Implementing Explain

Denis Lynch
TRW Business Intelligence Systems
dml@bis.trw.com

Abstract

The Explain facility is the primary mechanism for Z39.50 clients to discover servers' capabilities. Explain-based clients can dynamically configure their user interface (or other search capabilities) to exactly match individual servers. This allows generic clients to access a wide range of Z39.50 server, and allows any client to adjust to changes in server configuration.

The Explain facility is defined by an abstract record structure and attribute set, with no additional protocol mechanisms. Still, effectively using Explain is a relatively large undertaking. This paper describes the most important issues to be considered, and suggests the most important features to implement first.

Introduction

The Z39.50 protocol allows clients and servers to work together to provide users with tailored access to information. This flexibility presents a challenge to client developers: how will a client know how to deal with a specific server? The traditional choices have been:

- Make the client very general, but very simple, providing only "least common denominator" access.
- Build the client with specific knowledge of the server(s) it will access, providing tailored access to a limited set of information sources.

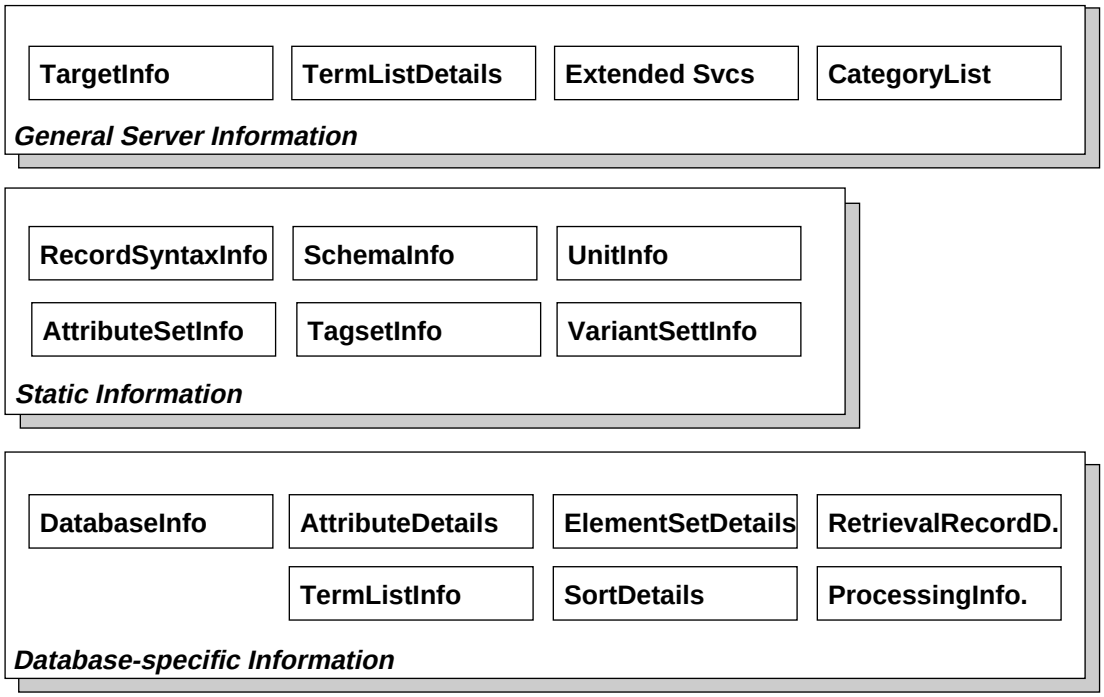
Early in the development of Z39.50 it became apparent that the least common denominator was not useful for most applications. On the other hand, building server knowledge into clients is cumbersome and dangerous: even the most stable environments change; a client with obsolete server knowledge will give its users nasty surprises. The Z39.50 environment reveals this problem more than some others precisely because of the specificity that it supports. For example a user might have a request like "Return Canadian MARC records for all the items in this database with MESH subject starting with 'symptom'". But while such a specific request will get exactly what the user wants when the server supports the specific search access point and record syntax, few servers do so.

