World Library and Information Congress:
71th IFLA General Conference and Council

"Libraries - A voyage of discovery"

August 14th - 18th 2005, Oslo, Norway

Conference Programme:
http://www.ifla.org/IV/ifla71/Programme.htm

June 8, 2005

Code Number: **065-E**
Meeting: **121 UNIMARC with Information Technology**

Delivering MARC/XML records from the Library of Congress catalogue using the open protocols SRW/U and Z39.50

Mike Taylor
Index Data, London, UK

Adam Dickmeiss
Index Data, Copenhagen, Denmark

Abstract

The MARC standard for representing catalogue records and the Z39.50 standard for locating and retrieving them have facilitated interoperability in the library domain for more than a decade. With the increasing ubiquity of XML, these standards are being superseded by MARCXML and MarcXchange for record representation and SRW/U for searching and retrieval. Service providers moving from the older standards to the newer generally need to support both old and new forms during the transition period. YAZ Proxy uses a novel approach to provide SRW/MARCXML access to the Library of Congress catalogue, by translating requests into Z39.50 and querying the older system directly. As a fringe benefit, it also greatly accelerates Z39.50 access.

Standards for representation and searching of catalogue records

The advantages of standards for representing catalogue records have been well understood for some time. The universal adoption of the MARC family of standards (Library of Congress (2005)) and the ISO 2709 syntax (ISO (1996)) has enabled a degree of co-operation and collaboration between libraries that we now take for granted, but which is undreamed of in other fields of endeavour such as collection management in museums.

What has not been so widely recognised is the value of standards for *searching* catalogues and retrieving catalogue records over networks. The key standard in this area over the last decade has been ANSI/NISO Z39.50 (ANSI/NISO (2003)). First approved in 1992 and expanded in 1995, Z39.50 was ratified as international standard ISO 23950 in 1998 and reaffirmed by NISO in 2003. The data structures transmitted in Z39.50 sessions are described by ASN.1 and encoded using BER – technologies which provide a compact, efficient encoding, but which are considered outdated in some circles. The protocol provides a set of powerful operations that bibliographic clients can use to to search in, and retrieve catalogue records from, remote repositories. Z39.50 servers have been implemented by most major libraries including the Library of Congress and the British Library, enabling users around the world to benefit from their cataloguing efforts. While Z39.50 has also been used in many other application domains, library catalogues have remained very much its “home turf”.

The combination of MARC for representing catalogue records and Z39.50 for locating and obtaining them enables a wide range of bibliographic applications to be built achieving levels of semantic coherence not possible in most other application domains. The free availability of Z39.50 programming toolkits such as YAZ (Index Data (1995)) and client API specifications such as ZOOM (Taylor et al. (2004)) has nurtured a dynamic community of Z39.50 application implementers.

XML standards for representation and searching

With the increasing ubiquity of XML as a canonical data-interchange metaformat, there is a perceived need to re-cast the battle-proven MARC formats in terms of XML. The first such effort to gain widespread acceptance was the Library of Congress's MARCXML (Library of Congress (2003)), an XML Schema designed to represent MARC21 records. Because this format is limited to the American version of MARC, and is unable to represent the various national formats, the International Standardisation Committee for Information and Documentation, ISO TC46 decided in May 2003 to prepare a generalisation of MARCXML able to represent all the historic MARC formats. This more general initiative, MarcXchange (ISO (2005)), has the desirable property that it is a superset of MARCXML: every valid MARCXML document is also a valid MarcXchange document with the same meaning. The MarcXchange specifications are currently at the NWI (New Work Item) stage of ISO certification, with the current draft having been recommended to proceed straight to voting for adoption as a Draft International Standard.

