DLF ILS Discovery Interface Task Group (ILS-DI) Technical Recommendation

An API for effective interoperation between integrated library systems and external discovery applications

Revision 1.1

Draft for public comment, October 30, 2008

ILS-DI Task Group Members

John Mark Ockerbloom, Univ. of Penn. (chair)

Terry Reese, Oregon State Univ.

Patricia Martin, California Digital Library

Emily Lynema, North Carolina State Univ.

Todd Grappone, Univ. of Southern California

Dave Kennedy, Univ. of Maryland

David Bucknum, Library of Congress

Dianne McCutcheon, National Library of Medicine

Table of Contents

1. Introduction

Today's libraries are outgrowing their traditional discovery tools. Information about the resources available to library users is commonly maintained in an Integrated Library System (ILS) that manages acquisitions, cataloging, circulation, and reporting. The ILS also provides a discovery interface (commonly known as the Online Public Access Catalog or "OPAC") that enables patrons to search for resources. The integrated functions of the ILS have helped streamline library operations, and the data the ILS manages gives valuable information about a library's collections. In recent years, however, users have grown to expect more. They want to be able to see resources available outside the scope of traditional ILS holdings, including journal articles, resources available at nearby institutions, and interactive forums. They want options for finding relevant content beyond traditional author/title/subject searching or generic keyword searches. When they do find something of interest, they want to use the library's services from wherever they are to obtain it. They want to integrate their library research and discovery with the other ways they carry on research and education, which often involve a wide variety of applications and online services.

The public interfaces currently provided by most ILS's cannot by themselves meet the demands of users in a world where the availability and sophistication of digital resources and web applications has increased significantly. This does not simply reflect badly designed interfaces; it reflects the fact that users now need a wider variety of capabilities than any one software package can be expected to provide. At the same time, the bibliographic data and services that the ILS manages are crucial for the effective use of libraries. These trends imply that the ILS needs to become a platform that supports appropriate interfaces for discovery applications living on top of it instead of trying to do everything on its own.

At present, a number of ILS vendors provide proprietary methods for accessing their underlying data stores. These may consist of command-line API tools accessible only for a special fee to trained users, or direct SQL queries against the ILS's database tables. While these methods provide much needed hooks into the ILS, making library resources more widely usable requires a larger, standards-based API. To enable outward integration, organizations will require that ILS's adopt a more standardized method for providing API access to the data store, moving away from traditional library-centric protocols like Z39.50 toward an XML-based web services API model. Such a model will enable developers outside of the library community to more easily access the information stored within the ILS, creating opportunities for greater integration with non-library applications like course management tools.

In the summer of 2007, the Digital Library Federation convened the ILS Discovery Interface Task Group (ILS-DI) to analyze issues involved in integrating integrated library systems (ILS's) and discovery applications, and to create a technical proposal for accomplishing such integration. This document is the technical recommendation of that group. It includes

- An overview of our methodology and approach.
- A summary of a survey of the needs and discovery applications implemented and desired by both academic and public libraries.

- A set of abstract functions that external discovery applications need to be able to invoke on ILS's and/or their data to support discovery and delivery.
- Recommendations for concrete bindings (specific protocols, data standards, etc.) that can be used to implement these functions in or on top of existing and/or future ILS's.
- More detailed specification of a specific subset of functions, the "Basic Discovery Interfaces", that have been agreed to by a group of ILS and discovery application developers and vendors.

Providing a complete set of binding specifications and reference implementations is beyond the scope of this small, short-term group, but we hope to provide sufficient requirements and details to allow others to produce appropriate bindings and implementations for the full set of recommendations we provide.

The initial recommendation of the group ("revision 1.0") was published in June 2008. In August 2008, the Digital Library Federation convened a meeting of ILS and discovery application developers to discuss implementing the "Basic Discovery Interfaces" subset of the recommendations. One of the outcomes of the meeting was a small set of recommended changes to our initial Basic Discovery Interfaces specifications to make them clearer and more useful to implement. Revision 1.1, this version of the recommendation, incorporates a number of those recommended changes. We also have added a summary of the requirements for full support of Basic Discovery Interfaces, and made some slight updates to reflect publication of some related standards that we cite in this document.

2. Methodology

The ILS-DI group is intended to be a small group working over a short duration: eight library professionals from various research libraries across the US, working from mid-2007 to early 2008. We aim to move quickly and work with the resources available to the library community to produce simple, effective, and practical recommendations. Towards that end, we aim in our recommendations to

1. **Improve discovery and use of library resources via an open-ended variety of external applications that build on the data and services of the ILS**. Our goal is not to specify or implement the applications themselves, but to specify interfaces that the applications can use. These applications may be local or remote, and may interact with more than just a single ILS.

2. **Articulate a clear set of expectations** so that ILS and discovery application developers know what services to provide and how they will interact. This includes describing specific functions and including their requirements, inputs, and outputs at a level detailed enough that implementers understand what to implement and clients understand what to expect. For example, instead of simply saying "support networked search" we describe the kinds of queries and responses that should be supported in a search.

3. **Make recommendations applicable to both existing and future systems and technologies**. Technologies, protocols, and standards are changing rapidly, but the basic functional requirements for discovery applications are in many ways independent of particular technologies. Therefore, we aim to specify both abstract functions that describe desired services in a technology-independent manner, and concrete bindings that implement these functions using a particular technology or standard. For example, we might specify an abstract function for extracting bibliographic records from an ILS and then define a particular binding of that function that provides MARCXML data via the OAI-PMH protocol. Note that bindings, as we define them, specify the interactions between the client and the implementation of a function, but do not dictate how the function is implemented internally.

4. **Ensure that the recommendations will be feasible to implement**, in whole or in significant part, in reasonable time and cost. Our aim is to keep our recommendations as simple and lightweight as possible for the desired functionality, and where possible be compatible with existing ILS's or a near-term revision of an ILS. The ILS may well undergo substantial redesign in the future, but we do not want to make a recommendation that cannot be implemented without completely reinventing the ILS. To support near-term prototyping, we suggest at least one binding for each function that uses current technologies. We also specify various levels of interoperability to support incremental adoption of the API recommendations. We are also specifying and promoting specific interfaces for an initial "Basic Discovery Interfaces" level of interoperability that supports the most crucial services for external discovery applications, and that we hope can be implemented and verified rapidly.

5. **Support interoperation and cooperation with applications outside the traditional library domain**. Library clientele today use a wide variety of applications, some designed by libraries and some created by unrelated entities, to locate interesting and relevant content. Once they find content, they use additional applications to store, analyze, and reuse it. We want to make sure that library resources can integrate well in this broader environment, and avoid having them isolated in a "content silo". We also want to recognize and reuse the standards and tools that are already in use or development outside libraries and could work well with library content and services. We do not want to needlessly limit these recommendations by requiring the use of standards like MARC or Z39.50 that have very limited use outside the library domain.

6. **Be responsive to the user and developer community**. To that end, we have conducted a survey of library developers and decision-makers, which is summarized in this report and described in more detail in a separate document. We have also met with the developers of ILS's and discovery applications to determine the most important functions for interoperation that ILS vendors could support. The outcome of that meeting was the "Berkeley Accord" (see Appendix 7), which includes the functions agreed on. These functions are represented in this document as the Basic Discovery Interfaces, or Level 1, interoperability profile.

We hope that these recommendations address many of the needs and desires exposed in the survey responses, and that the functional definitions supply a common reference point for rapid prototyping of interfaces and applications and discussion of present and future ILS capabilities. The recommendations in our report are meant as a starting point. As new applications and implementations are developed, we expect that new functions and bindings may be defined and publicized. Where appropriate, we hope that these can be incorporated into later documents building on the recommendations we make here.

3. Summary of survey results

In September 2007 the ILS-DI group conducted a library survey to measure community interest in and current work toward enhancing interaction between the ILS and external discovery applications. We received well over 100 responses.

Of the responders, more than 40% are considering adopting a new integrated library system (ILS) within the next 2 years. Of those considering a new ILS, 35% are evaluating open source options. 77% of responders are currently using external (non-ILS) discovery applications to supplement the functionality of their OPAC (this number includes integration of catalog results into a metasearch tool). Only 13% of responders indicated that their institution has no plans over the next 2 years to implement an external discovery application that utilizes catalog data. For institutions already using an external discovery application in addition to (or instead of) the traditional OPAC, the most common technique for accessing data managed within the ILS is through data export (27%). Application handoff, where the external discovery application links back to the OPAC, is commonly used to access functionality available only within the ILS (20%).

The results of the survey show a strong desire to move beyond the current OPAC functionality provided as part of the ILS. There was an overwhelming response that the current ILS search was inadequate and that data within these systems is difficult to work with, leading to slow adoption of new technology. Responses to open-ended questions identified several areas as problematic for the current ILS systems. Recurring themes from the responses include:

- Current systems are built for managing print collections and inventory. The functionality important for this aspect of collection management is not adequate for digital resources.
- Current OPACs are limited in their support for multiple metadata standards and lack support for Functional Requirements for Bibliographic Records (FRBR).
- The OPAC is limited in that it searches only items owned by the subscribing institution.
- The OPAC interface is difficult to use and is not intuitive compared to other search tools (particularly search engines and e-commerce sites). The more powerful features of the catalog search are mostly hidden or exposed in such a way as to confuse the users.
- Exploratory searching is difficult, and OPACs often lack basic features like spell checking and good relevance algorithms. Functionality does not encourage browsability or serendipity.
- Searching for known items can also be problematic, if users do not know exact titles or filing rules.

While there were a few (less than 5) responders who thought that the OPAC experience was adequate to good, the overwhelming response was to the contrary. A common thread in the responses was that the "siloing" of information in the OPAC was driving people to use services like Google and Amazon. The siloing effect refers to the fact that catalog searching is generally limited to locally cataloged resources, excluding licensed resources or other relevant external content. There were also a number of responders already looking to broader bibliographic search tools as a viable alternative to their OPAC. Some of the tools identified were vendor provided,

some were open source search applications, and some were web-based services such as OCLC's WorldCat.

More detailed survey results are described in a separate document, which will be made available at the ILS-DI task group's website.

4. Levels of interoperability, functions, and bindings

The remainder of this document contains specific recommendations for abstract ILS functions that promote interoperability with external discovery applications, and possible bindings for those functions. Since we do not expect that every ILS will immediately support all the recommended functions, and since some functions are more immediately needed, this section defines a series of usage profiles that represent increasing levels of interoperability. These profiles were derived from our own experiences building discovery tools, from needs expressed in the survey responses, and from feedback from ILS and discovery application vendors. This section also shows how the recommended functions are grouped into these profiles, and explains the difference between abstract functions and concrete bindings.

4.1 Levels of interoperability

- **Level 1: Basic discovery interfaces (BDI):** This level represents a minimal set of functions that are easily implemented and essential to support applications that provide discovery outside the ILS. The focus is on enabling external systems that provide new methods of discovery while still relying on the ILS for other traditional OPAC functionality. Some important functions are not included at this level, and we encourage vendors and developers to go beyond it where possible.

- **Level 2: Elementary OPAC supplement:** This level describes a set of functions needed for a reasonably broad range of practical discovery applications that operate in tandem with the OPAC. We assume here, to be appealing and straightforward for users, a supplement would need to support essentially the same breadth of discovery as the elementary OPAC alternative profile below. However, it might not need to support all the delivery functions, if those were better handled by the underlying OPAC. This use case requires functions for seamlessly passing control between the OPAC and external discovery applications, in both directions (not just to the OPAC, as in Level 1).

- **Level 3: Elementary OPAC alternative:** This level describes a set of functions needed for a practical discovery application that can operate completely independently of the OPAC. Such an application would need the essential discovery and delivery features of an OPAC, including search and browse, real time availability information, delivery, and patron services. While not all of the OPAC's functionality has to be replicated in the application, enough has to be available to make it attractive to users as an alternative to the normal OPAC interface.

- **Level 4: Robust/domain specific discovery platforms:** This level describes functions required to build useful discovery applications beyond the elementary level. It includes domain-specific functions that might not apply to all libraries, but that might be important for particular kinds of libraries (such as academic libraries dealing with course reserves or public libraries dealing heavily in e-commerce to handle fine transactions.)

Several ILS and application vendors and developers have pledged native support for Level 1 as described in this document. We therefore give special attention to the

specifications of Level 1 functions in order to support rapid and uniform implementation, in addition to summarizing these specifications in section 9.

4.2 List of functions by level

Level 1: Basic Discovery Interfaces

- HarvestBibliographicRecords (Data Aggregation, section 5.3.1)
- HarvestExpandedRecords (Data Aggregation, section 5.3.2)
- GetAvailability (Real Time Search, section 6.3.1)
- GoToBibliographicRequestPage (OPAC interaction, section 8.3.1)

Level 2: Elementary OPAC supplement

All of the above, plus

- HarvestAuthorityRecords (Data Aggregation, section 5.3.3)
- HarvestHoldingsRecords (Data Aggregation, section 5.3.4)
- GetRecords (Real Time Search, section 6.3.2)
- Search (Real Time Search, section 6.3.3)
- Scan (Real Time Search, section 6.3.4)
- GetAuthorityRecords (Real Time Search, section 6.3.5)
- Either OutputRewritablePage or OutputIntermediateFormat (OPAC Interaction, sections 8.3.2 and 8.3.3)

Level 3: Elementary OPAC alternative

All of the above, plus

- LookupPatron (Patron Functionality, section 7.2.1)
- AuthenticatePatron (Patron Functionality, section 7.2.2)
- GetPatronInfo (Patron Functionality, section 7.2.3)
- GetPatronStatus (Patron Functionality, section 7.2.4)
- GetServices (Patron Functionality, section 7.2.5)
- RenewLoan (Patron Functionality, section 7.2.6)
- HoldTitle (Patron Functionality, section 7.2.7)
- HoldItem (Patron Functionality, section 7.2.8)
- CancelHold (Patron Functionality, section 7.2.9)
- RecallItem (Patron Functionality, section 7.2.10)
- CancelRecall (Patron Functionality, section 7.2.11)

Level 4: Robust/domain specific discovery platforms

All of the above, plus

- SearchCourseReserves (Real Time Search, section 6.3.6; for academic libraries)
- Explain (Real Time Search, section 6.3.7)

- Both OutputRewritablePage and OutputIntermediateFormat (OPAC Interaction, sections 8.3.2 and 8.3.3)

4.3 Abstract functions and bindings

As noted earlier, this recommendation specifies both abstract functions (which do not specify a particular technology to use, but impose requirements on what any technology implementing them should do) and possible concrete bindings (which specify particular technologies and interface details specific to those technologies). For some bindings, we specify full details; for others, we simply note the possibility of creating such bindings. Since the functions are abstract and multiple bindings are possible, any ILS interface that implements this recommendation must fully and openly specify its function bindings and how they can be used.

In the sections that follow, we group similar functions into 4 broad categories for comprehensibility (Data aggregation, Real Time search, Patron functionality, and OPAC interaction). Most of these categories contain functions across several levels in the usage profiles.

This section describes the basic template we use for the description of abstract functions and bindings.

Rationale and general issues

Each category section opens with an explanation of the purpose of the category of functions. Where appropriate, example use cases for these functions are also discussed. These are intended to aid in understanding the rationale and use of the functions in the section.

Abstract functions

Each abstract function specification should include the following pieces of information.

The name of the abstract function
This is an abstract name used for reference in this specification. Bindings do not necessarily have to use this name, but it can simplify matters if they do.

Summary
A short synopsis of what the function does.

Parameters
For each possible input to the function, we give its name, what kind of parameter it is, whether it is required or optional, and a short summary of its purpose.

Returns
A summary of the output of the function, if any.

Exceptional conditions
For each applicable exceptional condition, we give an abstract exception name, and a summary of the conditions under which the exception may arise. Note that bindings

do not have to actually model these as exceptions in their underlying protocol or programming language, nor do they have to use the exact names for the exceptions. Alternative mechanisms for dealing with exceptions may include null or otherwise distinguished return values, auxiliary functions that signal the occurrence or possibility of exceptional cases, or even appropriate interface documentation. Bindings may also implement more specific exceptions than the ones defined here.

Side effects
If the function has any noticeable or lasting effect other than returning information, that should be noted here. If not, "None" should be stated here.

Rationale
A short summary of why the function is needed. This should be relatively short, if it appears at all; the rationale may already be covered in the general overview.

Notes
Paragraphs under the Notes heading go into more detail about the required behavior of the function. They may also note some of the required data and services implied by the functions, give some examples of how the function might be implemented or used in practice, and give other hints for implementation and use.

Possible Bindings

We here name and briefly describe one or more ways in which this function could be bound to a particular technology. Some of these may be more fully specified in the general Binding details subsection below.

Binding details

We may have a general Binding details subsection after the abstract function descriptions in each section. It can be useful to distance the bindings specifications slightly from the function specifications, both because it helps avoid unnecessarily tying abstract functions to a particular technology and because the same general binding details may be used for multiple functions in a given category.

5. Data aggregation

Many external discovery applications need to maintain external copies of ILS data. In this section, we define standard functions for extracting, or harvesting, ILS data in bulk.

5.1 Rationale and general issues

Many external discovery applications need data managed by the ILS to build their own independent index of metadata. They may, for instance, build indexes that support rapid search and retrieval techniques not supported by the ILS itself. They may need a selective index of ILS metadata, or an aggregated index that combines ILS metadata with information from other sources. Bibliographic metadata is of particular interest, though authority, holdings, and other item metadata (such as circulation information) can also be used by external applications.

