Open discovery of library resources: The Digital Library Federation's ILS-Discovery interface recommendations

John Mark Ockerbloom, University of Pennsylvania

Available at: https://works.bepress.com/john_mark_ockerbloom/11/
[notes prepared for my presentation. What I actually said in the live presentation differs slightly from what is shown here.]

Last year, I worked with a Digital Library Federation task group to produce a recommendation of standard machine interfaces (or APIs) that an ILS should provide to support information discovery applications.
By the end of this talk, I hope you understand
* why we saw a need for such a recommendation
* what the recommendation basically says, and how it came about
* how the recommendation can be implemented
* and what the community can do to further advance interoperability of the resources in our ILSs

So, first of all: Why have standard discovery APIs in the first place?
Basically, our traditional ILS is much too limited. From our perspective, it’s a very important tool for managing library resources. But from the perspective of our users, it looks increasingly like one of those information silos we’ve decried, and not even a very big one. The OPAC that comes with a typical ILS does not come close to supporting the rich range of interactions our users now expect from the Web. And it often doesn’t include much information about useful resources online that they could use, even when those are resources our library has bought, or resources that can be requested from our peers.

Now, most ILS providers recognize the limitations of the traditional OPAC, and many of them now provide, or at least are promising, new “next generation” catalog interfaces.
For instance, they might offer a federated search product that lets you search the catalog and other library databases at the same time.
Or they might offer faceted browsing so you can search the catalog much like you might search Amazon or other e-commerce sites.

But these sorts of applications take a while to develop, and there are lots of new ways of discovering information popping up on the Web all the time.
For instance, maybe you want to apply some of the new semantic web linked data that’s now coming online, and let your users go beyond facets to explore subjects. In this application, for instance, users see books on cooperative cataloging right alongside books on related subjects like intellectual cooperation. They can freely browse among these concepts, even when there’s no facet relationship.

This isn’t the kind of application that a typical ILS vendor is going to be able or willing to roll out to its customers on short notice.

But if it’s possible for anybody to write an application that talks to any ILS in standard ways, then we have a much larger pool of developers that can build applications that showcase our library’s resources.
For instance, if your users find tagging useful, your library’s resources could be tagged in whatever social tagging environment your users prefer. They don’t have to use the one that your ILS vendor might provide.
And, we could embed links out to that tagging system from within the OPAC.
We can also introduce our own applications that show off the special features of our own libraries. For instance, at the University of Pennsylvania, we have an extensive collection of videos that support our programs in areas like Cinema Studies. We created a special application that lets our users look for videos in our catalog much like they would browse Netflix or IMDB. And usage of our video collection has soared. We had to write this application ourselves with custom code to extract information for our ILS. But if a program could easily talk to any ILS, we’d probably see off-the-shelf video catalogs we could have adopted, instead of having to write our own.

Now, you’ve just seen Andrew’s presentation, so you know that the nature of the ILS itself is evolving. Some libraries will probably move to using a more network-based, centrally managed library system.
Some may adopt a centralized framework...

Mockup shamelessly fabricated by the author

John Mark Ockerbloom  UNIVERSITY OF PENNSYLVANIA LIBRARIES  June 10, 2009

Maybe it will look something like this.

Maybe some of your patrons are already using services like this instead of your own OPAC and applications.
And there are also other ways to re-imagine the ILS. The Mellon-funded OLE project, for instance, is redesigning the ILS as a set of components in an enterprise-scale architecture. I could easily see many libraries in higher education or corporate settings adopting a framework like this.
But whatever form your ILS takes in the future, it's going to manage information resources that you want to offer to your users. We should be able to offer these resources to our users in whatever ways best serve their patrons. Or to put it another way, we should be able to shape the destiny of our data.
That’s the purpose of the DLF’s ILS-DI recommendation. Specifically, we recommend standardized services that an ILS provides to applications supporting information discovery. These applications help users find information resources in our library (perhaps by searching), acquire them (perhaps by making a loan request), and manage their usage (perhaps through renewing loans).

