Biodiversity Information Standards (TDWG)

TDWG Life Sciences Identifiers (LSID) Applicability Statement

Date:
17-Dec-2007

Status:
TDWG Draft Standard

Permanent URL:
http://www.tdwg.org/standards/150

Task Group:
TDWG Globally Unique Identifiers Task Group (GUID)

Contributors:
Ricardo Pereira (TDWG Infrastructure Project)
Donald Hobern (Global Biodiversity Information Facility)
Roger Hyam (TDWG Infrastructure Project)
Lee Belbin (TDWG Infrastructure Project)
Kevin Richards (Landcare Research)
Stan Blum (California Academy of Sciences)

Abstract:
This document
1. Provides guidance on how to use LSID to meet specific requirements of the biodiversity information community and
2. Defines how to identify shared data objects in biodiversity information applications using Life Sciences Identifiers (LSID).

Legal Notice:

Disclaimer:

Table of Contents

Index of Recommendations

LSID Assignment

Reuse of Existing LSIDs

LSID Authority Identification

Motivation

The TDWG Globally Unique Identifiers Task Group (TDWG GUID) [1][2], after meeting twice in 2006, recommended the use of the Life Sciences Identifiers (LSID [3]) to uniquely identify shared data objects in biodiversity information applications wherever appropriate.

The LSID specification will support applications within the Life Sciences domain. There are points within the specification where implementers may choose the most appropriate of several options. This flexibility means that, to achieve maximum compatibility within any sub-domain, implementers have to develop and maintain applications that support all available options. Choosing a subset of options from the specification that is appropriate for the biodiversity informatics community enables timely and efficient roll out of LSIDs that maintain full compatibility with the broader LSID community.

This applicability statement specifies the subset of options for use in the biodiversity information community.

Terminology and Definitions

This specification assumes that the reader is familiar with the LSID specification [3] (http://www.omg.org/cgi-bin/doc?dtc/04-05-01) and the terminology used in that document.

Throughout this document we use the term **object** to refer to an entity or information about it. We refer to the organizations which disseminate objects as **providers**.

An **LSID HTTP proxy** is a web service that resolves LSIDs by returning the results of the getMetadata() call via **HTTP GET**. The **proxy version** of an LSID is created by concatenating the proxy web address (such as http://lsid.tdwg.org/) to the LSID, as in:

```
http://lsid.tdwg.org/urn:lsid:authority:ns:obj:rev
```

The key 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 [4].

Throughout the document we present each recommendation inside a box followed by the rationale behind the recommendation as in the example below.

1. Recommendation statement

Statement of reasons behind the recommendation.

1. LSID Assignment

> 1. An LSID **must** be assigned to a single object[1].

By definition, an LSID uniquely identifies a single object [3]. If the same LSID is assigned to more than one object, it is impossible to determine which object the LSID identifies by only inspecting the identifier.

> 2. Only one globally unique identifier **should** be assigned to each object.

Assigning more than one LSID to a single object is counter-productive because-

- it makes it more difficult for clients to check object identity and detect duplicates;
- it increases the costs of maintaining the identifiers (e.g., more records are needed, more effort is required to prevent and correct assignment errors).

> 3. Providers **should only** assign LSIDs to objects they are the authorities for.

By assigning an LSID to an object, a provider is stating that it is responsible for it. Clients are able to retrieve attribution information about the object by resolving its LSID. That creates a strong bond between the object and its owner.

Providers should express object attribution using the Dublin Core [5] metadata term **creator**.

Examples of objects in the biodiversity information domain that **should** be assigned LSIDs-

- scientific names,
- taxonomic concepts,
- species observations,
- specimens,
- collections,
- images of type specimens,
- images, videos, and sound recordings of specimens.

> 4. Providers **should not** assign LSIDs to objects that already have more widely accepted identifiers such as publications with DOIs.

Assigning an LSID to an object that already has a more widely accepted identifier violates recommendations 2 and 3 (when the provider assigning the LSID is not the object authority). Moreover, clients are likely to use the more accepted identifier to refer to the object, so the LSID only adds ambiguity to the system.