While data harvested from the ILS might not be as current as the data within the ILS, it is good enough for many purposes. This is particularly true since bibliographic and authority metadata does not change frequently. Applications that need up-to-the-minute ILS data can use real time search and query functions (the subject of the next section). Real time queries can use the identifiers of harvested records to retrieve additional or updated information related to harvested records as needed. These identifiers may exist outside the actual metadata records (for example, a bibid not included in a MARC record). Harvesting identifiers that persist over time is often important for supporting further discovery and services on exported metadata.

Harvesting all of the relevant data from an ILS can be an expensive operation. Selective harvesting, including incremental harvesting of data that has been added or changed since a certain date or time, can greatly reduce the cost and should be supported along with full harvesting. Selective harvesting based on pre-defined sets may also be useful.

Selectively filtered harvesting may be necessary in some cases if metadata records in the ILS have been licensed from a third party that does not allow redistribution. If this is the case, the client doing the harvesting needs to be aware of such filtering.

There may be multiple formats for metadata records. For example, some bibliographic records may be stored in MARC and others in MODS. It is also useful to allow a variety of export formats for some records, such as native MARC 21, MARC-XML, or Dublin Core.

5.2 Sample use cases

Some possible use cases include

- Building a duplicate index of ILS data; for example, a Lucene index of bibliographic records that can be searched with facets using Solr.
- Making a specialized index of selected material from the ILS; for example, a catalog of video recordings that supports special searches based on actors, directors, and other video-specific features.

- Making an aggregated index that includes records from multiple ILS's and other databases that can be searched in a single operation, without requiring slower or less reliable federated search mechanisms.
- Building an index of authority records to enable subject-based browsing, name and subject suggestion features, and other discovery aids.
- Harvesting both bibliographic and licensing information (which may be managed by the ILS or by another application such as an ERMS) to support appropriately-scoped discovery services for different audiences and uses of content.
- Harvesting holdings, item, and circulation information for services tailored to a specific library environment, such as date-sensitive citation resolvers, or relevance rankings weighted by usage. While circulation status on an item can change minute to minute, the status of a typical item does not change frequently. (Some books stay on the shelves for years without circulating, for instance).
- Harvesting recently added or changed bibliographic records for current awareness services.
- Harvesting for external metadata transformation, cleanup, relationship (FRBRizing), vocabulary mapping and other processing services.

5.3 Abstract Functions

5.3.1 HarvestBibliographicRecords (Level 1)

Summary
Returns a set of bibliographic records (and their identifiers) that are relevant for discovery in the ILS. When dates are specified for incremental harvesting, records that are newly available for discovery, have been changed, or are no longer available for discovery in the ILS are returned.

Parameters
- **from** (type date or time; optional): Only include records added or that have changed content or status since the specified time.
- **until** (type date or time; optional): Only include records added or that have changed content or status up to a specified time.
- **format** (type enum; optional): Specifies the metadata format to be returned.
- **set** (type string; optional): Only include records in the specified set. (In many cases, this will effectively be an enumerated type.)

Returns
A list of identifiers, each accompanied either by a bibliographic record or a notice that the associated record is no longer available for discovery.

Exceptional conditions
- **NotSupported**: The underlying system cannot accurately answer the query with the supplied parameters.
- **InvalidRequest**: The underlying system considers the supplied parameters invalid.

Side Effects
None

Rationale
Many discovery systems need to index bibliographic records independently of the ILS. This function allows all or part of an ILS' bibliographic records to be exported for aggregation. The whole catalog can be exported, or just a selection, such as records that have been changed since an earlier harvest.

Notes

The records should be in a well-specified format, and have all the details that are relevant to discovery. If MARC records are maintained or produced by the ILS, then records should be available in MARC or the semantic equivalent. However, the records may be returned in alternative representations (and some bindings require this). For example, a MARC record stored as relational table elements could be returned in some procedural bindings as native Marc21 binary format, but in other bindings, such as the OAI-PMH binding, the "marc21" MARC XML schema would be returned.

This function should return records that are available for discovery. Records that are in the ILS but are suppressed from user display, for example, are not available for discovery. In incremental harvesting, records not available for discovery should be identified as deleted records. It is permissible for records to be included that are available for discovery, but have restricted export conditions. The documentation or binding used for this interface should make it clear whether or not such records are included, and under what conditions.

Each record must have a unique identifier, and that identifier is assumed to persist (if not forever, then as long as the records are managed by the underlying ILS, and the ILS undergoes no major changes), so that it can be used in later queries and services. The ILS' "bib id" might serve as this identifier if it is reasonably stable. The identifier is unique within the ILS, not necessarily globally.

To support the from and until parameters for incremental harvesting, the underlying system must keep track of when bibliographic records were last changed (or added for the first time). Any change that results in a different bibliographic record value, or a different status value (such as suppressed to unsuppressed) should be tracked, so that harvesters may keep up to date records. Tracking deleted records (or records withdrawn from discovery) is necessary at least for a time. The ILS should document how long such records are tracked, and track them at least long enough that they will be noted in a reasonably scheduled incremental harvest..

The NotSupported condition may be needed to signal the caller that a date or set restriction can't actually be calculated correctly. The caller may be able to get an answer by removing the parameter that can't be handled. There may also be an InvalidRequest response indicating that the system recognizes the request to be invalid (which is distinct from not supporting the request). These two conditions may be further specialized, into messages indicating unsupported sets, time formats, metadata formats, etc. We do not at this time specify the full set of conditions.

The set parameter may be relevant for exporting well-defined subsets of the catalog. For example, for supporting a video catalog, an ILS might place some bibliographic records in the "video" set, and support full or incremental harvesting of just those records.. We do not here specify which sets are defined, and how they are defined,

but making it possible and convenient for special-purpose subsets to be available for applications that need them is a useful ILS feature.

It may be useful for there to be functions that return the sets and metadata formats available for harvesting, but it is not required in this profile. (In the OAI-PMH binding, the ListSets and ListMetadataFormats verbs are suitable for this.)

Possible Bindings

- **OAI-PMH binding (recommended)**: The functionality above maps fairly straightforwardly to OAI-PMH. Detailed specification will be given in an OAI-PMH binding profile (specified in section 5.4 below, "Binding details"). This binding is required for full Basic Discovery Interface support. This implies that BDI implementations must make Dublin Core records available, as required by OAI-PMH, as well as MARC XML records where applicable.
- **Other bindings**: Since this is an expensive, data-intensive operation, it may also be useful to have more specific library implementations closer to the ILS. A Java or Perl object library could be more efficient, for example. However, if it is not unacceptably inefficient, a web service binding (whether OAI-PMH or some other implementation) is likely to be more portable and robust.

5.3.2 HarvestExpandedRecords (Level 1)

Summary

Returns a set of bibliographic records and supplementary information about the associated holdings and items that are relevant for discovery in the ILS. When dates are specified for incremental harvesting, records that are newly available for discovery, have been changed, or are no longer available for discovery in the ILS are returned.

Parameters

- **from** (type date or time; optional): Only include records added or that have changed content or status since the specified time.
- **until** (type date or time; optional): Only include records add or that have changed content or status up to a specified time.
- **format** (type enum; optional): Specifies the metadata format to be returned.
- **set** (type string; optional): Only include records in the specified set.

Returns

A list of identifiers, each accompanied either by an expanded record or a notice that the associated record is no longer available for discovery.
The expanded records must include (where available):

- The bibliographic identifier
- The bibliographic record (identical to what is available via HarvestBibliographicRecords)
- Any associated MARC holdings records (and their identifiers)
- Identifiers for any items associated with this record

For each of these items (or, where more suitable, for the entire record), the expanded records should include (where available):

- Location (library building, and location within building
- Call number and scheme (e.g. LC, Dewey, SuDoc, NLM...)
- Format

- Availability (available, not available, unknown). This may further include the following:
    - Status message (if not already part of location, 'checked out' vs. 'on order', for example)
    - Whether the item circulates
    - Due date
    - Number of holds
- Barcode
- Item notes
- Item creation date/time
- Total number of loans
- Item last activity date/time
- Other special information about the item kept by the ILS and important for discovery

Exceptional conditions

- **NotSupported**: The underlying system cannot accurately answer the query with the supplied parameters.
- **InvalidRequest**: The underlying system considers the supplied parameters invalid.

Side Effects

None

Rationale

Searchable indexes to library holdings often need information beyond the bibliographic record. Searchers may want to filter or sort by location, availability, usage, and other ancillary data kept by an ILS but not present in the bibliographic record. We therefore need ways of extracting this information as well.

Notes

Each record should have a unique bibliographic identifier, and that identifier is assumed to persist (if not forever, then as long as the records are managed by the underlying ILS, and the ILS undergoes no major changes), so that it can be used in later queries and services. The ILS' "bib id" might serve as this identifier, for instance, if it is reasonably stable. In some ILS's, holdings and item identifiers do not persist as long as bibliographic identifiers, but they should be stable enough to normally be consistent in the time period between harvests.

To support the from and until parameters for incremental harvesting, the underlying system must track when the bibliographic and/or marc holding records were last changed (or added for the first time). Any change that results in a different bibliographic record value, or a different status value (such as suppressed to unsuppressed) should be tracked, so that harvesters may keep up to date records. In addition, new or changed information associated with a bibliographic record, such as an item's call number, location, or circulation status, should also trigger that record to be included in the next incremental harvest. Tracking deleted records (or records withdrawn from discovery) is necessary at least for a time. The ILS should document how long such records are tracked, and track them at least long enough that they will be noted in a reasonably scheduled incremental harvest.
Of the optional extended information examples given above, the most important elements, based on the libraries and applications we have surveyed, appear to be call number, location, format (if maintained at the item level), and availability status,

as these are commonly used for display or filtering. Although availability status can change from minute to minute, some discovery applications use it in their search indexing to provide quick filtering of relevant materials. (In most libraries, only a small fraction of items change their availability status on any give day.) A discovery application can supplement harvested availability status with real time queries, or frequent incremental record harvesting, to keep discovery displays up to date.

It is possible that an implementation of HarvestExpandedRecords may offer multiple record formats in the return values, with more or less expanded details provided in each. The more information is returned, the more useful the records may be for the clients, but the more work is required both to pull together the records.

Possible Bindings

- **OAI-PMH binding (recommended):** The functionality above maps fairly straightforwardly to OAI-PMH. Detailed specification will be given in an OAI-PMH binding profile (specified in section 5.4 below). This binding is required for full Basic Discovery Interface support. This implies that BDI implementations must make Dublin Core records available, as required by OAI-PMH, as well as MARC XML records where applicable.
- **Other bindings:** Since this is an expensive, data-intensive operation, it may also be useful to have more specific library implementations closer to the ILS. A Java or Perl object library could be more efficient, for example. However, if it is not unacceptably inefficient, a web service binding (whether OAI-PMH or some other implementation) is likely to be more portable and robust.

5.3.3 HarvestAuthorityRecords (Level 2)

Summary

Returns a set of authority records (and their identifiers) that are relevant for discovery in the ILS. When dates are specified for incremental harvesting, records that are newly available for discovery, have been changed, or are no longer available for discovery in the ILS are returned.

Parameters

- **from** (type date or time; optional): Only include records added or that have changed content or status since the specified time.
- **until** (type date or time; optional): Only include records added or that have changed content or status up to a specified time.
- **format** (type enum; optional): Specifies the metadata format to be returned.
- **set** (type string; optional): Only include records in the specified set. (In many cases, this will effectively be an enumerated type.)

Returns

A list of authority records and their identifiers.

Exceptional conditions

- **NotSupported**: The underlying system cannot accurately answer the query with the supplied parameters.
- **InvalidRequest**: The underlying system considers the supplied parameters invalid.

Side Effects
None

Rationale
ILS authority records give important supplementary information to the bibliographic records. For instance, they provide alternative forms of names and subjects, which are very important to support in a robust search (since users may well use the alternate forms rather than the "authorized" forms). They also include a wealth of other information, such as notes on scope and relationships between entities that can be indexed externally and used to suggest authoritative forms of names and subjects for users working with natural language searches.

While some authority records are available from third parties, these records are not easily downloadable in many cases. Furthermore, the authority records stored directly in the ILS often form a customized set of information about authorized and related headings of particular relevance to the resources managed by that ILS.

Notes
The records should be in a well-specified format, and have all the details that are relevant to discovery. The exact representation of the records may change, however. For example, a MARC record stored as relational table elements could be returned as native marc21, or in the "marc21" XML schema used by OAI-PMH.

Each record should have a unique identifier, and that identifier is assumed to persist (if not forever, then as long as the records are managed by the underlying ILS, and the ILS undergoes no major changes), so that it can be used in later queries and services. The ILS' "authority id" may serve as this identifier if it is reasonably stable.

To support the from and until parameters, the underlying system must track when authority records were last changed (or added for the first time).

The NotSupported condition may be needed to signal the caller that a date or set restriction can't actually be calculated correctly. The caller may be able to get an answer by removing the parameter that can't be handled. There may also be an InvalidRequest response indicating that the system recognizes the request to be invalid (which is distinct from not supporting the request). These two conditions may be further specialized, into messages indicating unsupported sets, time formats, metadata formats, etc. We do not at this time specify the full set of conditions.

It may be useful to define sets for different kinds of authorities, so that, for instance, name and subject authority records can be harvested separately if desired.

Possible Bindings
- **OAI-PMH binding**: The functionality above maps fairly straightforwardly to OAI-PMH. Using the OAI-PMH binding, authority records could be defined as a specific set or class of records to be harvested. Detailed specifications of an OAI-PMH binding can be found below in section 5.4, "Binding details".
- **Other bindings**: Since this is an expensive, data-intensive operation, it may also be useful to have more specific library implementations closer to the ILS. A Java or Perl object library could be more efficient, for example. However, if it is not unacceptably inefficient, a web service binding (whether OAI-PMH or some other implementation) is likely to be more portable and robust.

5.3.4 HarvestHoldingsRecords (Level 2)

Summary
Returns a set of structured (generally MARC) holdings records (and their identifiers) that are available for discovery in the ILS. When dates are specified for incremental harvesting, records that are newly available for discovery, have been changed, or are no longer available for discovery in the ILS are returned.

Parameters
- **from** (type date or time; optional): Only include records added or changed since the specified time.
- **until** (type date or time; optional): Only include records added or changed up to the specified time
- **format** (type string; optional): Specifies the metadata format to be returned
- **set** (type string; optional): Only include records in the specified set.

Returns
A list of holdings records and identifiers for holdings and associated bibliographic records.

Exceptional conditions
- **NotSupported**: The underlying system cannot accurately answer the query with the supplied parameters.
- **InvalidRequest**: The underlying system considers the supplied parameters invalid.

Side Effects
None

Rationale
Many discovery applications can use data from MARC holdings records to provide such information as call numbers, location of materials and extent of serial holdings. This function allows all or part of an ILS' MARC holdings records to be harvested for aggregation. The full set of holdings in the catalog can be harvested, or just a selection, such as records that have been changed since an earlier query. While holdings information may be returned in HarvestExpandedRecords (see above), it can also be useful to have a function specifically for retrieving holdings information.

Notes
The records should be in a well-specified format (generally based on MARC holdings) and have all the details that are relevant to determine the extent of holdings inasmuch as the data from the ILS can provide. The exact representation of the records may change, however. For example, a MARC record stored as relational table elements could be returned as a MARC holdings record in native MARC21 or MARC XML or in ISO 20775, an XML holdings schema.

Each holdings record should have a unique identifier, and that identifier is assumed to persist (within the underlying ILS), so that it can be used in later queries. In order for external applications to correctly associate these holdings records with a bibliographic record, the unique identifier of the associated bibliographic record must also be included.

To support the from parameter for incremental harvesting, the underlying system

must track when holdings records were last changed (or added). The NotSupported exception may be needed to signal the caller that a date or set restriction can't actually be calculated correctly. The caller may be able to get an answer by removing the parameter that can't be handled.

If the implementation of HarvestExpandedRecords includes holdings records, it is permissible for this function to be implemented by the same underlying method, as this function effectively returns a subset of the data in HarvestExpandedRecords in that case.

Possible Bindings

OAI-PMH binding: The functionality above can be implemented in OAI-PMH. Holdings export support could be implemented as a set within the larger context of available records. Detailed specification will be given in an OAI-PMH binding profile (specified in the following section).

5.4 Binding Details

OAI binding
OAI-PMH gives a fairly straightforward binding for the Data Aggregation functionality.

The HarvestBibliographicRecords function, for instance, can be bound to OAI-PMH's ListRecords function, with the since parameter modeled by ListRecords's from element, and the set parameter modeled by ListRecords's setSpec element. The exceptions above would be implemented by the more specific error messages of OAI-PMH (badArgument, noRecordsMatch, etc.)

In the return value, the OAI-PMH record header identifier would consist of a constant prefix followed by the bibliographic identifier, and the record metadata element would consist of an XML encoding of the bibliographic record. In the common case where the ILS maintained MARC 21 records, OAI-PMH's "marc21" MARC XML record schema could be used. Note that OAI-PMH requires export of unqualified Dublin Core versions of the records as well.