Along with this shift from historic binary formats towards XML in the representation of catalogue records has come an analogous shift from binary protocols towards XML-based transport for these records. The use of ASN.1 and BER in Z39.50 is widely perceived – whether rightly or not – as mounting a barrier to implementation. In response to this perception, work began in December 2000 on recasting the powerful and expressive Z39.50 semantics in terms of mechanisms more readily understood in the contemporary information environment. The result of this effort is a family of two new protocols: SRW (the Search/Retrieve Web-service) and SRU (Search/Retrieve URL). Their specifications are administered by the SRW Editorial Board, which is made up eleven members from institutions including commercial companies, libraries and universities.

SRW (SRW Editorial Board (2004a)) uses SOAP to deliver structured query payloads from client to server and to return responses including zero or more records from server to client. The requests and responses are both expressed in XML. Several operations are defined, each consisting of a request-response pair. These including "searchRetrieve", "explain" (in which a client asks a server to describe its capabilities) and "scan" (for browsing index entries). The specifications for a related Update operation are under development and will be released during 2005.

SRU (SRW Editorial Board (2004b)) is semantically equivalent to SRW, but uses a simpler, REST-like, mechanism as its transport. SRU requests are expressed as URLs with query parameters that carry information equivalent to that in the corresponding SRW-request XML documents. SRU response payloads are identical to those of SRW, but are returned directly as the content of the HTTP response rather than being wrapped in a SOAP envelope as in SRW.

Note that SRW and SRU are both based on HTTP, and can be legitimately seen as profiles of that protocol. They represent a refinement of, and greater precision in, what information is transmitted over HTTP and how that information is interpreted. Thus, while HTTP is not itself suitable for IR, it is a very suitable substrate for a protocol that facilitates rich IR.

Although SRW was initially seen as the more suitable vehicle for serious IR usage, uptake of SRU has been quicker and broader. SRU implementations seem in general to outperform related SRW implementations due to the additional XML-parsing overhead in the SOAP-based protocol. Since they are, however, in all respects semantically equivalent, we use the term "SRW" hereafter to refer to both protocols in the SRW/U family, except where it is clear from the context that the more specific interpretation is intended.

A very important aspect of structured search-and-retrieve protocols such as Z39.50 and the SRW/U family is the provision of a structured query language in which rich queries can by expressed. Z39.50 uses an esoteric binary query format known as the "Type-1 query", which need not concern us here. SRW/U improves on its predecessor by using a textual query format which is at once intuitively understandable and powerfully expressive - capable of representing all the queries that can be expressed in Z39.50's Type-1 notation. This format is known as CQL (SRW Editorial Board (2004c)), the

Common Query Language, and is exemplified by queries such as the following:

```
dinosaur
title=dinosaur
title=(dinosaur or pterosaur) and author=martill
dc.title=*saur and dc.author=martill
title exact "the complete dinosaur" and date < 2000
name=/phonetic "smith"
fish prox/distance<3/unit=sentence frog
```

This query language allows substantially greater precision than the more traditional type of IR query consisting only of a “bag of words” not related in any specific way or tied at any particular part of the documents being searched.

Three approaches to providing XML access

Whenever new technologies are introduced to replace older ones, the chicken-and-egg problem of adoption arises. No-one wants to abandon the old standard in favour of the new until the community as a whole supports the new standard; but the community will never support it until individual implementers do. Thus, inevitably, there is always a period of overlap during which old and new standards co-exist. The challenge for service providers is to assist users through the period of transition with minimal disruption, which inevitably means providing both services simultaneously. This is the scenario that the Library of Congress faced when the decision was taken in 2003 to deliver MARCXML records over SRW as well as MARC21 records over Z39.50. There are three approaches to solving this problem.

The first approach is to run two essentially separate services, driven from two separate databases using different software to provide the differently formatted records via the different protocols. While the two-databases-two-services approach has the virtue of conceptual simplicity, it is beset by operational difficulties: the two databases need to be kept up to date with respect to each other, which requires that conversion be possible in at least one direction. In general, duplication of data is to be avoided for several reason, including the waste of resources, the the increased maintenance burden and the potential for discrepancy between the versions.