---

[1] From the LSID specification.

> 5. Aggregators **should** assign new LSIDs to derived objects.

Aggregators add value by collecting and integrating data from distributed, heterogeneous sources. Added value may come from:

- Integrating objects into homogeneous datasets;
- Verifying consistency;
- Georeferencing locality descriptions;
- Checking spelling or
- Resolving ambiguities

Aggregators then serve the value-added objects to clients and become the authority for the modifications made to the original objects. According to recommendation 3, aggregators **should** assign LSIDs to value-added objects they create. They should also reference the aggregated objects (see recommendation 9).

> 6. LSIDs **should** be resolvable.

LSIDs are not guaranteed to be resolvable. To attain all benefits of LSIDs however, such as source attribution, providers of LSIDs **should** resolve their LSIDs indefinitely. Providers should be discouraged from not resolving or temporarily resolving an LSID.

Clients who rely on persistent object data or metadata could make copies.

> 7. The labels *urn* and *lsid,* and the authority identification for an LSID **should** all be lowercase.

This allows clients to verify object equivalence (or lack of) using simple case-sensitive comparisons of identifiers. This comparison can be case-sensitive, as long as this recommendation is followed.

This recommendation allows RDF toolkits and *reasoners* to merge RDF graphs properly when LSIDs are used as node identifiers.

The namespace, object and revision identifications may be expressed uppercase or lowercase.

2. Reuse of Existing LSIDs

> 8. Information systems **should** use LSIDs when available to refer to external objects.

Information systems keep relationships between objects from different sources. The reference to the original object is usually lost or weakened due to the lack of a standard object identifier. LSIDs allow information systems to retain persistent references to the original sources of information.

> 9. Aggregators **should** use LSIDs and the Dublin Core metadata term **source** to link derived objects to their sources.

Using LSIDs to link value-added objects to their sources is important to-

- Give the value-added object proper attribution. Clients may use the LSID of the source to retrieve information about the original object and its creator.
- Let clients detect duplicates of the original object that may have been modified and served by different providers.

The Dublin Core term **source** is most appropriate to express this relationship because it has the appropriate meaning and is part of the most popular metadata vocabulary available.

3. LSID Authority Identification

The **authority identification** is a string, usually a domain name, used to identify the authoritative source of a set of LSIDs. The **authority identification** part of an LSID is underlined below:

```
urn:lsid:authority.org:namespace:object:revision
```

> 10. A provider **should** use a domain name registered to it as an authority identification.

Using a domain name registered to the provider as an LSID authority identification increases the provider's ability to continue to resolve the identifiers. By using a domain name registered to another organization, a provider is delegating control of their LSIDs to that organization. Delegation will reduce the longevity of the original provider's issued LSIDs.

> 11. A provider **should** plan to control the domain names it uses as authority identifications for as much time as possible.

Loosing control over a domain name used as an authority identification renders all LSIDs under that authority unresolvable. Providers should choose authority identifications and domain names that are likely to remain under their control for a long time.

> 12. A provider **should** transfer control of domain names to a successor if the names are forgone.

To ensure that LSIDs remain resolvable even after a provider ceases to exist, a provider must make arrangements to transfer control of the domain names it uses as LSID authority identifications. When the provider is dissolved, its successor is able to set up an LSID Authority to resolve the identifiers it inherits.

> 13. Organizations susceptible to name changes **should** use domain names that will remain effective as authority identifications through reorganisation changes.

Government reorganizations may render departments incapable of maintaining domain names that they may have used as LSID authority identifications.

Organizations should register neutral and stable domain names such as project names for authority identifications. Alternatively, those organizations may request a TDWG authority identification (see next recommendation).

> 14. If a suitable domain name is not available or likely to be unstable, request an authority identification from TDWG

There are cases, such as in government department reorganizations, in which a provider may loose control of a domain name used as LSID authority identification. If a provider cannot register for a more persistent domain name, it **should** apply for a TDWG authority identification by following the instructions at: http://www.tdwg.org/activities/online-services/lsid-authority-ids/.

