Interoperability and the Search/Retrieve via URL Protocol

T. Michael Silver
LIS 600 – Capping Exercise
Winter 2010

Originally submitted:
August 31, 2009

For:
Dr. Ali Shiri
LIS 599: Directed Study in Interoperability
School of Library and Information Studies
University of Alberta
Edmonton, Alberta

Abstract

Explores the Search / Retrieve via URL (SRU) protocol, tracing its roots in a Z39.50, a previous information retrieval protocol to its use in present day applications. Provides a high-level overview of the three SRU operations: `searchRetrieve`, `scan` and `explain`. Touches briefly on some related standards such as CQL, Z39.91 and Z39.92. Reports on the creation of a test system using a Linux distribution in a virtual machine with the YAZ proxy software from Index Data. Provides a description and example configuration for the YAZ proxy software. Includes data and program files which may be used to experiment with the test server, including VMware Player, Mercury Z39.50 Client, and a rudimentary web interface.

keywords: interoperability, SRU, Z39.50, information retrieval, standards

Table of Contents

Introduction

The Internet has wrought many changes in how the world operates, affecting almost all areas of human activities. Although these changes include entire new ways of communicating and conducting business, nowhere are the changes more evident than in the way individuals and organizations access information. When the author began his first university degree, accessing scholarly information involved a trip to the university library, searches through physical card catalogues and journal indices, obtaining physical items and manually taking notes. The process has changed significantly in the intervening twenty-nine years. Physical card catalogues are a thing of the past and most journals are indexed, searched and accessed using digital tools. Digital formats for both metadata and actual items have enabled users to access information and have provided institutions with a new way to make their information available to other institutions and users.

The problem facing users and the challenge facing the institutions is that significant amounts of information are stored in silos - single purpose databases and repositories which were not designed to communicate with other systems. Interoperability aims to allow disparate systems to interact without requiring special knowledge of remote systems. A variety of standards have been developed by working groups consisting of representatives from a variety of public, private and commercial organizations. This paper discusses interoperability within the context of a specific protocol.

The Search / Retrieve via URL (SRU) protocol which is the focus of this paper was developed as a way of increasing the level of compatibility between library systems and other information sources. It builds on lessons learned in the use of previous protocols, adding

compatibility with current web standards. The goal is easier integration of information sources between libraries and digital information sources available on the Internet.

History and Literature Review

This section presents the history of information retrieval standards, the development of SRU and the current literature on the SRU protocol.

MARC and Z39.50

Information protocols and standards for libraries have existed for decades. Arguably the most used standard for information interchange in the library community is the MAchine Readable Cataloguing (MARC) record which began as a pilot project by the Library of Congress in 1966 (Avrams, 1975). This standard has developed over the years, and currently is used in a variety of national and international formats. MARC records revolutionized the way libraries exchanged information, allowing catalogue records to be re-used without requiring each library to manually enter the data from each record into their own systems.

As computers became more commonplace, the integrated library system (ILS) became a fact of life in many libraries. Large academic libraries adopted them first, followed by larger public institutions. Today, almost all libraries in Canada and the United States operate some form of ILS, most of which provide networked access through the Internet. The network availability raised the issue of increased access to information at other libraries, both for patrons and for activities such as Inter-Library Loans (ILL).

The National Information Standards Organization in the United States established a working committee in 1979 to develop an information retrieval protocol. Z39.50-1988 was the

result of that group's efforts. Since that time, Z39.50 has been revised numerous times to address short-comings and add new features, with the most recent version published in 2003. The Z39.50 protocol gave users have the ability to search multiple databases using the familiar interfaces available in their local software or online public access catalogue (OPAC). This protocol enables verification of bibliographic information, ILL and other information access related activities. Some federated search sites such as CRCnetBase and union catalogues such as TAL Online leverage the Z39.50 protocol to provide a single display of resources available from multiple institutions.

Z39.50 is not without criticisms. It uses protocol-specific methods for both communication and data exchange, forming an obstacle to implementation for many organizations. Although both the protocol and record formats are standardized, they are not widely used outside the library community. In addition to the unique nature of the software and interchange formats used, the networking ports used by the protocol are often blocked by municipal or corporate security policies. The protocol provides a wide range of configuration options for both servers and clients. The concomitant complexity raises further obstacles as both the server and client configurations must complement each other for full functionality. A mismatch in configuration can result in anything from unpredictable or incomplete results to a total lack of functionality. The Z39.50 protocol did provide for an explain operation geared towards assisting with configuration issues. However, the operation was never widely adopted due to its complexity and the lack of a common data exchange format.

SRU

The SRU protocol grew out of the Z39.50 protocol. Implementers desired a protocol that

would address a number of issues with the existing protocol. The two primary goals were to use standard Internet protocols and communication formats for information interchange, thereby removing obstacles to implementation by information providers outside the traditional library community. The World Wide Web communicates using hypertext transport protocol (HTTP) and HTTP Secure (HTTPS) for communication. Adopting these protocols for communication removes the need for implementing specialized protocols simply to successfully connect with remote systems and removes the challenges of adding unusual communication ports to local security policies. Extensible Markup Language (XML) has quickly evolved into a widely used information interchange format, and the SRU developers adopted it as the basis for information exchange. XML has the added advantage of being easily adapted for use with a variety of types of information.

A technical committee was formed in 2001 to develop the new protocol. Because the changes in the information retrieval protocol were designed to allow greater integration on the web, the initial name for the protocol was Search / Retrieve Web Service using the initialization SRW. The terms SRW and SRU were used to differentiate between the methods available for web based communication. SRW communication using SOAP[1] based access, while SRU uses the Representational State Transfer (REST) approach. The actual protocol operation is the same regardless of the communication method, and the current version of the protocol uses SRU to refer to both methods. The literature continues to contain references to both SRW and SRU.

The first version of the protocol was released in 2002, version 1.1 was released in 2004 and version 1.2 was released in 2007. Version 2.0 is currently being developed, with the latest draft released in July 2009 for comment by the community (Denenberg, 2009b). This work is

[1] SOAP originally stood for Simple Object Access Protocol, but the acronym was dropped in version 1.2, originally published in June 2003.

being carried out by the Search Web Services Technical Committee of the Organization for the Advancement of Structured Information Standards (OASIS).

Literature Review

The available literature regarding the SRU protocol falls into broad categories. The largest group is descriptive in nature, either providing an introduction to the protocol or comparing it to other search protocols. A second group of articles reports on development activities, while a third group provides information on specific implementations. The literature dealing with SRU does so within discrete contexts, reporting on activities or projects rather that subjecting the protocol to qualitative or quantitative research.

Morgan (2004) provides an introduction to the protocol which is widely cited. Morgan opens by discussing the challenges the protocol is intended to address, and continues to describe the three basic operations of the protocol, complete with sample responses. He also provides a section on a sample application used to list journals in a repository. The final section of his article discusses how SRU and the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) complement each other, a theme further expanded upon by Sanderson, Young and LeVan (2005). Morgan concludes his article with the statement “If index providers were to expose their services via [SRU], then the content of the 'hidden Web' would become more accessible and there would be less of a need to constantly re-invent the interfaces to these indexes” (summary). While it seems evident that such a standardization would be in the best interests of users of these indexes, research should verify this assumption given the resources necessary to implement SRU interfaces to large vendor databases.

Reiss (2007) examines the role SRU could play in metasearch, or “one-search access to

multiple electronic resources" (Reitz, 2007) After providing an introduction to the protocol and placing it within the context of the "emerging bibliographic infrastructure" (p. 374), he describes hypothetical systems which could search bibliographic, archival and current events information on the Internet through a single SRU server implementation. Other protocols which may compete or complement the SRU protocol are briefly discussed after a review of three projects relating to metasearch. He concludes by stating "SRU is an important piece of the entire upgrade that the library community needs to undergo in order to modernize our bibliographic infrastructure in order to improve and expand the search and retrieval services that libraries need to provide our users" (p. 384). Much of the literature on SRU showcases two projects, one being the TEL project (described below), both of which are described in Reiss' article. The constant citation of these projects makes one wonder what other projects are using the protocol and why these two specific projects are so frequently mentioned.

Taylor and Dickmeiss (2006) report on the implementation of an SRU service at the Library of Congress as a front-end to their traditional Z39.50 search service. Their presentation deals mainly with the delivery of bibliographic records in an XML format, demonstrating the advantages of using open standards for the interchange of data. They report ancillary benefits as the implementation of the new service also improved the performance of the library's Z39.50 service. The authors discuss three different approaches to providing XML access to the bibliographic data, explaining the advantages and disadvantages as well as the reasoning behind the method chosen at the Library of Congress. Taylor and Dickmeiss are employees of Index Data, the company chosen to implement this solution and developers of well-known open-source software toolkits for Z39.50, metasearch and SRU. In their conclusion, they report that "there are several installations in use worldwide" but fail to provide details. As mentioned above,

information on other installations would be useful in ascertaining the actual adoption of SRU.