A better approach, where possible, is to run both services from a single master database. The master database may hold the records in either of the formats to be supported (e.g. MARC21 and MARCXML) provided that on-the-fly conversion is possible to the other format; alternatively, the records may be maintained in a completely different form – e.g. as rows in a relational database table – and translated on the fly into both of the formats to be delivered. This one-database-two-services model reduces redundancy, potential for error and redundancy.

The one-database-two-services model works well when there is full access to all components of the system, so that both services can obtain records from the master database. However, in many real-world situations, this is not the case. For example, the Library of Congress's existing Z39.50 server was part of an intergrated library system provided by Endeavor Information Systems Inc. The bibliographic database at the heart of this system is not generally accessible through an API, so building an SRW server to provide MARCXML version of its records was not possible. It was this problem that led us to the third possible approach – a gateway that acts as an SRW/MARCXML server and fulfils the requests made of it by also acting as a Z39.50/MARC21 client. Once this scheme is arrived at, several advantages are immediately apparent:

- No duplication of data is necessary.
- No co-operation at all is required from the existing back-end installation.
- The use of the Z39.50 and MARC21 standards at the back-end of the SRW/MARCXML server means that the same SRW server can be used to front any compliant Z39.50/MARC21 server.

This last point bears some exposition, as it is responsible for much of the power of this approach. An SRW-to-Z39.50 gateway working atop a pre-existing Z39.50 server can be considered as a special case of the one-database-two-services model, with the SRW server's API to the database being Z39.50 itself. Using that model in its native form, each SRW server implementation would require a new back-end, depending on what kind of database system is responsible for the master records. But by using the standardised Z39.50 protocol itself as its back-end, the gateway sidesteps this multiplicity of implementations and gains leverage from the widespread adoption of the very protocol it intends to replace.

It is this third approach that the Library of Congress chose to adopt, contracting Index Data to provide an SRW-to-Z39.50 gateway to sit in front of their existing Endeavor Z39.50 server. In accordance with our standard business practice, we at Index Data released the completed gateway under an open source licence (the GNU GPL), and it is now freely available to anyone who wishes to use it. We provide commercial support where this is desired.

The SRW gateway is a somewhat novel application of Z39.50, as the protocol's primary intended use is in communication between remote clients and servers. In the gateway approach, instead, the Z39.50 client and server typically reside on the same machine, or at least on “nearby” machines that are on the same local area network: the protocol is used purely as an API. Network protocols such as Z39.50 are superior to conventional linking APIs, however, in that they do not dictate the physical configuration of the software components. For example, although it is conventional for the gateway to be close to the Z39.50 server, it need not be so: it is perfectly possible for a third party in Denmark to host an SRW-to-Z39.50 gateway that allows SRW clients in England to access a Z39.50 server in America.

It might be expected that the extra stage introduced into the SRW service process by the gateway would result in a degradation in performance. In practice, the opposite is the case, due to performance-enhancement features discussed below.

The YAZ Proxy

Index Data's SRW-to-Z39.50 gateway is named “YAZ Proxy” (Index Data (2004)) because it also functions as a Z39.50 protocol – that is, Z39.50-to-Z39.50 gateway – a role that was as important to the Library of Congress in their original deployment as the SRW gateway role. Why is it useful to have a Z39.50 proxy? For the same reasons that HTTP proxies such as Squid are useful: a proxy can reduce load on a server, improve client performance through caching and the re-use of already open back-end sessions, protect the back-end by sanitising client requests, and balance load over multiple back-end servers. The YAZ Proxy installation at the Library of Congress in fact does all these things, and so provides a significantly improved Z39.50 service to clients. Crucially, it also delivers MARCXML records via Z39.50, translating the back-end's MARC21 records on the fly. In this paper, however, we will concentrate on the SRW functionality that it provides (which includes support for both SRW in the restricted sense and SRU).