3.1. Using Multiple Authority Identifications to Separate Sets of LSID

> 15. Providers **should** use separate authority identifications for objects that are likely to be moved to different servers or transferred to new owners.

An organization may offer to resolve the identifiers on behalf of a provider who is unable to for technical reasons. The host organization should use a distinct authority identification for those LSIDs to make it easier to move the resolver to a separate server. Separate DNS records are set for each category and requests are routed independently.

Consider that the original provider and the organization resolving LSIDs on its behalf have the domain names `provider.org` and `host.org` respectively registered to them. There are three possibilities for authority identifications-

1. A sub-domain of `host.org`, such as `provider.host.org` (**not recommended** – see justification below);

2. The domain `provider.org` or a sub-domain thereof, such as `my-lsids.provider.org`;

3. An LSID authority identification assigned by TDWG, such as: `provider.lsid.tdwg.org`.

We **do not recommend** alternative #1 because it ties the LSIDs to the holder of domain `host.org`. This assignment limits the possibilities of transferring the LSIDs to a new owner or back to the original provider.

We **do recommend** alternative #2 when the original provider has complete control over its domain name (`provider.org` in the example) and it is not subject to constraints from recommendation 13. This alternative is not feasible in most cases however, because one of the barriers to setting up an LSID resolver is the lack of control over a registered domain name. In this situation, we **recommend** alternative #3.

> 16. Providers **should not** use separate authority identifications to split LSIDs by categories such as departments, collections and data types - unless the objects are likely to be transferred to new owners or served from different servers. Otherwise, LSID namespaces **should** be used to split LSIDs by categories.

It is possible to separate LSIDs by departments, collections, data types, for example, using authority identifications instead of namespaces. This practice is only advised if there is a chance the objects will be handed to a new owner. If that is not the case, we **recommend** that providers use a single authority identification and multiple namespaces to partition LSIDs across different categories.

4. LSID Namespace Identification

Namespace identifications are used to partition objects across different categories. The namespace identification is the underlined part of the LSID below:

`urn:lsid:authority.org:`**`namespace`**`:object:revision`

> 17. Providers **should** use namespace identifiers to split LSIDs across different categories.

A provider can use LSID namespaces to split identifiers across different categories, such as object type, scientific or taxonomic discipline, departments, collections and projects. Namespaces help distinguish objects of different types that have the same identifier.

5. LSID Object Identification

The **object identification** is the part of an LSID used to distinguish objects within the same namespace. The object identification must be unique for all objects in the same namespace.

The **object identification** part of an LSID is underlined below:

```
urn:lsid:authority.org:namespace:object:revision
```

> 18. Providers **should** use well-established locally unique and immutable object identifiers as LSID object identifiers.

Many providers already tag their objects with well established unique identifiers, such as:

- GenBank accession numbers;
- Integrated Taxonomic Information System (ITIS) Taxonomic Serial Number (TSN); and
- Catalogue of Life (CoL) Taxon ID.

These identifiers are good candidates for LSID object identifications. In the absence of well established locally unique identifiers, providers should create locally unique identifiers for their objects and use them as the LSID object identification.

> 19. LSID Authorities **should not** use the primary key of relational database tables as object identifications. Providers **should** create an extra column in the table (or a separate table) to manage the LSID independently of the primary key.

Providers implementing LSIDs often consider primary keys of relational database tables as LSID object identifications. We advise against that practice because primary keys may change if the database is reorganized or the data is transferred elsewhere.

Database administrators may create a new column on the affected table and copy the original primary key values into that column. Alternatively, administrators may create a completely separate table to manage the object identifications and relate both tables using foreign keys. The system that manages that database table must generate unique object identifications and store them into that new column or table. The LSID Authority must in turn use the new column or table when resolving LSIDs instead of the primary key.

6. LSID Revision Identification and Versioning

The revision identification is an optional part of an LSID that is used to manage revisions of a single object that varies over time. The optional revision identification is underlined below:

```
urn:lsid:authority.org:namespace:object:revision
```

> 20. LSID Authorities **should** use the revision identifier to manage object that are revised over time.