Van Veen and Oldroyd (2004) describe the implementation of a “co-operative framework ... for integrated access to the major collections of the European national libraries” (abstract). This project exemplifies many of the advantages of open standards. Commercial portals were considered, but TEL decided that “the SRU protocol proved to be more promising that the commercial portals” (Section 2, para. 8). One of the reasons for their choice was the low cost to entry, while another was the flexibility afforded by the tools. The variety of encoding schemes, languages and metadata sets between the libraries all had to be accommodated. The initiative described in this article also demonstrates the need for further work on interoperability. TEL developed its own registry for metadata activities to track metadata elements in use, under consideration or that have been rejected. This article describes a unique portal solution as the final portal software actually runs as JavaScript in the end user's web browser. Their conclusion is that this solution “offers a number of advantages ...[including]... scalability, functionality, low barrier of entry into TEL, and increased control of functionality for users, data providers and service providers. Last but not least, ... there is no longer a need for a central portal” (conclusion). Of the articles reviewed, this one provides the greatest detail on both the implementation process and the challenges faced when integrating diverse information resources.

Hafezi (2007) proposes a solution for national interoperability in Iranian libraries. Although the article is very hypothetical and exploratory in nature, it provides empirical data on the supported interchange formats in use in Iranian libraries. It discusses the challenges inherent in dealing with multiple languages as Persian, English and Arabic materials are common. The article also discusses the reliance on vendors for adherence to standards. The author concludes that the use of XML and SRU would aid in interoperability, ultimately resulting in lower costs

and more satisfied users (p. 733). The article demonstrates that interoperability issues apply across cultures rather than being a strictly Western concern.

The use of SRU as a data source used in conjunction with unAPI is mentioned briefly in Chudnov et al. (2006), and is described in detail in Binkley (2009). Basically, unAPI "provides the few basic operations necessary to perform simple clipboard-like copy of content objects across all sites" (Chudnov, 2006). Binkley uses it to offer metadata records to unAPI-aware applications from within a search interface, in this case a university library's catalogue. If the user has an unAPI-aware extension loaded in the web browser, a link appears on the item record display. Clicking on the link presents the user with the choice of available record formats. Once the user selects the desired format, the unAPI server retrieves the record from the SRU service and returns the record to the user.

These articles deal primarily with the use of SRU in metasearch, data delivery, and portal applications. The use of SRU in TEL establishes the effectiveness of the protocol in connecting disparate systems, while the unAPI example demonstrates the usefulness of the protocol in providing data within web-based environments. The Iranian example shows the expanding need for interoperable systems around the globe. A protocol which leverages the ubiquity of the HTTP/HTTPS protocol and XML data format greatly increases the likelihood of adoption beyond the library community.

Protocol Operations

The need to conform to web-based communication has shaped the specification. Querying an SRU server involves the same considerations as submitting any web-based form. In its simplest form, a connection URL for an SRU service looks like

`http://hostname/database?version=`*`versionNumber`*`&operation=`*`operation`*. The use of the `database` as part of the URL allows access to multiple databases or record sets, depending on the configuration of the SRU server. In the current version of the protocol, `version` and `operation` are both required. Different operations may require additional parameters, for example a `query` parameter for `searchRetrieve` operations.

The heart of the SRU `searchRetrieve` operation is the Contextual Query Language[2] (CQL), which is discussed in detail in the Related Standards section.

Because the results of all operations are valid XML, web applications can make direct use of the data. An example of this is the OCLC Open SiteSearch Documentation site. The search interface is built using the XML results of the `explain` operation using XSL. Results of searches are likewise displayed using XSL.

Operations

There are three basic operations in SRU:

- `explain`, which provides an XML description of the functionality of the service, including supported access points, record sets and features
- `searchRetrieve`, which performs searches and retrieves records (similar to standard keyword searches and record requests)
- `scan`, which provides a list of available terms in an index (similar to browse lists)

By comparison, the Z39.50 specification lists nine different operation types, many of which are poorly supported by software implementations.

[2] Contextual Query Language is the current basis for CQL. Prior to version 1.2, CQL stood for Common Query Language.

Additional Parameters

In addition to the `version` and `operation` parameters, the specification defines additional request, response, record and result set parameters specific to each operation. The response, record or term, and result set parameters are returned as part of an XML structure in keeping with the use of XML for data exchange.

Request Parameters

As mentioned above, the `searchRetrieve` operation uses requires a `query` parameter. The `scan` operation requires a `scanClause` parameter. The number of available request parameters varies - `explain` has three optional, `scan` has one mandatory and four optional, and `searchRetrieve` has one mandatory and seven optional parameters in version 1.2. These additional parameters control various features of the records returned, such as record schema, number of records and position in the result set. For example, an SRU search `http://z3950.loc.gov:7090/voyager?version=1.1&query=%22tai%20chi%22&maximumRecords=5&operation=searchRetrieve` would search (`operation=searchRetrieve`) the Library of Congress (`http://z3950.loc.gov:7090/voyager`) for items in the default index containing the term "tai chi" (`query="tai chi"`) and return at most five records (`maximumRecords=5`).

Response Parameters

The response, record or term, and result set parameters are provided as part of the XML document returned from an operation request. Many parameters are specific to the type of operation requested. For example, the `explain` operation does not return search results, so it has no record or result set parameters. The `scan` operation returns a list of terms, not records. This

parallels the structural differences in the request parameters.

Software Applications

Most of the existing software applications in the library community build off existing Z39.50 products and services. One of the most popular tools for providing SRU functionality is the YAZ proxy package from Index Data. It provides a middle-tier between clients and servers. In the provision of SRU services, it accepts incoming queries from clients, then connects to a native Z39.50 interface to retrieve information and then serves the responses back to the client using the appropriate XML structure.

The YAZ proxy does not have to operate in such a fashion. The open-source ILS Evergreen provides a native SRU interface. If the library wishes to provide a Z39.50 service, it uses the YAZ software combined with a perl module to provide one which connects to the SRU service (evergreen-admin, 2009).

The majority of server and client applications related to SRU are designed to be integrated within an existing server-based infrastructure. Index Data's YAZ tools provide a range of options, including C++, Java and PHP modules. Mike Taylor provides a number of Java-based tools, and the Cheshire3 project implements an XML-based search engine which emphasizes the use of standards, including SRU.

Options for desktop client software are limited for both Z39.50 and SRU. Most uses of SRU integrate with existing web sites, federated search products or cataloguing tools. For testing purposes, the yaz-client command-line program from Index Data and the Mercury Z39.50 Client (which includes SRU functionality) from Basedow Information Systems were used.

Server-based client systems do exist. As an example, Drupal (www.drupal.org), a popular

open-source content management system, has a module available that integrates Z39.50 and SRU searches into web site content. Built using the YAZ PHP toolkit from Index Data, it significantly reduces the level of skill needed to integrate searches into an existing web site.

Related Standards

The SRU protocol operates within the context of a multitude of other standards. The communication and data formats mentioned above are the basic building blocks of the protocol, but many other information standards affect the protocol's operation and implementation. Given that the whole purpose of the protocol is to facilitate connections between information sources and users, it is not surprising that the protocol has connections to a myriad of other standards. This section describes a few of the relevant standards.

Contextual Query Language (CQL)

Any discussion of SRU is incomplete without the inclusion of CQL, the query language used by SRU. CQL attempts to fill the gap between powerful query languages with a steep learning curve such as SQL or XQuery and the intuitive interfaces such as those Google and other search engines provide.

In its simplest form, a CQL query can be a term search with no other syntax. The responding service will provide a response based on the configured default index. In this manner, it achieves the simplicity and ease of use associated with popular search engines. More complex searches can be built by defining the index to be searched, the relationship between the index and the search term, Boolean and proximity relationships and modifiers as well as sorting modifiers. Taylor (2003) provides "A Gentle Introduction to CQL" which provides examples and details on

the use of the language which are more accessible than the official documentation hosted as part of the Library of Congress' official SRU site. The final search example provided by Taylor demonstrates the power of the language: `dc.author=(kern* or Ritchie) and (bath.title exact "the c programming language" or dc.title=elements prox///4 dc.title=programming) and subject any/relevant "style design analysis"`. Assuming the SRU server supports all of the features and context sets used in the query, it would find records whose author includes either a word beginning with `kern` or the word `ritchie`, which have either the exact title `the c programming language` or a title containing the words `elements` and `programming` within four words of each other, and whose subject is related to at least one of the words `style`, `design` or `analysis`.

That search example also demonstrates why the name of the query language was changed from common to contextual. The use of `bath.title` and `dc.title` are examples of placing search terms within a specific context. These contexts define how searches will be interpreted. In the example search, the `dc` context refers to the Dublin Core contextual set, while `bath` refers to the Bath Profile which has developed for bibliographic searches using Z39.50. Other context sets are listed on the SRU web site at the Library of Congress. While many are connected to bibliographic data sets, some are not. The Music Context Set extends the existing Dublin Core and Bath sets to add music-specific contexts such as arranger, performer and instrumentation. Other context sets exist for such diverse applications as network information in the Network Context Set or card-based games in the Collectable Card Games Context Sets.

XML Schema

Records may also be provided in a number of different ways. Unlike Z39.50, SRU has

standardized data interchange using XML. In itself, XML only defines a method of encoding data without specifying fields or the rules for how those fields will be populated. The record schemas listed on the Library of Congress web site are published and available for use. However, new schemas can be developed and published for local or public use. This extensibility is one of the many advantages the use of XML brings to the SRU protocol.