Collectively the Z39.50 Implementors' Group (ZIG) determined that dynamic discovery of server capabilities was the most promising way to get good client behavior. The best way to communicate the dynamic data from servers to clients was seen to be the very search and retrieval facilities that are the core of the Z39.50 protocol. Server characteristics were divided into categories, and database record structures were defined to contain the information in each category.

Explain records organize human-readable information as well as information to be used internally by client software. For example, the server description record (TargetInfo) includes separate human-readable items for an overall description, usage restrictions, and operating hours as well as items client software can use directly such as network addresses and the maximum number of result sets supported.

Capturing server information in a searchable database makes supporting Explain relatively easy, since both clients and servers will necessarily have most of the required capabilities. It also meant that the ZIG's energies could be devoted to defining the dynamic information requirements without concern for new protocol capabilities.

Implementation experience has proven that sophisticated clients can be built that use only Explain information to configure their user interface and key operating characteristics.



Overview of the Explain information structure

The remainder of this paper describes the Explain database structure and how clients and servers can use Explain information effectively. The paper does not exhaustively cover the Explain database structure. Readers are referred to the 1995 Z39.50 standard, in particular section 3.2.10, Appendix 3 section ATR.2, and Appendix 5 section REC.1.

The Explain Database

The Explain database operates within the Z39.50 protocol exactly like any other database. Each server's primary Explain database is named IR-EXPLAIN-1; a server may have additional Explain databases as well (see the Surrogate Explain section below). Because predictable behavior and content are important to the internal workings of clients, Explain databases are searched using a dedicated attribute set, and records are retrieved in a dedicated ASN.1 syntax. (Other alternatives, such as the Generic Record Syntax, were considered, but the concreteness and specificity of ASN.1 and the Explain-1 attribute set were compelling.)

An Explain database contains three basic kinds of records:

- Static information that should not vary among servers, e.g. attribute set definitions
- Database-specific information, e.g. available record syntaxes
- Information that applies generally to the server, e.g. extended services available.

General server information

Four types of records apply across databases:

- The TargetInfo record describes the server as a whole
- The CategoryList lists the kinds of Explain records contained in the database
- ExtendedServicesInfo records describe each Extended Service supported by the server
- TermListDetails describe each term list supported in the target (term list names appear in TermListInfo records for specific databases; multiple databases can share the more detailed TermListDetails descriptions).

Static Information

One of the sources of Z39.50's power and flexibility is the degree to which important concepts are modularized into entities that may be defined externally to the standard. Two such entities—Attribute Sets and Record Syntaxes—are central to using Z39.50. Four additional entities play important roles in Version 3: unit systems, database schemas, record tag sets and variant sets. Instances of all six of these entities are identified by an object identifier or a unique name.

Since the definition of each of these entities (for example a specific attribute set) is intended to be universal, clients and servers could have complete *a priori* knowledge of the definitions. In practice, though, clients can allow users to use servers without having built-in knowledge of the semantics of these entities. For example, a client can use information from a UnitInfo record to allow a user to enter a numeric term and specify that it is a length in furlongs, where the unit type and unit (i.e. "length" and "furlongs") are transparent to the client.

The static information categories contain enough details about the entities to support this kind of transparent client behavior.

Like all Explain records these static descriptions can contain human-readable text that clients simply display to users, allowing users to understand aspects of the server's operation that the client software does not necessarily understand.

Database-specific Information

Most of the dynamic content of an Explain database relates to a specific database on a server. Each database is described by a DatabaseInfo record; additional records describe specific aspects of the database:

- AttributeDetails lists the attributes that can be used in a Type-1 or Type-101 query, and how those attributes can be used in combination.
- TermListInfo lists the term lists ("indexes") that apply.
- ElementSetDetails and RetrievalRecordDetails describe record retrieval options.
- SortDetails lists the available sort options.
- ProcessingInformation allows a server to provide clients with specific instructions such as search forms and record formatting.

