

# METASEARCH

## Long-term Search Options

### Abstract

Search options for Metasearch services promote efficient multi-target and multi-resource searching to enable fast retrieval of the most meaningful results. This paper introduces the need for Metasearch tools, states the problems that must be overcome, and reviews the strengths and weaknesses of existing solutions with respect to critical search options.

### Introduction

The typical library of today offers access to well over one hundred electronic resources. These resources are provided by a large variety of vendors. Each resource will typically have a proprietary UI and offer access to one database per search. The challenge for libraries is twofold. First and foremost, is to match users (patrons) to these resources so they can get good results in a short period of time with minimal manual (librarian) intervention. Second, they must meet user community expectations for an overall search experience like the popular Web search engines, such as Google, which offer meaningful results for lots of different kinds of content with just one simple search.

Metasearch services have emerged to help libraries meet these needs. The Metasearch tools strive to simulate the Google 'one search' experience by offering a single UI to the end user that masks the variety of vendor specific UI's and databases. Behind the scenes, the Metasearch tool translates the user's search into the idiom of each vendor's resources. These translated queries are then simultaneously broadcast to each resource. Results are received, collated and then transformed from vendor specific formats into a single uniform presentation to the end user.

## Problems

There are a number of intrinsic difficulties with this approach, which makes it difficult to scale:

- **Lack of standards**

With each vendor UI, query language, and results presentation different, the Metasearch tool must develop a transformation profile for each. (The process of parsing a vendor's HTML result list presentation is often referred to as 'screen scraping'.) This isn't insurmountable by itself; however, each vendor modifies and enhances their proprietary interfaces and adds new content types to their databases in regular release cycles. The Metasearch vendor must be constantly vigilant in order to catch modifications, which would in turn break their transformation profiles.

Note that previous attempts at standardization, notably Z39.50 though broadly supported by traditional library aggregation vendors, have not been universally adopted, particularly by newer players such as primary publishers.

- **Search server overload**

If each vendor only had one database to search, the broadcast search of the Metasearch tools would work fine. However, today's electronic resource vendor usually offers several searchable databases that a library subscribes to. Broadcast searches for several of the same vendor's databases are then funneled to the vendor's search servers all at the same time. Typically, each database requires a Login (Authentication and Authorization step) and then builds a user session before the search can proceed. During non-busy times of day or time of the year, this approach provides good response time. However, during busy periods when several end users submit queries to several Metasearch tools which in turn translate them into several vendor specific queries, the number of searches (and corresponding Logins) can multiply to the point where the vendor's servers are temporarily overloaded. The possibility of server overload grows greater with each additional library that adds a Metasearch tool.

- **Information overload**

Most Metasearch tools offer the end user the capability to filter results by broad subject classification and sometimes by other mechanisms such as document type and electronic availability of text. However, the end user is still often left with a pile of similar results, particularly when searching large periodical databases. The proprietary search and results formats make it difficult to organize results from similar databases and remove duplicates.

## Solution Criteria

The solution to these problems would obviously adhere to published industry standards, be parsimonious and friendly wrt to content vendor search server resources, and provide mechanisms to reduce information overload to end-users. In particular and to summarize what was stated above, an effective set of Search Options should provide:

- ✓ Near Stateless access,
- ✓ Lightweight identification, authentication and authorization (e.g., a token),
- ✓ Search in a standard Query syntax,
- ✓ Allow several Resources to be searched by one Request,
- ✓ Preference for Start record and max Records per Request,
- ✓ Preference for Sort order of Result Set,
- ✓ Preference for Record syntax of Result Set, and
- ✓ Error handling and a preferred Diagnostic syntax and level.

## Possible Solutions

Here is a brief survey of existing and evolving solutions with suggestions for next steps:

- **Z39.50**

Z39.50 pre-dates the Web. (See <http://lcweb.loc.gov/z3950/agency/markup/01.html>). Though it is solid in terms of providing standardized access, today the protocol (connection-oriented, session-oriented) is no longer main stream and is seen as an impediment to the entry of new information providers. To a novice, the query and response syntax is both obscure and arcane, and building a reliable non-HTTP based Internet service is a cost that is difficult to justify.