The web site currently lists twenty-four schema, including the Metadata Object Description Schema (MODS), MARCXML, and Dublin Core. In addition, Denenberg from the Library of Congress has authored an Internet Draft that includes schema for MODS, Metadata Authority Description Schema (MADS), Metadata Encoding and Transmission Standard (METS), MARCXML and the SRU response schema as media types for use in Internet communications (Denenberg, 2009a).

Z39.91/Z39.92

Version 1.2 of the SRU protocol replaced the explain schema used previously with NISO *Z39.92 Information Retrieval Service Description Specification*. That specification also references *Z39.91 Collection Description Specification*. An understanding of the concepts embodied in these specifications is necessary for the effective implementation of the explain operation. Replication of the contents of those specifications within this document is not warranted, as they mainly present the structure and model for collection and service descriptions.

Other Standards

As mentioned above, examples of the interaction between SRU and other metadata standards such as OAI-PMH (Sanderson, LeVan, & Young 2005) and unAPI (Binkley, 2009; Chudnov et al., 2006) are already being presented within the literature. As SRU gains more

widespread acceptance and use, the author expects the number of interactions to increase, both in terms of number of implementations and number of different standards involved.

Project Details

One of the goals of this project was the implementation of an SRU client or server system. Installing and configuring such a system would provide the author with practical experience and knowledge beyond what can be gleaned through reading articles and specifications. Initially, the author hoped to be able to develop a production SRU server for the University of Alberta Libraries. As mentioned above, they have an SRU server that serves a specific function within the library's service model. Because its development was driven by a single specific need, there are changes that could be made to the system to improve its functionality and bring it more into line with organizational standards. The library was approached with this proposal and was genuinely interested and supportive of the project. The lack of suitable staff to supervise such a project within the time frame of the course precluded working on their production environment. Permission was obtained to access the library's Z39.50 server for this project.

The network environment at the University of Alberta presents a number of challenges for server-based applications. Any computer connected to a student network, whether wired or wireless, must first authenticate using a Campus Computing ID (CCID) and password. As well, in-bound traffic to the computer is regulated, rendering the establishment of a sandbox server infeasible. Because the various software packages have a multitude of dependencies, installation on the university's General Purpose Unix (GPU) servers was also not feasible. The author initially used his personal computer as a test environment, but was unable to devise a suitable

method to make it accessible for evaluation.

To address these concerns, the author opted to use a virtual appliance. The virtual appliance uses bridged networking through the host computer. As the virtual appliance is started manually after a network connection is successfully made from the host computer, the authentication issues no longer exist. Connecting to the service using software on the host computer bypasses any restrictions present on the university network. The remainder of this section discusses the project, providing details on the virtual machine set up, configuration and access. Appendix F provides details on the use of the virtual machine and other associated files.

Virtual Machine Selection and Preparation

CentOS 5.2 was installed on a virtual machine running under the VMware Player application. CentOS is a Linux distribution that is binary-compatible with Red Hat Enterprise Linux. Instead of creating a virtual machine from scratch, an existing appliance was downloaded from Linhost (Ventura, 2009). In preparation for the installation, several required packages were installed using the native package manager. The dependencies were:

- `gcc` and `gcc-c++`, compilers for C and C++ programs
- `libstdc++` and `libstdc++-devel`, libraries for C++ programs
- `libxml2` and `libxml2-devel`, libraries for XML support
- `libxslt` and `libxslt2`, libraries for XSL support
- `bison`, a scripting language

The CentOS distribution comes with security features enabled which interferes with communication on certain ports by unknown programs. For a production environment, the

software should be configured to recognize and allow traffic related to your specific services. For testing and experimentation in a non-production environment, the simplest solution is to disable the Security Enhanced Linux (SELinux) program. This can be accomplished a number a ways, but the easiest is to use the `system-config-securitylevel` program to disable the program. This must be done while logged in using the `root` account.

YAZ Software Installation

The next stage was downloading and installing the `yaz`, `yaz++`, and `yaz-proxy` packages from the software page at Index Data's web site. The standard CentOS software repositories do not include these software packages, so they had to be downloaded, configured and compiled on the virtual appliance. This was accomplished using the normal process which includes `configure` (automated detection of system properties and setting of compiler properties), `make` (automated compilation of programs) and `make install` (automated installation of programs, libraries and documentation within the system structure). If any unsatisfied system dependencies exist, error messages will be displayed during the `configure` and `make` steps.

Installation packages are available on the Index Data web site for specific Linux distributions. However, compiling from source was necessary in this case because packages did not exist for either CentOS or Red Hat. Additionally, compilation from source is an option on most operating systems, and the author felt the exercise was worth the additional steps.

Client Software for Testing

Three different software clients were used to connect to Z39.50 and SRU

implementations during this project. The command line tool `yaz-client` was used to test basic connectivity. The Mercury Z39.50 Client was used to test searches. Finally, a web browser was used to test URL-based access to the server.

yaz-client

Designed for use as a command-line client, the `yaz-client` program provides a range of options for testing. It supports both Z39.50 and SRU searches, and allows the user to make quick changes to connection strings and queries while receiving immediate feedback.

The `yaz-client` has numerous options, but the simplest form of starting the program is `yaz-client` *`target`*, where *`target`* is the connection information for the database. By default, the target is interpreted as Z39.50 server, but if the connection information includes the http protocol information, it switches to SRU. The command `yaz-client z3950.loc.gov:7090/voyager` connects to the Library of Congress Z39.50 server, but `yaz-client http://z3950.loc.gov:7090/voyager` connects to the SRU server. In this case, both services are being provided by the same program on a single port.

Conducting a simple search involves issuing a `find` command. Retrieving records is accomplished using the `show` command. Examples of both Z39.50 and SRU searches are shown in the Appendix A and Appendix B respectively. These code listings also show the difference between PQF and CQL queries.

Mercury Z39.50/SRU Client

Basedow Information Systems provides a graphical client program which provides search capability using both Z39.50 and SRU. The client comes pre-configured for a number of institutions around the world, but adding new targets is not difficult. Like `yaz-client`, the way

to define an SRU target is to include the protocol in the Z39.50 URL field. Figures in Appendix C and Appendix D show the configuration for Z39.50 and SRU targets and the results of example searches.

Web Browser

Because SRU is a web-based protocol, a web browser can be used to query SRU targets. In the absence of specifically programmed client routines, the queries can be handled by either constructing the URL by hand or using an HTML form to submit the queries to the server. The author was able to construct a form to query the service, but was unable to develop an interface as polished as the OCLC Open SiteSearch Documentation site. Further study and programming ability is required to accomplish that task.

Nonetheless, the web browser was an extremely useful tool in exploring the SRU service. One of the major benefits of SRU is its ability to be accessed by normal web methods. Both Mercury and the yaz-client are programs designed to operate with these protocols. The ability to successfully query the server using only a simple web form in a standard web browser demonstrates conformance to the HTTP protocol, and the results displayed as XML demonstrate conformance to that ubiquitous data format.

YAZ Configuration

This project uses YAZ proxy to provide an SRU interface by connecting to the University of Alberta's Z39.50 service. As mentioned above, most SRU interfaces to library catalogues are provided by this type of arrangement. Taylor and Dickmeiss (2006) provides details on the Library of Congress implementation which follows the same model.

The YAZ proxy configuration is controlled by a text file named `config.xml`. A

complete example of this file is provided in Appendix E. As indicated by the file extension, XML and associated technologies are used for most of the configuration of the program. The records are formatted using XSL style sheets. The main exception to the use of XML is the `pqf.properties` file which maps CQL and Perfect Query Format (PQF) notation to each other. PQF was introduced as part of the YAZ toolkit, and has been adopted by other Z39.50 tools as an alternative to type-1 or reverse polish notation (RPN) (Taylor & Dickmeiss, 2009).

The YAZ proxy software can be run by providing arguments on the command line. This is useful for preliminary testing, especially when having difficulties with connection information. The syntax from the command line is `yazproxy -t` *`hostname`*`:`*`port/database`* `@:`*`localport`* where *`hostname`*, *`port`* and *`database`* are the connection parameters for the backend server and *`localport`* is the port to be used for the SRU and Z39.50 services. For example, the command `yazproxy -t z39.50.loc.gov:7090:voyager @:210` would create a service on port 210 of the local machine that proxies information from the Z39.50 target located at the Library of Congress. By default, this will only provide a useful connection for Z39.50 as the configuration elements necessary to create an SRU service are missing from the command.

config.xml

In its simplest form, creating a target in the `config.xml` file involves providing connection information, most notably the host, port and database names for the target. However, there are a number of features that may be defined which extend the functionality of the service. The full details are given in the *YAZ Proxy User's Guide and Reference* but the following elements are of particular importance.

Proxy Configuration Header

`Config.xml` is an XML file, and as such must be well-formed. It must start with the XML declaration and must have a single root element. In this case, the root element is the proxy element.

```
<?xml version="1.0" encoding="UTF-8"?>
<proxy xmlns="http://indexdata.dk/yazproxy/schema/0.9/">
     <!-- remainder of configuration goes here -->
</proxy>
```

Target Element

The `target` element defines one target. A YAZ proxy server can proxy for multiple targets, so the `target` element may be repeated. All configuration elements for a particular target must be contained within the same `target` element. The `name`, `database` and `default` attributes define the name which will be used in the connection URL, the database used to connect to the backend target, and whether the target is the default for searches if a name or database is not explicitly defined by clients.

