

The big question is whether the LSID need is viewed by the community as being simply a "Life Science Dewey Decimal Number" to access a Life Science object fragment (ex: JPG or word document) or is the expectation and need broader, where the LSID should be capable of being optionally extended to be a Life Science "Object Reference"? The current proposal fulfills the former but, because it lacks context (and doesn't provide the user with the meaning of what is pointed to and how it relates to other objects), it relies on the existence of a semantic web to make it truly appealing. Because the semantic web does not yet exist, we should perhaps consider extending the LSID spec to be a little broader than it currently is.

The LSID as an object reference is something like a URL with attached arguments that contain qualifiers, like the sort of thing that Google returns a list of in response to a search query or that results after a web site log in. The client doesn't know or care about the format of these extra arguments (state or URL subpage directions) but they index the associated page, and will be interpreted at the server end when the URL request is issued. So, two scientists could then have LSIDs that were "equal" (same URL) but each had different argument extensions ... so the access rights, timeouts, and extents of data accessible COULD be different if the owner decided to make them so at LSID creation. The IBM case (subject to extensibility modifications) could then still fall out as the rather trivial "zero extension" case.

Sun proposes:

1. The LSID has the potential to be far more than the simple Life Science Dewey Decimal number.
2. It should be extended to support optional opaque (non-client decipherable) extensions. This would extend the LSID into a full fledged Life Sciences Object Reference ... and make the OMG synergy complete.
3. Extra capabilities that could then be supported include (but are NOT limited to):

Client Identification (and associated access rights – so different levels of information can be provided to different accessors)

Object accesses "leases" with defined expiration (all good things must end someday)

Encryption of qualifier contents (so no unauthorized changes to LSID) and whether these extensions have been added by the object owner, they are INVISIBLE to the accessing client. The standard just has to allow such extensions, not define them specifically.

4. Then, since the semantic web doesn't exist yet, you really want to leave the hooks in for the LSID to provide some aspects of Life Sciences context. Otherwise, a wonderful opportunity will be lost. We require that WSDL supported functions on LSID must include returning an XML file which can optionally describe the context of the LSID:

What the corresponding object is?

What defined WSDL functions are supported?

Whether the corresponding object is the latest revision?

What it is related to, and THEIR LSIDs?

or, instead of the last one, two additional optional functions:

LSID getParentLsid ()

LSID[] getChildrenLsids ()

might be used in those cases where the object was part of a tree hierarchy.

5. Replace the low-level untyped data stream capability with one based upon multi-attachment MIME packaging. In this way, a single LSID could reference ALL the views of an object at once, rather than representing only a fragment (a JPEG of a molecule, OR a word document whitepaper on it OR an Excel protein spreadsheet of it (?) or whatever).

SPECIFICALLY, the comments are detailed below:

1. URN Namespace for LSID's

The spec is not extensible and most of the defined fields should be opaque, but they are not. As such, it is flawed in overall design.

LSID PROBLEMS

A. No Extensibility

The last field, Revision ID, is optional! This means that NO OTHER FIELDS may be appended to the back of an LSID, without causing a great deal of confusion ... since how would you know if the new field (ex: Timestamp of when LSID becomes invalid) is actually the optional revision #??)

It would be much better to force the last character of an LSID to be a ":" field separator. If the revision # is not specified, the LSID ends with a "::". This is extensible and thus new fields can be added cleanly to the back.

Equality checking would then only be done against the defined fields.

Also, having a "*" as the Revision ID should always indicate that the latest version is wanted. This is precluded by the Revision # ID structure, which is defined to start with a "letternumber" but that can be easily changed. It is adding the wildcard concept to the current specification that represents the bigger impact, and it will be even more difficult to add into the open source code of the IBM LSID resolution service.

B. Overspecification

Addressing the problems defined in A. is more straightforward than the problem defined here (which is the major problem with this spec). ANY attempt to specify things beyond the point of URL/"LSID"/ (i.e. machine and port) is broken.

Past that point, things should be opaque. Perhaps this could be examined within the OMG (do some research into the CORBA object reference which provoked the same problems but was obviously resolved by a totally different solution).

Based on the role of the service to create the LSID and to give it out. Asking a resource provider to use a "standard" LSID assigning service is not the best approach, and the problem is compounded by taking the approach to implement a web service with a single "sprintf" function used to concatenate several supplied LSID subfields into a ":" delineated string.