The revision identification allows independent management of object revisions. While the object identifier is used to uniquely identify an object within a namespace, the revision identifier is used to distinguish between revisions of an object that is modified over time.

> 21. Clients **must not** try to infer relationships between objects based on the revision identification or any other part of an LSID. Instead, clients **must** dereference the LSID and retrieve any assertions about revisions from the returned metadata..

Clients may be tempted to infer relationships between objects associated with LSIDs that differ only on the revision identifier. This practice **is not encouraged** because the semantics of revision identifiers is not defined in the LSID specification. Clients cannot interpret the meaning of revision identifiers on LSIDs alone.

Therefore it is **bad practice to**:

- Remove the revision part of an LSID and dereference the resulting identifier to get the most update version of a object, or
- Perform any arithmetic operation (i.e. sum, subtraction) to the revision identification of an LSID and deference the resulting identifier to get the next or previous revision of an object.

Instead, clients must dereference the LSIDs and retrieve any information regarding versioning from the metadata associated with the object.

> 22. LSID Authorities **should** use appropriate metadata properties to represent relationship between revisions of an object.

LSID Authorities should use the following Dublin Core RDF and OWL properties to represent relationships between revisions of an object-

- **dcterms:replaces** – Points to the revision superseded by the revision at hand.

- **dcterms:isReplacedBy** – Points to a newer revision that supersedes the revision at hand.

- **dcterms:hasVersion** – Links an object to its revisions, regardless of whether it supersedes or is superseded by the other revisions.

- **owl:versionInfo** –String with information about the revision, such as the LSID revision identification and RCS/CVS keywords.

For example, the International Plant Names Index (IPNI) keeps in its database several revisions of the scientific name for *Gentlea costaricensis*. Suppose that the revisions are labelled **r1**, **r2** and **r3** and that each of these revisions, except the original revision **r1**, were originated from changes to the previous revision. A possible RDF document representing these revisions could be the following.

```
<rdf:RDF>
   <tcs:TaxonName rdf:about="urn:lsid:ipni.org:names:907328-1:r3">
     <dc:title>Gentlea costaricensis Lundell</dc:title>
     <owl:versionInfo>r3</owl:versionInfo>
     <dc:replaces>urn:lsid:ipni.org:names:907328-1:r2</dc:replaces>
   </tcs:TaxonName>

   <tcs:TaxonName rdf:about="urn:lsid:ipni.org:names:907328-1:r2">
     <dc:title>Gentlea costaricensis Lundel</dc:title>
     <owl:versionInfo>r2</owl:versionInfo>
     <dc:replaces>urn:lsid:ipni.org:names:907328-1:r1</dc:replaces>
     <dc:replacedBy>urn:lsid:ipni.org:names:907328-1:r3</dc:replacedBy>
   </tcs:TaxonName>

   <tcs:TaxonName rdf:about="urn:lsid:ipni.org:names:907328-1:r1">
     <dc:title>Gentlea costaricensis C.L.Lundell</dc:title>
     <owl:versionInfo>r1</owl:versionInfo>
     <dc:replacedBy>urn:lsid:ipni.org:names:907328-1:r2</dc:replacedBy>
   </tcs:TaxonName>
</rdf:RDF>
```

In some instances, LSID authorities may want to assign an LSID to the most up-to-date version of an object that changes over time. These object revisions are called "versionless" or "hub" objects.

As depicted in the example below, authorities should use-

- the property **versionedAs** from the TDWG LSID Vocabularies to refer to a revision that represents the current state of a "hub" object; and

- the Dublin Core term **hasVersion** to link an object to its revisions.

```
<rdf:RDF>
   <tcs:TaxonName rdf:about="urn:lsid:ipni.org:names:907328-1">
     <dc:title>Gentlea costaricensis C.L.Lundell</dc:title>

     <tdwg:versionedAs>urn:lsid:ipni.org:names:907328-1:r3</tdwg:versionedAs>

     <dc:hasVersion>urn:lsid:ipni.org:names:907328-1:r1</dc:hasVersion>
     <dc:hasVersion>urn:lsid:ipni.org:names:907328-1:r2</dc:hasVersion>
     <dc:hasVersion>urn:lsid:ipni.org:names:907328-1:r3</dc:hasVersion>
   </tcs:TaxonName>
</rdf:RDF>
```

