DRUG DISCOVERY TODAY BIOSILICO

The impact of Life Science Identifier on informatics data

Sean Martin, Moses M. Hohman and Ted Liefeld

Since the Life Science Identifier (LSID) data identification and access standard made its official debut in late 2004, several organizations have begun to use LSIDs to simplify the methods used to uniquely name, reference and retrieve distributed data objects and concepts. In this review, the authors build on introductory work that describes the LSID standard by documenting how five early adopters have incorporated the standard into their technology infrastructure and by outlining several common misconceptions and difficulties related to LSID use, including the impact of the byte identity requirement for LSID-identified objects and the opacity recommendation for use of the LSID syntax. The review describes several shortcomings of the LSID standard, such as the lack of a specific metadata standard, along with solutions that could be addressed in future revisions of the specification.

Bioinformatics research has been characterized by the creation of multiple databases that contain DNA and protein sequences [1–5]. The content and number of such databases is continually growing because new techniques and technologies create more data, as well as different types of data. If a researcher is interested in the identification of genes or proteins connected to human disease [6], they will need to cross-reference many databases to extract information about biological function.

Biologists intent on performing computational biology analysis across many databases have to work with a myriad of different identifiers, data formats and tools. It depends upon the knowledge and skill of the biologist as to how easily they can recognize and map identifiers and data formats between the different databases. To make this process consistent and repeatable, and to reduce the burden on the researcher, the databases should be integrated so that they can be accessed and used in a consistent manner.

Integrating databases occurs at multiple levels and should ideally include the use of common definitions, attributes and formats for domain objects. This is hard to achieve because of the resistance to structural change of large existing databases, the lack of consistent definitions in the scientific community for objects such as gene representations and the constant addition of and the need to integrate new, previously unknown object types. Therefore, we presume that there will always be structural differences between databases. However, federated integration is still possible given a consistent identification mechanism for assigning and recognizing identifiers. The goal of the Object Management Group's (OMG, www.omg.org) Life Science Identifier (LSID) specification is to provide such an identification system [7].

The LSID specification was initially defined by a working group from the Interoperable Informatics Infrastructure Consortium (I3C). The I3C submitted the specification to the OMG and it was published as an OMG specification in October 2004. The OMG's Life Sciences Research Domain Task Force (www.omg.org/lsr) now manages the standard. Other groups currently influencing the evolution of

Sean Martin*
IBM Corporation,
1 Rogers Street,
Cambridge,
MA 02142, USA
*e-mail: sjmm@us.ibm.com
Moses M. Hohman
Bioinformatics Core,
Robert H. Lurie
Comprehensive Cancer Center
of Northwestern University,
676 N. St Clair Street, Suite 1200,
Chicago,
IL 60611, USA
Ted Liefeld
The Broad Institute of MIT and
Harvard,
320 Charles Street,
Cambridge,
MA 02141, USA

1359-6446/05/$ – see front matter PII:S1359-6446(05)03651-2

LSID include the BioPathways Consortium (http://lsid.biopathways.org) and the World Wide Web Consortium (W3C) Technology and Society Domain (www.w3.org/2004/10/swls-workshop-report.html).

Clarke *et al.* [8] provide a detailed introduction to the definition and use of the LSID, which is recommended to readers seeking a deeper technical understanding of the specification.

The LSID specification [9] defines three main concepts for its use: the LSID syntax, LSID assignment and LSID resolution.

LSID syntax

The LSID syntax defines a uniform resource name (URN) namespace [10]. LSIDs are persistent, globally unique, location-independent identifiers.

The LSID syntax is:

<LSID>::= 'urn:' 'lsid:' <AuthorityID> ':' <AuthorityNamespaceID> ':' <ObjectID> [':' <RevisionID>]

In every LSID, the URN Namespace is 'lsid'. The remainder of the LSID defines the authority issuing the LSID, the authority-specific namespace of the LSID, the unique identifier of an object within this namespace and an optional revision identifier.

LSID assignment

The LSID specification was developed specifically to permit data providers to represent existing identifiers in the LSID syntax. The specification does not require new identifiers for existing objects or concepts, only that identifiers are packaged according to the LSID syntax. For example, if the NCBI assign LSIDs to GenBank objects a GenBank accession would be written as:

URN:LSID:ncbi.nlm.nih.gov:GenBank.accession:NM_010424:2

For an organization to assign LSIDs, it must define its unique AuthorityID (typically an internet domain name under its control) and define the namespace(s) for objects for which it intends to publish LSIDs.