The `url` element defines the connection for the proxied target.

```
<target name="neos" default="1" database="Unicorn">
     <!-- remainder of target configuration goes here -->
     <url>ualapp.library.ualberta.ca:2200</url>
     ...
</target>
```

Attribute Element

The `attribute` element specifies valid combinations of attribute pairs, based on the Z39.50 connection attributes for use, relation, position, structure, truncation and completeness. This element can repeat. It can also be omitted if the backend target handles rejection of unsupported attribute types. It can also be used to supply an error message for unsupported attributes, which can be useful if the backend server does not gracefully handle unsupported

types.

```
<attribute type="1" value="1-1016" />
<attribute type="1" value="*" error="114" />
```

Syntax Element

The `syntax` element specifies valid record syntax requests from a client. This element may provide, among other things, information on schema, designate error messages in a similar fashion to the attribute element and specify style sheet information for XSL transformation.

```
<syntax type="usmarc" />
<syntax type="xml" marcxml="1"
     identifier="info:srw/schema/1/marcxml-v1.1">
     <name>marcxml</name>
<syntax>
<syntax type="xml" marcxml="1"
     identifier="info:srw/schema/1/dc-v1.1"
     stylesheet="/usr/local/share/yazproxy/MARC21slim2SRWDC.xsl">
     <name>marcxml</name>
<syntax>
<syntax type="*" error="238" />
```

Cql2rpn Element

The `cql2rpn` element defines the location of a file which provides the details of CQL to RPN conversion. This is required for SRU searches to operate with backend Z39.50 servers that do not natively support CQL queries. Most Z39.50 servers require this element to function properly. The *YAZ User's Guide and Reference* discusses this in detail, but the `pqf.properties` file provided with YAZ proxy is generally sufficient.

```
<cql2rpn>/usr/local/share/yazproxy/pqf.properties</cql2rpn>
```

Explain Element

The `explain` element provides the response to explain operation requests from clients. In a perfect world, the software would automatically provide the explain response based on the software configuration. However, this is not feasible for most current implementations. Usually, YAZ proxy or similar software is used to connect to an existing Z39.50 server. As the proxy has

no way of knowing how the remote server is configured, it cannot provide the configuration and feature details. The server administrator must manually configure this response.

The explain record can be as detailed as the administrator chooses. It can include the bare minimum, such as connection information, or can include a wealth of information about the context sets, record schema and indexes provided by the service. A more complete example of an explain record is included as part of the `config.xml` example in the appendices.

```
<explain xmlns="http://explain.z3950.org/dtd/2.0/">
     <serverInfo>
          <host>192.168.213.128</host>
          <port>9000</port>
          <database>neos</neos>
     </serverInfo>
</explain>
```

Conclusion

Z39.50 is an undeniably complex protocol designed for and used by a very specialized community. Like the traditional MARC record, it works well in the library context, but fails to forge connections between the library community and other information resources. In the perspective of the Internet, it isolates large amounts of data by presenting obstacles to access which most individuals and organizations are unable or unwilling to overcome. The development of the SRU protocol addresses many of these obstacles by implementing widely adopted standards for communication and data exchange. In addition, it offers a standardized query language that allows for the simplicity of search-engine style queries while providing the power and sophistication desired by advanced searchers.

On the other hand, SRU relies on organized metadata and a system of context sets which can be shared and mapped. It still borrows heavily on the attributes and structures of the Z39.50 protocol. This is most likely due to the continued use of SRU as an overlay on native Z39.50

interfaces. As more native SRU interfaces are deployed, the dependence on Z39.50 structures should weaken.

The aim of this project was to prepare for research involving the Explain operation and the factors involved in its use. The author's experience with the complexities of Z39.50 configuration made the availability of a detailed description of a service very exciting. After studying the protocol and its implementation in the YAZ proxy software, that excitement has been tempered by reality. While the OCLC Open SiteSearch Documentation interface demonstrates the feasibility of advanced use of the explain operation, the requirement that server administrators manually create and maintain the explain records results in a great disparity in the level of detail available from any specific service. The usefulness of a standard method of acquiring information about a service should not be denigrated, but it is only one step in the process. The explain operation will be most useful when native SRU interfaces to information resources are developed that generate the explain record directly from the service configuration without requiring additional actions on the part of the administrator. The author hopes that his future research into the factors surrounding the use of the explain operation and its relationship with organizational goals will assist in future developments of the protocol.

Reference List

anarchivist. (n.d.). Z39.50/SRU Client. *drupal.org.* Retrieved from http://drupal.org/project/z3950 (accessed 20 August 2009).

Avram, H.D. (1975). *MARC: Its History and Implications.* Library of Congress: Washington.

Basedow Information Systems. (2009). Mercury Z39.50 Client. *Basedow Information Systems.* Retrieved from http://www.basedowinfosys.com/projects/mzc (accessed 1 August 2009).

Binkley, P. (2009, July 3). unAPI over SRU with Cocoon. *Quædam Cuiusdam: Morning Postings from Route 66.* Retrieved from http://www.wallandbinkley.com/quaedam/?p=69 (accessed 20 August 2009).

Chudnov, D. (2006). *unAPI.info - For All Your unAPI needs*. Retrieved from http://unapi.info/ (accessed 21 August 2009).

Chudnov, D., Binkley, P., Frumkin, J. Giarlo, M.J., Rylander, M., Singer, R., & Summers, E. (2006). Introducing unAPI. *Ariadne*(48)(July). Retrieved from http://www.ariadne.ac.uk/issue48/chudnov-et-al/ (accessed 8 December 2009).

Collectable Card Games Context Sets. (n.d.). Retrieved from http://srw.cheshire3.org/contextSets/ccg/ (accessed 24 August 2009).

Denenberg, R. (2009, May 7). *The Media Types application/mods+xml, application/mads+xml, application/mets+xml, application/marcxml+xml, application/sru+xml.*Retrieved from http://www.loc.gov/standards/sru/internet-drafts/draft-denenberg-mods-etc-media-types-00.html (accessed 25 August 2009).

----. (2009, July 22). OASIS Drafts of SRU and CQL. *SRU (Search and Retrieve via URL) Implementors zng@loc.gov* (accessed 23 July 2009).

evergreen-admin: sru and z39.50. (2009, July 1). *Evergreen DokuWiki.* Retrieved from

http://www.open-ils.org/dokuwiki/doku.php?id=evergreen-admin:sru_and_z39.50 (accessed 18 August 2009).

Federated search instructions. (n.d.). in CRCnetBase. Retrieved from http://www.crcnetbase.com/federated-search-instructions/index.asp (accessed 3 July 2009).

Hafezi, M.A. (2008). Interoperability between library software: A solution for Iranian libraries. *The Electronic Library* 26, (5): 726-34.

Hammer, S., Dickmeiss, A., Taylor, M., & Levanto, H. (2009). *YAZ User's Guide and Reference*. Index Data. Retrieved from http://www.indexdata.com/yaz/doc/yaz.pdf (accessed 3 June 2009).

Index Data. (2009). Software. *Index Data*. Retrieved from http://www.indexdata.dk/ (accessed 12 August 2009).

Morgan, E.L. (2004). An introduction to the Search/Retrieve URL service (SRU). *Ariadne*(40) (Jul).

Music Context Set. (2004, March 6). Retrieved from http://www.ceridwen.com/srw/music-contextset.html (accessed 21 August 2009).

National Information Standards Organization. (2005). *Collection Description Specification.* NISO Press: Bethesda, MD. Retrieved from http://www.niso.org/kst/reports/standards?step=2&gid=None&project_key%3Austring%3Aiso-8859-1=62d54e7e0dc6f3d14be8f5f262ac8ee1c524d4b4 (accessed 3 July 2009).

----. 2005. *Information Retrieval Service Description Specification.* NISO Press: Bethesda, MD. Retrieved from http://www.niso.org/kst/reports/standards?step=2&gid=None&project_key%3Austring%

3Aiso-8859-1=9a76e7558ecd4d62575b9d8adc60e802a7d7d32f (accessed 3 July 2009).

Network Context Set. (n.d.). Retrieved from http://srw.cheshire3.org/contextSets/net/1.0/ (accessed 24 August 2009).

OCLC Research. (n.d.). *Open SiteSearch Documentation.* Retrieved from http://alcme.oclc.org/srw/search/SiteSearchDocumentation?operation=explain&version=1.1 (accessed 15 August 2009).

Open Archives Initiative - Protocol for Metadata Harvesting - v.2.0. (n.d.). Retrieved from http://www.openarchives.org/OAI/openarchivesprotocol.html (accessed 15 August 2009).

Reiss, K. (2007). SRU, open data and the future of metasearch. *Internet Reference Services Quarterly* 12, (3-4): 369-86.

Reitz, J.M. (n.d.). ODLIS: Online dictionary for library and information science. 2007. Retrieved from http://lu.com/odlis/index.cfm (accessed 9 March 2009).

Sanderson, R., Young, J., & LeVan, R. (2005). SRW/U with OAI: Expected and unexpected synergies. *D-Lib Magazine* 11, (2) (Feb). Retrieved from http://www.dlib.org/dlib/february05/sanderson/02sanderson.html (accessed 8 December 2009).

SRU: Search/Retrieval via URL -- SRU, CQL and ZeeRex (standards, Library of Congress). (n.d.). Retrieved from http://www.loc.gov/standards/sru/ (accessed 15 July 2009).