If a single OAI-PMH interface binds multiple aggregation functions, it may need to use set prefixes to distinguish object spaces. If both bibliographic and authority records are returned via the same interface, for example, the setSpec "bib" could be used for bibliographic records, and the setSpec "auth" for authority records. Then, sets in each function would be modeled by subsets; for example, the set "video" in the HarvestBibliographicRecords function, if defined, would be modeled in the OAI-PMH interface by the set "bib:video". As an alternative to function-specific sets, harvesting functions could be handled by different OAI-PMH base URLs, so that bibliographic records are harvested in one place, authority records in another, and so on. As another alternative, HarvestBibliographicRecords and HarvestExpandedRecords could be implemented for the same sets, but return different metadata formats.

A special-purpose element in the OAI-PMH Identify return value would describe the functions supported, the set namespaces used, if applicable, and other exceptional conditions and implementation details a client should be aware of. (This is how the NotSupported exceptional condition would be handled in this case, since OAI-PMH

does not support general-purpose exception handling.)

We define the element as follows:

- **ilsharvest** is the top level element. It contains one or more of the following elements:
    - **harvestcollection** element, identifying a particular collection of records available for harvesting via this service.
        - It has the attribute **type** which describes the type of collection. The following attribute values are supported here:
            - "bibliographic" for bibliographic records
            - "authority" for authority records
            - "expanded" for expanded records
            - "holdings" for holdings records
        - harvestcollection also has an optional **set** element. The content is a string that identifies the setspec that should be used to harvest the collection, if any. (If omitted, it's assumed that all the records in the service are from this collection.)
        - harvestcollection also has an optional **fullmdformats** element. The content is one of more **fullmdformat** elements that contain strings that identify the recommended metadata formats for harvesting full information from this collection. This is to distinguish them from other metadata formats listed in the OAI ListMetadataFormats request for the set that might contain reduced information. For example, marc21 format may contain information that's stripped out of the oai_dc version.
        - harvestcollection also has an optional **embargoed** element, that specifies what is done with records that should not be exported externally.
            - The embargoed element has an attribute **included** that says whether or not embargoed records are included. Defined values are "true" and "false".
            - The suppressed element also has an optional **inset** element. Its content is the name of the set used to harvest only embargoed records, if such a set exists. If available, this would ordinarily be a subset of the set used for this collection.
            - The embargoed element also has an optional **outset** element. Its content is the name of the set used to harvest only non-suppressed records, if such a set exists. If available, this would ordinarily be a subset of the set used for this collection.
        - harvestcollection also has an optional **notes** element. Its content includes notes about what kinds of records should be expected in this collection, and other notes about the collection of interest to harvesters. One possible use of this is to describe what kinds of information is in the "expanded" set, for instance. It might be useful to have a structured description of that, but in this specification we just make it a free-text element.

For HarvestBibliographicRecords, the individual records should be available in the oai_dc format, and should also be provided in the marc21 format, in addition to

other possible formats. These may either be returned as top-level elements, or enclosed inside dlf:record elements, as described in section 6.4.

We do not here fully specify the exact format to use for HarvestExpandedRecords, but it should return XML elements that include both the bibliographic record and the associated holdings and/or item data for that record. The bibliographic records embedded in the expanded records should be the same as those returned by HarvestBibliographicRecords. The top-level structure of the expanded records is described in section 6.4.2, (Putting it all together), with examples of these records in Appendixes 2 and 3 (each using different lower-level standards for bibliographic and item information.) We are not aware of a current standard that includes all of the information we suggest for HarvestExpandedRecords at all levels, but the XML structures returned should include references to applicable schemas, documented so that client implementers know how to retrieve relevant data. Data elements used in multiple parts of this recommendation, such as in real time availability queries and search results, should be consistent across functions wherever possible.

6. Real Time Search

6.1 Rationale and general issues

While more and more organizations are looking to build discovery tools that make use of locally harvested and indexed content from the ILS, accessing real time data is necessary to integrate these tools with the ILS. Information such as an item's circulation status or availability are sufficiently ephemeral to require real time data access to accurately represent an item's current state. The presence of a real time data access API will complement large scale data harvesting mechanisms provided by the ILS. Record identifiers that can be used by external discovery applications to correlate their harvested metadata with metadata stored in the ILS are important requirements of this functionality.

The capacity to perform rich, real time searches also remains an important feature for organizations looking to integrate ILS content into other types of applications. Current generation library tools such as federated search rely on the ability to perform real time searches against an institution's collections, including the catalog. Although ILS vendors have provided access to some of this data via Z39.50, the protocol's library-centric nature makes it marginally useful when dealing with non-library groups interested in using ILS data.

Developed before the advent of XML or web services, Z39.50 has traditionally provided the library community with a unified method for retrieving data across ILS's. However, as the use of XML and XML-based protocols became more commonplace, the U.S. Library of Congress issued a revision to the Z39.50 protocol commonly known as the SRU/W standard. Adopted around 2000, the SRU/W protocol replaces the antiquated Z39.50 protocol, providing access to one's bibliographic content via an XML-based query service. In this specification, we encourage adoption of the newer SRU/W standard, which encompasses the functionality of z39.50. We also recommend that an SRU extension be implemented for OpenSearch, allowing non-library applications that use the increasingly common OpenSearch specification to easily include ILS data.

For most of the real time search functions, metadata could be returned in multiple formats. While MARC, MARCXML, or MODS may be sufficient for bibliographic information, other standards must be used for availability information and other item-related data. Dublin Core may also be useful, particularly for course reserve records. Where possible, API functions should allow a user to request a specific metadata format, and API providers will need to identify the metadata formats they support.

6.2 Sample use cases

Some possible use cases include:

- Enabling the ILS as a target for metasearching via a standard federated search product or other discovery tool. Capabilities should include result paging, sorting, and a minimum array of query types and limits. Ideally, the search interface should be at least as feature-full as the OPAC's own search interface.

- Providing real time access to requested bibliographic record(s) via record identifiers, including holdings, availability, and circulation information as needed. For example, an external catalog search tool that uses a Lucene index for bibliographic metadata might also display current item availability retrieved via a real time query.
- Integrating course reserve lists maintained in the ILS with course management systems, via a script or web service that retrieves a list of reserves by course and/or instructor.

6.3 Abstract Functions

6.3.1 GetAvailability (Level 1)

Summary
Given a set of bibliographic or item identifiers, returns a list with availability of the items associated with the identifiers.

Parameters
- **id**: (type string; required): list of either bibliographic or item identifiers
- **id_type**: (type string; required): defines the type of record identifier ("bib" or "item") being used in the request.
- **return_type** (type string; optional): requests a particular level of detail ("bib" or "item) in reporting availability.
- **return_fmt** (type string: optional): requests a particular format or set of formats in reporting availability.

Returns
- A list of item availability objects that represent all the availability of the items associated with the requested bibliographic / item identifiers.
- If no items are associated with a requested system bibliographic identifier, this may return an empty list or RecordNotFound message as the value in the hash for that bibliographic identifier. Availability may be reported at the bibliographic level if a bibliographic identifier is used in the request and there is not a return_type parameter requesting item-level availability. Returning bibliographic-level availability if requested is optional for Basic Discovery Interfaces functionality. The dlf:simpleavailability schema binding supports bibliographic-level as well as item-level availability, and is described in Section 6.4.1.
- If no records match any of the requested identifiers, return null or RecordNotFound message.

Exceptional conditions
- **NotSupported**: The underlying system does not support this type of request.
- **RecordNotFound**: Identified record was not found.

Side Effects
None

Rationale
Enables external discovery systems that maintain their own index of bibliographic information to display real time availability of results, rather than availability data

that is stale.

Notes

- Availability may be shown at the bibliographic level (if the availability data structure supports this), at the item level, or both. If the identifiers requested are of id_type "item" or the return_type parameter specifies "item", then availability information must be shown at the item level if it exists.
- Data returned from this function for each item should include the following information, where it exists. (All of this information can be included in a dlf:availability element.)
  - **bibliographicIdentifer** (required, type string)
  - **itemIdentifier** (required, type string)
  - **dateAvailable** (required when applicable, type dateTime
  - **status** (required, type string). The vocabulary for this return value depends on the binding; in this recommendation, we show examples from vocabularies defined for NCIP, ISO 20775, and dif:simpleavailability.
- It is also useful for this function to return additional data on request, such as the data elements below. Some of this information is included in dlf:availability; some can only be returned if other return formats are requested::
  - **location** (optional, type string, repeatable) - use case: patron going to the stacks to pick something up
  - **call number** (optional, type string) - use case: patron going to the stacks to pick something up
  - **circulating** (optional, type boolean) - whether material generally circulates to the primary user population
  - **holdQueueLength** (optional, type int) - number of holdings placed on title/item

Possible Bindings

- **Recommended binding:** REST call and XML response
  - REST URL template: GET http://{service}/availability/?id={identifier}+{identifier}+{identifier}&id_type=[item|bib]&return_type=[item|bib]&return_fmt={format}
  - This binding is required for full Basic Discovery Interface support.

The return value should be a dlf:collection XML element with appropriate availability elements. The default availability element is a dlf:simpleavailability element. The structure of the dlf:collection element and the dlf:simpleavailability elements are described in section 6.4.1. It is possible to include the dlf:simpleavailability element at the bibliographic record level as well as the item level. Examples of valid GetAvailability responses using dlf:simpleavailability can be found in Appendix 4.

As described in section 6.4.1, multiple representations of availability may be returned. The simplest possible response, and the one required for full Basic Discovery Interface support, uses the dlf:simpleavailability element. For more detailed availability, NCIP or ISO 20775 Holdings schemas could be used. Examples of valid GetAvailability responses using NCIP, and ISO Holdings (in combination with dlf:simpleavailability) can be found in Appendix 5.

Other Bindings

- **SRU/W**: While SRU/W may not be a perfect match for this functionality, the ability to define and create context sets for rendering data elements should provide a great deal of flexibility, allowing SRU/W to accommodate the requested data.
- **OpenURL**: An OpenURL structure could be used to define a structured URL to request information, although OpenURL does not define a response format for the requested data. We do not define a particular OpenURL structure to use in this document.

6.3.2 GetRecords (Level 2)

Summary

Given a list of record identifiers, returns a list of record objects that contain bibliographic information, as well as associated holdings and item information. The caller may request a specific metadata schema for the record objects to be returned. Individual API providers will need to enumerate the supported metadata schemas. This function behaves similarly to HarvestBibliographicRecords and HarvestExpandedRecords in Data Aggregation, but allows quick, real time lookup by bibliographic identifier.

Parameters

- **id**: (type string; required) - list of system record identifiers
- **schema** (type enum; optional) - Defines the metadata schema in which the records are returned (ex: MARC21, MODS, Dublin Core). If not specified, a default will be defined by the API provider.

Returns

- A list of record objects that represents the bibliographic record, as well as associated holdings and item information for each requested bibliographic identifier (unless the specified return schema does not support it). If a hash is returned, the bibliographic identifier would be the key.
- If no records match any of the specified identifiers, return an empty list or RecordNotFound message.
- If no record matches a specific bibliographic identifier, return null or RecordNotFound message as the corresponding value for that bibliographic identifier.

Exceptional conditions

- **NotSupported**: The underlying system does not support this type of request.
- **UnsupportedSchema**: The requested metadata schema is not supported by the underlying system.
- **RecordNotFound**: No records were found when expected.

Side Effects

None

Rationale

For external discovery systems that do not maintain full MARC records in their local indexes, this function allows quick lookup of this information. Discovery applications could pass this detailed information to other tools like RefWorks or display the MARC record directly to users without handing them off to a separate application. This

function also allows external discovery applications to pull records from the ILS individually if needed to replace data in local indexes.

Notes

- Returned data should include item identifiers and bibliographic identifiers.
- Maximum number of records that can be requested per query may need to be locally defined.

Possible Bindings

- **SRU/W** - searchRetrieve operation
  - The record schema used must be able to handle more than just bibliographic information
  - The service must allow a request to search by bibliographic identifier via CQL
- **OAI/PMH** - GetRecord function
  - this function retrieves only 1 record with 1 record identifier at a time
- **Web Service Call**
  - Could be implemented as a web service directly on top of the ILS.
  - RESTful URL template: GET http://{service}/records?id=<identifier>+<identifier>+<identifier>... [&schema=<x>]

6.3.3 Search (Level 2)

Summary

Returns a list of records in the ILS matching a search query.

Parameters

- **query** (type string; required): Query string; needs to include both index and search terms.
- **profile** (type enum; required): Defines the type of query syntax being utilized. For example, in SRU/W, users might define the profile as CQL. The API provider would need to enumerate supported profile types.
- **schema** (type enum; optional): Defines the metadata schema in which the records will be returned (ex: MARC21, MODS, Dublin Core). If not specified, a default will be defined by the API provider.
- **recordType**: (type enum; optional): Specifies the type for the returned records [brief, expanded]. The API provider would need to enumerate supported and default record types.
- **offset**: (type int; optional): First record in a search result list to return.
- **max**: (type int: optional): Specifies the maximum number of records to be returned in the query response. The API provider will need to establish a default.
- **sort**: (type string; optional): Specifies a sort type (e.g. title, date, relevance, etc.)

Returns

- A list of record objects that match the query. The level of information included in those records is defined by the recordType parameter.
- When requested, the expanded record type should return records that include all available bibliographic, holdings, and item/circulation information (the same metadata returned for the GetRecords function).

- When requested, we recommend that the brief record type should return records that include title, author, imprint, isbn/issn, and bibliographic identifier.
- If no records match query, returns an empty list or a RecordNotFound message.

Exceptional conditions

- **NotSupported**: The underlying system does not support this type of request.
- **UnsupportedProfile**: The requested profile is not supported by the underlying system.
- **UnsupportedSchema**: The requested metadata schema is not supported by the underlying system.
- **UnsupportedType**: The requested record type is not supported by the underlying system.
- **UnsupportedQuery**: The query as requested (index, truncation, etc.) is not supported by the underlying system.
- **RecordNotFound**: No records were found.

Side Effects

None

Rationale

Some external discovery applications, metasearch for example, may want to obtain results from the ILS on the fly for integration, rather than maintaining a separate index of ILS data. Enabling machine-readable access to ILS data on the fly is crucial for sending catalog data out of the catalog. Some external discovery applications (particularly non-library applications) may be able to interact more easily with a real time search service that provides for a simple query mechanism. Searching and accessing course reserves materials is particularly relevant to integration with campus course management systems at institutions where reserves are managed via the ILS.

Notes

- Returned data should include item identifiers and bibliographic identifiers.
- Maximum number of records that can be requested per query may need to be locally defined.
- To properly support Unicode searching, the API provider should specify whether the indexed data uses composed or decomposed Unicode characters (provider may support search for both) and whether diacritics have been stripped in indexing process.
- Optimally, the API provider should perform the same transformation on the incoming query string as used when data is being indexed.
- In addition to more traditional indexes like title or author, support for instructor and course/section indexes would allow this function to retrieve records by their association with reserves. However, to obtain course reserves records that contain reserves information in addition to bibliographic information, the SearchCourseReserves function is recommended.

Possible Bindings

- **SRU/W** searchRetrieve operation
    - SRU: GET/POST http://{service}/sru?operation=searchRetrieve&version=1.2&query=title=science\[&maximumRecords=20&recordSchema=dc\]

    - Needs to support instructor/course indexes if ILS supports course reserve functionality
- **OpenSearch**
    - Could be implemented as a web service directly on top of the ILS.
- **Web Service call**
    - Could be implemented as a web service directly on top of the ILS.
    - RESTful URL template: GET http://{service}/search?query=<query>&profile=<x>[&schema=<x>&recordType=(expanded|brief)&sort=<x>&max=<x>&offset=<x>]

6.3.4 Scan (Level 2)

Summary
Given a query, returns the index entries (and number of matching bibliographic records) that are nearest to where the query would appear in the index.

Parameters
- **query** (type string; required): query string; needs to include both index and search terms
- **profile** (type enum; required): defines the type of query syntax being utilized. For example, in SRU/W, users would define profile as CQL. The API provider would need to enumerate supported profile types.
- **schema** (type enum; optional): Defines the metadata schema for the records to be returned (ex: SRU, <record>, RSS/Atom). If not specified, a default will be defined by the API provider.
- **position**: (type int; optional): The position within the list of returned index entries where the requested term(s) should occur. If position = 1, the requested term should be the first term in the list returned. The default value is 1.
- **max**: (type int: optional): Maximum number of index entries to be returned. If not specified, a default will be defined by the API provider.

Returns
A list of index entries with number of matching bibliographic records near the user's query term[s]. SRU/W has a pre-defined schema for a Scan response that can be extended through use of the <sru:extraTermData> element. See Appendix 6 for an example of how this might look.

Exceptional conditions
- **NotSupported**: The underlying system does not support this type of request.
- **UnsupportedQuery**: The query as requested (index, etc.) is not supported by the underlying system.
- **UnsupportedProfile**: The requested profile is not supported by the underlying system.
- **UnsupportedSchema**: The requested metadata schema is not supported by the underlying system.

Side Effects
None

Rationale
Sometimes it is desirable to browse a list of possible headings, rather than view a list of individual records. For example, browsing the shelf by call number is a popular feature that can help mimic the physical experience of browsing the shelves without going to the library. Scan can also help support direct lookup of a title or author to help determine for sure whether that title or author is part of a library's collection.

Notes
The API provider must specify which indexes are available to scan. We recommend that a minimum set of indexes would include title, author, subject, and call number. For the author and subject indexes, this recommendation assumes that the index has been enriched with entries that represent cross-references and see also headings from authority records.

