MAGAZINE

The Z39.50 Information Retrieval Standard

Part I: A Strategic View of Its Past, Present and Future

Clifford A. Lynch
Director, Library Automation
Office of the President
University of California
Oakland, California
clifford.lynch@ucop.edu

D-Lib Magazine, April 1997

ISSN 1082-9873

Non-technical article summarizing the architecture and the concepts behind the Z39.50 protocol.

Contents

Introduction

The Z39.50 standard for information retrieval is important from a number of perspectives. While still not widely known within the computer networking community, it is a mature standard that represents the culmination of two decades of thinking and debate about how information retrieval functions can be modeled, standardized, and implemented in a distributed systems environment. And - importantly -- it has been tested through substantial deployment experience.

Z39.50 is one of the few examples we have to date of a protocol that actually goes beyond codifying mechanism and moves into the area of standardizing shared semantic knowledge. The extent to which this should be a goal of the protocol has been an ongoing source of controversy and tension within the developer community, and differing views on this issue can be seen both in the standard itself and the way that it is used in practice. Given the growing emphasis on issues such as "semantic interoperability" as part of the research agenda for digital libraries (see Clifford A. Lynch and Hector Garcia-Molina. Interoperability, **Scaling, and the Digital Libraries Research Agenda, Report on the May 18-19, 1995 IITA Libraries Workshop**, <http://www- diglib.stanford.edu/diglib/pub/reports/iita-dlw/main.html>), the insights gained by the Z39.50 community into the complex interactions among various definitions of semantics and interoperability are particularly relevant.

The development process for the Z39.50 standard is also of interest in its own right. Its history, dating back to the 1970s, spans a period that saw the eclipse of formal standards-making agencies by groups such as the Internet Engineering Task Force (IETF) and informal standards development consortia. Moreover, in order to achieve meaningful implementation, Z39.50 had to move beyond its origins in the OSI debacle of the 1980s. Z39.50 has also been, to some extent, a victim of its own success -- or at least promise. Recent versions of the standard are highly extensible, and the consensus process of standards development has made it hospitable to an ever-growing set of new communities and requirements. As this process of extension has proceeded, it has become ever less clear what the appropriate scope and boundaries of the

protocol should be, and what expectations one should have of practical interoperability among implementations of the standard. Z39.50 thus offers an excellent case study of the problems involved in managing the evolution of a standard over time. It may well offer useful lessons for the future of other standards such as HTTP and HTML, which seem to be facing some of the same issues.

This paper, which will appear in two parts, starting with this issue of **D-Lib**, looks at several strategic issues surrounding Z39.50. After a relatively brief overview of the function and history of the protocol, I will examine some of the competing visions of the protocol's role, with emphasis on issues of interoperability and the incorporation of semantics. The second installment of the paper will look at questions related to the management of the standard and the standards development process, with emphasis on the scope of the protocol and how that relates back again to interoperability questions. The paper concludes with a discussion of the adoption and deployment of the standard, its relationship to other standards, and some speculations on future directions for the protocol.

This paper is not intended to be a tutorial on the details of how current or past versions of Z39.50 work. These technical details are covered not only in the standard itself (which can admittedly be rather difficult reading) but also in an array of tutorial and review papers (see <http://lcweb.loc.gov/z3950/agency> for bibliographies and pointers to on-line information on Z39.50). Instead, the paper's focus is on how and why Z39.50 developed the way it did, and the conceptual debates that have influenced its evolution and use. While a detailed technical knowledge of the operation of Z39.50 is certainly helpful, it should not be necessary in order to follow most of the material here.

Some disclaimers are in order. I have been actively involved in the development of Z39.50 since the early 1980s and have been a participant -- and on occasion, even an instigator -- of some of the activities described here. This paper is an attempt to make a critical assessment of the current state of Z39.50 and a review of its development with the full benefit of hindsight. It recounts a number of debates that occurred within the developer community over the past years. In many of these, I advocated specific positions or approaches, sometimes successfully and sometimes unsuccessfully. What is presented here is one person's perspective - mine --, which is sometimes at odds with the current consensus with the developer community; I've tried to represent opposing views fairly, and to differentiate my opinions from fact or consensus. However, others will undoubtedly disagree with many of the comments here.

