Z39.50 Profile for Simple Distributed Search and Ranked Retrieval

Preliminary Draft 5
March 10, 1997

Comments on this preliminary draft solicited through March 24, following which Draft 5 will be issued.

0. Introduction

0.1 Background and Status

This Z39.50 profile, ZDSR, specifies Z39.50 procedures to support distributed searching and ranked retrieval. It stems from the *Stanford Protocol for Internet Search and Retrieval* (STARTS), an initiative of the Stanford Digital Library Project. The STARTS project developed requirements for distributed searching and ranked retrieval. This Z39.50 profile is based substantially on those requirements. It has been developed by Z39.50 implementors, participants in the STARTS project, and other interested parties.

In this profile, for purposes of searching, Z39.50 database records are documents (with associated metadata); for retrieval purposes, Z39.50 retrieval records are document descriptors. ZDSR assumes that queries pertain to *documents*, and for each document there is a *document descriptor* consisting of metadata about the document including a pointer which may be used to retrieve the document; document retrieval is otherwise out-of-scope of this profile.

0.2 Requirements and Assumptions

Following is an informal list of requirements and/or modelling assumptions.

1. This profile is intended to support distributed searching and ranked retrieval. In the distributed search model a client sends a query to a intermediary, or meta-searcher, which relays the query to several real information sources, integrates the results and presents a single, logical result set to the client. The end-client and intermediary together constitute the client, from the perspective of this profile. The profile does not address how servers are selected. It does not specify procedures for merging and ranking results, though it does support the exchange of information intended to facilitate merging and ranking.
2. The query includes a Restriction component and a Ranking component. The Restriction component is a boolean query specifying the documents that qualify for the answer. The ranking component is a list of boolean expressions (any of which may be single operands, effectively single terms, any of which may or may not be included within the restriction component). Each expression may be assigned a relative weight for ranking purposes.

   Note: The restriction component is represented by the type-1 query included in the body of the Z39.50 Search request. The ranking component is represented by a sequence of type-1 queries included in the additionalSearchInformation parameter of the Search request.
3. Searching by title and date-last-modified must be supported. Support for searching by author, language, url, and body of text, relevance feedback, stem and phonetic searching, and truncation, is recommended. Search results may be restricted by threshold score, or maximum number of documents. A query operand may indicate the language of the term, that a term is case-sensitive, that thesaural

expansion is desired, or request that the server not treat any words within the term as a stop word. The server may indicate, for each term in a query, how many documents contained the term.

4. Boolean Operators And, Or, And-not, and Proximity (specifying distance and whether *order* is significant) are supported.
5. A client may specify a sort order for the retrieved document descriptors. The client may specify sort criteria along with a search, or the client may request that the server sort the result set following completion of a Search operation (including successful execution of the query and the creation of a result set of document descriptors at the server).
6. Searches pertain to *documents*, however, the Z39.50 records retrieved are *document descriptor*s (corresponding to the documents) consisting of metadata about the document (see section 8), including a pointer which may be used to retrieve the document. Document retrieval is otherwise out-of-scope of this profile.
7. Servers provide results for a well-known sample document collection and a set of well-known queries. This is intended to allow a client to calibrate document scores from different sources.
8. Metadata for a Database (i.e. Z39.50 databaseInfo Explain information) is supported (see section 9).
9. Support for Z39.50 encapsulation is recommended. See section 11.

0.3 Z39.50 Services

This profile specifies the use of the following Z39.50 services: Init, Search, Present, Sort, and Close, as well as the use of Encapsulation.

1. Initialization

1.1 Protocol version

Support for version 3 of Z39.50 as specified by Z39.50-1995 is required.

1.2 Id/authentication

A client should support the capability to provide a user id and password by implementing the IDAuthentication parameter of the Init request, using the format provided in the commented description of idauthentication within the Z39.50 ASN.1. The client should implement 'userId' and 'password' of 'idPass', as well as 'anonymous'.

This requirement is imposed on the client based on the expectation that there will be servers implementing the profile who require the client to supply a userid and password. No such requirement is imposed on the server.

1.3 Message Size

For values of preferred-message-size and exceptional-record-size, the client must accept values of zero in an Init Response, meaning the server indicates that the client must be prepared to accept arbitrarily large records and arbitrarily large messages.

In addition, the client may, but is not required, to supply values of zero for both parameters, explicitly indicating "no preference".

2. Search

2.1 Attributes

A client must be able to send, and a server accept, a properly constructed type-1 query composed from the attributes listed in section 10.