Z39.50 continues to be a viable search and retrieval protocol that is better than ‘screen scraping’. Metasearch engines that already support it should continue to do so, and content vendors that already support it should make new databases available on it.

- **SRW/SRU**

SRW/SRU is very close to the optimal solution. It seeks to recreate many of the search and retrieval standards of Z39.50 by leveraging new Web services (e.g., XML, WSDL, SOAP) standards and technologies.

It offers a nearly stateless API, only introducing state into the protocol where it is critically needed (e.g., authentication). Features include search requests by URL or SOAP/HTTP POST methods, a standard and human readable query language “CQL”, standard result list record syntax with support for several record schemas, a WSDL based

Explain service, record sort options, and diagnostics. See <http://www.loc.gov/z3950agency/zing/srw/>.

The CQL query language (<http://zing.z3950.org/cql/intro.html>) offers a rich set of features including relational, boolean, and proximity operators; field specific search qualifiers; relation modifiers; pattern matching; and word anchoring. These features make it possible for Metasearch tools to implement reliable information filtering queries for targeted resources. Though queries would still need to be tuned to optimize access to each content vendor's databases, with CQL Metasearch vendors could afford to spend more time reducing information overload instead of just getting a 'screen scraping' script to work at all.

One drawback of CQL is its very richness. Supporting all features may be difficult for all content vendors. Work should be undertaken to define a minimum set that can be supported by all.

Another valuable feature of the protocol is the support for an authentication token. It serves much the same purpose as the authentication cookies that Web applications send to browsers, indicating that the request should be considered to be from the same user. Its use by a Metasearch tool could allow all anonymous traffic from a particular library to submit searches with the same authentication token (assuming business models are adjusted).

SRW/SRU does have one serious shortcoming wrt Metasearch tools. While seeking to simplify or eliminate some of the complexities of Z39.50, the notion of searching multiple databases has been dropped. The SRW background paper states, "... the philosophy is that the database distinction is just another search criterion, ..." If the databases were free this would be true, but the reality is that each one has revenue and royalty related business rules associated with it and must therefore be distinguished from the query. As a result, each database must be searched with a separate request, leading to potential server overload.

The potential for overload is, however, greatly reduced in comparison to 'screen scraping' approaches. The heavyweight login, session creation, and application activation are all eliminated. Only the vendor's search servers and Web servers need to be scaled to handle the load.

With support for searching multiple Resources (Databases) at a Target site, a Metasearch tool need only submit one search request via the SRW protocol. The activity of optimizing searching the Resources in the request is, for the most part, transparent to the protocol and left to the content vendor.

Offering a preference option for Complete or Partial results and a Timeout period in the Search request could leave the content vendors some wiggle room. If a search of 10 databases was requested and 7 had returned results by the end of the timeout period (e.g.,

5 seconds), the response could indicate a Partial result and provide a mechanism to get the rest in a subsequent request.

For example, with the addition of a couple new elements, instead of the current single Resource SRW query shown here:

```
<SRW:searchRetrieveRequest
xmlns:SRW="http://www.loc.gov/zing/srw/v1.0/">
  <SRW:query> (bath.title "metasearch engine" prox///5
reviews)</SRW:query>
  <SRW:sortKeys>/record/date,
"http://www.loc.gov/zing/srw/dcschema/v1.0/",0
</SRW:sortKeys>
  <SRW:startRecord>1</SRW:startRecord>
  <SRW:maximumRecords>10</SRW:maximumRecords>
  <SRW:recordSchema>http://www.loc.gov/mods/
</SRW:recordsSchema>
</SRW:searchRetrieveRequest>
```