In the example above, the object described (identified by **urn:lsid:ipni.org:names:907328-1** is the "hub" object, and **urn:lsid:ipni.org:names:907328-1:r3** points to its latest revision (a copy of itself). The objects **r1**, **r2** and **r3** are revisions of the "hub" object. It is not required that the hub object links to all previous revisions using **hasVersion** although it is desirable. The **versionedAs** property points to the start of the linked list of revisions and so gives access to all revisions.

7. LSID Opacity

23. Clients in general **must** consider LSIDs as opaque strings (even though they may not be opaque). Clients **must** dereference an LSID if they need any information about the object.

The LSID specification and this Applicability Statement embed meaning into identifiers. For example, a user may be able to infer the source of the LSID from its authority identifier, or its type from the namespace. Software developers may feel compelled to extract information from LSIDs to avoid the cost of dereferencing them. This process is considered **bad practice** because the semantics of the last 3 parts of LSIDs (namespace, object, and revision identifications) are not defined.

While clients may be able to interpret the meaning of these strings for some LSIDs, it is not guaranteed that they will be able to do so for all LSIDs.

The following are the exceptions to the requirement of opacity:

- An LSID authority has to interpret each part of their LSIDs to work properly.

- LSID resolvers have to use the authority identification part of LSIDs to locate the LSID authority. Resolvers do not need to use the other parts of the LSID individually to dereference them.

- Clients aware of LSIDs use the label "*urn:lsid:*" to recognize the string as an LSID.

- Clients not aware of the LSID specification use the label "*urn:*" to recognize the string as a Universal Resource Name, i.e., a generic, non-locatable identifier.

8. LSID Data

> 24. LSID data **must never** change.[2]

According to the LSID specification, data associated with an LSID (i.e., the content returned by the getData method) must never change. This is a requirement for the getDataByRange method to work as expected. That method has two parameters that define the starting point and the length of the subset of the data to be returned. If the data associated with the object changes, subsequent calls to getDataByRange may yield different results, and this is not permitted in the specification.

> 25. Providers **should not** encode data in formats such as XML that may change the exact sequence of bytes.

Data exchanged in biodiversity information systems are commonly encoded in XML. Those data however, may not be returned by the getData method if the sequence of bytes changes.

If XML is used to encode LSID data, the provider must ensure that the sequence of bytes returned (i.e., the XML serialization) remains constant. This assumption may not be guaranteed when the provider uses a third party XML library to output data because the serialization may change from one version of the library to the next.

When using XML to encode LSID data, we **recommend** that the provider store the XML data as binary data (i.e., not processing nor parsing it) to avoid undesired changes to the sequence of bytes that are returned in getData calls.

> 26. Providers **should not** return irrelevant data in LSID getData calls.

Some providers feel compelled to return something in the LSID getData even if it is not appropriate to do so. Core objects in biodiversity information systems such as taxonomic names, concepts, specimens and observations are not associated with immutable sequence of bytes that could be returned in the LSID getData call. It is reasonable not to return anything in the getData method.

> 27. Providers **should not** use getData just to assert that some attributes of objects are immutable.

It is considered **bad practice** to use getData to inform clients that some object attributes are immutable. Such assertions are best expressed in the metadata using appropriate predicates.

---

[2] As defined in the LSID specification

9. LSID Metadata

> 28. LSID metadata **may** change.

According to the LSID specification, metadata associated with an LSID may change. Clients who need metadata about an object to persist must keep a copy of it.

> 29. The default metadata response format **must** be RDF serialized as XML.

Metadata associated with an LSID is returned by the getMetadata() call. If no format is specified in the request, LSID authorities resolving identifiers associated with biodiversity objects **must** return metadata in RDF format by default. Other formats may be returned if supported by the authority. Format is negotiated between client and authority via the parameter *accepted_formats* of the *getMetadata* call.