Surrogate Explain Databases

A single Explain database describes a single information service. There are many reasons that a Z39.50 server might wish to provide Explain data about a service other than itself (for example if the other service does not have an Explain database). We refer to such a server as a "surrogate Explain server," and the corresponding database as a "surrogate Explain database." The mechanism for this is straightforward:

- The surrogate Explain server provides its own Explain data in the standard place: database IR-EXPLAIN-1.
- The surrogate Explain server provides Explain data for additional servers as separate databases. By convention the names of surrogate Explain databases should be the URLs for the server they explain. (That is not a requirement, and there are times when other names *must* be used.)
- Each surrogate Explain database should naturally be described in the IR-EXPLAIN-1 database. Only the DatabaseInfo record is required, and two of its elements are especially important to clients:
  - explainDatabase (a Boolean flag) informs the client that the database is an Explain database
  - lastUpdate gives clients an indication that cached data may be obsolete.

When a server follows these conventions a client can determine what surrogate Explain data is available by searching IR-EXPLAIN-1 for

ExplainCategory = "DatabaseInfo"
AND
ExplainDatabase AlwaysMatches (*any term*).

The client is likely to need the list of databases in any case, so it can simply search for ExplainCategory = "DatabaseInfo"; then Brief records will identify all the server's "normal" databases and surrogate Explain databases.

Using Explain in a Client

Even the most basic Z39.50 client can use dynamic information from Explain to great advantage. The most obvious application is search attributes, but the opportunities are much greater.

- Network addresses. If the Explain data is coming from a surrogate Explain server, the surrogate Explain server might recommend a network address different than the destination server name. It is possible that a non-surro-

gate server would provide alternative addresses—but these would only be discovered after a connection to the original address had succeeded!

- Target operation parameters. The TargetInfo record contains more detail about the server than Init negotiation provides. A client can use the namedResultSets and multipleDBsearch flags for named result sets and multi-database searching to avoid sending searches that the server will fail. (Clients can learn that named result sets are not supported during initialization if Version 3 is negotiated, but not if Version 2 is negotiated.)
- Searching.
  - Whenever a TermListInfo record is available for a database a client should use the listed termLists as the primary search access points. For each access point (or index) to a database, the TermListInfo record includes a title for each term list, intended for display to the user (possibly in multiple languages), an indication of the cost of using that access point, and whether the term list is scannable. The attributes in a term lists's TermListDetails record specify which attribute combinations address that term list.

    A server administrator identifies term lists as specific ways to search a database, so distinct term lists can be presumed to be truly different. On the other hand, the same administrator will list as many attribute values as possible in AttributeDetails, even though some are treated identically.
  - AttributeDetails lists all legal attributes for a database; its attributeCombinations lists how those attributes can be combined in a single operand. Attribute names and descriptions can be found in AttributeSetInfo records; database-specific descriptions may be included in the AttributeDetails.
  - The associatedDbs element of a database's DatabaseInfo record lists the databases that can be searched in combination with that database. (Similar information may be available in the TargetInfo record's dbCombinations element.)
- Retrieval. The ElementSetDetails and RetrievalRecordDetails records associated with a database identify the record syntaxes and element requests that are sensible for that database. Knowing the full list of alternatives allows a client to choose appropriate element set names, even if the standard "F" and "B" are not appropriate. (This could happen, for example, if "F" and "B" are supported only for a record syntax the client can't accept.)
- Scan. Scan requests use attribute combinations to address term lists. A client may choose to attempt to scan with any supported attribute combination, but using only term lists with the scanable flag set will avoid unnecessary errors.
- Processing instructions. These are currently usable only by private agreement between the client and server. General purpose formats will be defined in the future.
- Extended Services. Each Extended Service supported by a server is described by an ExtendedServicesInfo record. Clients can use this record to learn whether ES packages will be retained in the ES database, and whether the request can be issued with the wait flag.
- Access requirements. Both the TargetInfo record and DatabaseInfo records may contain AccessInfo elements. Clients can use this access information to determine if they can usefully access the server and specific databases. The most important element in this regard is restrictedAccess. If restrictedAccess indicates that access to a server or database is restricted, the client will be able to access the server or database successfully only if it supports one of the listed accessChallenges. In some cases a client might learn that a database doesn't support any of the query types the client supports. (This is unlikely, since essentially all clients and servers support Type-1 queries.)
- Human-readable descriptions. The various description fields—particularly in TargetInfo and DatabaseInfo records—should be available from a client's Help system. The information in these fields may explain to a user why operations behave as they do, as well as describing the contents of the databases.

