Biodiversity
Information
Standards
TDWG

Biodiversity Information Standards (TDWG)

www.tdwg.org

TDWG Life Sciences Identifiers (LSID) Applicability Statement

Date:
3-Sep-2009

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

Abstract:
This document
1. Should be read in conjunction with the GUID Applicability Statement;
2. Provides guidance on how to use LSID to meet specific requirements of the biodiversity information community; and
3. 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

Motivation

The LSID specification supports applications within the Biodiversity Informatics 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 of the Object Management Group (OMG) [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** (elsewhere called entity, thing, resource), a term that is hard or impossible to define, but it is meant to subsume anything that can be named, including specimens, locations, agents, and data records, metadata records, and publications.

We refer to the organizations that disseminate data objects, or metadata records about objects, as **providers**.

An LSID HTTP proxy is a web service that resolves LSIDs by returning the results of the `getMetadata()` method 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>

<Rationale for the recommendation.>

1. LSID Assignment

R1. 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. 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 in bold below:

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

R2. A provider **should** use a domain name registered to it as authority identification.

A provider maximises the longevity of the LSIDs it has issued by using a domain name registered to it for the authority identification. By using a domain name registered to another organization, a provider is delegating control of associated LSIDs to that organization. Delegation will reduce the longevity of the original provider's issued LSIDs.

R3. A provider **should** plan to control the domain names it uses as authority identifications for as long as possible.

Losing control over a domain name used as authority identification renders all LSIDs under that authority irresolvable, or at least results in unreliability. Providers should choose authority identifications and domain names that are likely to persist for a long time.

R4. 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 should 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.

R5. 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 TDWG authority identification (see recommendation 6).

R6. If a suitable domain name is not available or likely to be unstable, request an authority identification from TDWG.

There are cases such as government department reorganizations, where a provider may lose 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 TDWG authority identification by following the instructions at: http://www.tdwg.org/activities/online-services/lsid-authority-ids/.

2.1. Using Multiple Authority Identifications to Separate Sets of LSID

R7. Providers **should** use separate authority identifications for objects where there is any reasonable possibility of a future need to separate the namespace.

An organization may offer to resolve the identifiers on behalf of a provider who is unable to resolve them for technical reasons. The host organization should use 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). 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.

R8. 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 or data types, for example in using authority identifications instead of namespaces. This practice is only advised if there is a chance the objects will be transferred 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.

3. LSID Namespace Identification

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

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

R9. 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 object identifier.

4. 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 within the same namespace.

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

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

R10. 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)

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.

R11. 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, unless the primary key values are also well-established unique identifiers, such as those indicated in Recommendation R10.

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 **should** generate unique object identifications and store them into that new column or table. The LSID Authority **should** in turn use the new column or table when resolving LSIDs instead of the primary key.

5. LSID Revision Identification and Versioning

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

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

R12. Clients **must not** try to infer relationships between objects based on the revision identification or any other part of an LSID. Instead, clients **must** retrieve the information for the LSID and use any assertions about revisions found in 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 retrieve the information for the resulting identifier to get the most update version of an object, or
- Perform any arithmetic operation (i.e. sum, subtraction) to the revision identification of an LSID and retrieve the information for the resulting identifier to get the next or previous revision of an object.

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

R13. LSID Authorities **should** use appropriate metadata properties to represent relationships 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 revision control system (e.g. Subversion) keywords.

6. LSID Opacity

R14. Clients in general **must** consider LSIDs as opaque strings (even though they may not be opaque). Clients **must** retrieve the value of an LSID from the network 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.
- 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 Uniform Resource Name, i.e., a generic, non-locatable identifier.

7. LSID Data

R15. LSID *data* **must never** change.

As defined in the LSID specification, data is treated—and must be accessed differently—from metadata. Data associated with an LSID is returned when an LSID-compliant application requests the data portion of the LSID object from the network by using the `getData()` method. This data must never change. To reinforce the immutability of LSID data to consumers, techniques could be adopted that demonstrate this permanence attribute, such as a checksum. See also: R19, R21.

R16. Providers **should not** dynamically encode data in formats such as XML, which 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 calls to the `getData()` method.

R17. Providers **should not** return irrelevant data in LSID `getData()` method calls.

Some providers feel compelled to return something in the LSID `getData()` method call 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()` method call. It is reasonable not to return anything in the `getData()` method call.

R18. Providers **should not** use the `getData()` method call just to assert that some attributes of objects are immutable.

It is considered **bad practice** to use the `getData()` method call to inform clients that some object attributes are immutable. Such assertions are best expressed in the metadata using appropriate predicates. One reason for this is due to the recommendation—R17 above—to provide an empty result when LSID data is queried, if it refers to something that is not a piece of data.

8. LSID Metadata and Data

R19. 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 will need to keep their own copy of it. However, it must be remembered that an LSID (or GUID) must only refer to one object, so although some properties of that object may change, and hence the metadata changes, the GUID must continue to refer to the same object. See also: R15

R20. The default metadata response format **must** be RDF serialized as XML.