To properly support Unicode searching, the API provider should specify whether the indexed data uses composed or decomposed Unicode characters (provider may support search for both) and whether diacritics have been stripped in indexing process.

Optimally, the API provider should perform the same transformation on the incoming query string as used when data is being indexed.

Possible Bindings
- **SRU/W** scan operation
  - SRU: GET/POST http://{service}/sru?operation=scan&version=1.2&scanClause=subject=deforestation&responsePosition=1&maximumTerms=25
  - A pre-defined SRU schema exists for this response. See http://www.loc.gov/standards/sru/specs/scan.html. In this scenario, one could use the <sru:extraTermData> element combined with MADS elements for each term returned to list an identifier for the authority, and to enable information like 'see also' or 'see related headings'. For example, the <mads:related> element would be included if an index entry represents an authorized subject heading with several 'see also' headings. The <mads:authority> element would be included if an index entry represents an un-authorized 'see' heading that has been included in the index. In that scenario, the <mads:authority> element would recommend the authorized heading that should be used in place of the index entry.
- **OpenSearch**
- **Web Service Call**
  - Could be implemented as a web service directly on top of the ILS.
  - RESTful URL template: GET http://{service}/scan?query=<query>&profile=<x>[&schema=<x>&max=<x>&position=<x>]

6.3.5 GetAuthorityRecords (Level 2)

Summary
Given a list of authority record identifiers, returns a list of record objects that contain the authority records. The function user may request a specific metadata schema for

the record objects. Individual API providers will need to enumerate the schemas supported in that system.

Parameters

- **id** (type [string]; required) - list of authority record identifiers
- **schema** (type enum; optional) - Specifies the metadata schema of records to be returned (ex: MARC21, MARCXML, MADS, Dublin Core). If not specified, a default will be defined by the API provider.

Returns

- A list/hash of record objects that represents the full authority record for each requested authority record identifier. If using a hash, the authority record identifier would the key.
- If no records match any of the specified identifiers, return an empty list or RecordNotFound message.
- If no record matches a specific authority record identifier, return null or RecordNotFound message as the corresponding value for that identifier.

Exceptional conditions

- **NotSupported**: The underlying system does not support this type of request.
- **UnsupportedSchema**: The requested metadata schema is not supported by the underlying system.
- **RecordNotFound**: No records were found for this query.

Side Effects

None

Rationale

Some external discovery tools are built using authority records. This function allows these systems to retrieve a full authority record on-the-fly, even if the full record is not stored locally. It also allows locally cached records to be updated on an individual basis.

Notes

- Returned data should include authority record identifiers.

Possible Bindings

- **SRU/W** - searchRetrieve operation
  - must be able to make a request to search by authority record identifier via CQL
- **Web Service Call**
  - Could be implemented as a web service directly on top of the ILS.
  - RESTful URL template: GET http://{service}/authorities?id=<identifier>+<identifier>+<identifier>...[&schema=<x>]

6.3.6 SearchCourseReserves (Level 4)

Summary

Retrieves course reserve records by specifying instructor and/or course and section. API providers must enumerate the types of course reserve searches supported.

Parameters

- **query** (type string; required) - Query string; needs to include both index and search terms
- **profile** (type enum; required) - Defines the type of query syntax being utilized. For example, in SRU/W, users would define profile as CQL.
- **recordType**: (type enum; optional) - Specifies the type for the returned records. For example, brief or expanded. The API provider would need to enumerate supported record types.
- **schema** (type enum; optional) - Specifies the metadata schema of records to be returned (ex: Atom, RSS, Dublin Core, MARCXML). If not specified, a default will be defined by the API provider.

Returns

A list of course reserve record objects that matched the user's query. The level of information included in those records is defined by the recordType parameter.

- When requested, the expanded record type should return objects that include all available information about course, section, and instructor, as well as a list of bibliographic records with associated holdings/item information that are associated with those courses.
- When requested, we recommend that the brief record type should return records that include all available information about course, section, and instructor, and a summary of the number of items on reserve for that course/section/instructor.
- If no records match query, returns an empty list or a RecordNotFound message.

Exceptional conditions

- **NotSupported**: The underlying system does not support this type of request.
- **UnsupportedProfile**: The requested profile is not supported by the underlying system.
- **UnsupportedQuery**: The requested search type is not supported by the underlying system.
- **UnsupportedType**: The requested record type is not supported by the underlying system.
- **UnsupportedSchema**: The requested metadata schema is not supported by the underlying system.
- **RecordNotFound**: No records could be found matching the request.

Side Effects

None

Rationale

Searching and accessing course reserve materials is particularly relevant to integration with campus course management systems. While not all course reserve systems are managed through the ILS, this section of the API is relevant for those institutions where it is. It may be possible to utilize course and/or instructor querying via the Search function defined previously, but the records returned in that case will simply be a list of bibliographic records. This function provides the ability to retrieve course and instructor information in combination with bibliographic records.

Notes

- It would be useful to be able to search by department, course, section, and instructor in combination (combining this with keyword searching would allow even greater flexibility).
- Course level records should return a system identifier for a course.

Bindings
- **OpenSearch**
- **SRU/W** searchRetrieve operation
    - Only if course/instructor indexes are searchable in SRU/W implementation. Also need to define the XML response to come back.
    - http://{service}/sru?operation=searchRetrieve&version=1.2&query=course=eng*\[&recordSchema=dc\]
- **REST Web Service Call**
    - http://{service}/reserves?query=<query>&profile=<x>[&schema=<schema>&recordType=(brief|expanded)]
    - Specific course: http://{service}/reserves?course=eng201&profile=CQL&schema=dc&recordType=brief
    - More RESTful access to a specific course (not using a query language): http://{service}/reserves/course/eng201?schema=dc&recordType=brief

6.3.7. Explain (Level 4)

Summary
Explain, in machine-readable format, the real time search functionality that is supported by a given ILS/metadata provider.

Parameters
None

Returns
Summary of real time search functionality supported. This should describe the functions, metadata schemes, character encoding, indexes, and other default parameter values.

Exceptional conditions

- **NotSupported:** The underlying system does not support this type of request.

Side Effects
None

Rationale
Since some API providers may only implement some of the functionality described, this is a standard way of identifying which parts of the functionality are available.

Bindings
- **SRU/W** explain operation
- **OpenSearch**
    - May be able to provide a simplified version of this via an OpenSearch description document.

    - http://www.opensearch.org/Specifications/OpenSearch/1.1#OpenSearch_description_document
- **REST Web Service Call**
    - http://{service}/explain/

6.4 Binding Details

- **SRU/W**: http://www.loc.gov/standards/sru/
- **OpenSearch**: http://a9.com/-/company/opensearch.jsp

6.4.1 Metadata Schemas

In the abstract functions specified for the Real Time Search section, there are several different types of metadata that might be transmitted between the ILS and the requesting application. Where possible, it seems most efficient and re-usable to take advantage of pre-existing defined schemas to represent this data. The members of this task group are making recommendations for possible schemas that could describe the desired information, however, we do not want to limit API implementers and users to a single metadata schema. In some cases, the method of implementation (ex: SRU vs. OAI-PMH) may constrain the data schemas that can be used.

In general, we need to be able to describe bibliographic information, structured MARC holdings information, item information, availability information, authority information, and course reserve information in a standardized format. We must also consider what metadata should be included in the response for a Scan request.

Bibliographic information
Libraries have been exchanging bibliographic information for a long time, so there are many available schemas. Full bibliographic metadata should be returned in the GetRecords and Search (when full record type is specified) functions for Real Time Search. We strongly encourage API implementers to provide access to bibliographic information in an XML format, in addition to (or instead of) the traditional marc21 (z39.2/ISO 2709). Other recommended schemas include:

- **MARCXML**: http://www.loc.gov/standards/marcxml/
    - schema: http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd
- **MODS**: http://www.loc.gov/standards/mods/
    - schema: http://www.loc.gov/standards/mods//v3/mods-3-3.xsd
- **Dublin Core**: http://dublincore.org/
    - schema: http://dublincore.org/schemas/xmls/simpledc20021212.xsd

Structured Holdings information
MARC Holdings information has been less frequently included (and in less standardized ways) through common data interchange protocols like z39.50. While there is a stable marc21 holdings format, other XML options for fully describing holdings information are still being standardized. Full holdings information should be returned in the GetRecords and Search (when expanded record type is specified) functions for Real Time Search. Other recommended schemas include:

- **MARCXML**: http://www.loc.gov/standards/marcxml/
    - schema: http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd

- **ISO holdings** (ISO 20775): http://www.loc.gov/standards/iso20775/
  - For our examples, we used the schema at http://www.loc.gov/standards/isohold/N130_ISOholdings_v6_1.xsd
- **MODS**: http://www.loc.gov/standards/mods/v3/mods-holdings.html
  - Since MODS only allows for structured enumeration and chronology by bringing in information specified with an external schema, we recommend use of MARCXML or ISO Holdings rather than MODS where possible.

Item information
In addition to structured MARC holdings, bibliographic records are also usually associated with item information describing physical items. Most ILS's maintain this valuable information in an un-standardized format not associated with traditional MARC formats. Very few standardized schemas exist that can fully describe item information and it has not been historically exchanged between libraries. Full item and circulation information should be returned for the GetRecords and Search functions (when expanded record type is specified), as well as for the HarvestExpandedRecords function specified in section 5.3.2. This information could also be returned for GetAvailability queries. Recommended schemas include:

- **ISO holdings** (ISO 20775): http://www.loc.gov/standards/iso20775/
  - For our examples, we used the schema at http://www.loc.gov/standards/isohold/N130_ISOholdings_v6_1.xsd
- **NCIP** Item element (see the Item Element Type in section 6.3): http://www.niso.org/kst/reports/standards?step=2&gid=None&project_key=88add52c35a51f8acb31225a533599dfc679f913
    - xsd schema: http://ncip.envisionware.com/documentation/ncip_v1_0.xsd
    - dtd schema: http://ncip.envisionware.com/documentation/ncip_v1_0.dtd

Availability information
Availability information is a specific subset of item information that describes the current circulation status of a physical item. While both NCIP and ISO Holdings schemas can express most desired availability information, these standards are too complex for a quick implementation of Level 1 GetAvailability functionality. As an alternative, we define a simple availability schema that is used to describe the required data elements for a GetAvailability response. This schema is required for full Basic Discovery Interface support and can also be returned for functions such as GetRecords, Search, and HarvestExpandedRecords, preferably in addition to a more detailed item element.

- <**dlf:simpleavailability**> - used to enclose basic information about availability. Depending on the context where it is used, it might describe availability at the level of a particular physical item or at the level of the bibliographic record.
  - <**dlf:identifier**> - encloses the identifier of the resource described. (required, minOccurs=1, maxOccurs=1)
  - <**dlf:availabilitystatus**>: encloses one of the following string values: "unknown", "available", "not available", or "possibly available". These correspond to status codes in the ISO Holdings schema. For errors (such as an unknown identifier or an unreachable database), "unknown" may be used. For resources that may only be partially available (as can occur for multi-item resources) or whose availability

may be conditional on the status of the user, "possibly available" may be used. (required, minOccurs=1, maxOccurs=1)

- <**dlf:availabilitymsg**>: Encloses a human-readable message with more information on the item's availability, such as instructions for obtaining the item, reasons that the item is not available, or explanation of errors or unclear availability status. (optional)
- <**dlf:location**>: Encloses a human-readable message with more information on the item's location. (optional)
- <**dlf:dateavailable**>: For items expected to be available at a future date (such as the due date), this contains the date or the date and time at which it is expected to be available. This element does not need to be included when describing availability at the bibliographic record level where multiple items might have different due dates. Date and time formats should correspond to the W3C profile of ISO 8601 given at http://www.w3.org/TR/NOTE-datetime. Examples of valid strings in this profile are "2008-04-21" and "2008-04-21T13:00-05:00". (required when applicable and known, minOccurs=0, maxOccurs=1)

Scan information
The Scan function represents a different type of search functionality than provided by the other Real Time Search functions, in that it returns index entries, rather than full records. We strongly encourage API implementers to enrich these index entries with related or authorized forms of headings when users are scanning controlled indexes, such as author and subject. One schema that we recommend is the SRU/W Scan record format for returning index entries, with additional term data added to use recording using MADS or MARCXML.

Appendix 6 shows an example of what such a record might look like.

Authority information
Full authority records should be returned in the GetAuthorityRecords function for Real Time Search. Like bibliographic and holdings information, authority information can be expressed using the marc21 authority format (http://www.loc.gov/marc/authority/). However, we strongly encourage API implementers to make the metadata available in an XML format. Other recommended schemas include:

- **MARCXML**: http://www.loc.gov/standards/marcxml/
  - schema: http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd
- **MADS**: http://www.loc.gov/standards/mads/
  - Schema: http://www.loc.gov/standards/mads/mads.xsd

Course reserve information
As some institutions maintain course reserve records within the ILS, there is a need to access these records as well. While individual items on reserve could be represented with standard bibliographic / item information, course reserves also involve information about courses and instructors. Course reserve records should be returned in the SearchCourseReserves function for Real Time Search.

The members of this task group have not discovered a previously existing, openly available metadata schema that will handle course reserve information. We

encourage the development and promulgation of such schemas.

A data schema used to describe course reserve information could be hierarchical in nature, based at the course at the root level, and would need to take the following metadata into account.

Course

- Code (ex: eng221)
- Department
- Name (Introduction to Biology)
- Section (may be multiple Section elements per course)
  - Number (ex: 001)
  - Instructor
  - Time Period (ex: Spring Session 2007-2008)
  - Number of reserve materials
- Record - if recordType=expanded, include 1 or more record elements that represent expanded bibliographic records associated with a section

6.4.2 Putting it all together

Since no single metadata schema recommended above specifies all the information we would like to see made available for the GetRecords and Search functions (as well as HarvestExpandedRecords in the Data Aggregation section), a technique is needed to combine data elements from multiple schemas. We propose the use of very simple XML elements to help bring these schemas together, nested according to the indentation shown below. This schema can also be used in the GetAvailability response.

- <**dlf:collection**> - use to enclose a group of records
  - <**dlf:record**> - use to enclose an expanded record that may include bibliographic, marc holdings, and item/circulation information
    - <**dlf:bibliographic**> - use to enclose the bibliographic description in the record. This element should include an "id" attribute that contains the bibliographic identifier. In the response to functions that do not require bibliographic descriptions to be returned, such as GetAvailability, the element may be empty (but should still include the id attribute). (required, minOccurs=1, maxOccurs=1)
    - <**dlf:simpleavailability**> - use to enclose basic information about availability. At this level, it describes the availability of the bibliographic resource in general. The same element can also be used under dlf:item (see below) to describe the availability of a particular accessible item within this bibliographic resource. The structure of this element is described in section 6.4.1 above.
    - <**dlf:holdings**> - use to enclose structured holdings information described with schemas such as MARC holdings and ISO Holdings. (optional, minOccurs=0. maxOccurs=1)
      - <**dlf:holdingset**> - use to represent a set of holdings, that represent the entire record or a specific set of items within that record. If there are multiple holdings records that refer to different sets of items, they should appear

in separate dlf:holdingset elements. (required, minOccurs=1)
        - <**dlf:holdingsrec**> - use to enclose a representation of a specific holdings record for a particular set of items. (required, minOccurs=1). If ISO Holdings schema is used, this will contain a <holdingsSimple> element to describe individual items or a <holdingsStructured> element to describe marc holdings information for continuing resources. If MARC holdings is used, this element will contain an appropriate MARC xml representation, such as marc21.
        - <**dlf:items**> - use when a structured holdings record is associated with a particular set of items. When used in a dlf:holdingset element, contains a list of empty dlf:item elements, each with an id attribute that contains the item id of an item associated with this particular holding. These items are described in detail in an element within the dlf:holdingsrec (such as holdingsSimple) or in a dlf:items element outside of the dlf:holdings element (optional, minOccurs=0)
- <**dlf:items**> - use to enclose metadata for individual physical items associated with the bibliographic record. No more than one dlf:item element may be included per item in this element. (optional in general, but required by some functions minOccurs=0, maxOccurs=1)
    - <**dlf:item**> - use to declare and/or describe individual physical items. Must include an id attribute with the item id. It may also contain metadata about the item. (Some functions, like GetAvailability, require availability metadata to be returned here, for example).. The dlf:item element may contain multiple elements describing the item in different formats. Types of elements used for descriptions include holdings:copyInformation (defined for ISO Holdings), ncip:LookupItemResponse (defined for NCIP), or dlf:simpleavailability (defined above).

Some examples of how a response including expanded records might look are given in Appendix 2 (MARCXML bibliographic record + NCIP item information) and Appendix 3 (MODS bibliographic record + ISO Holdings item information).

The XML schema definition for the elements described in this section can be found at http://diglib.org/architectures/ilsdi/schemas/1.1/dlfexpanded.xsd

7. Patron functionality

Discovery applications intended as alternatives to the OPAC will need to allow patrons to access their account information in the OPAC as well as provide the circulation and delivery functionality available in the OPAC. In this section, we will define standard functions for interacting with ILS's circulation/delivery mechanisms and for accessing patron account information.