Now, there are other kinds of ILS integration that are useful as well. In this webinar, you’ve heard about integration possibilities for electronic resource management, workflows, and shared metadata management. Those are important too, but they’re not what our group dealt with.
The folks who worked on the recommendation represent many of the major research libraries in the US, including the Library of Congress, the National Library of Medicine, Harvard, and the University of California, as well as smaller institutions.

The ILS-DI task group

- John Mark Ockerbloom, Penn (chair)
- David Bucknum, Library of Congress
- Todd Grappone, USC
- Dave Kennedy, University of Maryland
- Emily Lynema, NC State
- Patricia Martin, California Digital Library
- Dianne McCutcheon, National Library of Medicine
- Terry Reese, Oregon State

The ILS-DI steering group

- Peter Brantley, DLF (chair)
- Dale Flecker, Harvard
- Marty Kurth, Cornell
- Terry Ryan, UCLA
- Robert Wolven, Columbia
After the task group was formed in 2007, one of the first things we did was conduct a survey of library professionals, both inside and outside of information technology, where we asked what they liked and disliked about their ILS, and what they most wished they could do with it.

We took that information, and the experiences of our task group, to compose an initial draft of recommended interfaces. We presented it to the DLF members for discussion in November.

After some revisions, we then met with ILS vendors and other developers in March to discuss the most important services needed from the ILS, and appropriate technologies for implementing those services. And we drafted the “Berkeley Accord”, where a number of ILS vendors and developers agreed to support the most critically needed services, using our recommended interfaces.

Shortly after that meeting, we published our first official recommendation in June. But when we started considering implementations, we realized there needed to be bit further discussion. So before we ended our activities, we had another developer summit, and publish a slightly revised version of the recommendation in December.
A layered recommendation

- Service requirements
  - 4 basic types of services: Data aggregation, Real-time search and queries, Patron services, OPAC interaction
  - E.g. “We need ways to extract bibliographic data from the ILS so it can be indexed and searched in other applications”

- Functions
  - 25 functions specifying particular behaviors, input, outputs, semantics for desired services
  - But in a technology-independent way
  - E.g. HarvestBibliographicRecords, HarvestExpandedRecords functions specify ways to get bibliographic data

- Bindings
  - Specific technologies, interfaces for functions (at least 1 for each)
  - Specifies protocols and data standards in implementation-independent way
  - E.g. OAI-PMH profiles for the Harvest...Records functions
  - Multiple bindings can exist for a given function, but it’s useful to recommend one preferred binding

So: What do these recommendations look like? At a high level, we tried to produce a layered recommendation that started with a set of requirements and expectations from libraries, and from there worked down to a specific set of functions and technologies that would meet those requirements.

Our recommendation doesn’t give a full specification of all of those technologies and standards. That would have taken too long to develop, and probably would have been obsolete the day it was published. Instead, we gave descriptions of the services and functions we found were needed, and then went into more detail about specification in the most critical areas.

We knew from our survey, for instance, that the applications our libraries want to use required 4 basic kinds of services. They need to be able to extract data from the ILS so that it can be indexed and searched. They need to query the ILS in real time, so that they can search for resources the ILS manages, and find out if they are currently available. They need to provide patron services, so that once a patron finds an item of interest, they can use it. And, because the ILS OPAC will be with us for a while, applications need to be able to guide the user to an appropriate OPAC context when necessary. We also want to be able to lead users out of the OPAC into richer discovery environments when that makes sense.

We didn’t use the term “service oriented architecture” in our recommendation, but that’s essentially what we’re recommending: basically, ILS components that provide well-defined services to discovery components.

From our four general service requirements, we came up with a set of 25 specific functions that provide these services. Functions specify how a service should interact with its clients, in a technology-independent way. For example, our description of the HarvestBibliographicRecords function says that, given a date and a supported format, the function should return all of the bibliographic records that have become available for discovery, or that have changed since that date, in the requested format. So in a function definition, we specify what inputs, outputs, and semantics are needed for an interaction with the ILS, but we don’t specify exactly how that interaction takes place.