The authority organization is responsible for ensuring that the namespace, object identifier and revision ID portion of the LSID are all unique and specify only one object. More than one LSID can be defined for an object, although this is not recommended because the overall objective of LSID adoption is to aid in the reduction of ambiguity and the number of 'keys' by the introduction of unique names. For example, an accession number and a GI number can refer to the same GenBank record:

URN:LSID:ncbi.nlm.nih.gov:GenBank.gi:31981696

URN:LSID:ncbi.nlm.nih.gov:GenBank.accession:NM_010424:2

For objects identified by LSIDs all changes to the underlying byte representation must be reflected in a new LSID, typically by incrementing the RevisionID. This byte identity requirement enables exact digital-experiment reproducibility and for middleware services, such as caches, to identify that data has changed and re-caching is required. LSIDs that do not name associated byte representations are used to name concepts.

An authority publishing an object via LSID is not required to provide LSID resolution for that object or the named data at any time, present or future. Additionally an LSID, once assigned, can never be reassigned to a different object.

LSID resolution

LSID authorities can provide an LSID resolution service (LSRS) for their LSIDs. LSRS specification defines a standard interface for retrieval of data and metadata regarding identified objects or named concepts via multiple protocols. Metadata can be used to describe further every named data object or concept directly and also to convey semantically meaningful relationships among the objects and concepts referred to by LSIDs.

LSID resolution consists of four steps: a client acquires the LSID to an object of interest; the client locates an LSRS for the LSID through *a priori* knowledge or use of the LSID Resolution Discovery Service (LRDS); the client sends a getAvailableServices() request to the LSRS providing the LSID as a parameter and the LSRS returns locations and protocols for services capable of retrieving data or metadata about this LSID; the client selects one of the services and sends it a getData() or getMetaData() request.

The full interface of the LSRS describes additional methods to permit efficient transmission of the data and metadata available for an LSID.

The LRDS is an additional interface that authorities can implement to enable a client application to find a resolution service for an LSID. It provides a single method returning a list of LSRSs that can resolve the given LSID. Publicly available LRDS implementations exist that are based on dynamic delegation discovery systems (DDDS) and/or domain name systems (DNS) [11]. Implementing a local LRDS allows organizations to control the resolution of LSIDs locally and to direct clients to local copies of wrapped data sources (GenBank, for example) instead of resolving the LSID over the internet.

Early adopters of LSIDs

Several organizations and communities have become early adopters of the LSID. The following brief descriptions and information displayed in Table 1 illustrate the various uses that five organizations have found for the standard. For additional practical examples of LSID usage the reader can explore the Biopathways Consortium website (http://lsid.biopathways.org).

Broad institute

GenePattern (www.genepattern.org) is a flexible analysis platform, developed to support multidisciplinary biomedical research. It provides an environment for the rapid development, deployment and sharing of new analytical

TABLE 1

Early adopters of named data and named concept LSIDs

Application	Named data	Named concept
Broad GenePattern	✓	
Broad GeneCruiser		✓
myGrid	✓	✓
BioMOBY		✓
IBM SLRP	✓	✓
CVIT	✓	

techniques. GenePattern uses LSIDs to support reproducible research by uniquely identifying computational analysis algorithm implementations and methods. The specification of an absolute (byte-wise) identity allows GenePattern to stipulate particular algorithm implementations, enabling precise reproduction of research results. Methodologies and algorithms have been published with LSIDs in several research papers including Brunet *et al.* [12], Golub *et al.* [13], and Lu *et al.* [14].

GeneCruiser (www.broad.mit/edu/cancer/software/genecruiser) [15] is a web service and web application that allows researchers to annotate genomic data by mapping microarray-feature identifiers to gene identifiers from public databases, such as UniGene (www.ncbi.nlm.nih.gov/UniGene) and LocusLink (www.ncbi.nlm.nih.gov/LocusLink), as well as into other web resources, such as the UCSC Genome Browser (http://genome.ucsc.edu). GeneCruiser addresses the problem of identifiers for probe sets on a microarray not being biologically meaningful and having to be mapped to more-informative identifiers, such as gene identifiers, chromosomal locations or molecular functions, to be interpreted by a biologist. GeneCruiser solves this problem by maintaining a database-mapping probe set identifiers to multiple public databases. LSIDs in GeneCruiser are used to include an identifier's issuing authority and namespace with the identifier in the LSID string. This allows GeneCruiser to create more-efficient queries and to return identifiers that are unambiguous in their context. GeneCruiser can accept non-LSID identifiers as strings and this forces the application to query the identifier against all known databases, which drastically reduces performance.