7.1 Rationale and general issues

As an increasing number of institutions implement external discovery applications and move away from reliance on the integrated OPAC, they are remembering that patrons use the OPAC for more than just discovery - they also use it to manage their account and request delivery of discovered materials. In order for discovery applications to be used as OPAC alternatives, they need to be able to provide the same level of discovery to delivery functionality that the out-of-the-box OPAC provides. In addition, it is becoming increasingly desirable to enable patrons to access account information outside of the library web site in the online environments where they spend time: places like the campus portal, learning management systems, or even Facebook. Discovery applications will need to interact with the ILS in order to perform these functions. There are three categories of these functions that are specified in this section: patron authentication, patron account retrieval, and circulation/delivery transactions.

Patron authentication: Discovery applications that make use of ILS patron information or allow a patron to perform some circulation activity will first need to authenticate with the ILS. This is necessary to obtain the ILS patron identifier that will be required to perform these patron account and circulation functions. There are a few different use cases for obtaining the patron identifier from the ILS. For instance, the discovery application may utilize the ILS as the authoritative source of user information, and so may pass user credentials to the ILS, so that the ILS can verify the credentials, and provide a persistent patron id back to the discovery application. Or, the discovery application may use a different authoritative source of user data, such as a campus directory, and may need the ILS to resolve the directory id into the patron's ILS id. Or, the discovery application may need to verify that a patron has an account in the ILS.

Patron account retrieval: Discovery applications may need to retrieve information about a patron, after that patron has been authenticated in the ILS. For instance, the discovery application may show a patron their library fines, current loans and status of hold requests.

Circulation/delivery: The ILS can be configured with complex circulation logic and policies that affect how an ILS handles these types of requests. For instance, the ILS may be able to determine what the 'best' copy of a title is when a patron places a hold request. Or the ILS may determine that an item cannot be picked up at a certain location due to paging policies. Or a consortial ILS may be able to determine from which location to place a hold request when a title is held by multiple consortial members. To be clear, discovery applications should not try to replicate this logic. Rather, discovery applications should request the ILS to perform these functions for them. These functions are already implemented in the OPAC interface of the ILS. The ILS should also expose the same logic to discovery applications via API requests.

The functions that are being discussed in this section may alter data in the ILS. In addition, these functions will be performed with respect to individual patrons in the ILS. It is therefore very important that security and patron privacy are taken into consideration when implementing these functions.

7.2 Abstract Functions

7.2.1 LookupPatron (Level 3)

Summary
Looks up a patron in the ILS by an identifier, and returns the ILS identifier for that patron, aka the patron identifier.

Parameters
- **id** (type string; required): an identifier used to look up the patron in the ILS
- **id_type** (type string; optional): the type of the identifier, as defined in the ILS; e.g. Barcode, local_id

Returns
The ILS patron identifier that matches the lookup query is returned.
If no match is found, a PatronNotFound message is returned.

Exceptional conditions

- **NotSupported**: The function is not supported by the ILS.
- **PatronNotFound**: The requested patron could not be identified.

Side Effects
None

Rationale
Discovery applications will need to know the patron's primary ILS identifier in order to perform other patron functions.

Notes
- The id types will be defined by the ILS and may vary between ILS vendors and also between ILS installations. Therefore, there should be no constraints placed on the id_type field.
- The ILS may choose whether or not to use id_type based on how ids are stored in the ILS.

Possible Bindings
- **NCIP**: Lookup User Service
  - must use either Unique User Id or Visible User Id
- **SIP2**: there is no binding for this function in the SIP2 protocol

7.2.2 AuthenticatePatron (Level 3)

Summary

Authenticates a user's login credentials and returns the identifier for the patron.

Parameters
- **username** (type string; required)
- **password** (type string; required)

Returns
- The ILS patron identifier that matches the patron credentials. If the username is not found or the password does not match, PatronNotFound message is returned.

Exceptional conditions
- **NotSupported** - the function is not supported by the ILS
- **NotAuthorized** - the client attempting to use this function is not authorized by the ILS. Note that this condition refers to authorization of the client application, not the end user.
- **PatronNotFound**: The requested patron could not be identified.

Side Effects
No direct required side effects, though some ILS's may consider the application to be authorized after this function is invoked.

Rationale
The discovery application may use the ILS as its authoritative source of user data.

Possible Bindings
- **NCIP**: Authenticate User service
- **SIP2**: Login command

7.2.3 GetPatronInfo (Level 3)

Summary
Returns specified information about the patron, based on options in the request. This function can optionally return patron's contact information, fine information, hold request information, loan information, and messages.

Parameters
- **patronId** (type string; required): the unique patron identifier in the ILS; the same identifier returned by LookupPatron or AuthenticatePatron
- **showContact** (type boolean; default true; optional): whether or not to return patron's contact information in the response
- **showFines** (type boolean; default false; optional): whether or not to return fine information in the response
- **showHolds** (type boolean; default false; optional): whether or not to return hold request information in the response
- **showLoans** (type boolean; default false; optional): whether or not to return loan information in the response
- **showRecalls** (type boolean; default false; optional): whether or not to return recall information in the response
- **showMessages** (type boolean; default false; optional): whether or not to return patron messages in the response

Returns
- Requested information is returned about the patron
- If no match is found on the patronId, return PatronNotFound message
- Response should be well formatted to include identifiers for holds and loans.

Exceptional conditions
- **NotSupported** - This function is not supported by the ILS.
- **NotAuthorized** - The client attempting to use this function is not authorized by the ILS.
- **PatronNotFound**: The requested patron could not be identified.

Side Effects
None

Rationale
Discovery application may need to display user's account information.

Notes
- Response should be well structured so that discovery application can retrieve hold and loan identifiers from the response, in order to perform RenewLoan and CancelHold functions.

Possible Bindings
- **NCIP**: Lookup User Service partially implements this function
  - Has the ability to return fines, holds and loans.
  - Does not have the ability to return messages or recalls.
- **SIP2**: Patron Information command partially implements this function
  - Has the ability to return fines, holds, loans, and recalls.
  - Does not have the ability to return messages.

7.2.4 GetPatronStatus (Level 3)

Summary
Returns a patron's status information from the ILS.

Parameters
- **patronId** (type string; required): the unique patron identifier in the ILS; the same identifier returned by LookupPatron or AuthenticatePatron

Returns
- Returns a message with patron's borrower type, status, and expiration information.
- If no match found, returns PatronNotFound message

Exceptional conditions
- **NotSupported** - the function is not supported by the ILS
- **NotAuthorized** - the client attempting to use this function is not authorized by the ILS
- **PatronNotFound**: The requested patron could not be identified.

Side Effects
None

Rationale
Discovery applications may need to know patron's status in order to make authorization decisions.

Notes
- The response may be a subset of the response for the GetPatronInfo function.

Possible Bindings
- **NCIP** - Lookup User Service
- **SIP2** - Patron Status Request command

7.2.5 GetServices (Level 3)

Summary
Returns information about the services available on a particular item for a particular patron.

Parameters
- **patronId:** (type string; required): the unique patron identifier in the ILS; the same identifier returned by LookupPatron or AuthenticatePatron
- **itemId:** (type string; required): system item identifier

Returns
- An enumerated list of services available for a particular patron and a particular item.

Exceptional conditions
- **NotSupported** - the function is not supported by the ILS
- **NotAuthorized** - the client attempting to use this function is not authorized by the ILS
- **PatronNotFound**: The requested patron could not be identified.
- **RecordNotFound**: The requested item could not be identified.

Side Effects
This function has no side effects.

Rationale
Which services are currently available to a patron at a particular moment in time for a particular item is a complex calculation made by the ILS for every single transaction. This function calculates whether the patron has the ability to perform certain services (for example, a hold, recall, request, or check out), which may be depend on whether the patron's account is blocked (for unpaid fines or other reasons). Service availability may also depend on whether the patron's status in the system (for example, grad student, faculty, undergraduate) has the ability to perform the requested service for the requested item. (For example, circulation policies may be calculated based on patron and item type.)

Returns
The response should include multiple <availableFor> elements to describe the services available to the patron. Examples of services it might be useful to identify

are: loan, request, recall, in building use, reproduce (some special collections materials can't be photocopied).

- Both NCIP and ISO Holdings (ISO 20775) specify a possible list of enumerated values that could serve as the starting point for a vocabulary of available services. The existing values are listed below.
  - **ISO Holdings** (http://www.loc.gov/standards/iso20775/)
    - loan
    - physical copy
    - digital copy
    - online access
    - reference look-up
    - other
    - unspecified
  - **NCIP** http://www.niso.org/kst/reports/standards?step=2&gid=None&project_key=88add52c35a51f8acb31225a533599dfc679f913
    - Available For Supply Without Return
    - In Library Use Only
    - Limited Circulation, Long Loan Period
    - Limited Circulation, Normal Loan Period
    - Limited Circulation, Short Loan Period
    - No Reproduction
    - Not For Loan
    - Overnight Only
    - Renewals Not Permitted
    - Supervision Required
    - Term Loan
    - Use Only In Controlled Access
    - User Signature Required

Possible Bindings

- This functionality is not specified by NCIP or SIP2.
- **Web service call** - this function could be implemented as an XML based web service on top of the ILS.
  - RESTful URL template: GET http://{service}/patron/<patronid>/item/<itemid>/services

7.2.6 RenewLoan (Level 3)

Summary

Extends the due date for a patron's existing loan.

Parameters

- **patronId**: (type string; required): the unique patron identifier in the ILS; the same identifier returned by LookupPatron or AuthenticatePatron
- **itemId**: (type string; required): system item identifier
- **desired due date** (type date, optional): the date the patron would like the item returned by

Returns

New loan message with the new due date.
If no items match the item identifier, returns RecordNotFound message.

Exceptional conditions
- **NotRenewable** - the item is not able to be renewed by the ILS
- **NotSupported** - the function is not supported by the ILS
- **NotAuthorized** - the client attempting to use this function is not authorized by the ILS
- **PatronNotFound**: The requested patron could not be identified.
- **RecordNotFound**: The requested item could not be identified.

Side Effects
Date and loan are both updated.

Rationale
Patrons will want to renew their loans through their accustomed library interface.

Notes
- In discovery applications that do not support patron and circulation functions (and thus are not OPAC alternatives), it may still be useful to hand off control to the underlying OPAC to process renewals. Such application handoff is described in section 8.

Possible Bindings
- NCIP - Renew Item Service
- SIP2 - Renew command

7.2.7 HoldTitle (Level 3)

Summary
Creates, for a patron, a title-level hold request on a given bibliographic record in the ILS.

Parameters
- **patronId** (type string; required): the ILS identifier for the patron for whom the request is placed
- **bibId** (type string; required): the ILS identifier for the bibliographic record on which the request is placed
- **requestLocation** (type string; required): IP address where the end user request is being placed
- **pickupLocation** (type string; optional): an identifier indicating the location to which to deliver the item for pickup
- **neededBeforeDate** (type date or time; optional): date after which hold request is no longer needed
- **pickupExpiryDate** (type date or time; optional): date after which item returned to shelf if item is not picked up

Returns
- If a hold request is successfully placed, a response message indicates where the item may be picked up and the date it expects the item to be available
- A restricted access message - qualified success message that title is not holdable, but is accessible under certain conditions (i.e. does not circulate)

- Title level hold not available message - message indicating the this title is not requestable, but item level hold requests may be available
- NotHoldable message - error message that title is not holdable by this patron

Exceptional conditions
- **NotSupported**: the function is not supported by the ILS
- **NotAuthorized**: the client attempting to use this function is not authorized by the ILS
- **NotHoldable** - the title cannot be held by the patron

Side Effects
A hold request is placed on a title in the ILS.

Rationale
Patrons will want to place hold requests on the titles that they discover. The discovery application should not have to replicate the functionality that the ILS provides in terms of determining best item. This function allows discovery application to initiate a hold request with a patron id and bib id. A title-level hold request is a hold request that is placed on the bibliographic record, rather than on an individual item record, in the ILS.

Request location is required in order to allow ILS to determine pickup locations and apply paging rules.

Possible Bindings
- **NCIP** - Request Item Service (though it needs to allow for specifying pickup location)
- **SIP2** - Hold command
- **Web service call** - this function could be implemented as an XML based web service on top of the ILS

7.2.8 HoldItem (Level 3)

Summary
Creates, for a patron, an item-level hold request on a specific item of a bibliographic record in the ILS.

Parameters
- **patronId** (type string; required): the ILS identifier for the patron for whom the request is placed
- **bibId** (type string; required): the ILS identifier for the bibliographic record on which the request is placed
- **itemId** (type string; required): the ILS identifier for the specific item on which the request is placed
- **pickupLocation** (type string; optional): an identifier indicating the location to which to deliver the item for pickup
- **neededBeforeDate** (type date or time; optional): date after which hold request is no longer needed
- **pickup expiry date** (type date or time; optional): date after which item returned to shelf if item is not picked up

Returns

- If a hold request is successfully placed, a response message indicates where the item may be picked up and the date it expects the item to be available
- A restricted access message - qualified success message that title is not holdable, but is accessible under certain conditions (i.e. does not circulate)
- NotHoldable message - error message that title is not holdable by this patron

Exceptional conditions
- **NotSupported** - The ILS does not support this type of hold request.
- **NotHoldable** - the item cannot be held by the patron

Side Effects
Hold request is placed on an item in the ILS.

Rationale
Patrons will want to place hold requests on the items that they discover This function allows the user to determine which item is the best option to request.

An item-level hold request is a hold request that is placed on a specific item, rather than on the bibliographic record, in the ILS.

Possible Bindings
- **NCIP** - Request Item Service (though it needs to allow for specifying pickup location)
- **SIP2** - Hold command
- **Web service call** - this function could be implemented as an XML based web service on top of the ILS

7.2.9 CancelHold (Level 3)

Summary
Cancels an active hold request for the patron.

Parameters
- **patronId** (type string; required): the unique patron identifier in the ILS; the same identifier returned by LookupPatron or AuthenticatePatron
- **itemId**: (type string; required): system item identifier

Returns
Confirmation message indicating success or failure.
If no patron records match the patron identifier, returns PatronNotFound message.
If no items match the item identifier, returns RecordNotFound message.

- **NotSupported**: the function is not supported by the ILS
- **NotAuthorized**: the client attempting to use this function is not authorized by the ILS
- **NotCanceled**: the hold could not be canceled by the ILS
- **PatronNotFound**: The requested patron could not be identified.
- **RecordNotFound**: The requested item could not be identified.

Side Effects
Hold request is canceled.

Rationale
Patrons will want to cancel holds as well as make them.

Possible Bindings
- **NCIP** - Cancel Request Item Service
- **SIP2** - Hold command

7.2.10 RecallItem (Level 3)

Summary
Initiates a recall of an item that is loaned out.

Parameters
- **itemId** (type string; required): the ILS identifier for the specific item on which the recall is placed
- **desiredDueDate** (type date or time; optional)

Returns
- If recall is successfully placed on the item, a message will be returned indicating an estimated delivery date
- If not eligible for recall, an error message is returned to this effect.

Exceptional conditions
- **NotSupported** - the function is not supported by the ILS
- **NotAuthorized** - the client attempting to use this function is not authorized by the ILS

Rationale
If patrons discover items they want that are on loan, they may want to recall them.

Possible Bindings
- **NCIP** - Recall Item Service
- **SIP2** - there is no binding for this function in the SIP2 protocol

7.2.11 CancelRecall (Level 3)

Summary
This function cancels an active recall request for the patron.

Parameters
- **patronId** (type string; required): the unique patron identifier in the ILS; the same identifier returned by LookupPatron or AuthenticatePatron
- **itemId**: (type string; required): system item identifier

Returns
Confirmation message indicating success or failure of the cancellation.
If no patron records match the patron identifier, returns PatronNotFound message
If no items match the item identifier, returns RecordNotFound message.

Exceptional conditions
- **NotSupported** - the function is not supported by the ILS

- **NotAuthorized** - the client attempting to use this function is not authorized by the ILS
- **NotCanceled** - the item was not able to be canceled by the ILS
- **PatronNotFound**: The requested patron could not be identified.
- **RecordNotFound**: The requested item could not be identified.

Side Effects
Recall is canceled

Rationale
Patrons may need to cancel as well as make recalls.

Possible Bindings
- **NCIP** - Cancel Recall Item Service
- **SIP2** - there is no binding for this function in the SIP2 protocol

7.3 Binding Details

- **NCIP** - http://www.niso.org/committees/committee_at.html
- **SIP2** - http://multimedia.mmm.com/mws/mediawebserver.dyn?6666660Zjcf6lVs6EVs66S0LeCOrrrrQ-

8. OPAC interaction

In this section, we define standard functions and behaviors for interactions between discovery applications and the traditional ILS OPAC. These include functionality for transferring the user from the discovery application to an appropriate context in the OPAC for further requests or information on discovered content. They also include functionality for transfers in the other direction, from the OPAC to discovery applications, and for embedding external information within an OPAC application.

8.1 Rationale and general issues

The ILS today includes a public catalog interface (the OPAC) that will continue to be provided and used in many libraries. Users of the OPAC may want to access the additional information and services provided by external applications, either jumping out from the OPAC to these applications, or by embedding application content into OPAC displays, to provide seamless discovery services.

In addition, the OPAC may continue to be the easiest place to provide certain kinds of library services (such as patron requests). In such cases, one might need to be able to carry a user context from an external application into the OPAC, and later back to the external application.

8.2 Sample use cases

Some possible use cases include

- Adding links to external resources from within the OPAC (e.g. "See comments on this book; buy on Amazon; try this search again on WorldCat")
- Embedding additional information on items in the OPAC (e.g. tags, LibraryThing-like services)
- Embedding alternate formats for reuse (e.g. formatted citations, links to download cites into citation manager, etc.)
- Linking back to an OPAC's book information page, request page, or other results page from an external application.

