available: <http://www.nla.gov.au/wgroups/ISO2146/n197.doc>

OWL: *OWL Web Ontology Language Overview*. W3C Recommendation 10 February 2004. <http://www.w3.org/TR/owl-features/>

RFC 2119: Bradner, S. 1997. *Key words for use in RFCs to Indicate Requirement Levels*. <http://www.ietf.org/rfc/rfc2119.txt>

Copyright © Monash University



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.