2.2 Database Names

The Search request should specify a single database name.

2.3 Boolean Operators

Boolean Operators And, Or, And-not, and Proximity are to be supported. Proximity is expressed in terms of *words*, that is, whether the two terms occur within the specified number of words. The server should support the 'ordered' flag in the proximity expression, indicating whether order is significant.

2.4 Named Result sets

Support for named result sets is not required.

2.5 Term Type

Servers should support search terms of ASN.1 types OCTET STRING (binary), INTEGER (number), GeneralString (characterString), and GeneralizedTime (dateTime).

2.6 Ranking Component

The search request may include a ranking component (included within the additionalSearchInformation parameter). The ranking component is a list of boolean expressions (each is a type-1 query).

For example, if the ranking expression consists of a set of terms, this means, informally, that a documents with more of these terms would be assigned a better score than a document with less of the terms. In addition, any term within any expression in the ranking component may be assigned a weight, via the weight attribute (see 10.3.5).

2.7. Ranking Algorithm Id

The Search request may include a ranking algorithm identifier (included within the additionalSearchInformation parameter). If so, the client requests that the server use the identified algorithm for ranking results.

> *Note: The Z39.50 Maintenance Agency will establish and maintain a register of public ranking algorithm identifiers. Currently, the list is empty.*

When the client does supply this parameter, it is assumed that the client knows that the identified algorithm is supported for the database (it may learn which are supported from the Explain databaseInfo record). It is not assumed however that the client knows anything about the algorithm identified, other than its identifier.

This capability is provided for the circumstance where a query is to be sent to multiple servers, and the client has determined that a specific algorithm is supported by all of the servers, so it might request all of the servers to use that algorithm, so that ranking consistency across the servers might be improved.

The server is not obligated to use the identified algorithm, nor even to recognize or acknowledge that it was requested. The server may, optionally, indicate in the response (within the additionalSearchInformation parameter) the identifier of the actual ranking algorithm used. This profile recognizes that some servers will always use their own proprietary algorithm, which might not have a public identifier.

2.8 AdditionalSearchInformation in Search Request

The Search request may optionally include the AdditionalSearchInformation parameter. The information to be included in this parameter is one or both of the following:

- The ranking component of the query (as described in 2.6).
- A ranking algorithm id (as described in 2.7).

2.9 OtherInfo in Search Request

The Search request may optionally include the OtherInfo parameter, containing encapsulated PDUs. See section 11. In particular, a Sort APDU may be encapsulated, to specify sort criteria. See section 3.

2.10 Retrieval Records in Search Response

A server is not required to include retrieval records in the Search Response. If the combination of values of Search request parameters (Small-set-upper-bound and Large-set-lower-bound) and Search response parameter Result-count are such that Retrieval records would (according to the procedures in the standard) be included, the server may chose to supply the records, or may instead supply a value of 'failure' for the parameter Present-status, along with an appropriate diagnostic.

2.11 AdditionalSearchInformation in Search Response

The Search response may optionally include the AdditionalSearchInformation parameter, including one or more of the following:

- Actual restriction component executed. If the search was successful, the server may supply the actual type-1 query (restriction component) that was executed (which may or may not be identical to the restriction component supplied on the request).
- Recommended restriction component. The server may include a recommended restriction component. This may be supplied if the search failed, or even if the search was successful but the restriction component supplied on the request differed from that which the server is recommending.
- Actual ranking component used. If the search was successful, the server may supply the actual ranking component used (which may or may not be identical to the ranking component supplied on the request).
- Recommended ranking component. The server may include a recommended ranking component. This may be supplied if the search failed, or even if the search was successful but the ranking component supplied on the request differed from that which the server is recommending.
- Identifier of actual ranking algorithm. See 2.7.

2.12 OtherInfo in Search Response

The Search response may include the OtherInfo parameter, containing encapsulated PDUs if applicable. See section 11.

3. Procedures for Specifying Sort Criteria

A client may request that the results of a search be sorted, either by specifying sort criteria along with a search (see 3.1) or, following the search, by requesting that the result set be sorted (see 3.2).

3.1 Sort Criteria Encapsulated in Search Request

The client may include a Sort APDU encapsulated within the OtherInfo parameter of a Search request.

3.2 Post-Search Sort Request

Following completion of a Search operation including successful execution of the query and the creation of a result set of document descriptors at the server, the client may request that the server sort the result set, by sending a Sort APDU.