TAL Online. (n.d.). Retrieved from http://www.talonline.ca/searchalberta/ (accessed 12 June 2009).

Taylor, M. (2003). A Gentle Introduction to CQL. *ZING.* Retrieved from http://zing.z3950.org/cql/intro.html (accessed 9 July 2009).

Taylor, M. & Dickmeiss, A. (2006). Delivering MARC/XML Records from the Library of

Congress Catalogue Using the Open Protocols SRW/U and Z39.50. *International Cataloguing and Bibliographic Control* 35, (1) (Jan 2006-Mar): 7-10.

----. 2009. *YAZ Proxy User's Guide and Reference*. Index Data. Retrieved from http://www.indexdata.com/yazproxy/doc/yazproxy.pdf (accessed 3 June 2009).

The European Library 2.1. (n.d.). Retrieved from http://search.theeuropeanlibrary.org/portal/en/index.html (accessed 9 March 2009).

van Veen, T., & Oldroyd, B. (2004). Search and retrieval in The European Library: A new approach. *D-Lib Magazine* 10, (2) (Feb). Retrieved from http://www.dlib.org/dlib/february04/vanveen/02vanveen.html (accessed 8 December 2009).

Ventura, L. (2009). VMware. *Linhost.info: Notes from an IT Engineer.* Retrieved from http://linhost.info/vmware/ (accessed 24 August 2009).

VMware. (2009). VMware Player: Run Virtual Machines on Linux or Windows PCs for Free. *VMware.com.* Retrieved from http://www.vmware.com/products/player/ (accessed 23 August 2009).

What's New in Version 1.2? (2008, March 4). (standards, Library of Congress). Retrieved from http://www.loc.gov/standards/sru/whatsnew.html (accessed 22 August 2009).

Z39.50 maintenance agency page. (n.d.). Retrieved from http://www.loc.gov/z3950/agency/ (accessed 4 June 2009).

ZeeRex: The explainable ``Explain" service. (n.d.). Retrieved from http://explain.z3950.org/ (accessed 9 March 2009).

Reflections for LIS 600

I completed this Directed Study as a preparation for research completed in LIS 597, Seminar in Advanced Research Methods. My choice of topic was driven by previous experiences with administering a Z39.50 server at a regional library in Alberta. Our Z39.50 server was set up using the defaults created by the integrated library system (ILS) vendor, and was largely ignored after the initial setup. I suspect that many Z39.50 servers are administered in the same manner.

I first realized that there were a number of options available when our technical services department started using Z39.50 to connect to remote servers, including paid subscription services, for copy cataloguing. Configuring the Z39.50 client within our ILS was challenging, largely due to the complexity of the Z39.50 protocol and the differences in configuration chosen by the various target systems. During this process, I encountered Z39.50 profiles, including the Bath profile and various national profiles.

This experience led me to consider researching profile compliance of Z39.50 servers for my LIS 505 (Research Methods for Library and Information Studies) project. However, as I explored this topic, I realized that the SRU protocol was arguably more relevant. As mentioned in the body of the paper, I was especially intrigued by the Explain operation. Once I realized that there was a lack of research in this area, I chose this as my topic for the LIS 505 proposal. I used that proposal as the basis for a successful application for a Queen Elizabeth II Scholarship.

Because of the lack of research, I chose a grounded theory methodology to discover what variables were influencing SRU use. Part of grounded theory methodology is a need for sufficient knowledge to recognize variables of interest, which is termed theoretical sensitivity. In addition to a need to develop this sensitivity, I was also driven by personal desires – I did not

want to waste my participants' time become of my ignorance, and I did not want to be embarrassed by my lack of knowledge. A Directed Study in the SRU protocol was a natural response to these needs.

The Directed Study certainly fulfilled those needs, but it also allowed me to explore collegial connections and the use of virtual machines to test new software.

While choosing my topic for LIS 505, I contacted Dr. William Moen of the University of North Texas. His comments and guidance were invaluable in choosing my final topic. As mentioned in the body of the paper, Sandra Shores and Dr. Peter Binkley of the University of Alberta Libraries were very generous with their time and assistance.

In my previous employment, I made heavy use of server-based virtual machines for both testing and production services. While I had experimented with desktop installations of virtual machines, my use had been fairly limited, and I had never experimented with the export of virtual machines for other people to use.

In the Fall of 2009, I used the knowledge gained in this Directed Study to design and complete a pilot study in the use of the Explain operation and SRU as part of LIS 597. This study touched on a number of broad issues beyond technology, including the needs of users, the challenges in information retrieval and the changing nature of libraries' online presence. I expect these concepts and challenges to inform much of my life as an information professional.

Acknowledgements

This project was conducted as a LIS 599 Directed Study in preparation for a research project completed in LIS 597 - Seminar in Advanced Research Methods. I am grateful to both Dr. Ali Shiri and Dr. Lisa M. Given for their support and guidance during the respective projects.

I would also like to thank my advisor, Dr. Dinesh Rathi, for his support and guidance during the completion of my MLIS program.

Appendices

Appendix A: Sample Z39.50 Search Using yaz-client

```
[root@localhost ~]# yaz-client z3950.loc.gov:7090/voyager
Connecting...OK.
Sent initrequest.
Connection accepted by v3 target.
ID     : 34
Name   : Voyager LMS - Z39.50 Server (YAZ Proxy)
Version: 2006.5.2/1.2.1.1
Options: search present
Elapsed: 0.579191
Z> find @attr 1=1016 "ali shiri"
Sent searchRequest.
Received SearchResponse.
Search was a success.
Number of hits: 2
records returned: 0
Elapsed: 0.223608
Z> show 2+1
Sent presentRequest (2+1).
Records: 1
[VOYAGER]Record type: USmarc
02462cam  2200349 a 4500
001 14441095
005 20070507172928.0
008 060707s2006    nyua     b    001 0 eng
906    $a 7 $b cbc $c orignew $d 1 $e ecip $f 20 $g y-gencatlg
925 0  $a acquire $b 2 shelf copies $x policy default
955    $a lh39 2006-07-07 $i lh39 2006-07-07 $e lh39 2006-07-07 to
Dewey $a aa20 2006-07-07 $a ps10 2007-05-01 1 copy rec'd., to CIP ver.
$f lk18 2007-05-07 Z-CipVer $g lk18 2007-05-07 to BCCD
010    $a   2006021354
020    $a 0789034522 (alk. paper)
020    $a 9780789034526 (alk. paper)
020    $a 0789034530 (pbk. : alk. paper)
020    $a 9780789034533 (pbk. : alk. paper)
035    $a (OCoLC)ocm70267096
035    $a (OCoLC)70267096
040    $a DLC $c DLC $d YDX $d BAKER $d BTCTA $d YDXCP $d DLC
050 00 $a Z696.D7 $b M68 2006
082 00 $a 025.4/31 $2 22
245 00 $a Moving beyond the presentation layer : $b content and
context in the Dewey decimal classification (DDC) system / $c Joan S.
Mitchell, Diane Vizine-Goetz, editors.
260    $a New York : $b Haworth Information Press, $c c2006.
300    $a xix, 239 p. : $b ill. ; $c 23 cm.
```

```
500    $a "Co-published simultaneously as Cataloging & classification
quarterly, volume 42, numbers 3/4 2006."
504    $a Includes bibliographical references and index.
505 0  $a Forty years of classification online : final chapter or
future unlimited? / Karen Markey -- The DDC relative index / Francis
Miksa -- Teaching the Dewey decimal classification system / Arlene G.
Taylor -- Classifying the popular music of Trinidad and Tobago /
Lorraine M. Nero -- Dewey decimal classification (DDC) at the Swiss
National Library / Patrice Landry -- DDC German--the project, the
aims, the methods : new ideas for a well-established traditional
classification system / Magda Heiner-Freiling -- Users browsing
behaviour in a DDC-based Web service : a log analysis / Traugott Koch,
Koraljka Golub, Anders Ardèo -- HILT : a pilot terminology mapping
service with a DDC spine / Dennis M. Nicholson, Alan Dawson, Ali Shiri
-- Resource discovery in the government of Canada using the Dewey
decimal classification / Deane Zeeman, Glenyss Turner -- DeweyBrowser
/ Diane Vizine-Goetz.
650  0 $a Classification, Dewey decimal.
700 1  $a Mitchell, Joan S.
700 1  $a Vizine-Goetz, Diane.
730 0  $a Cataloging & classification quarterly.
856 41 $3 Table of contents only $u
http://www.loc.gov/catdir/toc/ecip0616/2006021354.html

nextResultSetPosition = 3
Elapsed: 0.156460
```