What is Z39.50?

Z39.50 -- properly "Information Retrieval (Z39.50); Application Service Definition and Protocol Specification, ANSI/NISO Z39.50-1995" -- is a protocol which specifies data structures and interchange rules that allow a client machine (called an "origin" in the standard) to search databases on a server machine (called a "target" in the standard) and retrieve records that are identified as a result of such a search.

The rather forbidding name "Z39.50" comes from the fact that the National Information Standards Organization (NISO), the American National Standards Institute (ANSI)-accredited standards development organization serving libraries, publishing and information services, was once the Z39 committee of ANSI. NISO standards are numbered sequentially and Z39.50 is the fiftieth standard developed by NISO. The current version of Z39.50 was adopted in 1995, thus superseding earlier versions adopted in 1992 and 1988. It is sometimes referred to as Z39.50 Version 3.

Z39.50 had its roots in the OSI efforts of the 1980s. Within the OSI model, it is an application layer protocol. But at this point the only lower-layer service that it requires is a reliable full-duplex byte stream transport such as TCP. A TCP port number for Z39.50 is registered, and there is a Request for Comment (RFC) that specifies how to use Z39.50 over TCP (see Clifford A. Lynch. "Using the Z39.50 Information Retrieval Protocol in the Internet Environment," Request for Comments: 1729 [December 1994] <http://www.internic.net/rfc/rfc1729.txt>).. Abstract Syntax Notation One (ASN.1) is used to specify the contents of the protocol data units that are passed between client and server, and the Basic Encoding Rules (BER) are used to serialize the ASN.1 structures.

The protocol is stateful and connection-oriented. The protocol defines interactions between two machines only. While some groups are now developing "broadcast search" applications that permit a client to search multiple servers in parallel, these are applications that are built on top of Z39.50 and use multiple concurrent Z39.50 connections to multiple machines. Z39.50 does not specify an applications program interface (API) to the services of the protocol on either the client or the server. It deals only with the interactions between the client and server machines. In addition, Z39.50 does not address any of the issues involved in user interfaces that the client may present or any of the issues involved in database management at the server.

The basic architectural model that Z39.50 uses is as follows: A server houses one or more databases containing records. Associated with each database are a set of access points (indices) that can be used for searching. This is a much more abstract view of a database than one finds with SQL, for example. Relatively arbitrary server-specific decisions about how to segment logical data into relations and how to name the columns in the relations are hidden; one deals only with logical entities based on the kind of information that is stored in the database, not the details of specific database implementations.

One of the basic Z39.50 functions allows the client to transmit a search to the server (a SEARCH request). A search produces a set of records, called a "result set", that are maintained on the server; the result of a search is a report of the number of records comprising the result set. The standard is silent as to whether the result set is materialized or maintained as a set of record pointers, and as to how the result set may interact with database updates that may be taking place at the server. Result sets can be combined or further restricted by subsequent searches. Note that this is substantially different from SQL servers, which do not employ result sets.

Records from the result set can be subsequently retrieved by the client using PRESENT requests. The PRESENT request offers elaborate options for controlling the contents and format of the records that are returned. The PRESENT request indicates specifically which records from the result set are to be retrieved. There are facilities for managing buffer space in the presence of very large records and also for transferring very large numbers of records from server to client without the need for repeated PRESENT requests and responses (and hence many round-trip interactions between client and server).

Z39.50 also contains functions for search management. For example, a server can provide progress reports for an active search, or can ask the client for authorization to continue a resource intensive search; a client can abort an active search. The report for search completion can also return supplementary information such as how many records matched individual component terms in a search.

Z39.50 contains facilities for managing result sets, for sorting result sets, for browsing the values of access points associated with a database, for opening and closing connections, and also a general mechanism called "extended services", which is essentially an asynchronous remote procedure call mechanism that the client can use to invoke services on the server, optionally making reference to the contents of a result set as a parameter. Extended services (which will be discussed in more detail later) were originally intended as a means of saving result sets across sessions, queuing them for print or electronic mail processing at the server, or for registering and managing queries that would be executed periodically on the server.