We do that at the level of bindings. Bindings specify a particular technology to use for a functional interaction. For example, the OAI-PMH protocol can be used to service a HarvestBibliographicRecords request.

For each function, we cite a binding that can be used for it. Sometimes there are multiple bindings possible. You could for instance, export bibliographic records through Atom or RSS instead of through OAI. We separate our function definitions from our binding definitions so that ILS interoperability can easily adapt to different environments and new technologies. At the same time, we recognize that there’s some need for uniformity—you don’t want applications to have to deal with a half-dozen different protocols for requesting the same function. So we also try to recommend preferred bindings for each function.

Now, even at the level of bindings, we aren’t trying to dictate how an ILS should be implemented. Your bibliographic records can be stored in a relational database, or an RDF triplestore, or in a cloud-computing service, or anywhere else that’s useful. As long as applications can get those bibliographic records through a standard interaction, you can manage them however you like.
Even with that freedom of implementation, we recognize that ILS’s are not likely to implement all 25 functions we want overnight. So we classified our functions into different levels of interoperability, based on how critical they were for library needs. Our first level, which we call our Basic Discovery Interface, identifies 4 of those 25 functions that are most important. Using just these 4 functions, an application can extract data from an ILS so that it can build its own search services on that data. It can query the availability of an item in the ILS, so that a patron knows whether they can use it. And it can guide the user to the record of that item in the native OPAC, so that the user can make requests for it if necessary.

When we met with ILS vendors and developers, they agreed that they could support these functions, as long as there was a clear specification of the interaction. So at this level, we got quite specific about our recommended bindings.
Here, for instance, is an example of our recommended binding for GetAvailability. An application simply resolves a URL concerning a particular bibliographic record or item. The URL is based on a bibid, noted in red. That URL invokes a RESTful HTTP service that returns a simple XML structure reporting whether the item is available. It can optionally return additional information about how and where to get the item. In this case, we see there’s a copy of the item with the bibid we want in Van Pelt library. And that’s all there is to it.

Now, this binding specification was a bit unusual for us, because we actually defined some new XML elements for it. We generally prefer to just use existing standards for function bindings instead of inventing our own. But in this case, we wanted a simpler interaction than would be required if we used something like, say, NCIP. So we rolled our own, but we still reused some basic technologies like REST, HTTP, and XML to do it.
We hope that applications will be able to interact with the ILS in richer ways than those 4 basic functions allow. We’d like applications to be able to search the ILS in real time, for instance, or learn the ILS’s authority structures, or go back and forth between the OPAC and specialized applications. Those functions and others are defined as part of interoperability level 2.

Level 3 includes patron request functions that let an application act as an alternative to the native OPAC, and not just a supplement to it. And level 4 specifies a few additional functions that support rich discovery in domain-specific ways. For example, we define a function for searching course reserves, which is an important feature of many academic ILSs.

We don’t go into the same level of detail about bindings in these upper levels as we do with our basic level of interoperability. But we still describe the functions that are needed, we recommend one or more binding methods for each function, and we encourage people to develop their own implementations of these functions. Eventually, it may be useful to standardize bindings at these higher levels as well, but we sometimes felt that was premature.
Now, it’d be great if tomorrow we had a whole new generation of ILSs that support all of our recommendations. In reality, there are still going to be lots of legacy ILSs that don’t. Fortunately, it’s possible to implement the functions on top of existing ILSs. For instance, we can implement all of the 4 Basic Discovery Interface functions on our existing Ex Libris ILS we have at Penn. One of the functions already exists, in fact. Voyager 7 lets you link to a bibliographic record page based on a bib-id, and that’s enough to implement GoToBibliographicRequestPage. You just have to turn that feature on.