The default return type in the LSID specification is RDF. Parsing arbitrary formats make the implementation of client applications problematic. Using RDF as the default format does not preclude the use of other formats as the non-default return types as stipulated in the LSID specification.

> 30. HTTP GET **must** be the default binding for LSID *getMetadata* calls.

All LSID authorities resolving identifiers associated with biodiversity information objects must implement the HTTP GET binding for LSID *getMetadata* calls. This does not preclude binding to additional protocols. The *getData* call does not have to be bound to HTTP GET although it is desirable in most cases.

The *getData* LSID call does not need to be implemented, and in most cases will be expected to return nothing. Except for binary encoded objects such as images, audio, and video, most information in the biodiversity informatics domain should be served as metadata,

The LSID specification suggests three initial bindings (HTTP GET, SOAP and FTP) without specifying a default. If an LSID authority wants to ensure maximum availability of its data, it must implement all three bindings. If a client application wants to be sure it has access to all data it must implement all three bindings. The lack of a default implies that everyone has to implement all protocol bindings (HTTP GET, FTP, SOAP, etc).

All existing authorities bind to HTTP GET and very few bind FTP. We therefore mandate that the *getMetadata* call MUST be bound to HTTP GET. This formalises an existing 'lowest common denominator'. If this is done, all authorities and clients can be guaranteed to interoperate at the metadata level and a considerable implementation burden can be avoided.

> 31. Objects in the **biodiversity information domain** that are identified by an LSID **must** be typed using the **TDWG ontology** or other well-known vocabularies in accordance with the **TDWG common architecture**. .

Any objects identified by an LSID must be typed using the TDWG ontology [6] or other well-known vocabularies. Typing must follow TDWG common development architecture [7]. Entirely bespoke ontologies should not be used but existing ontologies should be extended where necessary.

Any objects referenced within a dataset must be referenced by the unique identifier of that object, whether this is an LSID or another existing GUID scheme. This reference may include existing objects such as literary references using a DOI. Text or literal versions of any referenced object should be avoided.

Machine and human clients that retrieve the metadata associated with an LSID will use the associated typing information to decide how to process the metadata and any associated data. If the type information is novel, processing may be difficult or impossible. Use of well known types allows the development and integration of applications that exploit the known types.

10. Presenting LSIDs to Clients

> 32. Providers **should** tag their objects with LSIDs and encourage clients to use LSIDs to refer to those objects.

When objects are tagged with their LSIDs, providers and their clients may attain the following benefits

- Clients (authors or other information systems) may refer to the object unambiguously;
- LSIDs give users access to provenance and attribution information;
- LSID metadata enables the integration of information

10.1. LSIDs Appearing in HTML Documents (Web Pages)

There are two common situations where LSIDs are presented to humans in HTML web pages:

1. When the LSID identifies the object being displayed
2. When the LSID identifies an object that is related to the main object being displayed.

Below are recommendations for presenting LSIDs for each case.

> 33. In an HTML document, an LSID appearing within the description of the object it identifies **should** be presented in plain text (i.e. not hyperlinked) and in its original form as in:
>
> **urn:lsid:authority.org:namespace:object:rev** `LSID`
>
> An icon linking to an explanation of the LSID **should** be present as in the example above. You may use the icon above and the following text as a template.
>
> "This is a Life Sciences Identifier (LSID), a permanent, globally unique identifier for this data item. Use this LSID whenever you need to refer to this data item."

34. In HTML web pages, LSIDs that refer to objects other than that being described **should** be presented as hyperlinks, with their **original form** as link text, and their **proxy version** as the link URL, such as-

<div align="center">

urn:lsid:authority.org:namespace:object:rev `LSID`

</div>

A link **should** be provided to explain what an LSID is wherever an identifier appears. You may use the text and icons provided here as a template.