The protocol also defines the following:

- A query language for specifying searches, which in turn builds upon registered definitions for attribute sets that specify the names of access points;
- Various record syntaxes that can be used for transferring records from the server to the client, including both some application domain specific syntaxes like MARC for bibliographic data and a massively complex, very general purpose syntax called Generalized Record Syntax One (GRS-1);
- A language for describing how to construct records that are to be transferred from a result set back to the client; and

- A facility called EXPLAIN which allows clients to obtain a wide range of information from a server about what databases are available, what access points are supported in each database, and the like. EXPLAIN is modeled as a special-purpose database which is searched using standard Z39.50 queries; the standard specifies the detailed structure of the records that can be retrieved from this database. EXPLAIN is intended to permit the development of clients that to at least some extent are dynamically self-configuring as they encounter various servers.

Z39.50 makes extensive use of registries for various types of objects, such as attribute sets used in queries and record syntaxes used in present requests. These are referred to via object identifiers which are used as parameters in the various protocol requests and responses that move between client and server. Some initial object identifiers are assigned by the standard; assignment of object identifiers on an ongoing basis is handled by the Z39.50 maintenance agency.

A Timeline of Z39.50 Standardization and Deployment

Z39.50 has its roots in efforts dating back to the 1970s to allow standardized means of cross-database searching among a handful of (rather homogeneous) major bibliographic databases hosted by organizations such as the Library of Congress, the Online Computer Library Center (OCLC), and the Research Libraries Information Network. At the time, the primary application was to support shared cataloging using a logical national bibliographic database constructed from this small number of bibliographic utilities rather than to offer end users a common view of large numbers of autonomously managed databases. This program was called the Linked Systems Project. Initially, the participants both wrote protocol specification and worked on implementation; however, by the early 1980s the focus of the Project had shifted to almost exclusively to implementation, and the work on the specifications had been moved into a formal standards development effort under the auspices of the National Information Standards Organization (NISO). (See Clifford A. Lynch and Cecilia M. Preston. "Internet Access to Information Resources," Annual Review of Information Science and Technology (ARIST) Volume 25 (New York, NY: Elsevier, 1990), pp. 263-312. for more details on the early history of Z39.50)

NISO committee D was established in 1979. It operated under the normal rules for traditional standards making bodies: as a small, closed committee of appointed experts who worked very much in isolation from the broader community until the final product of the committee went to ballot, and with a relatively weak connection between the protocol developers and those who would actually implement the resulting standard. After an unsuccessful ballot in 1984, the committee was finally successful in balloting "American National Standard Z39.50, Information Retrieval Service Definition and Protocol Specifications for Library Applications" in 1987; the standard was published in 1988. This document is probably best described, in hindsight, as an unimplementable abomination which should never have been adopted in the form it was. Rooted firmly but somewhat inarticulately within the OSI framework that was evidently mandatory for formal standards making bodies at the time, the context of the 1988 standard was, as its title suggests, information retrieval from bibliographic databases. To the best of my knowledge, outside of the Linked Systems Project context the only "implementation" of Z39.50-1988 was Brewster Kahle's work on the Wide Area Information Server (WAIS) project. The role of Z39.50-1988 in WAIS might best be describe as "inspirational" rather than that of a standard. WAIS never interoperated with anything except WAIS, and freely deviated from Z39.50 both in intent and specifics in the interests of producing a working system. It's a tribute to Kahle and his colleagues that they managed to produce anything useful based on Z39.500-1988.

By the end of the 1980s the community's view of goals for Z39.50 were beginning to change. Indeed, the community interested in using the standard had grown much larger and more diverse than the handful of institutions involved in the linked systems project. The concern was now with end user access to bibliographic and abstracting and indexing databases, and even more general classes of databases. The world now was being viewed as containing many clients and servers -- not just a handful of major bibliographic utilities -- in part because of the deployment of local on-line catalogs into libraries during the 1980s, and in part as a result of the implementation of access to abstracting and indexing databases for the general library patron community rather than specialist searchers. The rapid growth of network-accessible

computers also motivated this changing perspective. The typical application envisioned for Z39.50 at that time was to permit the implementation of a user interface, running either on a timeshared mainframe or a personal workstation (sometimes called a "scholar's workstation"), which provided uniform, consistent access to a range of networked servers hosting content resources.