The myGRID Consortium

The myGrid UK (www.mygrid.org.uk) e-Science project is a collaboration of five universities, the European Bioinformatics Institute (www.ebi.ac.uk) and industrial partners. It provides middleware for bioinformatics with a focus on grid technologies. LSIDs are used throughout myGrid for the identification of data objects from external sources, as well as internally created data [16]. Using LSID for identification allows for more-efficient and cohesive exchanges between many myGrid components.

The LSID-enabled components for myGrid include an LSID-assigning service for objects created by the Taverna (http://taverna.sourceforge.net) workflow system or the myGrid Information Repository (MIR); an LSID authority service that contains locations of other services, internal or external, that can provide data for a particular LSID-assigned data object and LSID data and metadata resolvers. The data service returns data for an LSID-associated object from the MIR storage system. The LSID metadata resolver service returns metadata associated with a particular data object. Metadata can be stored in several places including the MIR and in the Knowledge Annotation and Verification of Experiments (KAVE). The KAVE service allows myGrid components to store and retrieve information using the Resource Description Framework (RDF, www.w3.org/RDF) and is used in myGrid to record knowledge-provenance information during the execution of a Taverna workflow in a digital experiment. For example, KAVE records the LSIDs of input and output data in a workflow. This allows the detailed history of the entire flow to be recorded and verified.

BioMOBY

BioMOBY (www.biomoby.org) [17] is a popular open-source community-based development effort that defines an ontology-based interoperable messaging system for web services, as well as a registry that facilitates the discovery of task-appropriate services and analytical pipelines. LSIDs within BioMOBY are used to identify ontological nodes describing data and services, as well as individual web service instances. LSID resolution is primarily used to obtain the 'signature' of a service instance – an RDF document describing the nature of the service, its inputs, its outputs and its task- and quality-metadata.

IBM Corporation

The IBM Corporation adopted the LSID standard for support in future software solutions that it will market to the life sciences industry. It also makes use of LSID technologies in several research prototypes. One such system is the Semantic Layered Research Platform (SLRP) [18]. The SLRP system creates LSIDs to identify and retrieve binary objects (data files, images or papers), recorded by researchers and GRID [19] programs throughout the course of a life-sciences research project, by using an LSID-assigning service and a 'write once' permanent data-store. LSIDs are also used as keys to metadata, which describe the objects and concepts and their attributes and relationships to other objects that can be internal or external to the system. All metadata are recorded in an RDF database, which includes annotations and complete provenance data for all aspects of a research project. Information stored in SLRP is made available using an LSID authority service as well as accompanying data and metadata access services.

The Center for the Development of a Virtual Tumor

The Center for the Development of a Virtual Tumor (CViT, www.cvit.org) is a National Cancer Institute sponsored

effort promoting collaboration between groups of internationally based researchers attempting to create multi-scale mathematical models and computer simulations of cancerous tumor growth. CViT provides a collaborative space where the researchers will be able to upload source-code implementations of formulae and algorithms for review, annotation, discussion and reuse by collaborators. One system they use is the Concurrent Versions System (CVS, www.gnu.org/software/cvs) to manage file versioning. CViT has added an LSID authority to CVS that automatically provides LSIDs to source-files, projects and releases of tracked code. Researchers can use these LSIDs to accurately refer to particular versions, source-files, projects and the resulting programs that make up experimental simulation models on which they are collaborating. During reviews and discussions or when reproducing *in silico* experimental results, it is vital for these researchers to be able to pinpoint exactly which programs were executed as part of an experiment and exactly which source-code files combined to form those programs. The combination of a regular CVS system, an LSID authority and LSID data and metadata services provides this facility.

LSID common concerns and misconceptions

As organizations move to incorporate LSIDs into their IT infrastructure several concerns commonly arise. We collected these concerns from our work with the National Cancer Institute's Cancer Biomedical Informatics Grid (caBIG, http://cabig.nci.nih.gov) Architecture Working Group and from actual use in life science companies and academic organizations.

