Preface: Potential Usage & Users of Ranked List Query

The Type 102 Ranked List Query (RLQ) was originally intended as a natural language query. Because of the vast number of natural language searching methodologies being used today it would be virtually impossible to design a query type that adequately services all of them. Instead, the Type 102 RLQ has been specifically designed to accommodate the ranked searching technologies used by the majority of large-scale commercial information providers and information industry software vendors. This includes 80-90% of the mainstream commercial ranked searching technologies including those used by organizations such as:

- Chemical Abstracts Service
- Clearinghouse for Networked Information Discovery & Retrieval(CNIDR)
- Excalibur Technologies Corp.
- Knight-Ridder Information Services
- LEXIS-NEXIS
- National Institute of Health (National Library of Medicine)
- West Publishing Company

These organizations have actively participated in the Advanced Search sub-group of the Z39.50 Implementor's Group and expressed the usefulness of the Type 102 RLQ to their organizations.

All of the ranked searching methodologies used by mainstream relevance ranking search systems share a number of common aspects. The following common aspects have been generalized and used as the basis for the Type 102 RLQ definition:

1. Results Ranking: Ranking of search results based on various relevancy criteria and ranking functions is fundamental to all relevance ranking search technologies. The Type 102 RLQ supports mechanisms which allow the client/origin to provide a server/target with all of the query information it needs to perform it's particular ranking function.
2. User/client hints about the importance of individual query components: The Type 102 RLQ provides a wide range of methods for the client to indicate which components of the query are most significant to the user. This includes: the weighting of terms and operators, a variety of ranking operators, query reformulation options, etc.
3. Relevance Feedback: The Type 102 RLQ supports this industry standard mechanism which allows the user/client to 'seed' the server's search process by indicating records which are either precisely on-point or are totally off-the-mark depending upon the user's information 'need'.
4. Restriction of the search scope: The Type 102 RLQ definition provides the ability to restrict the set of records which are eligible for input to the ranked search process based on various criteria, including Boolean search restrictions. This mechanism, which is a particularly useful in the case of large source collections, is used in most ranked search environments to provide a concrete, deterministic way of limiting the input set.
5. Query Reformulation: The Type 102 RLQ allows the user/client to specify that they would like the server to expand the elements of the query based on their need for additional precision or recall. The client can specify a general need for query expansion, or can denote explicit dimensions in which to expand the query (such as thesaural or morphological expansions). The client may then ask the server to return the reformulated query so that the user can inspect it and make any necessary modifications before it is executed (or reformulated again). This provides a mechanism by which query generation is an iterative process complete with server recommendations.

6. Precision vs. Recall control: Many ranking mechanisms allow the user to determine whether precision or recall is more important. The Type 102 RLQ includes a mechanism by which the client can instruct the server to perform the search in such a way as to enhance either precision or recall.

The combination of these tools and mechanisms is aimed at producing a robust, extensible, and widely useful ranked query specification protocol. As stated above, it is applicable to a broad range of commercial information suppliers and vendors which serve customers across a number of different industries.

1.0 Ranked List Query Background & Introduction

In the current universe of information search & retrieval users & service providers, there is a wide diversity of different search techniques & methodologies. These range from simple document scanning tools such as those found in word processors, to Boolean search systems allowing a fine grain of user control, to various different forms of relevancy-based systems, to full natural language processing, modal logic, LSI, & connectionist search systems. Across this spectrum there are many variations in query syntax, & in the degree of control given to the user/client over the exactness of the interpretation of search terms, as well as over the precision & comprehensiveness of the results selected from the target collection(s).

In the interest of simplicity & clarity, the universe of this paper has been narrowed down to those search methodologies which support relevancy ranking of results. This ranking is based on the evaluation of selected records (restricted by specification of one or more collections and, optionally, an RPN query which resolves to a set of result records) & their ability to satisfy criteria specified by the components of a Type 102 Ranked List Query (RLQ). Further, we have restricted the problem space to be addressed by the Ranked List Query to those methodologies which apply the query expressions to each record in a collection, & select relevant records independently of any prior ranking/scoring of those records in the overall collection(s) being searched.

Once this evaluation has been made, it is further expected that each relevant record will be assigned an absolute ‘score’ (rank), & that these scores will be returned (or made accessible) to the client which initiated the search. For the Type 102 query, the results of record ranking will be communicated by way of an RSV (Retrieval Status Value) for each record in the Result Set (see discussion of model in Section 3 for further detail).

The currently defined Type 1 Query is designed to be versatile enough to handle a wide range of information searching requirements, including structured pairwise combinations of terms to which Boolean operators are applied, & explicit attributes to specify the search indexes to which each term is to apply. The usefulness of Type 1 has also been extended to support free-form text queries by adding a new structure attribute, & to support ranked results using metadata tags associated with the records in the result set.

Nevertheless, Type 1 has some limitations which would be quite difficult to overcome (as described in the comparison below). The Type 102 Ranked List Query was designed to overcome these limitations, while maintaining stability in the Type 1 query, & providing a test bed for experimental applications built using the new query type as a foundation.

Query Type 102 is designed to be a generalized Ranked List query syntax which will handle a wide range of relevancy-based searching needs.