There were also a messy set of standards issues emerging in the international arena. Parallel to NISO's committee D, an international committee, ISO Technical Committee 46 Subcommittee 4, had been working on a protocol called Search and Retrieve (SR), which was almost identical to Z39.50 except that it used ASN.1/BER as a protocol data unit encoding and omitted a few functions. It was defined by a pair of independently edited documents distinct from the work done in NISO. The international work was standardized in 1991 as ISO 10162/10163. The feeling in the USA was that it was essential that the next version of the US protocol not only be implementable and responsive to the evolving needs of the potential implementor community but that it also be at least compatible with the ISO work, though there was a sense that the requirements of the US community went beyond the functions available in the ISO version.

During 1989-1991, a major shift occurred in the way that Z39.50 protocol development was being handled in the US. The Library of Congress was appointed as the maintenance agency for the standard by NISO; this provided a focal point for the drafting of a revised standard. One of the early assignments of the maintenance agency was to harmonize USA developments with the ISO work. Ray Denenberg of the Library of Congress took on the role of editor for the revised USA standard. Committee D was disbanded, or faded away; functionally it was replaced by a self-selected unofficial group called the Z39.50 Implementorís group (ZIG) convened and chaired by Mark Hinnenbusch of the Florida State Center for Library Automation.

Initially, perhaps 15 organizations were represented on the ZIG, but the meetings were widely advertised and open to all interested parties. The group grew in size rapidly. For the first time, a public electronic mail list was also put in place to facilitate discussion of the revision of the standard, again opening up the process to a much larger range of interests. The process was much more akin to the kinds of standards development efforts one finds in the IETF, though the work was reconnected with the traditional process at the end through a formal ballot to the NISO membership.

The net effect of all of these events was that by 1991, a second version of Z39.50 had been prepared and put out for ballot. This became Z39.50 version 2 or Z39.50-1992. Unlike its 1988 predecessor, Z39.50-1992 had heavy input from a substantial number of people actually building implementations in various environments. It was a compatible superset of the ISO 10162/10163 work that had been done internationally. While still heavily driven by applications involving bibliographic and abstracting and indexing databases, influences such as the work of the WAIS project on full text databases, emerging SGML projects, and similar applications had broadened the sphere of applications and the version 2 standard was actually useable with a very broad range of datatypes, though it did not necessarily have all the flexibility one might want for dealing with them.

Perhaps the greatest problem with the 1992 version of Z39.50 was its continued explicit positioning of the protocol within the OSI framework. Worse, Z39.50-1992 wants to actually make use of certain relatively esoteric presentation layer services (which turned out not to be part of most of the available OSI protocol stack implementations). This was a major barrier to deployment. By 1992, it was already clear to most implementors that OSI had failed, but this was not yet a politically acceptable statement within international standards bodies or certain US government and library circles. There was at least one OSI-based implementation of Z39.50-1992, which was developed but never really much exercised because there was nobody to talk to -- and no way of talking to anyone. In order to move Z39.50 from theory to practice it was necessary to move it into the TCP/IP based environment of the Internet, despite the political controversy that this would entail.

In 1992-1993, a program called the Z39.50 Interoperability Testbed was launched under the sponsorship of the Coalition for Networked Information (CNI). The purpose of this project was to facilitate the development of a large number of interoperable implementations of Z39.50 which ran over TCP/IP and were accessible through the Internet. This effort was a substantial success, and led to a number of

demonstrable Z39.50 clients and servers which could be seen to communicate with each other at trade shows like the American Library Association's exhibits. This was a very novel experience for vendors and librarian-purchasers alike: they could actually put vendor claims of standards conformance to the test by trying to get a vendor's system to communicate with other vendor or university implementations. The vast majority of the implementors that participated in the testbed were library automation systems vendors offering access to bibliographic or abstracting and indexing databases, although universities and bibliographic database access providers also played major roles. In part as a result of the efforts of the interoperability testbed, Z39.50 gained a great deal of credibility in the library automation community and rapidly became part of the specifications most libraries used in the procurement of new library automation systems, thus further encouraging implementations. (The Z39.50 maintenance agency maintains a list of implementors; readers are invited to browse this to get a sense of the range of current implementations of the protocol).