Metadata associated with an LSID is returned by the `getMetadata()` method 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 `accepted_formats` parameter of the `getMetadata()` method call.

The default return type in the LSID specification is RDF. Parsing arbitrary formats makes 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.

R21. HTTP GET **must** be the default binding for LSID `getMetadata()` method calls.

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

R22. Non binary encoded objects **should** be served as LSID metadata.

The `getData()` LSID method 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 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, SOAP and FTP).

All existing authorities bind to HTTP GET and very few bind FTP. We therefore mandate that the `getMetadata()` method 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.

R23. Objects in the **biodiversity domain** that are identified by an LSID **should** be typed using the **TDWG ontology** or other accepted vocabularies in accordance with the **TDWG common architecture**.

Any objects identified by an LSID must be *typed* (by designating the object's type or class) using the TDWG ontology [6] or other accepted vocabularies. Typing must follow TDWG common development architecture [7]. Entirely bespoke (custom-made) ontologies should not be used where standard ontologies exist, 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.

9. Presenting LSIDs to Clients

R24. Providers **should** tag their objects with LSIDs and encourage clients to use LSIDs to refer to those objects.

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

- Clients may refer to the object unambiguously;
- Provenance and attribution information are accessible;
- LSID metadata enables the integration of information.

9.1. LSIDs Appearing in HTML Documents (Web Pages)

There are two common situations where LSIDs are presented in a human readable form in HTML web pages:

- When the LSID identifies the object being displayed on a web page (e.g. the main subject of a web page).
- When the LSID identifies an object that is related to the information object being displayed.

Below are recommendations for presenting LSIDs for each case.

R25. 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.

For example:

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

An icon, or hyperlink, 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 persistent globally unique identifier for this object. Use this LSID whenever you need to refer to this object.”

R26. 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.

For example:

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

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 an object 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 metadata records, indexing and making them available through popular search engines such as Google and Yahoo!.

9.2. LSIDs Appearing in Documents that Support Hyperlinks

R27. 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.

For example:

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

9.3. LSIDs Appearing in Printed Documents

R28. In printed documents, LSIDs **should** be presented in their original form.

For example:

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

A note **should** be added to explain what LSIDs are and about their 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 (http://www.omg.org/cgi-bin/doc?dtc/04-05-01). These are permanent, globally unique identifiers of objects 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/."

9.4. 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 outlined in Recommendations 26, 27, and 30, 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*.

R29. In RDF documents, objects **must** be identified by an LSID in its standard form using the `rdf:about` attribute.

For example:

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

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

In most cases, clients will not need to retrieve the information for the identifier in the `rdf:about` property because they will already have the RDF description 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's metadata at a later time, they can use the proxy version of the LSID provided according to Recommendations 26 and 27.

R30. The description of all objects identified by an LSID **must** contain an `owl:sameAs`, `owl:equivalentProperty` or `owl:equivalentClass` statement expressing the equivalence between the object identifier in its standard form and its proxy version.

For example:

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

Most semantic web clients are not able to retrieve the information for LSIDs in their original form. They may only retrieve a description of an object identified by an LSID if a proxy version of the LSID is provided. The recommendation above guarantees that most clients have an HTTP URL they can access. The equivalence should be denoted by the use of the appropriate predicate which is compatible with OWL-DL, namely, owl:sameAs for individuals, owl:equivalentProperty for properties, and owl:equivalentClass for classes.

R31. All references to objects identified by LSIDs using the `rdf:resource` attribute **must** use a proxy version of the LSID.

For example:

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

Standard web clients will have access to the metadata for 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>
```

9.5. The Fake Protocol Handler `lsidres`

R32. 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 need to be extended to retrieve the information for `lsidres` links or the links will appear to be broken. Standard web browsers, on the other hand, can use 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 21-Jan-2011)

[2] TDWG Globally Unique Identifiers Task Group (TDWG GUID) Wiki: http://wiki.tdwg.org/GUID/ (accessed 21-Jan-2011)

[3] Life Sciences Identifiers Specification. OMG Specification, 2004: http://www.omg.org/cgi-bin/doc?dtc/04-05-01 (accessed 21-Jan-2011)

[4] Internet Engineering Task Force (IETF). RFC 2119. Key words for use in RFCs to Indicate Requirement Levels. http://www.ietf.org/rfc/rfc2119.txt (accessed 21-Jan-2011)

[5] Dublin Core Metadata Initiative (DCMI): http://dublincore.org/ (accessed 21-Jan-2011)

[6] TDWG Ontology: http://wiki.tdwg.org/twiki/bin/view/TAG/TDWGOntology (accessed 21-Jan-2011)

[7] TDWG Technical Architecture Interest Group (TDWG TAG) Website: http://www.tdwg.org/activities/tag/ (accessed 21-Jan-2011)