Getting Explain records

Explain databases are accessed using the search and retrieval facilities of the Z39.50 service. The exp-1 attribute set includes USE attributes that are tailored for searching Explain databases. Each exp-1 attribute corresponds to a specific element in an Explain record. Some of the attributes correspond to elements that appear in more than one type of record (e.g. ExplainCategory and DatabaseName), while others correspond to elements that currently appear in only one type of record (e.g. ExtendedServiceOID).

Several attributes require specific search terms, as described in the definition of exp-1.

Exp-1 only defines USE attributes; attributes of other classes are the same as Bib-1 attributes.

Explain databases can be accessed effectively using the Version 2 present facility, in particular simple element set names. The Explain record syntax definition identifies the

elements that must appear in brief records (element set name "B"), these include all of the identifying elements of the record. Full records (element set name "F") must contain all elements present in the database. The element set names "description", "specificExplain" and "asn" are also defined in certain cases as detailed in the syntax definition.

Search

Finding desired records in an Explain database is generally very simple. The canonical search is:

ExplainCategory = *category*
AND
identifier = *value*

For example

ExplainCategory = "DatabaseInfo"
AND
DatabaseName = "CATALOG"

A few records have multi-field keys (e.g. ElementSetDetails and ProcessingInformation) which means that known-item searches require additional terms.

As with most things, it is best to keep the searches as simple as possible. TRW's Explain software, for example, never sends attributes other than USE, and always uses the general choice for encoding search terms.

Retrieval

The safest policy is to assume that a server has 100% recall, but unknown precision. Clients should therefore be prepared to accept records that weren't asked for. In particular, the desired record may not be the first one in a result set. (Perhaps even more annoying: the search may return many records, but *none* is the desired record!) It is wise for the client to remember the result set positions of these "unwanted" records in case they become wanted later (e.g. if a target returns many DatabaseInfo records when only one was wanted, the client can avoid later searches by retaining the positions of the "unwanted" records).

To minimize round-trip delays, all searches should request piggyback presents. In most cases the piggyback should only request brief records, but known-item searches (searches that expect to find no more than one record) could set a small-set size of one, and request full records for the small set.

Explain servers may not handle present requests exactly as issued. The two most obvious things clients should be prepared to handle are:

- Piggyback present is not required, and some common servers do not honor it. Clients should be prepared to issue present requests if a search response contains no response records.
- Brief records are not required, so clients should examine the records they have received before issuing a redundant present request. If any non-brief elements are present the record isn't brief. It would be most irregular for a server that doesn't provide brief records to make other element set distinctions, so in practice it is safe to assume that non-brief records received for "B" are full records.

Although it isn't strictly necessary, specifying the preferred record syntax as Explain is a reasonable safety measure.

Handling errors

The most common error received from Explain searches is "Database does not exist." This should be noted carefully—there's no reason to try again! A server may react in other less helpful ways, for example "Unsupported attribute set" or "No records syntaxes available." It is simplest to assume that almost any error means that the server doesn't support Explain at all. (Bib-1 error 27—Result set no longer exists—is one exception.)

Once connected, servers frequently time out idle connections. Since Explain is best handled "behind the scenes," the connection should simply be re-established the next time it is needed.