While Z39.50-1992 moved into widespread implementation, the ZIG began work in 1991 on Z39.50 version 3. Version 3 was much more ambitious than version 2. While version 2 built upon the functions of the 1988 version and the ISO work, Version 3 included everything that anyone participating in the implementor's group wanted. It was a consensus document in the sense that all proposed requirements were accommodated. By this time, however, the implementor's group was much larger and more diverse, including major information services providers like Lexis/Nexis, Dialog, and Chemical Abstracts, as well as the traditional constituencies. These new participants brought with them a vast range of new requirements and sometimes a fundamentally different view of the role of standards and interoperability. The resulting version 3 product, balloted in 1995, contained a number of important incremental changes like segmentation (important for high performance on fast networks), sorting, and access point browsing; it also introduced the EXPLAIN database. But version 3 also introduced very complex features like extended services and the generalized record syntax, which were major departures from previous protocol versions, and which were to raise more fundamental questions about the appropriate scope of the Z39.50 protocol and about the nature of interoperability one might expect from conformant implementations. These are discussed later from several different perspectives.

Version 3 was much larger than version 2, weighing in at about 160 pages (as opposed to about 40 for the earlier version). Yet the comparison is a little misleading; in version 2 very little was optional, while the vast majority of the new functionality and changes in version 3 were optional. The actual set of changes necessary to move from a version 2 implementation to a minimal conformant version 3 implementation are not very large, with much of the work for a server being to politely decline to perform various optional functions. An additional reason for the bulk and apparent complexity of version 3 was that it, in fact, included version 2. Version 3 was designed as a superset of version 2, which incorporated the ability to fall back to the older version 2 specification if the parties involved did not support version 3 for the sake of backwards compatibility with the existing base of implementations. This seemed like a good idea at the time. But in hindsight, it is not clear that the amount of confusion and complexity it created in the standard was really worthwhile.

Version 3 of the standard explicitly recognized the TCP/IP Internet environment in an appendix but also contained carefully crafted language which still permitted Z39.50 to be viewed as an OSI protocol by those who wished to do so. This, again, is confusing to today's reader, but was probably politically expedient at the time the standard was balloted.

Since the adoption of version 3 in the USA in 1995, developments have been proceeding in a number of directions. The independent international text of ISO 10162/10163 has been superseded by the international adoption of the NISO Z39.50-1995 text, meaning that there is now only one standards document to work with, rather than multiple documents describing what is hopefully the same protocol. International participation in the Z39.50 Implementor's group has grown substantially, with a particularly heavy representation from Europe but also now growing interest from Australia; the ZIG has been meeting abroad once a year for the past few years. Thus, in a real sense, the whole international Z39.50 community is directly involved in ongoing development of the standard, although through the peculiar mechanism of an unofficial advisory group to the maintenance agency for a US standard. Presumably future versions of

the standard will be balloted within NISO, and perhaps within ISO internationally as well, although they are not being developed within the normal standards development processes for these organizations. With the growth of international participation, there has been an increased focus on issues such as support of multiple character sets and languages.

Various groups have been developing Z39.50 profiles. The maintenance agency keeps a list of these, but the process by which they are approved and subsequently maintained remains somewhat unclear. Profiles are basically customizations of the standard to particular communities of implementors with common applications requirements. A profile may include a whole range of agreements: for example, agreements to use or not to use specific optional version 3 features; agreements on particular attribute sets and record syntaxes to be used (including perhaps the definition and registry of new attribute sets and/or record syntaxes to support the community in question); and even agreements on what extended services will be used (including, again, definitions of new extended services that the profileís community may want to use). Often it is doubtful how much meaningful interoperability will be possible between one Z39.50 implementation that is built according to a given profile and another which is not aware of the specific profile. Examples of profile work include GILS, the Government Information Locator System; the Museum Interchange Profile being developed by the Computer Interchange of Museum Information (CIMI) group; the Digital Collections profile under development by the Library of Congress; the (revised) WAIS profile; profiles for applications involving remote sensing and geospatial data, and a cataloging profile under development by the National Library of Australia.