3.3 Sort Criteria

In either case (whether the Sort APDU is encapsulated in a Search APDU or it is sent after the Search operation), the sort keys may include the following: rank, score, title, creator, author, publisher, contributor, publication date, date first created, date current form created, date last modified, date valid from, date valid to, url, mime type, token count, word count, byte count (these correspond to metadata elements listed in 8.2). The request must include a primary sort key and may include one or more ancillary sort keys.

The Sort request may also specify the sort order: ascending or descending.

3.4 Behavior when Sort Key is not Supported

If the server cannot support the primary sort key, it should fail the search. However, when the server can support the primary sort key but cannot support one of the ancillary keys, the Z39.50 base standard does not address the server behavior. It is a requirement of this profile that the client be able to specify such behavior; i,e. when the request includes one or more ancillary keys, the client may indicate:

(a) If the server cannot support all of the keys (primary as well as all ancillary keys) fail the Sort; or

(b) as long as the primary key is supported, do not fail the sort just because one or more of the ancillary keys is not supported.

The two object identifiers 1.2.840.10003.x.y and 1.2.840.10003.x.z respectively correspond to the semantics in (a) and (b). One or the other of these oids may be included in the OtherInfo parameter of the Sort request. (The OtherInformation parameter should omit 'category', select 'oid' for the CHOICE for 'information', and supply one of these oids as the value.)

4. Retrieval of Document Descriptors

Following successful completion of a Search operation and the establishment of a result set, where each result set item identifies a document descriptor (which in turn identifies a document) the client may use the Z39.50 Present service to retrieve one or more of the document descriptors.

Each document descriptor includes one or more of the elements listed in section 8, supplied according to the document descriptor schema and GRS-1 record syntax.

When requesting retrieval of document descriptors, the Present request should include the parameter compSpec, indicating the document descriptor schema and GRS-1 record syntax.

5. Retrieval of Documents

A document descriptor contains metadata about a particular document. One of the metadata elements may be a pointer (see element *'linkage'*). When the client has retrieved a document descriptor, the pointer (if supplied) is intended for the client's use in retrieving the actual document. However, document retrieval is otherwise outside the scope of this profile.

6. Retrieval of Database and Server Metadata

Servers will provide database metadata via the Z39.50 Explain facility. Servers will maintain an Explain database (database with name IR-Explain-1), support queries with attributes from the exp-1 attribute set (servers will support queries on the Explain database for the purpose of searching for DatabaseInfo Explain records; i.e. queries composed of a single operand where AttributeSetId = exp-1; Use attribute = ExplainCategory; Term = databaseInfo) and return explain records supplied according to the database descriptor schema (see section 9) and GRS-1 record syntax.

When requesting retrieval of database descriptors, the Present request should include the parameter compSpec, indicating the database descriptor schema and GRS-1 record syntax.

7. Character Set

Character strings (name and message strings) are to be Unicode sequences using UTF-8 encoding.

8. Document Descriptor Schema

8.1 Tag Types

For this schema a GRS-1 record will use the following tagTypes:

TagType	Usage
1	Elements from tagSet-M defined in Z39.50-1995. Appendix TAG, TAG.2.1.
2	Elements from tagSet-G defined in Z39.50-1995, Appendix TAG, TAG.2.2. Note: both tagSet-M and TagSet-G have been extended. See http://lcweb.loc.gov/z3950/agency.
3	Reserved for tags locally defined by a target.
4	Tags local to the abstract record structures defined in 8.2.

8.2 Abstract Record Structure

The table below defines the elements that may be included in a document descriptor. All elements are optional. When a server presents a GRS-1 retrieval record for this schemas the record may include any or all elements below.

In the tag path column below (as well as in section 9) the notation (x,y) means "element y from tagSet x"; the notation (x,y)/(z,w) means subelement (z,w) of element (x,y).

Element	Tag Path	Datatype	Note
rank	(1,10)	Integer	Range: 1 to resultCount.
score	(1,18)	Integer	Value from 0 to 100.
ddCreationDate	(1,15)	GeneralizedTime	of document descriptor
ddDateLastModified	(1,16)	GeneralizedTime	of document descriptor
title	(2,1)	GeneralString	
subjectThesaurus	(2,21)	GeneralString	
controlledSubjectTerm	(2,22)	GeneralString	repeatable
uncontrolledSubjectTerm	(2,23)	GeneralString	repeatable
pseudoAbstract	(2,17)	GeneralString	
creator	(2,36)	GeneralString	
author	(2,2)	GeneralString	
publisher	(2,37)	GeneralString	
contributor	(2,38)	GeneralString	
publicationDate	(2,4)	GeneralizedTime	of document
dateFirstCreated	(2,39)	GeneralizedTime	of document
dateCurrentFormCreated	(2,40)	GeneralizedTime	of document
dateLastModified	(2,41)	GeneralizedTime	of document
dateValidFrom	(2,43)	GeneralizedTime	of document
dateValidTo	(2,44)	GeneralizedTime	of document