Byte identity

The most popular LSID example in the literature identifies a GenBank record. From the example above, we have: URN:LSID:ncbi.nlm.nih.gov:GenBank.accession:NM_010424:2

The sequence portion of the GenBank record is uniquely associated with the GenBank accession number and version. All other fields in the record (e.g. sequence feature annotations) can change without requiring a change in the version, and thus in the LSID. This seems to fly in the face of the LSID byte identity requirement.

First, we point out that the GenBank example is unusual. GenBank and other public databases might never adopt LSID as a data identifier standard, although other organizations, such as Biopathways, could choose to provide LSIDs for GenBank-hosted data. The primary intention of this example is to show how LSIDs wrap existing identifiers for these data sources. It is insufficient just to wrap the identifier. Data providers must also satisfy the byte-identity contract required by the LSID specification, unless the LSID is used to name a concept without a byte representation.

The prevailing guidance for accommodating byte identity in this example includes only the GenBank sequence, accession number and version in the data associated with the LSID, all other fields are placed in the metadata. The LSID specification puts no restrictions on the metadata, so these fields can vary arbitrarily. This is an artificial solution because these other data fields are no closer to being metadata than they are to being data.

This raises a more general issue. Some members of the data identifiers community point out that biological objects are actually characterized by these 'dynamic' properties and their values can change without requiring a new identity for the enclosing object, and thus a new identifier. This concern arises from a conflation of data identity and conceptual identity. This conflation is understandable given the ambiguity of the terms 'identity' and 'identifier'. Two different but complementary approaches to LSID use are suggested:

(i) LSIDs provide identifiers (references or 'keys') for distributed data objects. These identifiers must satisfy the requirements of distributed data storage and retrieval, caching, distributed joins, among others. These requirements mandate byte identity. In this case, equal LSIDs signify byte identity between the two associated data objects.

(ii) The LSID syntax unifies the Babel of life science identifiers so the community can use a single identifier format to reference concepts currently referenced by accession numbers. For example, MAGE (www.mged.org/Workgroups/MAGE/mage.html) identifiers, ICD-9 codes [20] and proprietary identifier systems local to institutions. These LSIDs do not point to data, instead they point to metadata relationships (e.g. in RDF) among concepts, enabling semantic discovery. Equal LSIDs imply conceptual identity. Byte identity does not come into play because there is no associated data. Metadata can record relationships between these concepts and associated data objects (via the data objects' LSIDs). The associated data objects follow the byte identity requirement.

Thus, there is a dual use for LSIDs. To identify data it is associated in a byte-identical way with the LSID. The identification of a concept does not necessarily associate data with the LSID, instead relationships are put into the metadata. These two uses can complement each other in a single system, allowing robust and efficient distributed data retrieval and semantic discovery, where concepts and data are named, related and further described using metadata (see Figure 1). Data providers can house changing properties in byte-identical, versioned data objects and they can dynamically associate them with the concept of a biological object via metadata relationships.

Another useful way to look at the difference between byte-identical and dynamic data is in the terms of location independence. If a client can rely on byte identity, it does not matter whether the client retrieves data from the authoritative source of the data, from a mirror service closer to the client or from a cache. All clients that retrieve data associated with a particular LSID work with identical data so they can expect to achieve the same results when applying a given analysis. By contrast, data that do not

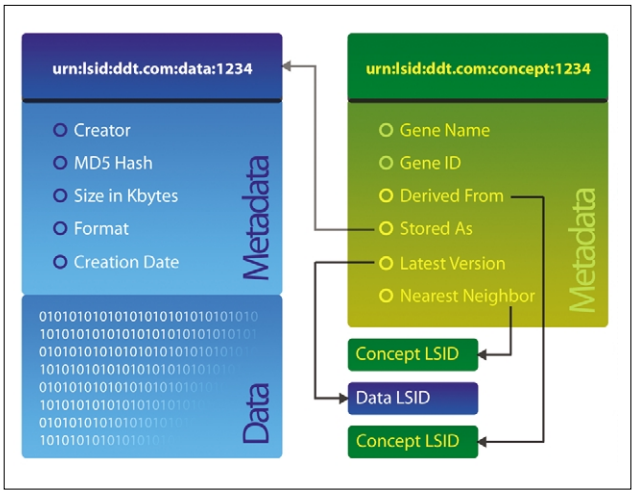

FIGURE 1

Relationship between named data and named concept LSIDs. LSIDs are used to name a set of data bytes and metadata (further describing those bytes). LSIDs without and associated set of data bytes usually have associated metadata that provides relationship linkages to other LSIDs, thereby creating an information graph.

obey byte identity are inherently location-dependent. Here the concept of authoritative source becomes very important. Clients will ideally want to retrieve these data from the authoritative source. Cached data gives some indication of freshness and clients can decide how comfortable they are working with data that are potentially out-of-date by a given time period. Clients at different sites cannot expect analyses performed on this sort of data to yield the same results over time because they have no way to check that they are using the same underlying data.