In some sense, the development of profiles signifies the fragmentation of the Z39.50 implementor community into more specialized and potentially insular sub-communities. To a degree, I believe that it is also a response to the interoperability problems raised by the vast number of optional or incompletely specified features in version 3 of the standard. Finally, one can also view profile development within the Z39.50 community as a response to the lack of other well-defined processes for establishing standards for attribute sets and record interchange syntaxes to support various semantic classes of information resources (such as museum information); these are developed as Z39.50 profiles rather than separate parallel standards that are used in conjunction with Z39.50.

There is work underway on linkages between Z39.50 and various other standards activities. URLs have been defined for Z39.50 database queries, for example. There is an active effort to incorporate SQL as an alternative query language with Z39.50 search requests, although a complete definition of the requirements, limitations, and expected benefits of such an integration remain somewhat unclear. People are beginning to think about how Z39.50 and CORBA might inter-relate.

And, of course, there is discussion about the possible development of version 4 of the standard, about what principles might guide the development of such a version, and what requirements might shape it. At present no consensus exists on such guidelines, and there is no firm commitment or timetable for a new version of the standard. There does seem to be a general feeling that it will be important to simplify and streamline future versions of the standard; that it is important to more rigorously separate semantic definitions that are specific to certain classes of databases, such as attribute sets and record syntaxes, from general protocol mechanisms that are relevant of all databases; and that the elaborate backwards compatibility requirements that characterized the transition from Z39.50-1992 to Z39.50-1995 may not be necessary in future.

The Role of Content Semantics in Z39.50

Z39.50 becomes linked to the semantics of the databases being searched in two primary areas: the attribute sets used to describe the access points being searched, and the record syntax (and related record composition control parameters in PRESENT) that are used to actually transfer records back from server to client. As indicated earlier, because these semantics are typically at the level of logical (intellectual) constructs for classes of databases, Z39.50 offers a much higher degree of abstraction than traditional database management system technology.

In many of the early applications scenarios for Z39.50, particularly in the bibliographic environment, there was a strong (though usually implicit) assumption that both client and server software really embodied deep and sophisticated understanding of data semantics, of the meaning of attribute sets like BIB-1 (which is used for queries against bibliographic data), and of record syntaxes such as MARC. Put another way, there was an implicit assumption that Z39.50 might be buried rather deeply underneath an application at both client and server: The client's user interface would likely do a good deal of processing to translate a user query into a Z39.50 query using the BIB-1 attribute set. Similarly, a substantial amount of work reformatting records received from the server for presentation to the end user would be required. At the server side, elaborate processing might be done in order to translate a Z39.50 query into one or more database queries, perhaps even post-processing the results of the database queries in order to fully implement the semantics of the Z39.50 query in cases where it did not map directly into the capabilities of the server's database system.

Here, attributes from the commonly known attribute set and fields within the commonly understood record transfer syntaxes are operating at a semantic level, independent of the implementation of a database on a given server. One speaks of intellectual ideas like author names and dates of publication, rather than of particular field or column names specific to a given implementation. Automatic configuration, in this environment, basically means that the client and server make reference to attribute sets and record syntaxes that are already mutually understood at a semantic level, and then characterize the specific capabilities of the server, where necessary, in terms of this common understanding of semantics.

By the time of version 3, an alternative model was gaining support in some quarters. In this view, client and server really understood very little of the semantics of the information being searched and retrieved. The responsibility for this was placed primarily on the human user of the client software. Here, the Z39.50 protocol interactions were a much more directly exposed to the user and shared semantics were only used at a mechanical level, for example to agree on the datatype of a particular data element. Neither the client or the server really understood the meaning of the information that was being searched and retrieved. Automatic configuration was not about adjusting client search strategies and Z39.50 query formulation for the peculiarities of a specific serverís capabilities, but about how to tailor a user interface.