(continued next page)

(Document Descriptor Abstract Record Structure -- continued)

Element	**Tag Path**	**Datatype**	**Note**
resourceType	(2,24)	GeneralString	
linkage	(4,1)	(structured)	repeatable
url	(4,1)/(2,33)	GeneralString	
mimeType	(4,1)/(2,32)	GeneralString	
relation	(2,35)	GeneralString	
source	(2,45)	GeneralString	
languageOfResource	(2,20)	GeneralString	
spatialCoverage	(2,46)	GeneralString	
temporalCoverage	(2,47)	GeneralString	
rights	(2,34)	GeneralString	
tokenCount	(4,3)	integer	
numberOfWords	(4,4)	integer	
numberOfBytes	(4,5)	integer	
termMetaData	(4,6)	(structured)	repeatable
term	(4,6)/(4,7)	GeneralString	
termFrequency	(4,6)/(4,8)	integer	
termWeight	(4,6)/(4,9)	integer	
private	(4,10)	GeneralString	repeatable

9. Database Descriptor Schema

9.1 Tag Types

For this schema a GRS-1 record will use the following tagTypes:

TagType	**Usage**
1-3	As in 8.1.
4	Tags local to the abstract record structures defined in 9.2.

9.2 Abstract Record Structure

The table below defines the elements that may be included in database descriptor. All elements are optional. When a server presents a GRS-1 retrieval record for this schemas the record may include any or all elements below.

Element	Tag Path	Datatype	Note
databaseName	(4,1)	GeneralString	
minScore	(4,2)	null or integer	null = - infinity
maxScore	(4,3)	null or integer	null = + infinity
rankingAlgorithmId	(4,4)	GeneralString	Repeatable
tokenizerId	(4,5)	GeneralString	Repeatable
samplePointer	(4,6)	GeneralString	
stopWordPointer	(4,7)	GeneralString	
contentSummaryPointer	(4,8)	GeneralString	
rankingExpressionSupport	(4,9)	boolean	

(continued next page)

(Database Descriptor Abstract Record Structure -- continued)

Element	Tag Path	Datatype	Note
filterExpressionSupport	(4,10)	boolean	
AttributeCombination	(4,11)	(structured)	Repeatable.
attributeSetId	(4,11)/(4,12)	Object Id	may be omitted if same as previous
attributeType	(4,11)/(4,13)	integer	may be omitted if same as previous
attributeValue	(4,11)/(4,14)	int. or GString	
subDb	(4,15)	GeneralString	Repeatable. Occurs when database is a logical db, really a combination of other, "real" dbs

10. Query Attributes

Queries are constructed from the following attribute sets.

- bib-1. See 10.1.
- gils. See 10.2.
- ZDSR. See 10.3.

10.1 Bib-1 Attributes

Following bib-1 Use attributes must be supported:

Use attribute	Term
date/time last modified	GeneralizedTime
Any	InternationalString
Title	InternationalString

Support for the following bib-1 attributes is recommended:

Use attribute	Term
Author	InternationalString
Body-of-text	InternationalString
Language	InternationalString
Subject	InternationalString
Publisher	InternationalString
Date-of-publication	GeneralizedTime

Structure Attributes	Truncation attributes	Relation Attributes	Relation Attr. (cont)
Document-text (see 10.4)	left-truncation	equal	not equal
	right-truncation	less than	stem
		greater than	phonetic
		greater than or equal	relevance (see 10.4)
		less than or equal	

10.2 GILS Attributes

The following GILS Use attribute must be supported:

Use attribute	**Term**
Linkage (url of document)	InternationalString

Support for the following GILS Use attributes is recommended:

Use attribute	**Term**
Linkage-type (mime type)	InternationalString
cross-reference-linkage (urls within document)	InternationalString

10.3 ZDSR Attribute Set

The ZDSR attribute set defines the following attribute types:

Attribute	Type
Use	1
Modifier	2
Language-of-term	3
Count	4
Weight	5

10.3.1 ZDSR Use Attributes

Use Attribute	Term	Description
score	integer	For example to restrict results based on a threshold sco re, Use: score; relation: GreaterOrEqual; term: the threshold score.
rank	integer	For example to restrict the results to N documents, Use: rank; relation lessOrEqual; term: N.
Contributor	InternationalString	
dateFirstCreated	GeneralizedTime	
dateCurrentFormCreated	GeneralizedTime	
dateLastModified	GeneralizedTime	
creator	InternationalString	
description	InternationalString	
ResourceType	InternationalString	
Relation	InternationalString	
Source	InternationalString	
spatialCoverage	InternationalString	
temporalCoverage	InternationalString	

10.3.2 ZDSR Modifier Attributes

Value	Meaning
1	case-sensitive
2	thesaurus
3	noStopWord

If case-sensitive is present, this indicates that the term is case-sensitive. If case-sensitive is not present, the server may assume that the term is not case-sensitive.

If thesaurus is present, this indicates that thesaural expansion is desired.

If noStopWord is present, the client is requesting that the server not treat any word within the term as a stop word.

These three modifier attributes are independent and may occur in combination. However, noStopWord may not occur in the returned query (either actual or recommended) under any circumstances.

10.3.3 ZDSR Language-of-term Attribute

The Language-of-term attribute value is a character string based on RFC 1766 (http://andrew2.andrew.cmu.edu/rfc/rfc1766.html).

10.3.4 ZDSR Count Attribute

The Count attribute is meaningful only in a returned query in the Search response. A Count attribute may be attached to any term in a returned query, and its value is the number of documents in which the term occurs.

Although the Count attribute is meaningful only in a returned query, it may occur in a *submitted* query but should be *ignored* by the server. (The server should not infer any semantics based on the occurrence of the Count attribute, however, nor should the server treat its occurrence as an error, because the client may have simply resubmitted a query previously returned by the server, where the server included a Count attribute.)

10.3.5 ZDSR Weight Attribute

The Weight attribute applies to terms in the ranking component only. It is the weight assigned to the term, for purposes of assigning scores to documents. It is an integer from 0 to 1000.

This attribute is intended primarily for inclusion in the ranking component of the Search request. However, the server may include this attribute in the returned ranking component, to reflect the actual value used when the query was executed (or to indicate a recommended value to use for a re-submitted query); the value may be the same as, or different from, the value in the submitted ranking component.

10.4 Relevance Feedback

This profile specifies *Relevance Feedback by Document Text*, RFDT. Other forms of relevance feedback, for example, relevance feedback by document id, are not addressed by this profile. RFDT applies when a client wishes to locate documents relevant to a specific document, and supplies the text of that document. An RFDT query is formulated as follows:

Use:	Any (bib-1)
Relation:	Relevance (bib-1)
Structure:	Document Text (bib-1)
Term:	the document text

Support for RFDT is not required, however a server is required recognize that a query formulated as such is an RFDT query. If a server receives an RFDT query and does not support RFDT, it should fail the search and supply an appropriate diagnostic.

11. Encapsulation

This profile recommends support for encapsulation, the Z39.50 feature permitting several operations to be performed with a single exchange of messages (i.e. a single round-trip) between client and server. Encalsulation may also permit the server to perform various optimizations. Examples:

- Using encapsulation a client can package a Search request together with a Sort request, in a single message (the Sort request is "encapsulated" within the Search request) and the server responds by packaging a Search response together with a Sort response in a single message (the Sort response is "encapsulated" within the Search response). In this example, the advantages of encapsulation are twofold: (1) the two operation are performed by a single exchange of messages rather than two exchanges; and (2) the client is able to specify sort criteria along with the search request as opposed to after-the-fact, and this may allow the server to optimize the search, knowing in advance the desired sort-order.
- A client may package an Init, Search, Present, and Close request in a single message (by encapsulating the Search, Present and Close requests within the Init request) and the server would respond by packaging an Init, Search, Present, and Close response in a single message (by

encapsulating the Search, Present and Close response within the Init response). In this example, the entire z-association is accomplished in a single Z39.50 operation, simulating a connectionless transaction.

- A client may package a Search and Present request, simulating a search with piggybacked present.
- A client can package a Search, Present, and Delete request together. In this example, the client is able to convey the fact that it seeks (for example) a single known record, and does not wish that the server maintain a result set. The server may, in this instance, optimize and not even create a result set.

For a definition and description of Z39.50 Encapsulation, see http://lcweb.loc.gov/z3950/agency/agree/encapsul.html.