"*This is a Life Sciences Identifier (LSID), a permanent, globally unique identifier for a data item related to the one being displayed. You may retrieve a description of this object by clicking on the hyperlinked LSID.*"

By providing proxy versions of the LSIDs, web crawlers and spiders may navigate through the network of objects indexing and making them available through popular search engines such as Google and Yahoo!.

10.2. LSIDs Appearing in Documents that Support Hyperlinks

35. In documents that support hyperlinks, such as Adobe PDF® or Microsoft Word®, LSIDs **should** be presented as hyperlinks with their **original form** as link text, and their **proxy version** as the link URL, such as-

<div align="center">

urn:lsid:authority.org:namespace:object:rev `LSID`

</div>

.

10.3. LSIDs Appearing in Printed Documents

36. In printed documents, LSIDs **should** be presented in their original form, as in:

<div align="center">

urn:lsid:authority.org:namespace:object:rev

</div>

A note **should** be added to explain what LSIDs are and about resolution using an on-line resolver such as http://lsid.tdwg.org/. A template is provided here:

"*The labels presented in this document that start with 'urn:lsid:' are Life Sciences Identifiers (citation). These are permanent, globally unique identifiers of data items used in the experiments reported in this article. You may retrieve a digital representation of each individual data item by typing its LSID in the form available at* http://lsid.tdwg.org/."

10.4. Recommendations for LSIDs Appearing in RDF

While independence of protocol and persistent association between object and identifier are benefits of the LSID identification scheme, standard Web software cannot consume LSIDs directly. Therefore, LSID adopters cannot take advantage of the wealth of software developed by the World Wide Web and the Semantic Web communities.

To work around that limitation and retain the benefits of the LSID specification, we recommend the use of **LSID HTTP proxies**, as outlines in the recommendations 37, 38, and 39, to simplify the resolution process for tools that do not yet handle LSID directly. These three recommendations together make up what was originally called the *LSID HTTP proxy usage recommendation*.

---

37. In RDF documents, objects **must** be identified by LSIDs in its standard form using the rdf:about attribute as in:

```
<rdf:Description rdf:about="urn:lsid:authority:ns:obj:rev">
```

LSIDs are Universal Resource Identifiers (URI) and as such can be used to identify resources in RDF documents via rdf:about property.

Standard clients do not need to dereference the identifier from the rdf:about property because they already have an RDF representation of the object. Thus, providers may use the LSID in its original form in the rdf:about property. If a client needs to retrieve the object at a later time, they can use the proxy version of the LSID provided according to the recommendation below.

---

38. The description of all objects identified by an LSID must contain an **owl:sameAs** statement expressing the equivalence between the object identifier in its **standard form** and its **proxy version** as in:

```
<owl:sameAs
rdf:resource="http://lsid.tdwg.org/urn:lsid:authority:ns:obj:rev"/>
```

Standard clients are not able deference LSIDs in their original form. They may only retrieve a representation of an object identified by an LSID if a proxy version of the LSID is provided. The recommendation above guarantees that standard clients have an HTTP URL they can dereference.

> 39. All references to objects identified by LSIDs using the **rdf:resource** attribute must use a **proxy version** of the LSID as in:
>
> ```
> <someProperty
>     rdf:resource="http://lsid.tdwg.org/urn:lsid:authority:ns:obj:rev"/>
> ```

Standard Web clients will have access to an object if a proxy version of every LSID is made available. Those clients may navigate through a network of objects if resources are linked by proxy versions of LSIDs.

Below are two sample RDF documents; the first example does not comply with the LSID HTTP proxy usage recommendation while the second does. Namespace declarations have been omitted for conciseness.