The principle of the YAZ proxy's SRW service is simple: for each SRW search-request it receives from the client, it performs the following steps:

- receives the SRW request, including a CQL query;
- translates that CQL query into the equivalent Z39.50 Type-1 query;
- embeds the translated query in a Z39.50 search-request;
- sends it to the back-end server;
- awaits the response;
- extracts the MARC records from the response;
- converts them into MARCXML;
- embeds them in an SRW search-response; and finally
- sends the search-response back to the client.

From the client's perspective, this is a perfectly normal SRW request-response pair, and from the server's perspective a normal Z39.50 request-response. As is so often the case, however, the devil is in the details: although SRW was designed as a direct equivalent of Z39.50, there is a certain amount of “impedance mismatch” between the protocols which the gateway has to absorb. We discuss some of these matters in the following sections.

- Session re-use – For some Z39.50 back-ends, including Endeavor's Voyager server, session initialisation is a very high-overhead operation, often taking an order of magnitude longer than actual searching. For long sessions, this overhead is amortised over many searches and so is not crippling, but typical usage patterns show that many sessions consist of a single search-and-retrieve operation, which therefore takes ten times as long as it ought. To ameliorate this problem, YAZ Proxy caches initialised sessions with the back-end server, and re-uses them for multiple clients. The details of how this is done are configurable: it is possible to specify the maximum amount of time that any back-end session may live, the maximum amount of data that may be sent over it, and the maximum number of requests that may be satisfied by it. These limits on session-length serve to mitigate the difficulties suffered by back-ends that leak resources during the course of long sessions. The gateway may in addition be configured to pre-initialise a pool of back-end connections at start-up time. When a client that has already used the gateway subsequently re-connects, it is matched up with the same back-end session it used before, provided it is still available. This enables better re-use of result-set caching in back-ends that support it.

- Query caching – Simple stateless clients often send identical searches in a relatively short period of time (e.g. in order to produce a results-list page, the next page, a single full-record, etc.) For many back-ends, it's much more expensive to produce a new result set than to reuse an existing one. The gateway accelerates such repeated searches by remembering the last query sent to the back-end, so that if an identical query is received next, it is turned into Z39.50 Present requests (which refer to the existing result set) rather than Search requests (which create new result sets).

- Query translation – The most complex and configurable part of the gateway's task is translating the CQL queries that come from its clients into equivalent Z39.50 Type-1 queries to be sent to the back-end. Part of this complexity arises because both of these query languages are themselves configurable through the use of "context sets" (CQL queries) and "attribute sets" (Type-1 queries). These may be defined by specific communities to enable queries to express domain-specific concepts such as "find weather stations between longitudes 79 and 82 degrees east". In general, then, mapping from CQL to an equivalent Type-1 query requires knowledge not only of the languages' syntax but of the particular context sets in use and the equivalent attribute sets. This knowledge, rather than being hard-wired into the gateway, is read from a dedicated configuration file which contains information about which Type-1 attributes should be used to express each CQL index, relation, wild-card pattern, etc.

- Load balancing – The gateway may be configured to associate multiple back-end Z39.50 servers with a single front-end SRW server that it provides. In this case, incoming client sessions are routed to whichever of the back-ends is currently serving the fewest initialised sessions. When this facility is used, it is the responsibility of the back-ends' administrator to ensure that they are kept synchronised.

- Record translation – The gateway includes native code to translate from MARC21 into MARCXML. In addition, it can be configured to use XSLT style-sheets to further transform the MARCXML documents so derived into other XML formats, including Dublin Core, MODS and METS.

- Record caching – The gateway may cache records retrieved for a search, and return its local copy if the same record is requested multiple times. This pattern of usage is characteristic for some classes of SRW client, but not all. The gateway may also be configured to pre-cache records at the time of the initial search, in anticipation of subsequent Retrieve requests.

- Logging – Logs can be generated that include either or both of requests made by the client and to the back-end server; these may be logged either in detail or in summary.