8.3 Abstract Behaviors and Functions

Practically speaking, the ability to embed external information and links into an OPAC application may be better understood as a behavior of an ILS OPAC than a function, and the recommended bindings reflect this understanding. (That is to say, they give a general mechanism for embedding content and links into a user interface, rather than specifying a particular output for a particular input.) From an abstract point of view, however, it can be useful to consider two kinds of functions that model these behaviors and take different kinds of parameters.

An incoming-oriented function that models how ILS applications can be linked to or embedded from external applications is essentially a behavior that takes request parameters, specifying the ILS function or display that is being linked to or embedded. Essentially, this kind of function points into the ILS from an external application.

An outgoing-oriented function that models how external content or links are embedded into a page is essentially a behavior that takes output configuration parameters, which control the final form of output. This could consist of the content to be embedded, for instance, or a second function that transforms the default content into the desired content. Essentially, this kind of function points out of the ILS into an external application.

The Level 1 (Basic Discovery Interfaces) profile requires that inbound links be possible to any item in the OPAC for further user requests.

The Level 2 (Elementary OPAC Supplement) profile requires that either OutputRewritablePage or OutputIntermediateFormat be implemented. The Level 4 profile calls for both of those two behaviors.

8.3.1 GoToBibliographicRequestPage (Level 1)

Summary
Outputs an OPAC display page for the bibliographic item with the supplied identifier, with links for making requests on that item.

Parameter
- **bibid** (type string; required): the ILS identifier for the desired bibliographic item

Returns
A web page with information about the identified item, with links for patron requests on it.

Exceptional condition
- **RecordNotFound**: No record was found for the specified ID.

Side Effects
None

Rationale
This function allows the OPAC to supply information and services for an item found in a discovery application. It is a simple, easy-to-implement way of providing patron services to users when it is not practical to do it directly in the discovery application.

Possible Bindings

URL template (recommended binding for level 1): All that is really necessary to implement this function is a stable URL that, given a bibliographic ID, will land the user on a bibliographic information page for the title with that ID. (The URL would also vary depending on the location of the OPAC.) Note that, in order for this to work, the URL cannot require any transient information (such as a session ID) and must work for all unsuppressed bibliographic records in the system. ISBNs, LCCNs, and other identifiers that may exist for some records but not others, are not appropriate here. For compatibility with other functions, it needs to use the same bibliographic identifiers returned in functions like HarvestBibliographicRecords.

In order for clients to use this function, ILS vendors should specify in some standard form the pattern to use for URLs of this type in their system, with parameters for the OPAC service and the bibliographic ID. Machine-readable patterns could be particularly useful, but not required. One possibly relevant machine-readable pattern is the OpenSearch URL template, which uses braces to define parameters. We recommend that templates use {service} to denote the particular ILS service being linked to (which will vary by ILS instance, and consist of a domain name and possibly a path prefix), and {bibid} to denote the bibliographic identifier for the item being linked to. For example, one ILS might declare the pattern

{service}/cgi-bin/Pwebrecon.cgi?DB=local&CMD=id"{bibid}"

and another ILS might simply use

{service}/{bibid}

Given a service parameter of "http://catalog.example.com" and a bibid parameter of "12309906" these would imply URL values of

http://catalog.example.com/cgi-bin/Pwebrecon.cgi?DB=local&CMD=id"12309906"

and

http://catalog.example.com/12309906

respectively.

OpenURL: An OpenURL could also be used for this functionality, with the idea that the same OpenURL structure could be used not only for links to a generic OPAC page, but also to more directly request specific patron information and services, by supplying appropriate service parameters. (For example, an OpenURL could specify that a particular user wanted to renew or hold a particular book, which could allow for a more targeted transition to an OPAC request page than a generic URL that simply lands on the book's information page.) An OpenURL that is in a form compatible with HTTP GET (i.e. a bare URL, not a POSTed form or XML-based data structure) could be one valid form of URL template, and thus satisfy the previous recommended binding. We would welcome the suggestion and adoption of an OpenURL profile for this function and more sophisticated OPAC transfers. We do not, however, require OpenURL implementations for Level 1.

8.3.2 OutputRewritablePage behavior

(Level 2 requires this or OutputIntermediateFormat; Level 4 requires both)

Summary
Outputs an OPAC display page in a form that includes embedded links or other content based on the display and on local customization.

Returns
A display with appropriate content embedded.

Sample associated outbound-oriented function

OutputRewritableBibliographicPage

Parameter: **bibid** (type string; optional)

Returns a bibliographic display that includes the bibid in an addressable location.

This function can be thought of as a refinement of the GoToBibliographicRequestPage function that uses the OutputRewritablePage behavior.

This function is given as an example; others may be defined as well by future interoperability standards.

Exceptional conditions

- **NotSupported**: The type of page requested is not rewritable.

Side Effects
None

Rationale
Outputting a page that can be rewritten allows a library to seamlessly add new services to its displays by embedding or linking them into the OPAC's own display pages.

Possible Bindings
Below are general notes on possible bindings. A particular ILS would need to specify a detailed binding describing exactly how an output page could be rewritten.

HTML/JavaScript binding (for outgoing behavior): If an ILS writes out HTML with the option of including library-added header content, then the library can load and execute its own JavaScript code to modify the page to include new links and content. The components of the page need to be addressable, which can be done in various ways, including unique and predictable identifiers on each component of the page that might be rewritten or used in a page rewrite. For example, the OutputRewritableBibliographicPage would need to include, at the very least, the bibid in a well-defined location that would be available for the JavaScript to use.

OpenURL binding (for incoming): OpenURLs may be a useful way of requesting a particular bibliographic, search, or other OPAC display. Details would need to be worked out.

8.3.3 OutputIntermediateFormat behavior

(Level 2 requires this or OutputRewritablePage; Level 4 requires both)

Summary
Outputs an OPAC display page in a semantically-oriented form that can be arbitrarily transformed to user output.

Returns

Information in an appropriate structured data format.

Sample associated outbound-oriented function

OutputIntermediateBibliographicPage
Parameter: bibid (type string; optional)
Returns a data structure that includes complete bibliographic information sufficient for translation into a user output page.

This function is given as an example; others may be defined as well in later versions of this recommendation.

Exceptional conditions

- **NotSupported**: The type of page requested is not available.

Side Effects
None

Rationale
Outputting a semantic format allows arbitrary rewriting to many different formats, including HTML for web browsers, web pages customized for mobile phones, RSS feeds, components that can be embedded in larger portal views, or XML elements for Ajax-driven applications.

Possible Bindings

Below are general notes on possible bindings. A particular ILS would need to specify a detailed binding describing exactly how an output page could be rewritten.

XML/XSLT binding (for outgoing behavior): Supporting output of OPAC pages in XML, using a server configuration (such as Cocoon) that allows the inclusion of custom XSLT stylesheets to transform the XML to HTML, is a common way of implementing this sort of behavior in a variety of systems. A fully specified binding would need to give a complete specification of the XML schema used for output.

OpenURL binding (for incoming behavior): OpenURLs may be a useful way of requesting a particular bibliographic, search, or other OPAC display. Details would need to be worked out.

9. Summary of Basic Discovery Interfaces compliance

In order to fully support the Basic Discovery Interfaces described in this document, an ILS must support (either on its own or with additional software) the Level 1 functions described in this document, using the recommended bindings for those functions. (In other words, the recommended bindings become required for full BDI compliance). To summarize these requirements:

HarvestBibliographicRecords and **HarvestExpandedRecords** must be supported for all records available for discovery via OAI-PMH. The implementation needs to realistically support incremental harvesting of record additions, updates, and deletions (where "deletions" include records that are no longer available for discovery). The use of the "deleted" OAI-PMH attribute is required to indicate records in this category, and the ILS should document how long "deleted" records are tracked. The records returned must include all the information in the underlying bibliographic records that are relevant to discovery. Required record formats include Dublin Core (which is required by the OAI-PMH standard), as well as MARC XML if the underlying ILS maintains or produces MARC records. For HarvestExpandedRecords, the underlying ILS must return item identifiers, and MARC holdings information where available, and the expanded records must reference a schema definition.

GetAvailability must be supported for all bibliographic records and items available for discovery, via a REST URL request and an XML response that conforms to the DLF XML schema, using the dlf:simpleavailability element, as defined in this document. If item-level identifiers or an item-level response are specified in the request, the return value must be at item-level granularity. Supporting bibliographic-level responses for availability in other cases is optional.

GoToBibliographicRequestPage must be supported for all bibliographic records relevant to discovery, using a templated URL. The ILS should document the form of this template. The OpenSearch URL template language is one form that this documentation can take.

10. Conclusion

This recommendation is just the start of the work needed to develop a comprehensive interoperability standard. It provides an overview of functionality that is known to be needed for enhanced interoperability at this point in time. However, it is not a fully specified standard; it does not have complete binding specifications for the functions that we define, and in some cases the suggested standards for bindings do not yet provide all of the functionality desired.

The functionality in this recommendation may be implemented within the ILS or by third-party packages on top of an ILS. In fact, some libraries and other third parties are already developing functionality surrounding record harvesting and real time availability as an overlay to their ILS. While it would be most convenient for the ILS to provide these interfaces natively, we also encourage libraries, vendors, and other developers to cooperate in designing and building suitable interfaces that can be implemented quickly. As there may be a variety of bindings for these abstract functions, the particular bindings used for implementation must have complete, openly accessible specifications without intellectual property restrictions on their use. To be fully compliant with a profile within this recommendation, the ILS interface must implement the functions and specify the bindings used in precise enough detail that users of the interface can use it with confidence.

We recommend a group review, accept, and publish qualifying bindings for existing functions, and proposed new functions in new documents following up this recommendation.

To be successful, these recommendations need to be followed up by implementation. We recommend that reference implementations be built for these functions soon. Ideally, these would be available as open source for public specification and reuse. At this writing, one prototype implementation exists for Level 1 functionality, and the Digital Library Federation intends to convene a workshop to assist developers that with to implement Level 1 capabilities for their ILS's and discovery applications. We hope that these activities will help develop a community that can support and build on the interoperability recommendations we make here.

This recommendation, along with related materials from the ILS-DI Task Group and other interested parties, is hosted at the Digital Library Federation's web site at <http://diglib.org/>. We hope that this recommendation helps lead to the development of ILS's that are more responsive to the needs of libraries and their users and make library content easier to discover and use effectively in the networked era.

Appendices

Appendix 1: Glossary of Terms

Abstract function
The description of a process in terms of its core characteristics, separated from the technologies that might be used to deliver its functionality.

Acquisition
The process of obtaining resources for the library's collection, typically including ordering, receiving and payment.

API
Application Programming Interface. A language and message format used by an application program to communicate with the operating system or some other control program such as a database management system (DBMS) or communications protocol. Functions in one program can also be called for by other programs and shared.

Atom
An XML-based file format that allows information from web-pages to be syndicated between applications.

Authority record
A record that shows the preferred form of a personal or corporate name, geographic region or subject. It also includes variant forms of the preferred form as cross references.

Barcode
A printed code, consisting of lines and spaces that can be read by a bar code scanner (reader), affixed to physical materials in a library collection to identify particular items for tracking and circulation.

Bath profile
A Z39.50 specification supporting library applications and resource discovery. It describes and specifies a subset of ANSI/NISO Z39.50-1995, Information Retrieval (Z39.50): Application Service Definition and Protocol Specification (ISO 23950). The profile defines searching across multiple servers to improve search and retrieval among library catalogues, union catalogues, and other electronic resources worldwide.

Bibliographic identifier
A unique identifier which unambiguously identifies a bibliographic record within an ILS catalog and is assumed to persistent, at least as long as the records are managed within the ILS.

Bibliographic metadata
Information about a resource that serves the purpose of discovery, identification and selection of the resource. Includes elements such as title, author, subjects, etc.

Bindings: see Concrete bindings

Call number scheme
A classification scheme, usually consisting of a numeric or alphanumeric notation, which categorizes or subdivides a subject area. Most classification schemes were originally intended to organize physical items on library shelves.

Catalog
A compilation of records describing the holdings of a given library or group of libraries.

Circulation
The process of lending library materials.

Concrete bindings
The specific technologies used to deliver specified abstract functionalities.

Course reserves
A selection of materials that instructors set aside, usually in the library, for the students in a class to read. Some ILS's have system modules that help manage circulation of the materials.

Discovery application
A computer application designed to simplify, assist and expedite the process of finding information resources.

Dublin Core
A fifteen element metadata set for use in resource description intended to facilitate discovery of electronic resources.

ERMS
Electronic Resources Management System. Used to manage a library's electronic resources, primarily e-journals and databases. Systems can include features to track trials, license terms and conditions, usage, cost, and access.

FRBR
Functional Requirement for Bibliographic Records. A conceptual model for the aggregation and display of bibliographic records. An entity-relationship model, with four primary entities - work, expression, manifestation, and item - which represent the products of intellectual or artistic endeavor.

Harvesting see Metadata harvesting

Holdings
Resources owned by a library. May be represented in MARC holdings and/or item records within an ILS.

ILS
Integrated Library System. A group of automated library subsystems working together and communicating within the same set or system of software to control such activities as circulation, cataloging, acquisitions, serial control and a public access catalog.

ISO 20775
An XML schema designed to express holdings, item, and availability information.

Item
A physical object in a library. Information about an item is recorded in an item record within an ILS which is used to manage circulation of the physical object.

Location
Where an item is kept in the library. Location codes are used in an ILS to show where the item is located in the library.

Lucene
An open source text search engine library written entirely in Java that can be used for applications that require fielded and full-text search, especially cross-platform.

MADS
Metadata Authority Description Schema. An XML schema for an authority element set that may be used to provide metadata about agents (people, organizations), events, and terms (topics, geographics, genres, etc.). MADS was created to serve as a companion to the Metadata Object Description Schema (MODS).

MARC holdings
A record structure and encoding standard to record holdings information for library resources, including serials and nonserial resources.

MARC 21
A recent version of standard data formats for MAchine-Readable Cataloging. A record structure and encoding standard for electronic bibliographic records developed by the Library of Congress which provides the mechanism by which computers exchange, use, and interpret bibliographic information. There are also MARC21 formats for authority data, holdings data, classification data and community information.

MARCXML
A metadata scheme for working with MARC data in a XML environment. The schema supports all MARC encoded data regardless of format. The recommended MARC XML schema is defined by the Library of Congress, and is known as the "marc21" format in the OAI-PMH implementation guidelines.

Metadata
Structured information that describes an information resource. "Data about data" which includes data associated with either an information system or an information object for purposes of description, administration, legal requirements, technical functionality, use and usage, and preservation.

Metadata harvesting
A technique for extraction of metadata from individual repositories for collection into a central catalog.

MODS
Metadata Object Description Schema. Intended to be able to carry selected data from existing MARC 21 records as well as to enable the creation of original resource description records. Includes a subset of MARC fields and uses language-based tags

rather than numeric ones, in some cases regrouping elements from the MARC 21 bibliographic format. Expressed in an XML schema language.

NCIP
NISO Circulation Interchange Protocol (NCIP). A standard which defines a protocol for the exchange of messages between and among computer-based application to enable them to perform functions necessary to lend and borrow items, to provide controlled access to electronic resources, and to facilitate co-operative management of these functions.

OAI-PMH
OAI-Protocol for Metadata Harvesting. Protocol for application-independent interoperability framework based on metadata harvesting, open standards HTTP (Hypertext Transport Protocol) and XML (Extensible Markup Language). Includes standards for data providers who administer systems that wish to expose metadata and service providers that use harvested metadata.

OPAC
Online Public Access Catalog. A library catalog which can be searched online and is a module of the ILS.

OpenURL
A protocol for "actionable" URLs that delivers appropriate data or services based on encoded resource metadata. It is often used to refer library users to an appropriate, context-sensitive accessible instance of a resource or library service.

OpenSearch
A collection of technologies developed by Amazon that allow publishing of search results in a format suitable for syndication and aggregation.

Open Source
A concept through which programming code is made available through a license that supports the users freely copying the code, making changes it, and sharing the results. Changes are typically submitted to a group managing the open source product for possible incorporation into the official version. Development and support is handled cooperatively by a group of distributed programmers, usually on a volunteer basis.

REST
Representational State Transfer. It generally attempts to emulate HTTP and similar protocols by constraining the interface to a set of well-known, standard operations (e.g., GET, PUT, DELETE) with a focus on interacting with stateful resources, rather than messages or operations. It can be used a simple way of implementing web services, often taking URLs as input and returning XML streams as output.

RSS
Really Simple Syndication. An XML format used for distribution or syndication of frequently updated Web content such as news items, blogs, and podcasts. Several versions of RSS exist.

SIP2
Standard Interface Protocol Version 2. Standard for the exchange of circulation data and transactions between different systems.

Solr
An open source enterprise search server based on the Lucene Java search library.

SRU
Search/Retrieve via URL. A standard search protocol for Internet search queries, utilizing CQL (Common Query Language), standard query syntax for representing queries. A variation of SRW, currently with more active development.

SRW
Search/Retrieve Webservice. A web services implementation of the Z39.50 protocol that specifies a client/server-based protocol for searching and retrieving information from remote databases. It specifies procedures and structures for a client system to search a database provided by a server, retrieve database records identified by a search, scan a term list, and sort a result set. The protocol addresses communication between corresponding information retrieval applications, the client and server (which may reside on different computers); it does not address interaction between the client and the end-user. It is a variation of SRU.