There are some concerns that, because byte identity is merely a service contract, data providers might ignore the contract unbeknownst to data clients. There is only one surefire way to prevent this problem. To be confident of data integrity, clients must compute a fingerprint and/or hash of the data object and compare hashes with other clients. Clients cannot trust server-provided hashes because they could be out-of-date or otherwise incorrect. This approach allows clients to take action that is appropriate to their degree of trust and need for data integrity.

Some clients insist that their data exist in a special format (e.g. XML) that can undergo byte changes without changing the data content (e.g. rearranging the order of the attributes of an XML element). They propose an alternative contract based on a more forgiving algorithm for determining data identity and they are concerned that requiring byte identity would be too restrictive and impose a burden on data providers. The authors of this review respectfully disagree with this suggestion. Byte identity is a straightforward requirement, simpler to implement and to enforce than the suggested higher-level identity comparisons. For LSIDs to serve as identifiers across disparate data sources, the algorithm for determining data identity must also be suited to every data format, not just a particular format used by a subset of applications.

Other notions of identity

Some clients wonder whether they should replace their existing object identifiers with LSIDs and discard, for example, their current accession numbers. It should be evident from the literature that this is not necessary. LSIDs can wrap existing identifiers (as in the GenBank example) and data providers can store dynamic properties in metadata, at least until a better solution has been invented.

Others decide to ignore the identifier 'opacity' requirement of the LSID specification and they start putting semantic information into the identifier. Data providers might store information about the type of tissue from which a particular measurement was derived, easing data categorization based on sorting identifiers: urn:lsid:authority.example.org:expression_data_from_liver_tissue:12345

This approach has some of the same drawbacks as using natural keys in relational databases (e.g. using patient name as a key for patient data). Inevitably, data providers discover errors in the semantic part of the LSID (e.g. the expression data turns out to be from pancreas) and at that point many other data sources and clients could be using inappropriate identifiers. There is no easy way to correct this situation because it is too difficult to coordinate a correction with all clients and external data sources that use the incorrect reference. Often it just goes uncorrected. As a result of this the semantic information in the LSID becomes unreliable and misleading. This problem is usually only one that is internal to data providers because client software never treats the LSID string as anything but opaque.

A common misconception is that LSIDs have something to do with patient health information (PHI) de-identification. De-identification is a data scrubbing process, whereby information that might reveal the identity of a patient under study is removed from data records before they are disclosed to nonclinical parties. LSIDs identify data objects and concepts and thus neither facilitate nor prevent the revelation of patient identity. Data providers currently performing de-identification who choose to use LSIDs must continue to scrub their data as they always have.

Resolution and gradual adoption

Identifier resolution refers to the process of retrieving the associated data (and/or metadata) for an identifier or for a list of identifiers. We contrast this with query, a much richer service for retrieving a list of objects that match a set of criteria.

Some use LSIDs entirely in-house with a proprietary resolution protocol (or none at all). In this context LSIDs serve mainly as organization-wide unique identifiers. This is a good first step towards intra- and inter-organizational data integration. Providing a reasonable authority string is chosen these identifiers are also globally unique.

The next step to wider integration is the adoption of a common resolution protocol among organizations. If the organizations already use LSIDs in accordance with the

rest of the specification they will experience a smoother path to data integration. For example, organization A can depend on the byte-identity service contract satisfied by organization B and cache B's data objects. If a standard resolution protocol for LSID takes hold, an organization that wants to access other organizations' data, or to provide access to its own data, can simply adopt the standard protocol.

Distributed query

Query services that join data from distributed data sources benefit greatly from byte identity. It permits query engines to retrieve lists of identifiers matching query conditions instead of lists of entire objects. Identifiers are then matched with each other to determine the result set. Retrieving and comparing only the identifiers gives a much better performance.