Apart from the specialised information in the CQL-to-Type-1 translation file discussed above, all YAZ Proxy configuration is specified in a single configuration file, expressed in XML; XInclude is used to allow subsidiary configuration files to be included if desired. The settings specified in the configuration file may be overridden by command-line arguments specified a run-time.

We include a short example configuration file:

```
<?xml version="1.0"?>
<proxy xmlns="http://indexdata.dk/yazproxy/schema/0.9/"
 xmlns:xi="http://www.w3.org/2001/XInclude"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://indexdata.dk/yazproxy/schema/0.9/yazproxy.xsd">
  <target name="localhost" default="1">
    <url>localhost:9999</url>
    <target-timeout>30</target-timeout>
    <client-timeout>60</client-timeout>
    <keepalive>
      <bandwidth>1000000</bandwidth>
      <pdu>1000</pdu>
    </keepalive>
    <limit><!-- per minute limits .. -->
      <bandwidth>2000000</bandwidth>
      <pdu>50</pdu>
      <retrieve>100</retrieve>
    </limit>
    <attribute type="1" value="1-11,13-1016"/>
    <attribute type="1" value="*" error="114"/>
    <syntax type="opac"/>
    <syntax type="usmarc"/>
    <syntax type="none"/>
    <syntax type="xml" marcxml="1"/>
    <syntax type="*" error="238"/>
    <preinit>0</preinit>
    <xi:include href="explain.xml"/>
    <cql2rpn>pqf.properties</cql2rpn>
    <query-charset>ISO-8859-1</query-charset>
  </target>
  <target name="*">
    <target-timeout>60</target-timeout>
    <client-timeout>30</client-timeout>
  </target>
  <max-clients>50</max-clients>
  <log>client-requests server-requests</log>
</proxy>
```

Conclusion

A novel use of the standard Z39.50 protocol enables YAZ Proxy to provide SRW access to the contents of Z39.50 servers. Configuration consists of writing a single XML file describing the operational parameters and the desired performance optimisations, and an additional file that specifies how CQL is translated into Z39.50 Type-1 queries. The gateway can be used to front any Z39.50 server. There are several installations in use worldwide, of which the highest profile is that of the Library of Congress. In addition to providing a way to obtain MARCXML records from the Library's catalogue of MARC21 records via SRW, the gateway also doubles as an optimising Z39.50 proxy that resolves several long-standing problems with the native Z39.50 server. YAZ Proxy is freely available under the GNU GPL, and is fully supported.

References

ANSI/NISO. 2003. Information Retrieval (Z39.50): Application Service Definition and Protocol Specification. http://www.niso.org/standards/resources/Z39-50-2003.pdf

ISO (International Standards Organization). 1996. ISO 2709: Information and documentation: format for information interchange. http://www.iso.org/cate/d7675.html

ISO (International Standards Organization). 2005. MarcXchange. http://www.niso.org/international/SC4/n569.pdf

Index Data. 1995. YAZ - Yet Another Z39.50 toolkit. http://indexdata.com/yaz/

Index Data. 2004. YAZ Proxy. http://indexdata.com/yazproxy/

Library of Congress. 2003. MARCXML. http://www.loc.gov/standards/marcxml/

Library of Congress. 2005. MARC Standards. http://www.loc.gov/marc/

SRW Editorial Board. 2004a. SRW - Search/Retrieve Web-service. http://www.loc.gov/z3950/agency/zing/srw/

SRW Editorial Board. 2004b. SRU - Search/Retrieve URL. http://www.loc.gov/z3950/agency/zing/srw/sru.html

SRW Editorial Board. 2004c. CQL - Common Query Language. http://www.loc.gov/z3950/agency/zing/cql/

Taylor, M., S. Hammer, A. Sanders, A. Dickmeiss, R. Sanderson, and A. Lav. 2004. ZOOM: The Z39.50 Object-Orientation Model, v1.4. http://zoom.z3950.org/