You could have something like this:

```
<SRW:searchRetrieveRequest
xmlns:SRW="http://www.loc.gov/zing/srw/v1.0/">
  <SRW:searchCollection>
    <SRW:search resource={1st database name or id}>
      <SRW:query> (bath.title "metasearch engine*" prox///5
reviews)</SRW:query>
      <SRW:sortKeys>/record/date,
"http://www.loc.gov/zing/srw/dcschema/v1.0/",0,0,
highValue</SRW:sortKeys>
      <SRW:startRecord>1</SRW:startRecord>
      <SRW:maximumRecords>10</SRW:maximumRecords>
      <SRW:recordSchema>http://www.loc.gov/mods/
</SRW:recordsSchema>
    </SRW:search>
    <SRW:search resource={2nd database name or id}>
      <SRW:query>(dc.title "metasearch engine*" )
</SRW:query>
      <SRW:sortKeys>/record/title,
"http://www.loc.gov/zing/srw/dcschema/v1.0/",1,0,
highValue</SRW:sortKeys>
      <SRW:startRecord>1</SRW:startRecord>
      <SRW:maximumRecords>10</SRW:maximumRecords>
      <SRW:recordSchema>http://www.loc.gov/mods/
</SRW:recordsSchema>
    </SRW:search>
    ...
    ...
  </SRW:searchCollection>
</SRW:searchRetrieveRequest>
```

Notice that the Metasearch engine has conditioned the user's query slightly differently for the Resources identified in the Search. The first database could be periodicals and the second could be comprised of books.

Similarly, the Results from a multi-Resource search could also easily be encapsulated in an SRW search response with the provision of just a couple new elements. For example, instead of the existing single Resource schema shown here:

```
<SRW:searchRetrieveResponse
xmlns:SRW="http://www.loc.gov/zing/srw/v1.0/"
xmlns:DIAG="http://www.loc.gov/zing/srw/v1.0/diagnostic/">
  <SRW:numberOfRecords>2</SRW:numberOfRecords>
  <SRW:resultSetId>8c527d60-c3b4-4cec-alde-
1ff80a5932df</SRW:resultSetId>
  <SRW:resultSetIdleTime>600</SRW:resultSetIdleTime>
  <SRW:records>
  ...
  ...
```

You could have:

```
<SRW:searchRetrieveResponse
xmlns:SRW="http://www.loc.gov/zing/srw/v1.0/"
xmlns:DIAG="http://www.loc.gov/zing/srw/v1.0/diagnostic/">
  <SRW:responseCollection>
    <SRW:response resource={1st database name or id}>
      <SRW:numberOfRecords>2</SRW:numberOfRecords>
      <SRW:resultSetId>8c527d60-c3b4-4cec-alde-
1ff80a5932df</SRW:resultSetId>
      <SRW:resultSetIdleTime>600</SRW:resultSetIdleTime>
      <SRW:records>
      ...
      ...
    </SRW:response>
    <SRW:response resource={2nd database name or id}>
      <SRW:numberOfRecords>2</SRW:numberOfRecords>
      <SRW:resultSetId>8c527d60-c3b4-4cec-alde-
1ff80a5932df</SRW:resultSetId>
      <SRW:resultSetIdleTime>600</SRW:resultSetIdleTime>
      <SRW:records>
      ...
      ...
    </SRW:response>
    ...
  </SRW:responseCollection>
```

Subsequent requests for ‘More’ records from a particular Resource could also be readily supported by referencing the <SRW:resultSetId> element.

The SRW WSDL Explain service would also need to be modified to handle multi-Resource search requests.

SRU could remain single Resource API.

- Proprietary APIs

Several content vendors have created proprietary APIs that, like SRU, allow a search to be submitted as a URL via HTTP and the response returned in XML format. Though not standardized, from the content vendor's perspective a proprietary API can be a lightweight investment since it likely uses the same query syntax and leverages services already built for proprietary end-user search applications. It can also be optimized to best fit the vendor's existing search and retrieval infrastructure.

Gale's HITS API (<http://www.gale.com/metasearch/index.htm>) and Books24x7.com's XML Gateway (<http://www.exlibris-usa.com/news1.asp?categoryId=155&admin=>) are examples of this approach. Both offer structured access to most of the features and functions of their existing end-user applications.

From a Metasearch provider's perspective, scripting access to these APIs should be less difficult and much more reliable and stable than the 'screen scraping' approach. However, it still means custom programming for each target site and non-uniform responses, which make it challenging to remove duplicates and properly rank results from multiple targets.

Content vendors could also be expected to evolve these APIs to accept search requests for multiple Resources and to offer duplicate item detection and removal services as the need arises.