Appendix B: Sample SRU Search Using yaz-client

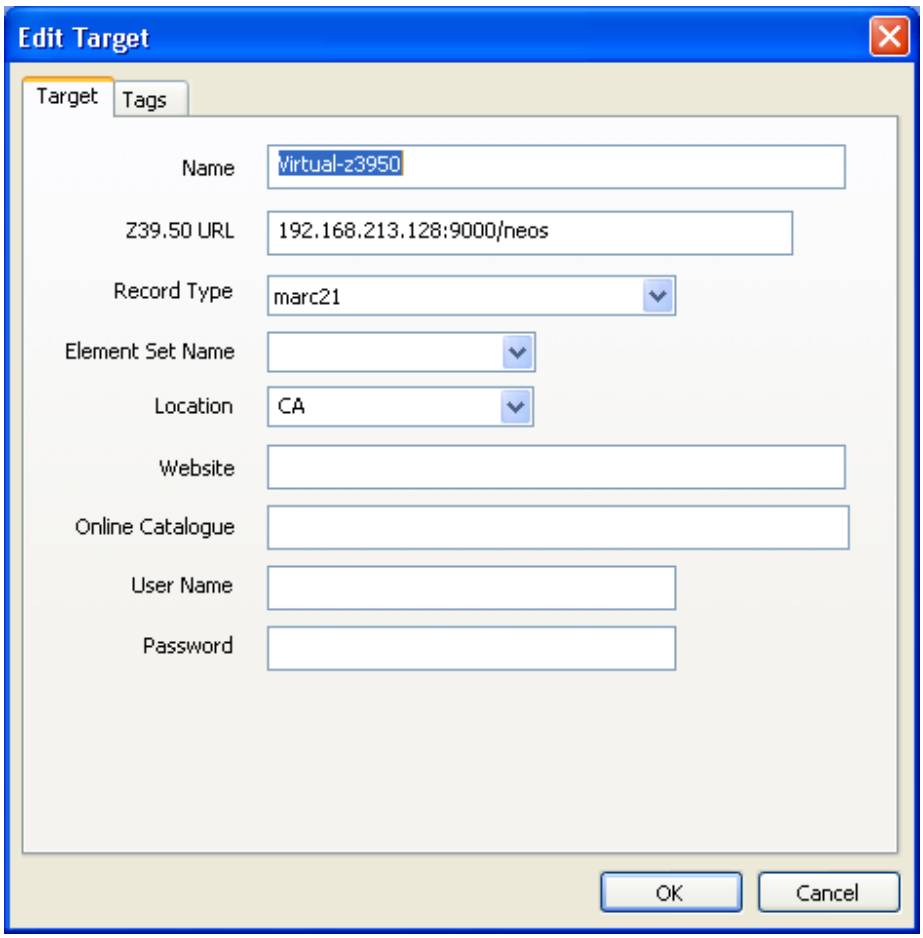
```
[root@localhost ~]# yaz-client http://z3950.loc.gov:7090/voyager
Connecting...OK.
Z> find "ali shiri"
Received SRW SearchRetrieve Response
Number of hits: 2
Elapsed: 0.198200
Z> show 2+1
Connecting...OK.
Received SRW SearchRetrieve Response
Number of hits: 2
pos=2 schema=info:srw/schema/1/marcxml-v1.1
<record xmlns="http://www.loc.gov/MARC21/slim">
  <leader>02462cam a2200349 a 4500</leader>
  <controlfield tag="001">14441095</controlfield>
  <controlfield tag="005">20070507172928.0</controlfield>
  <controlfield tag="008">060707s2006    nyua     b    001 0 eng
</controlfield>
  <datafield tag="906" ind1=" " ind2=" ">
    <subfield code="a">7</subfield>
    <subfield code="b">cbc</subfield>
    <subfield code="c">orignew</subfield>
    <subfield code="d">1</subfield>
    <subfield code="e">ecip</subfield>
    <subfield code="f">20</subfield>
    <subfield code="g">y-gencatlg</subfield>
  </datafield>
  <datafield tag="925" ind1="0" ind2=" ">
    <subfield code="a">acquire</subfield>
    <subfield code="b">2 shelf copies</subfield>
    <subfield code="x">policy default</subfield>
  </datafield>
  <datafield tag="955" ind1=" " ind2=" ">
    <subfield code="a">lh39 2006-07-07</subfield>
    <subfield code="i">lh39 2006-07-07</subfield>
    <subfield code="e">lh39 2006-07-07 to Dewey</subfield>
    <subfield code="a">aa20 2006-07-07</subfield>
    <subfield code="a">ps10 2007-05-01 1 copy rec&amp;apos;d., to CIP
ver.</subfield>
    <subfield code="f">lk18 2007-05-07 Z-CipVer</subfield>
    <subfield code="g">lk18 2007-05-07 to BCCD</subfield>
  </datafield>
  <datafield tag="010" ind1=" " ind2=" ">
    <subfield code="a">  2006021354</subfield>
  </datafield>
  <datafield tag="020" ind1=" " ind2=" ">
    <subfield code="a">0789034522 (alk. paper)</subfield>
  </datafield>
  <datafield tag="020" ind1=" " ind2=" ">
```

```
    <subfield code="a">9780789034526 (alk. paper)</subfield>
  </datafield>
  <datafield tag="020" ind1=" " ind2=" ">
    <subfield code="a">0789034530 (pbk. : alk. paper)</subfield>
  </datafield>
  <datafield tag="020" ind1=" " ind2=" ">
    <subfield code="a">9780789034533 (pbk. : alk. paper)</subfield>
  </datafield>
  <datafield tag="035" ind1=" " ind2=" ">
    <subfield code="a">(OCoLC)ocm70267096</subfield>
  </datafield>
  <datafield tag="035" ind1=" " ind2=" ">
    <subfield code="a">(OCoLC)70267096</subfield>
  </datafield>
  <datafield tag="040" ind1=" " ind2=" ">
    <subfield code="a">DLC</subfield>
    <subfield code="c">DLC</subfield>
    <subfield code="d">YDX</subfield>
    <subfield code="d">BAKER</subfield>
    <subfield code="d">BTCTA</subfield>
    <subfield code="d">YDXCP</subfield>
    <subfield code="d">DLC</subfield>
  </datafield>
  <datafield tag="050" ind1="0" ind2="0">
    <subfield code="a">Z696.D7</subfield>
    <subfield code="b">M68 2006</subfield>
  </datafield>
  <datafield tag="082" ind1="0" ind2="0">
    <subfield code="a">025.4/31</subfield>
    <subfield code="2">22</subfield>
  </datafield>
  <datafield tag="245" ind1="0" ind2="0">
    <subfield code="a">Moving beyond the presentation layer
:</subfield>
    <subfield code="b">content and context in the Dewey decimal
classification (DDC) system /</subfield>
    <subfield code="c">Joan S. Mitchell, Diane Vizine-Goetz,
editors.</subfield>
  </datafield>
  <datafield tag="260" ind1=" " ind2=" ">
    <subfield code="a">New York :</subfield>
    <subfield code="b">Haworth Information Press,</subfield>
    <subfield code="c">c2006.</subfield>
  </datafield>
  <datafield tag="300" ind1=" " ind2=" ">
    <subfield code="a">xix, 239 p. :</subfield>
    <subfield code="b">ill. ;</subfield>
    <subfield code="c">23 cm.</subfield>
  </datafield>
  <datafield tag="500" ind1=" " ind2=" ">
```

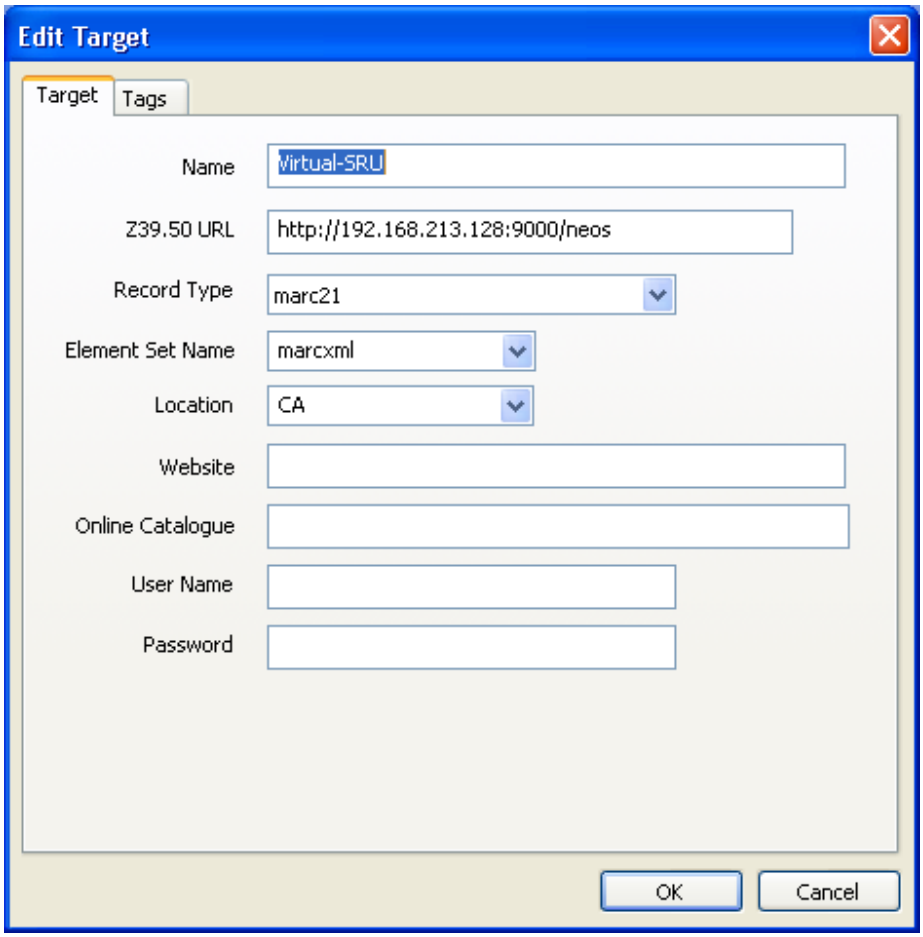
```
    <subfield code="a">&amp;quot;Co-published simultaneously as Cataloging
&amp;amp; classification quarterly, volume 42, numbers 3/4
2006.&amp;quot;</subfield>
  </datafield>
  <datafield tag="504" ind1=" " ind2=" ">
    <subfield code="a">Includes bibliographical references and
index.</subfield>
  </datafield>
  <datafield tag="505" ind1="0" ind2=" ">
    <subfield code="a">Forty years of classification online : final
chapter or future unlimited? / Karen Markey -- The DDC relative index
/ Francis Miksa -- Teaching the Dewey decimal classification system /
Arlene G. Taylor -- Classifying the popular music of Trinidad and
Tobago / Lorraine M. Nero -- Dewey decimal classification (DDC) at the
Swiss National Library / Patrice Landry -- DDC German--the project,
the aims, the methods : new ideas for a well-established traditional
classification system / Magda Heiner-Freiling -- Users browsing
behaviour in a DDC-based Web service : a log analysis / Traugott Koch,
Koraljka Golub, Anders Ardoì -- HILT : a pilot terminology mapping
service with a DDC spine / Dennis M. Nicholson, Alan Dawson, Ali Shiri
-- Resource discovery in the government of Canada using the Dewey
decimal classification / Deane Zeeman, Glenyss Turner -- DeweyBrowser
/ Diane Vizine-Goetz.</subfield>
  </datafield>
  <datafield tag="650" ind1=" " ind2="0">
    <subfield code="a">Classification, Dewey decimal.</subfield>
  </datafield>
  <datafield tag="700" ind1="1" ind2=" ">
    <subfield code="a">Mitchell, Joan S.</subfield>
  </datafield>
  <datafield tag="700" ind1="1" ind2=" ">
    <subfield code="a">Vizine-Goetz, Diane.</subfield>
  </datafield>
  <datafield tag="730" ind1="0" ind2=" ">
    <subfield code="a">Cataloging &amp;amp; classification
quarterly.</subfield>
  </datafield>
  <datafield tag="856" ind1="4" ind2="1">
    <subfield code="3">Table of contents only</subfield>
    <subfield
code="u">http://www.loc.gov/catdir/toc/ecip0616/2006021354.html</subfi
eld>
  </datafield>
</record>

Elapsed: 0.484132
```