Unicode
A universal character-encoding standard used for representation of text for computer processing. Unicode provides a unique numeric code (a code point) for every character, no matter what the platform, no matter what the program, no matter what the language. The standard was developed by the Unicode Consortium.

Web Service
Software system designed to support interoperable machine to machine exchange of data/information, typically using the XML, SOAP, WSDL and UDDI open standards.

XML
EXtensible Markup Language. An open standard for describing data from the World Wide Web Consortium. Used for defining data elements on a Web page, business-to-business documents, and other hierarchically structured text and data. XML uses a similar tag structure to HTML; however, whereas HTML specifically describes elements on web pages, XML can be used to describe arbitrary elements, in a more constrained syntax, and supports links to definitions of the structure and meaning of the elements.

XSL
EXtensible Stylesheet Language. An XML-based style sheet language.

XSLT
XSL Transformations. A language used to transform XML documents into XHTML documents or to other XML documents.

Z39.50
A NISO and ISO standard protocol that specifies a client/server-based protocol for cross-system searching and retrieving information from remote databases. It specifies procedures and structures for a client system to search a database provided by a server, retrieve database records identified by a search, scan a term list, and sort a result set.

Appendix 2: Expanded Record using MARCXML and NCIP

This response shows bibliographic and item information (assuming no MARC holdings exist) for 1 item associated with 1 bibliographic record. This response could be used with HarvestExpandedRecords in the Data Aggregation section or with Search or GetRecords in the Real Time Search section.

The NCIP schema examples we give in this document are not normative, but are based on our understanding of the schema at this time. ILS's and clients using the NCIP XML standards should use official published versions of their schemas, even if they differ from what we show here.

```
<dlf:collection xmlns:dlf="http://diglib.org/ilsdi/1.1"
    xmlns:marcxml="http://www.loc.gov/MARC21/slim"
    xmlns:ncip="http://ncip.envisionware.com/documentation/ncip_v1_0.xsd" >
  <dlf:record>
     <dlf:bibliographic id="92005291">
       <marcxml:record>
         <marcxml:recordTypeType>Bibliographic</marcxml:recordTypeType>
         <marcxml:leader>01142cam  2200301 a 4500</marcxml:leader>
         <marcxml:controlfield tag="001">   92005291 </marcxml:controlfield>
         <marcxml:controlfield tag="003">DLC</marcxml:controlfield>
         <marcxml:controlfield
tag="005">19930521155141.9</marcxml:controlfield>
         <marcxml:controlfield tag="008">920219s1993    caua   j      000 0 eng
</marcxml:controlfield>
         <marcxml:datafield tag="010" ind1=" " ind2=" ">
           <marcxml:subfield code="a">   92005291 </marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="020" ind1=" " ind2=" ">
           <marcxml:subfield code="a">0152038655 :</marcxml:subfield>
           <marcxml:subfield code="c">$15.95</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="040" ind1=" " ind2=" ">
           <marcxml:subfield code="a">DLC</marcxml:subfield>
           <marcxml:subfield code="c">DLC</marcxml:subfield>
           <marcxml:subfield code="d">DLC</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="042" ind1=" " ind2=" ">
            <marcxml:subfield code="a">lcac</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="050" ind1="0" ind2="0">
           <marcxml:subfield code="a">PS3537.A618</marcxml:subfield>
           <marcxml:subfield code="b">A88 1993</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="082" ind1="0" ind2="0">
           <marcxml:subfield code="a">811/.52</marcxml:subfield>
           <marcxml:subfield code="2">20</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="100" ind1="1" ind2=" ">
           <marcxml:subfield code="a">Sandburg, Carl,</marcxml:subfield>
           <marcxml:subfield code="d">1878-1967.</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="245" ind1="1" ind2="0">
           <marcxml:subfield code="a">Arithmetic /</marcxml:subfield>
           <marcxml:subfield code="c">Carl Sandburg ; illustrated as an
anamorphic adventure by Ted Rand.</marcxml:subfield>
         </marcxml:datafield>
         <marcxml:datafield tag="250" ind1=" " ind2=" ">
           <marcxml:subfield code="a">1st ed.</marcxml:subfield>
```

```
        </marcxml:datafield>
        <marcxml:datafield tag="260" ind1=" " ind2=" ">
          <marcxml:subfield code="a">San Diego :</marcxml:subfield>
          <marcxml:subfield code="b">Harcourt Brace
Jovanovich,</marcxml:subfield>
          <marcxml:subfield code="c">c1993.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="300" ind1=" " ind2=" ">
          <marcxml:subfield code="a">1 v. (unpaged) :</marcxml:subfield>
          <marcxml:subfield code="b">ill. (some col.) ;</marcxml:subfield>
          <marcxml:subfield code="c">26 cm.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="500" ind1=" " ind2=" ">
          <marcxml:subfield code="a">One Mylar sheet included in
pocket.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="520" ind1=" " ind2=" ">
          <marcxml:subfield code="a">A poem about numbers and their
characteristics. Features anamorphic, or distorted, drawings which can be
restored to normal by viewing from a particular angle or by viewing the image's
reflection in the provided Mylar cone.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="650" ind1=" " ind2="0">
          <marcxml:subfield code="a">Arithmetic</marcxml:subfield>
          <marcxml:subfield code="x">Juvenile poetry.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="650" ind1=" " ind2="0">

          <marcxml:subfield code="a">Children's poetry,
American.</marcxml:subfield>

        </marcxml:datafield>
        <marcxml:datafield tag="650" ind1=" " ind2="1">
          <marcxml:subfield code="a">Arithmetic</marcxml:subfield>
          <marcxml:subfield code="x">Poetry.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="650" ind1=" " ind2="1">
          <marcxml:subfield code="a">American poetry.</marcxml:subfield>
        </marcxml:datafield>
        <marcxml:datafield tag="650" ind1=" " ind2="1">

          <marcxml:subfield code="a">Visual perception.</marcxml:subfield>

        </marcxml:datafield>

        <marcxml:datafield tag="700" ind1="1" ind2=" ">
          <marcxml:subfield code="a">Rand, Ted,</marcxml:subfield>
          <marcxml:subfield code="e">ill.</marcxml:subfield>
        </marcxml:datafield>
      </marcxml:record>
    </dlf:bibliographic>
    <dlf:items>
      <dlf:item id="S02420717N">
        <ncip:LookupItemResponse>
          <ncip:UniqueItemId>
            <ncip:UniqueAgencyId>ncsu</ncip:UniqueAgencyId>
<ncip:ItemIdentifierValue>S02420717N</ncip:ItemIdentifierValue>
          </ncip:UniqueItemId>
          <ncip:HoldPickupDate>2008-04-13T12:00:00-05:00</ncip:HoldPickupDate>
          <ncip:DateRecalled>2008-04-01T12:00:00-05:00</ncip:DateRecalled>
          <ncip:ItemOptionalFields>
            <ncip:BibliographicDescription>
              <ncip:BibliographicRecordId>
```

```
<ncip:BibliographicRecordIdentifier>92005291</ncip:BibliographicRecordIdentifie
r>
                  <ncip:UniqueAgencyId>ncsu</ncip:UniqueAgencyId>
                </ncip:BibliographicRecordId>
              </ncip:BibliographicDescription>
              <ncip:ItemUseRestrictionType>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/itemuserestrictiontype/
itemuserestrictiontype.scm</ncip:Scheme>
                <ncip:Value>in-library use only</ncip:Value>
              </ncip:ItemUseRestrictionType>
              <ncip:CirculationStatus>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/circulationstatus/circu
lationstatus.scm</ncip:Scheme>
                <ncip:Value>On Hold</ncip:Value>
              </ncip:CirculationStatus>
              <ncip:HoldQueueLength>2</ncip:HoldQueueLength>
              <ncip:ItemDescription>
                <ncip:CallNumber>QA241 .S53 2007</ncip:CallNumber>
                <ncip:CopyNumber>copy 1</ncip:CopyNumber>
                <ncip:ItemDescriptionLevel>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/itemdescriptionlevel/it
emdescriptionlevel.scm</ncip:Scheme>

                  <ncip:Value>Item</ncip:Value>

                </ncip:ItemDescriptionLevel>
                <ncip:VisibleItemId>
<ncip:VisibleItemIdentifierType>Barcode</ncip:VisibleItemIdentifierType>
<ncip:VisibleItemIdentifier>S02420717N</ncip:VisibleItemIdentifier>
                </ncip:VisibleItemId>
              </ncip:ItemDescription>
              <ncip:Location>

                <ncip:LocationType>Permanent Location</ncip:LocationType>

                <ncip:LocationName>
                  <ncip:LocationNameInstance>
                    <ncip:LocationNameLevel>1</ncip:LocationNameLevel>
                    <ncip:LocationNameValue>D. H. Hill
Library</ncip:LocationNameValue>
                  </ncip:LocationNameInstance>
                  <ncip:LocationNameInstance>

                    <ncip:LocationNameLevel>2</ncip:LocationNameLevel>
<ncip:LocationNameValue>Stacks</ncip:LocationNameValue>
                   </ncip:LocationNameInstance>
                </ncip:LocationName>
              </ncip:Location>
              <ncip:SensitizationFlag />
            </ncip:ItemOptionalFields>
          </ncip:LookupItemResponse>
        </dlf:item>
      </dlf:items>
    </dlf:record>

 </dlf:collection>
```

Appendix 3: Expanded Record using MODS and ISO Holdings (ISO 20775)

This response shows bibliographic and item information (assuming no MARC holdings exist) for 1 item associated with 1 bibliographic record. This response could be used with HarvestExpandedRecords in the Data Aggregation section or Search or GetRecords in the Real Time Search section.

Note that we specifically give the item id of the item mentioned in the holdings in the dlf:holdingset element after the dlf:holdingsrec element. Among other things, this lets clients who don't know how to interpret the ISO holdings record still retrieve the item ID from the simpler enclosing DLF schema. We also give the item id of the items associated with the bibliographic record in a dlf:items element as a child of the dlf:record element. Normally, only the latter is required; in this case, we have a slightly redundant output to show how items would be associated with a particular holdings record when the ILS maintains this information. If there is additional information about the items to be shown, it would be included in the dlf:item elements of the dlf:items element directly underneath dlf:record, with only an ID specified in the dlf:item elements within the dlf:holdingset element.

The ISO Holdings schema examples we give in this document are not normative, but are based on our understanding of the schema at this time. ILS's and clients using the ISO holdings XML standard should use official published versions of that schema, even if they differ from what we show here.

```
<dlf:collection xmlns:dlf="http://diglib.org/ilsdi/1.1"
    xmlns:mods="http://www.loc.gov/mods/v3"
    xmlns:holdings="http://www.loc.gov/standards/iso20775/" >
  <dlf:record>
    <dlf:bibliographic id="11761548">
      <mods:mods xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="3.0"
xsi:schemaLocation="http://www.loc.gov/mods/v3
http://www.loc.gov/standards/mods/v3/mods-3-0.xsd">
        <mods:titleInfo>
          <mods:title>Sound and fury :</mods:title>
          <mods:subTitle>the making of the punditocracy /</mods:subTitle>
        </mods:titleInfo>
        <mods:name type="personal">
          <mods:namePart>Alterman, Eric</mods:namePart>
          <mods:role>
            <mods:roleTerm type="text">creator</mods:roleTerm>
          </mods:role>
        </mods:name>
        <mods:typeOfResource>text</mods:typeOfResource>
        <mods:genre authority="marcgt">bibliography</mods:genre>
        <mods:originInfo>
          <mods:place>
            <mods:placeTerm authority="marccountry"
type="code">nyu</mods:placeTerm>
          </mods:place>
          <mods:place>
            <mods:placeTerm type="text">Ithaca, N.Y</mods:placeTerm>
          </mods:place>
          <mods:publisher>Cornell University Press</mods:publisher>
          <mods:dateIssued>c1999</mods:dateIssued>
          <mods:dateIssued encoding="marc">1999</mods:dateIssued>
```

```
          <mods:issuance>monographic</mods:issuance>
        </mods:originInfo>
        <mods:language>
          <mods:languageTerm authority="iso639-2b"
type="code">eng</mods:languageTerm>
        </mods:language>
        <mods:physicalDescription>
          <mods:form authority="marcform">print</mods:form>
          <mods:extent>vii, 322 p. ; 23 cm.</mods:extent>
        </mods:physicalDescription>
        <mods:note type="statement of responsibility">Eric
Alterman.</mods:note>
        <mods:note>Includes bibliographical references (p. 291-312) and
index.</mods:note>
        <mods:subject authority="lcsh">
          <mods:topic>Journalism</mods:topic>
          <mods:topic>Political aspects</mods:topic>
          <mods:geographic>United States.</mods:geographic>
        </mods:subject>
        <mods:subject authority="lcsh">
          <mods:geographic>United States</mods:geographic>
          <mods:topic>Politics and government</mods:topic>
          <mods:temporal>20th century.</mods:temporal>
        </mods:subject>
        <mods:subject authority="lcsh">
          <mods:topic>Mass media</mods:topic>
          <mods:topic>Political aspects</mods:topic>
          <mods:geographic>United States.</mods:geographic>
        </mods:subject>
        <mods:subject authority="lcsh">
          <mods:topic>Television and politics</mods:topic>
          <mods:geographic>United States.</mods:geographic>
        </mods:subject>
        <mods:subject authority="lcsh">
          <mods:topic>Press and politics</mods:topic>
          <mods:geographic>United States.</mods:geographic>
        </mods:subject>
        <mods:subject authority="lcsh">
          <mods:topic>Talk shows</mods:topic>
          <mods:geographic>United States.</mods:geographic>
        </mods:subject>
        <mods:classification authority="lcc">PN4888.P6 A48
1999</mods:classification>
        <mods:classification edition="21"
authority="ddc">071/.3</mods:classification>
        <mods:recordInfo>
          <mods:recordContentSource>DLC</mods:recordContentSource>
          <mods:recordCreationDate
encoding="marc">990730</mods:recordCreationDate>
          <mods:recordChangeDate
encoding="iso8601">20000406144503.0</mods:recordChangeDate>
          <mods:recordIdentifier>11761548</mods:recordIdentifier>
        </mods:recordInfo>
      </mods:mods>
    </dlf:bibliographic>
    <dlf:holdings>
      <dlf:holdingset>
        <dlf:holdingsrec>
           <holdings:holding>
              <holdings:holdingsSimple>
                <holdings:copiesSummary>
                  <holdings:copiesCount>1</holdings:copiesCount>
                  <holdings:status>
                   <holdings:availableCount>0</holdings:availableCount>
                   <holdings:availableFor>1</holdings:availableFor>
```

```
                  </holdings:status>
  <holdings:reservationQueueLength>2</holdings:reservationQueueLength>
                  <holdings:onOrderCount>1</holdings:onOrderCount>
                </holdings:copiesSummary>
                <holdings:copyInformation>
                  <holdings:pieceIdentifier>
                    <holdings:value>S02420717N</holdings:value>
                    <holdings:typeOrSource>
                      <holdings:text>barcode</holdings:text>
                    </holdings:typeOrSource>
                  </holdings:pieceIdentifier>
                  <holdings:resourceIdentifier>
                    <holdings:value>11761548</holdings:value>
                    <holdings:typeOrSource>
                      <holdings:text>bibliographic</holdings:text
                    </holdings:typeOrSource>
                  </holdings:resourceIdentifier>
                  <holdings:form>
                    <holdings:value>text</holdings:value>
                    <holdings:typeOrSource>
                      <holdings:text>marc formats</holdings:text>
                    </holdings:typeOrSource>
                  </holdings:form>
                  <holdings:sublocation>D. H. Hill
Library</holdings:sublocation>
                  <holdings:sublocation>Stacks</holdings:sublocation>
                  <holdings:shelfLocator>QA241 .S53
2007</holdings:shelfLocator>
                  <holdings:note>Water damage to back cover.</holdings:note>
                  <holdings:availabilityInformation>
                    <holdings:status>
<holdings:availabilityStatus>2</holdings:availabilityStatus>
                      <holdings:availableFor>1</holdings:availableFor>
                      <holdings:dateTimeAvailable>2008-05-10T12:00:00-
05:00</holdings:dateTimeAvailable>
                    </holdings:status>
                    <holdings:reservationPolicy>1</holdings:reservationPolicy>
                     <holdings:reservationQueue>2</holdings:reservationQueue>
                  </holdings:availabilityInformation>
                </holdings:copyInformation>
              </holdings:holdingsSimple>
              <holdings:summaryPolicy>
                <holdings:form>
                  <holdings:value>text</holdings:value>
                  <holdings:typeOrSource>
                    <holdings:text>marc formats</holdings:text>
                  </holdings:typeOrSource>
                </holdings:form>
                <holdings:availability>
                  <holdings:availableFor>1</holdings:availableFor>
                   <holdings:text>loans for 30 days</holdings:text>
                </holdings:availability>
                <holdings:reservationPolicy>1</holdings:reservationPolicy>
              </holdings:summaryPolicy>
              <holdings:summaryHistory>
                <holdings:countPeriod>
                  <holdings:countPeriodStart>2001-01-15T12:00:00-
05:00</holdings:countPeriodStart>
                  <holdings:countPeriodEnd>2008-04-15T12:00:00-
05:00</holdings:countPeriodEnd>
                  <holdings:totalCirculation>
                    <holdings:totalCirculationCount>56
                    </holdings:totalCirculationCount>
                    <holdings:totalLoansCount>53</holdings:totalLoansCount>
                  </holdings:totalCirculation>
```

```
<holdings:totalReservationCount>10</holdings:totalReservationCount>
                  <holdings:copiesCount>
<holdings:totalCopiesHeld>1</holdings:totalCopiesHeld>
                  </holdings:copiesCount>
                </holdings:countPeriod>
              </holdings:summaryHistory>
              <holdings:lastActivityInfo>
                <holdings:lastActivityDate>2008-04-01T12:00:00-
05:00</holdings:lastActivityDate>
                <holdings:lastActivityType>
                  <holdings:value>loan</holdings:value>
                  <holdings:typeOrSource>
                    <holdings:text>something here</holdings:text>
                  </holdings:typeOrSource>
                </holdings:lastActivityType>
              </holdings:lastActivityInfo>
            </holdings:holding>
          </dlf:holdingsrec>
          <dlf:items>
            <dlf:item id ="S02420717N" />
          </dlf:items>
        </dlf:holdingset>
      </dlf:holdings>
    <dlf:items>
      <dlf:item id ="S02420717N" />
    </dlf:items>
  </dlf:record>
</dlf:collection>
```

Appendix 4: GetAvailability Responses with DLF:simpleavailability

Here we give example responses for 2 items associated with 1 bibliographic record. These responses satisfy the Basic Discovery Interfaces profile. Example 2, showing a bibliographic-level availability summary, is optional in the BDI profile.