Default configuration

The conventions described in this section are used by TRW's software. They have been proposed to the ZIG as possible implementors' agreements.

For servers that provide many similar databases it may be simpler to describe them only once. The easiest way to do this is to describe a database named "Default". A client that finds no record of a particular type (e.g. AttributeDetails) for a specific database should try to find the same record for the database "Default". In many cases that record will have already been retrieved as fallback for a different database.

A surrogate Explain server can provide default client configuration information for specific destination servers, as

well as providing fallback default behavior. To obtain information about a destination server from a surrogate Explain server, a client searches a sequence of Explain databases. (An Explain database is searched only if the surrogate Explain server's IR-Explain-1 database has a DatabaseInfo record for that Explain database.)

- The URL for the destination server is used as a database name (e.g. the database name "z39.50s://rlg.stanford.edu" is used when looking for information about RLG's server)
- A similar string is used to find a database of default information to be used for the specific access (e.g. "z39.50s://default" for Z39.50 access)
- A fallback database is used for generic default information ("default://default")

The default Explain databases are structured exactly like other surrogate databases.

Minimizing impact on servers

Explain searches require roughly the same amount of server attention as real information searches. This makes it important that clients behave reasonably! Here are a few techniques:

- Cache explain records, and don't ask the server for information that has already been retrieved. Generally it's not a good idea to "download" a server's whole Explain database—just cache the records that are retrieved in the normal course of operation.

  The biggest problem with cached records is knowing when to throw them away. The strategy TRW adopted is to delete all cached information about a database whenever the CommonInfo in a DatabaseInfo record has an update (or creation) date more recent than the corresponding DatabaseInfo record in the cache. This works well because the DatabaseInfo records are retrieved from the server on a regular basis (see the next item), and the CommonInfo is in brief records.
- Keep lists, and check the appropriate list before asking the server a question. This technique applies especially to DatabaseInfo records. A client can avoid searches by noticing that there is no DatabaseInfo record for the relevant database (e.g. when looking for AttributeDetails for a specific database, or for surrogate Explain information about a specific server). One scheme for managing these lists is as follows:
  - Initialize the list before any records are retrieved from the cache. (Each cache will have several lists.)
  - As records are retrieved from the cache, add them to the appropriate list.
  - When a specific record is needed, retrieve records from the cache until either the record is found or the whole cache has been retrieved.
  - When all the records have been retrieved from the cache and the needed record hasn't been found, ask the server. But instead of asking the server for a specific record, ask for the list (e.g. "all DatabaseInfo records"). As the records from this list are retrieved the local cache can be checked for out-of-date information.
  - After the server's list has been completely retrieved a record that isn't in the list doesn't exist—there is no need for additional searches.
- If a surrogate server is supplying the Explain data, check for the destination server's surrogate Explain database in the list of Explain databases before accessing the surrogate database.
- Remember result set positions of records to avoid redundant searches. As Explain records are encountered in Present responses, the client should retain the result set name, result set position, and identifying information (ExplainCategory plus, for example, DatabaseName). When the client needs a record that was incidentally received, this retained information lets the record be Presented from the existing result set without another Search.

Building an Explain Database

"Using Explain in a Client", above, points out most of the issues with populating an Explain database. This section points out a few additional considerations for server administrators.

- Make use of default configurations wherever possible. In a surrogate Explain system it makes sense to provide default TargetInfo records, for example to make clients presume that multiple database searching is not supported.
- Consider carefully what attributes to include in Explain. There is no reason to list attributes that are treated as synonyms just for compatibility. Clients that rely on dynamic configuration will be misled by these attribute aliases; clients that don't use the dynamic configuration won't notice that the aliases are missing.
- Provide term list information if at all possible.
- Include AttributeCombinations to let clients filter out illegal search requests. The AttributeCombinations structure is designed to avoid combinatorial explosion by specifying patterns that describe a set of similar combinations, for example:

- With USE attributes 4, 20, 21, 62 and 1000 the RELATION attribute may be omitted or value 3 may be supplied, and the STRUCTURE attribute may be omitted or values 1, 2 or 6 may be supplied
- For USE attributes 30, 31, 32, 1011 and 1012 the RELATION attribute may be omitted or values 1, 2, 3, 4 or 5 may be supplied, and the STRUCTURE attribute may be omitted or values 5 or 100 may be supplied.