Appendix C: Mercury Z39.50 Client Configurations

Z39.50 Target

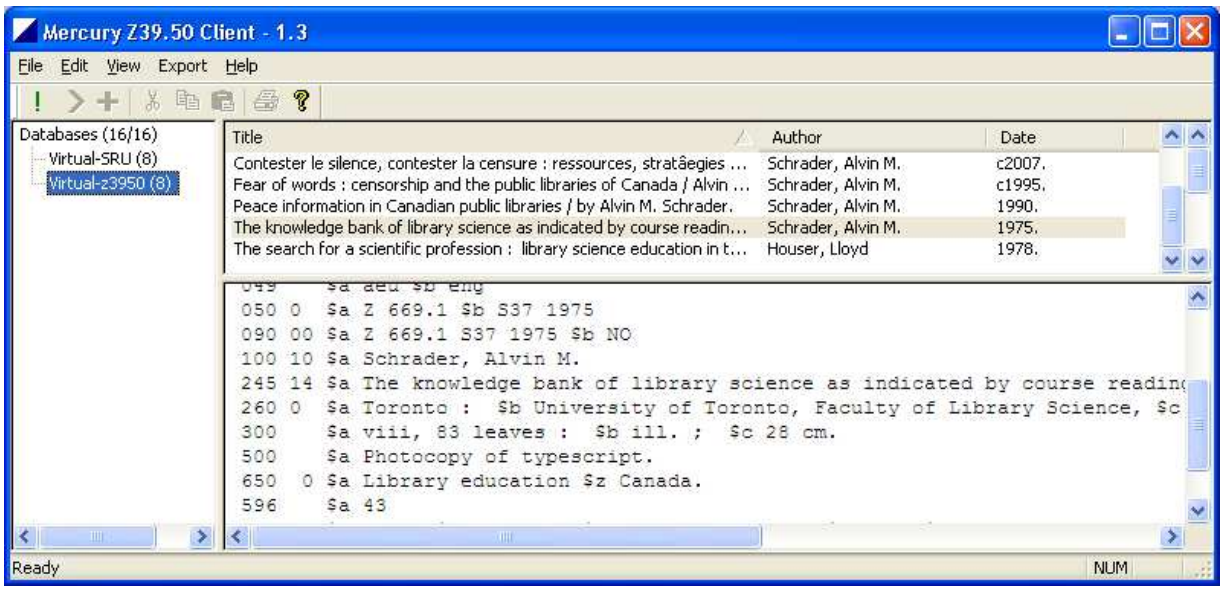


SRU Target

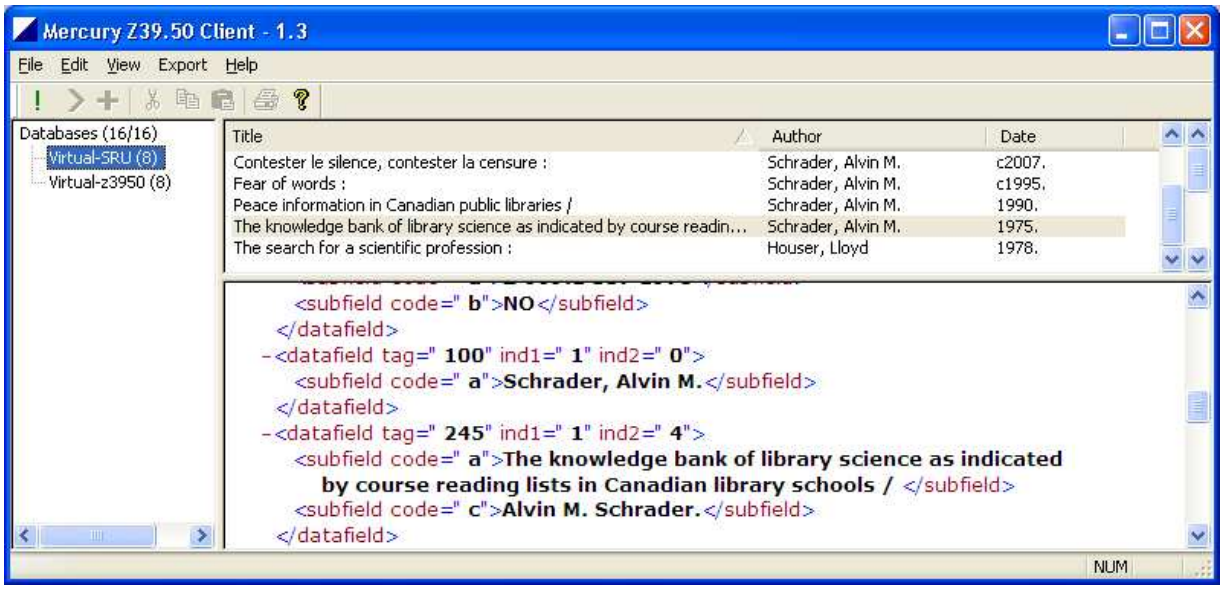

Edit Target
Target
Tags
Name
Virtual-SRU
Z39.50 URL
http://192.168.213.128:9000/neos
Record Type
marc21
Element Set Name
marcxml
Location
CA
Website
Online Catalogue
User Name
Password
OK
Cancel