The current spec assumes that ANYONE who is given an LSID (or who creates one in cleartext, or who finds the string on a disk, etc.) can FOREVERMORE retrieve ALL data about the object, with the rights indistinguishible from the object's owner, which we do not believe represents the Life Sciences business case this standard is trying to address.

Such resource providers will likely use their own INTERNAL ways of qualifying access to their internal data. Consider the present LSID with internals limited to "namespace, object and revision". How can an resource provider add the following functionality to the LSIDs it issues?

Add a Timeout: perhaps the LSID is only to be valid for a limited period? This (coupled with some of the additional LSID qualifications below), allows someone to sell (and someone to buy) such access.

Authorization: Confirms LSID is valid, for the data being accessed

Authentication: Confirms (with additional data supplied with the LSID) that the client seeking the resource is who they claim to be.

Encryption: Prevents external folks from changing things (as for example the timeout)

Restrictions on Use: Perhaps all the data should not be available to every LSID-wielding client. Are there any protected subobjects?

etc. etc.

The only way out of this, to the best of our knowledge, is to mandate that all LSID fields past the point needed to find the resource provider and indicate that this is an LSID, must be opaque to the standard (AND to the client who holds it).

2. Life Sciences Identifiers RFP Response

The resolver problems compound the problems in the proposal above.

The opening chapters introduce the reader to the limitations and mistakes of the LSID specification itself (see above), all of which are not challenged in the current form of these specs.

Then IBM proposes three LSID services. Let's look them.

A. LSID Resolution Service

This requires extensive discussion. There are several problems here.

Packaging:

The entire contents of ANYTHING retrievable by an LSID are assumed to be resolvable into nothing higher than a randomly accessible byte stream ... along with some metadata, describing what the stream data contains.

Query strings can only result in qualifying the metadata returned, and can not help qualify the stream data itself. So for example I can't ask the LSID to return me the Word document corresponding to the whitepaper describing the object I am retrieving. But I can ask the metadata to give me the start / stop bytes of that portion of the data stream containing the Microsoft word document. Then I have to make my best guess on how to store it in a way that anticipates how future Microsoft Word apps will be later understand. And the same for a compressed JPEG picture of the molecule (or whatever). And so on.

Instead, what should be done here is to use MIME the packaging and encoding technology that is already known everywhere where email is supported because it can

deal with multiple views of the same data and associated material as a series of typed attachments.

The IBM LSID Resolution Service doesn't appear to support an API higher than a data stream – is that correct? As such, the LSID Resolution Service is broken.

Context:

The next problem is related to context. If Lifesci objects are totally isolated, please disregard this section. If not, let's assume the objects referenced form the basis of a hierarchy. So given the hierarchy, I start with an LSID. I get the object data. Then what happens? What SHOULD happen, is that the API to the LSID Resolution Service should have an API which (perhaps optionally) covers:

LSID getParentLsid ()

LSID[] getChildrenLsids ()

Now (subject to my access rights) I can walk around the hierarchy of objects .. "browsing" for materials I need.

Presumably the same access rights, security and timeouts would be encoded in these new LSIDs ... but that is up to the discretion of the object provider. Should there be a requirement to encode such details?

A single "top level" object could be the bibliography for everything else ... one datatype returned is a picture of the object tree of the provider. Again, this doesn't have to be the case ... but it should at least be allowed.

Also ... it'd be REAL nice to be able to have:

bool isLsidLatestRev ()

LSID getLatestRev (LSID)

in the API.

Fixed Functions

The issue of WSDL and attribute definition needs to be expanded in this spec. For true expandability, the API of the LSID Resolution Service should support

```
getWSDL () // Returns an XML document which is the WSDL definition
           // of the services provided by this LSID (including
           // this one). No LSID argument needed

getContext (LSID) // Returns an XML file, whose schema is defined
by the           // LSID standards group, which supplies
object-specific // values for a common set of attributes
                //
                // Type, description, date, genome code (??),
....
                // (mostly keys you'd find in an XML database)
```

However, the Resolution spec does not support any of this. Its "product description" is:

ANYONE gives it a fixed format string (LSID). It allows them random access to a corresponding unformatted byte stream.

In short, it is understood that the lifesci community currently lacks any general methodology to identify and resolve its data, but we feel that the proposal before the OMG lacks the robustness required to make the tool useful and widely adopted by the community.

B LSID Assigning Service has some interesting points but is incomplete in this draft and needs to be expanded.

C LSID Resolution Discovery Service is not part of the specification but could probably best be done through something like UDDI.