- There is no need to include individual attributes in an AttributeSetInfo record if no databases on the server use those attributes.
- If a CategoryList record is provided, it should list only categories for which at least one record is available. The primary purpose of the CategoryList is to allow clients to learn about Explain extensions supported by the server, but it can also allow clients to skip searches for records that the server doesn't have.

Serving Explain Records

Any fairly capable Z39.50 server should be able to process Explain requests. Unlike user-initiated requests, Explain requests are generated from low-level client code. They will generally be fairly simple, and will not take advantage of Explain information themselves. Servers should therefore make every attempt to process Explain requests without returning the kind of diagnostics that might help a human user refine a request.

There are a few specific requirements:

- The Explain records syntax and attribute set must be supported.
- Since Explain searches will be generated automatically by the client software, support for named result sets is nearly mandatory. The cost of retaining the result sets will surely be less than the cost of re-executing searches.
- If Version 3 is negotiated, the search engine must be prepared to process search terms sent in the OBJECT IDENTIFIER choice.
- In Version 2 or Version 3, the search engine must be prepared to deal with object identifier terms sent as character strings as specified is Appendix ATR.2, note (4).
- Support of element sets other than "F" is not required. But a Brief record should *never* contain any non-brief elements: clients need to be able to look at a received record and determine whether it is Full. As described above, clients will reasonably presume that records with non-brief elements are non-brief records.
- Support for piggyback present is not required.
- Unlike processing user-requested searches, a server should simply ignore Explain search operands with unsupported attributes. If the server fails the search instead, the client will (at best) reformulate a broader query without the offending attribute. Ignoring the unsupported operand avoids this inevitable round trip.
- A useful Explain service can be provided with only a few attributes. The minimal set of attributes is very short:
  - ExplainCategory
  - DatabaseName
  - TermListName (if term lists are provided)

  The next attributes to add are the identifiers from other records supported by the server:
  - AttributeSetOID
  - RecordSyntaxOID
  - TagSetOID
  - SchemaOID
  - ExtendedServiceOID
  - VariantSetOID
  - UnitSystem

  The next tier of attributes is:
  - HumanStringLanguage (if there is more than one)
  - DateChanged
  - DateAdded
  - DateExpires

Only a very simple search engine and no specialized database are required. The Explain records can simply be stored as BER-encoded files. The TRW Explain Editor and server add some small twists to this:

- Each Explain database is stored as a separate directory.
- Within an Explain Database directory, subdirectories are created to store records pertaining to specific databases.
- Records that are not specific to a single database are stored in the top-level Explain Database directory.
- Two special files are created in every directory:
  - title.trw contains the title of the database. Our software makes no use of directory names because of character set and length restrictions.
  - contents.trw contains a list of all the files in the directory along with the values of the searchable fields (including creation/update dates). This isn't much data, but it allows the directory to be indexed or searched without accessing the actual Explain records.

Conclusion

The Explain database structure appears daunting to new readers in size and complexity: its ASN.1 definition is slightly larger than the definition of the body of the Z39.50 protocol, and the records are closely interrelated (like the protocol itself, the Explain database was designed for economy and modularity).

This paper has described how the most general parts of the Explain database are used in practice, and how software can be designed to provide and access Explain databases efficiently. The paper is based on experience gained in building a reasonably complete end-to-end Explain system: a graphical editor for creating and maintaining the data, server extensions to provide the data, and a client that relies exclusively on Explain records for configuration.

The author can be reached at dml@bis.trw.com.