To slightly exaggerate this approach: The client should be able to get from the server a list of supported access points along with textual labels for them which are suitable for display to a human user. Based on this, the client throws a search form up on the screen; the user fills in some of the blanks for some attributes, and the client mechanically translates this back into a Z39.50 query. Records are handled similarly, using some general syntax like GRS-1, where the client gets back a series of data elements and textual tags to display with them (perhaps using EXPLAIN to obtain the textual tags).

Clearly, in such an environment, the client software really cannot add much value. It can't help the user with the mapping of searches expressed at an intellectual level into appropriate Z39.50 queries for specific servers, and it cannot, for example, easily fuse data from multiple sources. In a way, the client is just acting as a programmable user interface which is configured by the server. Put another way, one can think of this latter approach as one in which database semantics are reduced to simple syntax. One simply makes reference to arbitrarily-named access points and record data elements, and as side information, users are provided some hints to help them interpret the actual meaning of these arbitrarily named fields.

Semantics are clearly intimately connected to interoperability. In the first view of semantics, that in which applications are very sensitive to content semantics, it is clear that a client encountering a database that employs an unknown attribute set and/or record syntax will be unable to interoperate meaningfully with the server. To make the client and server interoperate, there are three main approaches:

1. Extend the client to know about the characteristics of the new logical class of information (that is the record syntax and attribute sets);

2. Have the server automatically map the semantics of some attribute set already known to the client to the logical access points relevant for the new class of information, and then map the new information into a familiar record syntax for the client (which, in both cases, is likely to be only an

approximate, imprecise and probably incomplete mapping); or

3. Have the client obtain automatic configuration information from the server or a third party which allows similar mappings to be performed at the client -- in essence, use an existing set of semantics that the client knows as a basis for establishing at least approximate semantics for the new class of information.

All of these are hard, and require substantial work on someone's part in order to make a new class of information content broadly accessible to the base of Z39.50 clients. Further, they require community consensus on an understanding of the structure and attributes of various classes of information content. There is an unending list of such classes: bibliographic records, seminar announcements, course schedules, personnel records, gene sequences, descriptions of the properties of chemical compounds. In many cases it is not clear how to define the appropriate community to develop and document such a consensus on how a given class of content is to be structured. Shared semantics within the Z39.50 context presupposes that some basis for defined semantics of a given kind of information object already exists; that it has been codified so that it can be shared. This is not always a realistic assumption, and to some extent the need for it limits the situations in which Z39.50 can deliver its fullest value as a framework for the development of distributed information retrieval applications.

In the second view, where only mechanical semantics are supported, interoperability is not going to be conditional upon any kind of mutual understanding of semantics, except to the extent that special datatypes (for example, as might be needed for chemical structure searching) are required to fully exploit the search capabilities of the server, or specialized display routines might be needed to interpret the datatypes of some of the data elements in the records coming back (for example, animations). In these cases the client will need methods for handling input and/or display of the new datatypes. So, in one sense, there is much broader interoperability; more clients and servers can communicate without prior arrangement or knowledge. Further, it avoids the need for prior consensus on how the semantics of a given class of information should be structured.

Yet, in another sense the loss of abstraction renders this use of Z39.50 much closer to traditional low-level distributed database applications, with all of the limitations, scaling and maintenance problems that characterize these applications, and misses much of the point that motivated the development of Z39.50. At best, I believe that this should be used as an undesirable fallback applications scenario which is invoked only in cases where it becomes clear that there are no shared semantics to build upon, in order to ensure some minimum level of access to exotic information resources that are outside of a given client's basic design objectives.

It is interesting to note that recently there has been considerable work done in developing a very generic set of data elements called the Dublin Core which could be used both as the basis for an attribute set of access points and in the construction of a record syntax. While these could be used as an intellectual set of semantics that would provide wide interoperability (since they can be mapped to most more elaborately structured information resources), their very generality means that the amount of processing a client can perform based on an understanding of their semantics is very limited. Ultimately, they serve as rough classes to which access point values can be assigned, and as tags that can be used to label corresponding data elements when returned records are displayed to users. Z39.50 provides maximum leverage where there is a shared understanding between client and server of rich and specific information semantics.

This is the first of a two-part story; Part II will appear in the May 1997 issue of this magazine.

Editor's note, 1/11/99. Part II was not published as planned.

hdl:cnri.dlib/april97-lynch