Listing 1: The RDF document below **DOES NOT COMPLY** with the LSID HTTP proxy usage recommendation:

```
<rdf:RDF>
   <rdf:Description rdf:about="urn:lsid:ubio.org:namebank:11815">
     <dc:identifier>urn:lsid:ubio.org:namebank:11815</dc:identifier>
     <dc:creator rdf:resource="http://www.ubio.org"/>
     <dc:subject>Pternistes leucoscepus (Gray, GR) 1867</dc:subject>
     <dc:title>Pternistes leucoscepus</dc:title>
     <rdfs::type rdf:resource="http://rs.tdwg.org/ontology/voc/example#ScientificName"/>

     <gla:vernacularName rdf:resource="urn:lsid:ubio.org:namebank:954940"/>
     <gla:vernacularName rdf:resource="urn:lsid:ubio.org:namebank:954941"/>
     <gla:vernacularName rdf:resource="urn:lsid:ubio.org:namebank:1564236"/>
     <gla:objectiveSynonym rdf:resource="urn:lsid:ubio.org:namebank:12292"/>
   </rdf:Description>
</rdf:RDF>
```

Listing 2: The RDF document below **DOES COMPLY** with the LSID proxy usage recommendation:

```
<rdf:RDF>
  <rdf:Description rdf:about="urn:lsid:ubio.org:namebank:11815">
    <dc:identifier>urn:lsid:ubio.org:namebank:11815</dc:identifier>
    <owl:sameAs rdf:resource="http://lsid.tdwg.org/urn:lsid:ubio.org:namebank:11815" />

    <dc:creator rdf:resource="http://www.ubio.org"/>
    <dc:subject>Pternistes leucoscepus (Gray, GR) 1867</dc:subject>
    <dc:title>Pternistes leucoscepus</dc:title>
    <rdfs::type rdf:resource="http://rs.tdwg.org/ontology/voc/example#ScientificName"/>

    <gla:vernacularName rdf:resource="http://lsid.tdwg.org/urn:lsid:ubio.org:namebank:954940"/>
    <gla:vernacularName rdf:resource="http://lsid.tdwg.org/urn:lsid:ubio.org:namebank:954941"/>
    <gla:vernacularName rdf:resource="http://lsid.tdwg.org/urn:lsid:ubio.org:namebank:1564236"/>
    <gla:objectiveSynonym rdf:resource="http://lsid.tdwg.org/urn:lsid:ubio.org:namebank:12292"/>
  </rdf:Description>
</rdf:RDF>
```

10.5. The Fake Protocol Handler `lsidres:`

40. The fake protocol handler `lsidres:` **must not** be used in hyperlinks (except for debugging purposes). The proxy version of the LSID **must** be used instead.

Extensions were developed to enable popular web browsers to-

- Resolve LSIDs in hyperlinks; and
- Allow users to type LSIDs directly into the web browser address bar.

Since web browsers do not support the `urn:lsid:` resolution scheme, nor can they be extended to support `urn:` sub-schemes, LSID developers created a fake protocol handler called `lsidres:` to resolve LSIDs natively.

The `lsidres:` protocol however fails to provide interoperability between LSID and standard web browsers. Web browsers must be extended to dereference `lsidres:` links or the links will appear to be broken. Standard web browsers, on the other hand, can dereference proxy versions of LSIDs without modification. Therefore, we **recommend** using proxy versions of LSIDs instead of using the `lsidres:` protocol handler.

References

[1] TDWG Globally Unique Identifiers Task Group (TDWG GUID) Website: www.tdwg.org/activities/guid/ (accessed 27-Aug-2007)

[2] TDWG Globally Unique Identifiers Task Group (TDWG GUID) Wiki: http://wiki.tdwg.org/GUID/ (accessed 27-Aug-2007)

[3] Life Sciences Identifiers Specification. OMG Specification, 2004: http://www.omg.org/cgi-bin/doc?dtc/04-05-01 (accessed 27-Aug-2007)

[4] Internet Engineering Task Force (IETF). RFC 2119. Key words for use in RFCs to Indicate Requirement Levels. http://rfc.net/rfc2119.html (accessed 27-Aug-2007)

[5] Dublin Core Metadata Initiative (DCMI): http://dublincore.org/ (accessed 27-Aug-2007)

[6] TDWG Ontology: http://wiki.tdwg.org/twiki/bin/view/TAG/TDWGOntology (accessed 27-Aug-2007)

[7] TDWG Technical Architecture Interest Group (TDWG TAG) Website: http://www.tdwg.org/activities/tag/ (accessed 27-Aug-2007)