The other 3 functions are ones we or other libraries can implement. At Penn, we have an implementation of GetAvailability that queries our Voyager database tables. We hope to have it in production soon and distribute the code so that others can use it. Duke has also implemented GetAvailability on their Aleph system, using Ex Libris’ X-Services. Basically, they’ve used a vendor-specific API to implement the vendor-independent API that we recommend.

For the Harvest functions, there’s an OAI interface developed at the national library of Finland. It doesn’t quite have the exact semantics we call for, but it’s close enough that we hope we can adapt it to implement our specifications.

I’m told that Ex Libris is also developing its own OAI interface to support these functions.
And that’s really the sort of development we like to see. The ILS providers are the ones who have the best knowledge of their products, and they can make them easy to install, either directly built in to a new version of an ILS, or provided as an add-on.

As I mentioned earlier, a number of organizations, including all the ones you see here in red, have signed the Berkeley Accord, and agree to support the Basic Discovery Interfaces in future products. So you can ask them about what their plans are to provide these services. It's a good thing for customers and potential customers to do. There are a lot of standards out there, and it often takes a while for developers to get serious about supporting them fully. We’ve seen that with NCIP, for instance. But if vendors know that customers want these services, they’ll give them higher priority.

And you don’t want someone to just say “yes, we support this”. If the interface is going to be useful, a provider needs to fully document how it can be used in practical applications. They shouldn’t put unnecessary restrictions on what applications are allowed to use it, or how those applications can be distributed. And, while they can certainly implement the interfaces either in closed or open source, it’s very useful to have open source clients, or even open source service implementations, to make it easy to develop and test new applications on top of your ILS.
So that’s an overview of the recommendations. Now, what happens next?

As some of you may know, the Digital Library Federation is going away as an independent group at the end of this month; it’s being merged back into CLIR. But the ILS-DI recommendations are not going away. The task group completed its charge back in December, when we released our revised recommendation. That recommendation should give all the information needed to support the Basic Discovery Interfaces, and it points the way forward for higher levels of interoperability.

If we look at the ILS-DI process as a development spiral, we’ve moved from a specification stage to an implementation stage. A number of us in libraries are now developing implementations of the recommended APIs, and we’re hoping to see ILS-DI compliant services brought out by ILS providers as well. Some of us are extending some of the services to meet local needs; we’re doing that at Penn, for instance, for enhanced availability information in our video catalog.

And we’re not supposing that the current ILS-DI recommendation is the last word in interoperability. There are other projects working in similar or related areas, including Jangle, which has Atom-based APIs that implement similar functions to the ones we recommend. And I already mentioned OLE, a project looking to redesign the entire ILS around a service-oriented architecture.

You can also apply the recommendation to information resources that aren’t managed by an ILS. The first implementation I know of for the Basic Discovery Interfaces is one that I wrote for The Online Books Page. That’s not an ILS; it’s a catalog of free online books available anywhere on the Internet. But because it now supports the basic ILS-DI functions, discovery applications can take advantage of its metadata along with ILS metadata. Conceivably, you can have institutional repositories, electronic databases, and other collections of information also supporting ILS-DI functions, so that your library’s users can discover all kinds of resources available to them.

Now, as we see interoperability progress, it may well be useful to revisit the recommendation, to refine it as a standard, incorporate new technologies and standards, and to specify in more detail the higher levels of interoperability that we recommend. At that point, it will be good to have a trusted neutral party to sponsor further work, and make sure it progresses fairly and soundly. That might be someone like CLIR, or maybe NISO, or some similar organization.
But for now, there’s a lot that can be done. You can download and read the recommendation from the DLF website, and it will continue to be there even after CLIR takes over. We created a Google group where folks interested in the recommendation and its implementation can discuss it. I’ve also just created a new section on my blog site, Everybody’s Libraries, that will track new implementations of ILS-DI, as well as related activities, as I hear about them.

If you’d like to know more about ILS-DI and our work, I’d be happy to answer questions here or in email. And if you want to see support for them in your ILS and discovery applications, I encourage you to contact your providers.

Thank you.