Appendix D: Mercury Z39.50 Results Screens

Z39.50 Results



SRU Results



Appendix E: Example YAZ Proxy Configuration config.xml

```
<?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"
  >
<!-- Config can be checked with xerces-c++: PParse -n -s config.xml  -->
<!-- ********************************************************
           NEOS
*********************************************************-->
<!-- Defines service available at http://192.168.213.128:9000/neos -->
<target name="neos" database="Unicorn" default="1">
    <url>ualapp.library.ualberta.ca:2200</url>
    <target-timeout>240</target-timeout>
    <client-timeout>180</client-timeout>

    <!-- List of supported attributes -->
    <!-- use attributes -->
    <attribute type="1"
       value="1-9,13,16,20,21,29-31,33,50,51,54,58,60,62,63,1003,1004,1016"/>
    <attribute type="1" value="*" error="114"/>

    <!-- relation attributes -->
    <attribute type="2" value="1,2,3,4,5" />
    <attribute type="2" value="*" error="117" />

    <!-- position attributes -->
    <attribute type="3" value="1,3" />
    <attribute type="3" value="*" error="119" />

    <!-- structure attributes -->
    <attribute type="4" value="1,2,4,6,101" />
    <attribute type="4" value="*" error="118" />

    <!-- truncation attributes -->
    <attribute type="5" value="1,100" />
    <attribute type="5" value="*" error="120" />

    <!-- completeness attributes -->
    <attribute type="6" value="1,3" />
    <attribute type="6" value="*" error="122" />

    <!-- other attributes -->
    <attribute type="*" value="*" error="113" />

    <!-- list of allowed record syntaxes / schema -->
    <syntax type="usmarc"/>
    <syntax type="none"/>
    <syntax type="xml" marcxml="1"
      identifier="info:srw/schema/1/marcxml-v1.1" >
      <name>marcxml</name>
    </syntax>
    <syntax type="xml" marcxml="1"
```

```
stylesheet="/usr/local/share/yazproxy/MARC21slim2SRWDC.xsl"
      identifier="info:srw/schema/1/dc-v1.1">
      <title>Dublin Core</title>
      <name>dc</name>
    </syntax>
    <syntax type="xml" marcxml="1"
      stylesheet="/usr/local/share/yazproxy/MARC21slim2MODS3.xsl"
      identifier="http://www.loc.gov/mods/v3">
      <title>MODS</title>
      <name>mods</name>
    </syntax>

    <!-- reject other syntax requests with an error message -->
    <syntax type="*" error="238"/>

    <!-- defines a minimum number of spare sessions -->
    <preinit>0</preinit>

    <!-- explain record -->
    <explain xmlns="http://explain.z3950.org/dtd/2.0/">

      <!-- metainfo about this explain record -->
      <metaInfo>
        <dateModified>2009-08-29</dateModified>
      </metaInfo>

      <!-- server connection information -->
      <serverInfo>
        <!-- host should normally contain a FQDN, not IP -->
        <host>192.168.213.128</host>
        <port>9000</port>
        <database>neos</database>
      </serverInfo>

      <!-- database information -->
      <databaseInfo>
        <title lang="en" primary="true">Demonstration SRU Server
        </title>
        <description>An SRU interface to the records of the NEOS
          consortium hosted at the University of Alberta.
        </description>
        <agents>
          <agent type="creator">T. Michael Silver, michael.silver@ualberta.ca
          </agent>
        </agents>

        <!-- software information -->
        <implementation
          identifier="http://www.indexdata.com/yazproxy" version="1.3.4">
          <agents>
            <agent type="vendor">Index Data</agent>
            <agent type="contact">David Dorman, dorman@indexdata.com</agent>
          </agents>
          <title>YAZ proxy</title>
        </implementation>

        <!-- links information -->
```

```
  <links>
    <link type="www">http://www.library.ualberta.ca/</link>
    <link type="z39.50">
      z3950s://ualapp.library.ualberta.ca:2200/Unicorn
    </link>
  </links>
</databaseInfo>

<!-- info about access points -->
<indexInfo>
  <set identifier="info:srw/cql-context-set/1/cql-v1.1" name="cql" />
  <set identifier="info:srw/cql-context-set/1/dc-v1.1" name="dc" />
  <set identifier="http://zing.z3950.org/cql/bath/2.0/" name="bath" />
  <index id="1">
    <title>personalName</title>
    <map><name set="bath">personalName</name></map>
  </index>
  <index id="2">
    <title>corporateName </title>
    <map><name set="bath">corporateName</name></map>
  </index>
  <index id="3">
    <title>conferenceName</title>
    <map><name set="bath">conferenceName</name></map>
  </index>
  <index id="4">
    <title>title</title>
    <map><name set="dc">title</name></map>
    <map><name set="bath">titleKeyword</name></map>
  </index>
  <index id="5">
    <title>series</title>
    <map><name set="local">series</name></map>
  </index>
  <index id="6">
    <title>uniformTitle</title>
    <map><name set="bath">uniformTitle</name></map>
  </index>
  <index id="7">
    <title>isbn</title>
    <map><name set="bath">isbn</name></map>
  </index>
  <index id="8">
    <title>issn</title>
    <map><name set="bath">issn</name></map>
  </index>
  <index id="9">
    <title>lccn</title>
    <map><name set="local">lccn</name></map>
  </index>
  <index id="13">
    <title>ddc</title>
    <map><name set="local">ddc</name></map>
  </index>
  <index id="16">
    <title>lcc</title>
    <map><name set="local">lcc</name></map>
```

```
</index>
<index id="20">
  <title>localClass</title>
  <map><name set="local">localClass</name></map>
</index>
<index id="21">
  <title>subject</title>
  <map><name set="bath">subject</name></map>
  <map><name set="dc">subject</name></map>
</index>
<index id="29">
  <title>localSubject</title>
  <map><name set="local">localSubject</name></map>
</index>
<index id="30">
  <title>date</title>
  <map><name set="dc">date</name></map>
</index>
<index id="31">
  <title>pubDate</title>
  <map><name set="bath">pubDate</name></map>
</index>
<index id="33">
  <title>keyTitle</title>
  <map><name set="bath">keyTitle</name></map>
</index>
<index id="50">
  <title>govPubNum</title>
  <map><name set="local">govPubNum</name></map>
</index>
<index id="51">
  <title>musicNum</title>
  <map><name set="local">musicNum</name></map>
</index>
<index id="54">
  <title>language</title>
  <map><name set="dc">language</name></map>
</index>
<index id="58">
  <title>geographicName</title>
  <map><name set="bath">geographicName</name></map>
</index>
<index id="60">
  <title>coden</title>
  <map><name set="local">coden</name></map>
</index>
<index id="62">
  <title>description</title>
  <map><name set="dc">description</name></map>
</index>
<index id="63">
  <title>notes</title>
  <map><name set="bath">notes</name></map>
</index>
<index id="1003">
  <title>author</title>
  <map><name set="bath">author</name></map>
```

```
          <map><name set="dc">author</name></map>
          <map><name set="dc">creator</name></map>
        </index>
        <index id="1004">
          <title>authorPersonalName</title>
          <map><name set="local">authorPersonalName</name></map>
        </index>
      </indexInfo>

      <!-- information on record schema -->
      <schemaInfo>
        <schema identifier="info:srw/schema/1/marcxml-v1.1"
          sort="false" name="marcxml">
          <title>MARCXML</title>
        </schema>

        <schema identifier="info:srw/schema/1/dc-v1.1"
          sort="false" name="dc">
          <title>Dublin Core</title>
        </schema>

        <schema identifier="info:srw/schema/1/mods-v3.0"
          sort="false" name="mods">
          <title>MODS v3</title>
        </schema>
      </schemaInfo>

      <!-- server configuration information -->
      <configInfo>
        <default type="index">1016</default>
      </configInfo>

    </explain>

    <!-- location of file mapping PQF to CQL notation -->
    <cql2rpn>/usr/local/share/yazproxy/pqf.properties</cql2rpn>

  </target>

  <max-clients>20</max-clients>
  <!-- <max-connect>10</max-connect> -->
  <!-- <period-connect>10</period-connect> -->
  <!-- <limit-connect>5</limit-connect> -->
  <log>client-ip</log>
  <docpath>doc</docpath>
</proxy>
```

Appendix F: Using the Project Demonstration Files

> This appendix is included for completeness. However, only `query.html` is included because of licensing issues and the size of the files involved.

The USB drive included with this document contains the following items:

- VMware Player installation file for Microsoft Windows
- Mercury Z39.50 Client installation file for Microsoft Windows
- A web page (`query.html`) which demonstrates accessing SRU servers using a simple HTML web form
- The CentOS-based virtual machine which includes the YAZ proxy installation discussed in the paper

The first step in using these files is the installation of VMware Player and the Mercury Z39.50 Client. This is accomplished using the normal Windows software installation process. The setup files are located in the `Software` folder of the USB drive.

The next stage is copying the virtual machine files to the local hard drive. Because of file size limitation, the virtual machine cannot be run from the USB drive. The entire `centos52` folder should be copied to the hard drive.

Opening the virtual machine is accomplished similarly to opening any file. Open the VMware Player application (Start → VMware → VMware Player). If this is the first time the software has been run, the user must accept the license agreement. Select *Browse for available virtual machines* in the program interface. The `CentOS 5.2` file should be selected from the location it was saved earlier. A warning message will appear indicating the machine files have

been moved or copied. Select *I copied it* and continue. The screen will display the startup sequence for the image. Depending on local security settings, a warning may be displayed regarding the VMware NAT service attempting to connect to the network. This service must be allowed network access for the demonstration server to work as it must access the University of Alberta Z39.50 server.

The VMware Player creates unique private subnets when it is installed on individual computers. Unfortunately, that means that the configuration information for the Mercury Z39.50 Client will have to reflect the IP address of the running system, and the `form action` tags in the `query.html` file must be updated to reflect the correct address. The virtual machine has been configured to display the IP address and URL of the SRU service on start up. The `query.html` file contains comments that mark where the file must be updated with the correct URL.

In addition, the virtual machine may be accessed in the VMware Player console or by using an ssh or sftp client. The username and password for the machine are `root` and `ctuser`. The YAZ proxy files are located in the `/usr/local/share/yazproxy/` directory.

The Mercury Z39.50 Client may also be used to connect to the SRU server. The configuration screen is accessed by selecting File → Search, then navigating to the Databases section. The Add button creates a new entry. An example of the configuration settings is shown in Appendix C. The field label *Z39.50 URL* is misleading - entering the URL including the protocol will cause the program to interpret it as an SRU target. Again, the URL to be used will be displayed in the VMware Player window.

The SRU server may be accessed by other programs running on the local computer.

Because of the nature of networking used in this virtual machine, it is not possible to access the SRU server from a remote computer.

Shutting down the virtual machine gracefully may be accomplished by logging in to the machine either at the VMware Player window or using ssh and issuing the command `shutdown -h now`.