There is an interesting caveat here because the LSID specification allows (but does not recommend) data providers to assign multiple LSIDs to the same data object. This facilitates situations where a provider might want to wrap more than one natural identifier scheme (e.g. in the case of GenBank, GI numbers and accession numbers). However, if objects have more than one LSID, one of them should be assigned as the canonical identifier and only that identifier should be returned to query engines and stored in other data services. Otherwise, if query engines receive incompatible lists of identifiers from different sources they will not be able to match identical objects correctly.

What is missing from the current LSID standard?

Defining metadata standards

A significant shortcoming of the current LSID standard is the absence of any agreement on the specific information that a client might expect to see when they retrieve the metadata for an LSID-named data object or concept. Without agreement on specific, well-defined metadata elements that could or should be present, and the form in which they are serialized, software can do little to automatically take decisions and process the information accessible by the LSID resolution protocol. Whereas metadata for an object could convey useful information, such as the size, the digital signature or hash, the format or the programmatic context, general-purpose software will be unable to extract and use this information unless a standard is created for identifying these elements.

Providing an extensive list of metadata standards for form and content is probably the most important item that a revision of the standard can address because it would allow software to take the handling of information, named and retrieved using LSIDs, to the next level of automation. Existing metadata standards, such as the Dublin Core (www.dublincore.org), which provide a standardized metadata scheme for describing documents, can serve as examples of how to proceed with this development. Benefits of a standardized LSID metadata standard might include sophisticated automatic negotiations between clients and servers to locate and retrieve the information that is most appropriate to a user's current task (e.g. to locate the most recent version of a particular conceptual object, perhaps a gene or a protein, the binary expression of which, perhaps a sequence or an image, is created using a particular algorithm that is in a particular binary format that the client can render) and to give client software accurate context for processing the information that has been communicated to it.

Immutable metadata

Another problem of the current standard is that it has not made any provision for segmenting metadata that changes frequently from metadata that should never change. The specification states that, when even a single byte of an object named by an LSID changes, a new LSID should be created, preferably by incrementing the 'RersionID' part of the LSID. However, metadata provided for the object named by LSID can be rewritten at any time. This will often leave data providers using the LSID-naming scheme difficult choices as to classification in the data portion of the object and what to provide as metadata. It also means that metadata that will never change (e.g. the name of the creator, the date on which the object was created or an MD5 digital hash) will be treated in exactly the same way as metadata that frequently changes.

To understand the intentions behind the differentiation of data and metadata, we point out that metadata were originally envisioned as annotations (data that recorded additional local knowledge about a particular data object). This is why the specification imposes no global requirements on metadata. In the longer term applications will require more-sophisticated classification of data. The dynamic fields in the GenBank record are not really all metadata. One person's metadata are another person's data. Work needs to be done in this area to come up with a more-elegant solution.

Protocol inefficiencies

Finally, the LSID-resolution specification does not provide a method for set retrieval. The existing interfaces permit only 'object-at-a-time' resolution. Thus, if a distributed query returns a list of objects, the retrieval of each object in the list involves the overhead of a unique SOAP request each time. Building query services that interoperate with LSID resolution will require extensions of the LSID resolution protocol to include set retrieval.

Future mechanisms

The LSID standard defines how an object is uniquely named, as well as a protocol. The same protocol can be used to retrieve metadata information about the object from third parties, called foreign authorities, who use someone else's LSID as a key to their own annotations.

How to know which third parties have this annotated data identified (so that it can be requested from them) is a problem that must be addressed.

Three options are:

(i) One could collect a list of LSID authorities that are known to aggregate useful information indexed by LSID. Then, upon resolving an LSID, to ask all of the foreign authorities, from this list, if they have metadata beyond that already provided by the original authority.

(ii) To query a third party service that creates indices from all known LSID-accessible metadata using web-crawling techniques. There is no doubt that this could be a powerful search mechanism; we have seen how successful services like Google and MSN are at retrieving relevant information. If combined with web indices this might have the benefit of including results, not only from formal databases of objects named by LSID but also those that happen to include an LSID in a web-based document, presentation or spreadsheet, creating interesting cross-referencing.