Example 1: A response for 2 copies of the same book, showing item-level availability:

```
<dlf:collection xmlns:dlf="http://diglib.org/ilsdi/1.1">
  <dlf:record>
    <dlf:bibliographic id="92005291" />
    <dlf:items>
      <dlf:item id="S02420717N">
        <dlf:simpleavailability>
          <dlf:identifier>S02420717N</dlf:identifier>
          <dlf:availabilitystatus>not available</dlf:availabilitystatus>
          <dlf:availabilitymsg>Held for Patron (2
requests)</dlf:availabilitymsg>
          <dlf:location>D. H. Hill Library</dlf:location>
        </dlf:simpleavailability>
      </dlf:item>
      <dlf:item id="S01149512N">
        <dlf:simpleavailability>
          <dlf:identifier>S01149512N</dlf:identifier>
          <dlf:availabilitystatus>available</dlf:availabilitystatus>
          <dlf:location>D. H. Hill Library, Call # QA107 .S73</dlf:location>
          <dlf:availabilitymsg>library use only</dlf:availabilitymsg>
        </dlf:simpleavailability>
      </dlf:item>
    </dlf:items>
  </dlf:record>
</dlf:collection>
```

Example 2: A response for the same book, showing bibliographic-level availability:

```
<dlf:collection xmlns:dlf="http://diglib.org/ilsdi/1.1">
  <dlf:record>
    <dlf:bibliographic id="92005291" />
    <dlf:simpleavailability>
       <dlf:identifier>92005291</dlf:identifier>
       <dlf:availabilitystatus>available</dlf:availabilitystatus>
       <dlf:availabilitymsg>1 copy available, library use
only</dlf:availabilitymsg>
       <dlf:location>D. H. Hill Library, Call # QA107 .S73</dlf:location>
    </dlf:simpleavailability>
  </dlf:record>
</dlf:collection>
```

Appendix 5: GetAvailability Responses with NCIP, ISO Holdings (ISO 20775) and DLF:simpleavailability

Here we give example responses for 2 items associated with 1 bibliographic record, utilizing the more complex NCIP or ISO Holdings schemas in addition to the required dlf:simpleavailability response. Note that the Basic Discovery Interfaces profile requires using only the dlf:simpleavailability schema. These schemas, however, may be used if the client requests these formats.

The NCIP and ISO Holdings schema examples we give in this document are not normative, but are based on our understanding of the schema at this time. ILS's and clients using these standards should use official published versions of their schemas, even if they differ from what we show here.

Example 1: A response using only the NCIP schema, for the same book and items used in Appendix 4:

```
<dlf:collection
    xmlns:dlf="http://diglib.org/ilsdi/1.1"
    xmlns:ncip="http://ncip.envisionware.com/documentation/ncip_v1_0.xsd" >
  <dlf:record>
    <dlf:bibliographic id="92005291" />
    <dlf:items>
      <dlf:item id="S02420717N">
        <ncip:LookupItemResponse>
          <ncip:UniqueItemId>
            <ncip:UniqueAgencyId>ncsu</ncip:UniqueAgencyId>
<ncip:ItemIdentifierValue>S02420717N</ncip:ItemIdentifierValue>
          </ncip:UniqueItemId>
          <ncip:ItemOptionalFields>
            <ncip:BibliographicDescription>
              <ncip:BibliographicRecordId>
<ncip:BibliographicRecordIdentifier>92005291</ncip:BibliographicRecordIdentifie
r>
                <ncip:UniqueAgencyId>ncsu</ncip:UniqueAgencyId>
              </ncip:BibliographicRecordId>
            </ncip:BibliographicDescription>
            <ncip:ItemUseRestrictionType>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/itemuserestrictiontype/
itemuserestrictiontype.scm</ncip:Scheme>
              <ncip:Value>Limited Circulation, Normal Loan Period </ncip:Value>
            </ncip:ItemUseRestrictionType>
            <ncip:CirculationStatus>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/circulationstatus/circu
lationstatus.scm</ncip:Scheme>
              <ncip:Value>On Hold</ncip:Value>
            </ncip:CirculationStatus>
            <ncip:HoldQueueLength>2</ncip:HoldQueueLength>
            <ncip:ItemDescription>
              <ncip:CallNumber>QA241 .S53 2007</ncip:CallNumber>
              <ncip:VisibleItemId>
<ncip:VisibleItemIdentifierType>Barcode</ncip:VisibleItemIdentifierType>
<ncip:VisibleItemIdentifier>S02420717N</ncip:VisibleItemIdentifier>
              </ncip:VisibleItemId>
            </ncip:ItemDescription>
            <ncip:Location>
              <ncip:LocationType>Permanent Location</ncip:LocationType>
              <ncip:LocationName>
                <ncip:LocationNameInstance>
```

```
                  <ncip:LocationNameLevel>1</ncip:LocationNameLevel>
                  <ncip:LocationNameValue>D. H. Hill
Library</ncip:LocationNameValue>
                </ncip:LocationNameInstance>
                <ncip:LocationNameInstance>
                  <ncip:LocationNameLevel>2</ncip:LocationNameLevel>
<ncip:LocationNameValue>Stacks</ncip:LocationNameValue>
                </ncip:LocationNameInstance>
              </ncip:LocationName>
            </ncip:Location>
          </ncip:ItemOptionalFields>
        </ncip:LookupItemResponse>
      </dlf:item>
      <dlf:item id="S01149512N">
        <ncip:LookupItemResponse>
          <ncip:UniqueItemId>
            <ncip:UniqueAgencyId>ncsu</ncip:UniqueAgencyId>
<ncip:ItemIdentifierValue>S01149512N</ncip:ItemIdentifierValue>
          </ncip:UniqueItemId>
          <ncip:ItemOptionalFields>
            <ncip:BibliographicDescription>
              <ncip:BibliographicRecordId>
<ncip:BibliographicRecordIdentifier>92005291</ncip:BibliographicRecordIdentifie
r>
                <ncip:UniqueAgencyId>ncsu</ncip:UniqueAgencyId>
              </ncip:BibliographicRecordId>
            </ncip:BibliographicDescription>
            <ncip:ItemUseRestrictionType>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/itemuserestrictiontype/
itemuserestrictiontype.scm</ncip:Scheme>
               <ncip:Value>In Library Use Only</ncip:Value>
            </ncip:ItemUseRestrictionType>
            <ncip:CirculationStatus>
<ncip:Scheme>http://www.niso.org/ncip/v1_0/imp1/schemes/circulationstatus/circu
lationstatus.scm</ncip:Scheme>
               <ncip:Value>Available On Shelf</ncip:Value>
            </ncip:CirculationStatus>
            <ncip:ItemDescription>
              <ncip:CallNumber>QA107 .S73</ncip:CallNumber>
              <ncip:VisibleItemId>
<ncip:VisibleItemIdentifierType>Barcode</ncip:VisibleItemIdentifierType>
<ncip:VisibleItemIdentifier>S01149512N</ncip:VisibleItemIdentifier>
              </ncip:VisibleItemId>
            </ncip:ItemDescription>
            <ncip:Location>
              <ncip:LocationType>Permanent Location</ncip:LocationType>
              <ncip:LocationName>
                <ncip:LocationNameInstance>
                  <ncip:LocationNameLevel>1</ncip:LocationNameLevel>
                  <ncip:LocationNameValue>D. H. Hill
Library</ncip:LocationNameValue>
                </ncip:LocationNameInstance>
                <ncip:LocationNameInstance>
                  <ncip:LocationNameLevel>2</ncip:LocationNameLevel>
                  <ncip:LocationNameValue>Reference
Material</ncip:LocationNameValue>
                </ncip:LocationNameInstance>
              </ncip:LocationName>
            </ncip:Location>
          </ncip:ItemOptionalFields>
        </ncip:LookupItemResponse>
      </dlf:item>
    </dlf:items>
  </dlf:record>
</dlf:collection>
```

Example 2: A response using both the ISO 20775 and dlf:simpleavailability schemas, for the same book and items used in Appendix 4:

```
<dlf:collection xmlns:dlf=http://diglib.org/ilsdi/1.1
                    xmlns:holdings=http://www.loc.gov/standards/iso20775/ >
  <dlf:record>
    <dlf:bibliographic id="92005291" />
    <dlf:items>
      <dlf:item id="S02420717N">
        <dlf:simpleavailability>
          <dlf:identifier>S02420717N</dlf:identifier>
          <dlf:availabilitystatus>not available</dlf:availabilitystatus>
          <dlf:availabilitymsg>Held for Patron (2
requests)</dlf:availabilitymsg>
           <dlf:dateavailable>2008-05-10T12:00:00-05:00</dlf:dateavailable>
          <dlf:location>D. H. Hill Library</dlf:location>
        </dlf:simpleavailability>
        <holdings:copyInformation>
          <holdings:pieceIdentifier>
            <holdings:value>S02420717N</holdings:value>
            <holdings:typeOrSource>
              <holdings:text>barcode</holdings:text>
            </holdings:typeOrSource>
          </holdings:pieceIdentifier>
          <holdings:resourceIdentifier>
            <holdings:value>92005291</holdings:value>
             <holdings:typeOrSource>
                <holdings:text>bibliographic</holdings:text>
             </holdings:typeOrSource>
          </holdings:resourceIdentifier>
          <holdings:sublocation>D. H. Hill Library</holdings:sublocation>
          <holdings:sublocation>Stacks</holdings:sublocation>
          <holdings:sublocation>Held for patron</holdings:sublocation>
          <holdings:shelfLocator>QA241 .S53 2007</holdings:shelfLocator>
          <holdings:availabilityInformation>
             <holdings:status>
               <holdings:availabilityStatus>2</holdings:availabilityStatus>
               <holdings:availableFor>1</holdings:availableFor>
                <holdings:dateTimeAvailable>2008-05-10T12:00:00-
05:00</holdings:dateTimeAvailable>
             </holdings:status>
              <holdings:reservationPolicy>1</holdings:reservationPolicy>
             <holdings:reservationQueue>2</holdings:reservationQueue>
          </holdings:availabilityInformation>
        </holdings:copyInformation>
      </dlf:item>
      <dlf:item id="S01149512N">
        <dlf:simpleavailability>
          <dlf:identifier>S01149512N</dlf:identifier>
          <dlf:availabilitystatus>available</dlf:availabilitystatus>
          <dlf:location>D. H. Hill Library, Call # QA107 .S73</dlf:location>
          <dlf:availabilitymsg>library use only</dlf:availabilitymsg>
        </dlf:simpleavailability>
        <holdings:copyInformation>
          <holdings:pieceIdentifier>
            <holdings:value>S01149512N</holdings:value>
            <holdings:typeOrSource>
              <holdings:text>barcode</holdings:text>
            </holdings:typeOrSource>
          </holdings:pieceIdentifier>
          <holdings:resourceIdentifier>
            <holdings:value>92005291</holdings:value>
             <holdings:typeOrSource>
```

```
                <holdings:text>bibliographic</holdings:text>
              </holdings:typeOrSource>
            </holdings:resourceIdentifier>
            <holdings:sublocation>D. H. Hill Library</holdings:sublocation>
            <holdings:sublocation>Stacks</holdings:sublocation>
            <holdings:shelfLocator>QA107 .S73</holdings:shelfLocator>
            <holdings:availabilityInformation>
              <holdings:status>
                <holdings:availabilityStatus>1</holdings:availabilityStatus>
                <holdings:availableFor>6</holdings:availableFor>
              </holdings:status>
              <holdings:policy>In Library Use Only</holdings:policy>
               <holdings:reservationPolicy>2</holdings:reservationPolicy>
            </holdings:availabilityInformation>
          </holdings:copyInformation>
        </dlf:item>
      </dlf:items>
    </dlf:record>
  </dlf:collection>
```

Appendix 6: Enriched Scan Response with Subject Authority Index Entries

```
<sru:scanResponse xmlns:sru="http://www.loc.gov/standards/sru/"

xmlns:mads="http://www.loc.gov/standards/mads/">
<sru:version>1.1</sru:version>
  <sru:terms>
    <sru:term>
      <sru:value>Revolutionary War, American, 1775-1783</sru:value>
      <sru:numberOfRecords>0</sru:numberOfRecords>
      <sru:extraTermData>
       <mads:authority>

         <mads:topic>United States</mads:topic>
        <mads:topic>History</mads:topic>
        <mads:temporal>Revolution, 1775-1783</mads:temporal>
      </mads:authority>
      <mads:identifier type="lccn">sh85140139</mads:identifier>
      </sru:extraTermData>
    </sru:term>

    <sru:term>
      <sru:value>Revolutions</sru:value>
      <sru:numberOfRecords>236</sru:numberOfRecords>
      <sru:extraTermData>
         <mads:related type="narrower">
           <mads:topic>Coups d'e?tat</mads:topic>
         </mads:related>

          <mads:related type="other">
            <mads:topic>Government, resistance to</mads:topic>
         </mads:related>
         <mads:identifier type="lccn">sh85113507</mads:identifier>
      </sru:extraTermData>
    </sru:term>
    </sru:terms>
  <sru:echoedScanRequest>

    <sru:version>1.1</sru:version>
    <sru:scanClause>dc.subject="revolutionary war"</sru:scanClause>
    <sru:responsePosition>1</sru:responsePosition>
    <sru:maximumTerms>2</sru:maximumTerms>
  </sru:echoedScanRequest>
</sru:scanResponse>
```

Appendix 7: The Berkeley Accord, Spring 2008

The following agreement was announced on April 4, 2008 by Peter Brantley, executive director for the Digital Library Federation

ILS Basic Discovery Interfaces: A proposal for the ILS community.

On March 6, representatives of the Digital Library Federation (DLF), academic libraries, and major library application vendors met in Berkeley, California to discuss a draft recommendation from the DLF for standard interfaces for integrating the data and services of the Integrated Library System (ILS) with new applications supporting user discovery. Such standard interfaces will allow libraries to deploy new discovery services to meet ever-growing user expectations in the Web 2.0 era, take full advantage of advanced ILS data management and services, and encourage a strong, innovative community and marketplace in next-generation library management and discovery applications.

At the meeting, participants agreed to support a set of essential functions through open protocols and technologies by deploying specific recommended standards.

These functions are:

1. **Harvesting**. Functions to harvest data records for library collections, both in full, and incrementally based on recent changes. Harvesting options could include either the core bibliographic records, or those records combined with supplementary information (such as holdings or summary circulation data). Both full and differential harvesting options are expected to be supported through an OAI-PMH interface.

2. **Availability**. Real-time querying of the availability of a bibliographic (or circulating) item. This functionality will be implemented through a simple REST interface to be specified by the ILS-DI task group.

3. **Linking**. Linking in a stable manner to any item in an OPAC in a way that allows services to be invoked on it; for example, by a stable link to a page displaying the item's catalog record and providing links for requests for that item. This functionality will be implemented through a URL template defined for the OPAC as specified by the ILS-DI task group.

Next steps:

The DLF ILS-Discovery Interface (ILS-DI) committee will prepare a recommendation with a new interoperability profile, "ILS Basic Discovery Interfaces" or "ILS-BDI", that includes the functions above, along with specifications of the proposed technologies (or "bindings", in the language of the recommendation).

ILS and application developers and vendors will support the ILS-BDI using the recommended bindings in future products.

The DLF will publicize these recommendations, and encourage further enhancements and cooperation between libraries, vendors, and applications developers in building more advanced, interoperable architectures for bibliographic discovery and use.

We are all committed to providing the best library services for research and learning. The agreement we are making now is an important step in advancing these services for the library users of today and tomorrow.

- Digital Library Federation, March 2008

Signers of the Berkeley Accord as of April 4, 2008:

1. Talis
2. Ex Libris
3. LibLime
4. BiblioCommons
5. SirsiDynix
6. Polaris Library Systems
7. VTLS
8. California Digital Library
9. OCLC
10. AquaBrowser