(iii) A method that might be supported by a future revision of the standard is foreign authority notification [21]. This option allows every party possessing additional metadata for a particular LSID created elsewhere to let the authority for that LSID know where they can be contacted. This machine-readable contact information would be optionally provided by the original authority along with all metadata provided for that LSID when it is first resolved. The person or program resolving the LSID would then have the option to retrieve the third-party information, knowing that it was from a third party and not the original source. This mechanism would provide a means by which users of LSID-named information could swiftly obtain information about annotations and relationships between every named data item and information provided by third parties.

Conclusion

The LSID standard has seen good, early adopter interest since its publication in October 2004. For the most part, these initial users have found immediate value in applying the identifier standard to their own in-house or in-community data naming and sharing problems with the longer-term benefits of common software, data sharing and unambiguous metadata references in plain sight. Current adopters use the LSID for naming and identifying binary objects with strict byte-identity and related metadata. They also identify abstract concepts where the LSID is utilized to associate metadata to these concepts. The authors recognize this is an excellent first step towards intra- and inter-organizational data integration and look forward to increasing network effect benefits for the entire life sciences community as adoption accelerates and improvements to the standard are implemented.

References

1 Dayhoff, M.O. *et al.* (1980) Nucleic acid sequence bank. *Science* 209, 1182

2 Kneale, G. and Bishop, M. (1985) Nucleic acid and protein sequence databases. *Comput. Appl. Biosci.* 1, 11–17

3 Burks, C. *et al.* (1985) The GenBank nucleic acid sequence database. *Comput. Appl. Biosci.* 1, 225–233

4 Hamm, G.H. and Cameron, G.N. (1986) The EMBL Data Library. *Nucleic Acids Res.* 14, 5–9

5 George, D.G. *et al.* (1986) The protein identification resource (PIR). *Nucleic Acids Res.* 14, 11–15

6 Mootha, V. *et al.* (2003) Identification of a gene causing human cytochrome c oxidase deficiency by integrative genomics. *Proc. Natl. Acad. Sci. U. S. A.* 100, 605–610

7 Sollins, K. and L. Masinter (1994) Functional Requirements for Uniform Resource Names. *RFC* 1737

8 Clark, T. *et al.* (2004) Object Identification for Biological Knowledgebases. *Brief. Bioinform.* 5, 59–70

9 *The Object Management Group* (2004) Life Science Identifiers, OMG Adopted Specification. dtc/04-08-02

10 Moats, R. (1997) URN Syntax. *IETF RFC* 2141

11 Mealing M (2002) *Dynamic Delegation Discovery System (DDDS)* IETF RFC 3401,3402,3203,3404,3405

12 Brunet, J.P. *et al.* (2004) Metagenes and molecular pattern discovery using matrix factorization. *Proc. Natl. Acad. Sci. U. S. A.* 101, 4164–4169

13 Golub, T.R. *et al.* (1999) Molecular Classification of Cancer. Class Discovery and Class Prediction by Gene Expression. *Science* 286, 531–537

14 Lu, J. *et al.* (2005) MicroRNA Expression Profiles Classify Human Cancers. *Nature* 435, 834–838

15 Liefeld, T. *et al.* (2005) GeneCruiser: a web service for the annotation of microarray data. *Bioinformatics* 21, 3681–3682

16 Lord, P. *et al.* (2004) The Semantic Web: Service discovery and provenance in myGrid: A position paper for W3C.org Life Sciences Workshop http://lists.w3.org/Archives/Public/public-swls-ws/2004Sep/att-0016/semantic_web_for_life_sciences_position.pdf

17 Wilkinson, M. *et al.* (2005) BioMOBY successfully integrates disributed heterogenous bioinformatics web services. The PlaNet exemplar case. *Plant Physol* 138, 1–13

18 Martin, S.J. and Jackson, A. (2004) Knowledge Integrated Modeling (KIM), an application for the Semantic Layered Research Platform (SLRP): A position paper for W3C.org Life Sciences Workshop http://lists.w3.org/Archives/Public/public-swls-ws/2004Sep/att-0069/Semantic_Layered_Research_Platform.pdf

19 Foster, I. *et al.* (2001) The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications* 15, 200–222

20 *International Classification of Diseases* Ninth Revision (ICD-9) http://www.cdc.gov/nchs/about/major/dvs/icd9des.htm

21 Smith, D. and Szekely, B. (2005) A guide to deploying Life Science Identifiers: http://www128.ibm.com/developerworks/opensource/library/os-lsidbp