2.0 Key Differences between Type 1 & Type 102 Queries

The Type 102 Ranked List Query (RLQ) differs from Query Type 1 in the following key areas:

- Whereas Type 1 supports only pairwise combinations of operands (specified by Operators), Type 102 supports combinations of multiple operands (greater than 2 & up to some specified limit)
- Results obtained from applying the Type 102 query against one or more record collections are 'ranked' according to their ability to satisfy the criteria specified in the RLQ;
- Type 102 supports relevancy-based operators (such as rqAND, rqOR, rqIndep, and rqHeadRelation); Type 1 supports only operators which have explicit (concrete) semantic interpretations (e.g, "AND", "OR")
- Type 102 conforms to the standard Boolean-based attribute structure defined by Type 1, & even re-uses one of the Bib-1 attribute types (Relation). It also defines several new Attribute Types (Location in Record, Semantic Class, Content Authority, & Content Format).
- Type 102 provides the capability to specify relevancy-based limits within the ranked query itself, which restrict the query results by a fixed number (e.g, the top 20 ranked results) or by a relevancy score threshold (an absolute score - RSV - such as in WAIS)
- Parameters specified in the ClientServerInfo structure of the Type 102 query are used to describe the preferences for how the Origin/client would like the RLQ search server to interpret the query.

3.0 Ranked List Query Model & Description

3.1 High-Level RLQ Model

At the highest level, a Type 102 query can be viewed as being a need statement (or a set of need statements plus a combinator) specifying a user information need, with a description of what the desired results are.

These need statements are composed of three major parts:

1. A restriction set (component), giving a description of which records should be considered.
2. Information about what records are known to be helpful or not helpful to this need (relevance feedback)
3. A structured Ranked Query (RQ) giving terms and their relationship to each other.

Note that all of this is a description of information need; it is not instructions to the server to perform specific search operations.

3.2 Query Structure & Processing

The Type 102 structured query is a recursively defined structure of operators and weighted operands, with the operators giving the relationship between the operands. The query contains parameters specifying how the restriction and query components are to be interpreted and how the result set is to be organized. These parameters are specified as part of the ASN.1 query definition in Appendix A.

As a point of terminology, two different terms are used to reference query contructs in this document, and it is important to understand the differences between the two. The first term, Type 102 (sometimes referred to as the Ranked List Query or RLQ) is used to refer to the top-level query structure, which is the total composite query contained within the query parameter of the Z39.50 Search request. The Type 102 query may contain a number of optional parameters, but is required to contain a list (sequence) of Need Statements.

Within each Need Statement, there may be Restrictions, Relevance Feedback data, a relative weight for the Need Statement, and a Ranked Query (RQ). The RQ is the actual query to be evaluated against the specified collection(s). It contains a (potentially) recursive series of weighted operands and operators, plus a set of parameters specifying reformulation options and advice.

Each of the need statements submitted by the client optionally goes through an explicit query reformulation process by the server. The server applies all the information it knows about the databases, the query terms, general linguistics, relevance information, thesaurii, and so on, to arrive at a reformulated query that, in the server's opinion, better describes the user's information need in the context of each of these databases. If the client has so requested, processing can stop here with the reformulated query being shipped back to the client for further modification by client and user.

So in general terms, a client submits a query, and the server changes it and returns it to the client. The client (in cooperation with the user) modifies it and sends it back to be run exactly (or submits it for further reformulation). The server is free to do what it wants in the interpretation process; if either the client or the user disagree with what the end result is, then they can further modify the query.

In general, the user/client does not have as much control over the processing of a Type 102 query as they would using a Type 1 query. One of the chief reasons for using Relevancy Ranked searching is to take advantage of advanced query interpretation and reformulation processing which can only be performed by the search server, due to its intimate knowledge of the collections it is searching, the vocabularies native to those collections, & the most effective expansions of the query terms as related to the precision & comprehensiveness of the desired search results. Thus, when Type 102 queries are reformulated by the server, it is less likely that the user/client will be able to understand the whys & wherefores of the various modifications made to the query.

Conceptually, the (possibly reformulated) query is applied to each record that satisfies the restriction expression to produce an indication of how well the record matches the ranked query. For probabilistic models, this indication will generally be a probability or some function based on a probability. For vector space models this indication will be some function based on a distance between the record and the query. Other retrieval models may use other measures. This indicator is called a Retrieval Status Value (RSV). By convention, the RSV for the Ranked List Query must be a value in the range 0..1 (by using the IntUnit data type, whole integer values are specified along with a scale factor to form the RSV).

The result of evaluating a query against a set of collections is a set of records together with Retrieval Status Values. The desired size of the result set is defined by the ranked query parameters. The result set will generally be ordered by decreasing RSV, but other orderings (e.g., date) can be requested using the Z39.50 Sort service. A ranked query result set can be manipulated and presented in the same manner as a Boolean Type 1 result set.

Note: One important subclass of query is the single operand query, where the term consists of a natural language statement.

3.3 Options to control Processing & Reformulation of RLQ

A number of options are specified at the top level of the Type 102 query to control the processing & interpretation of the query, as well as the types of information returned in the search response & follow-up Present responses against the generated result set.

3.3.1 Search Output Request

This set of options specifies what information is to be accumulated by the server & possibly retrieved via a subsequent Z39.50 Present.

<u>returnResultSet</u>

This option instructs the server to perform the search & build a result set (if set to 'no', the search is not executed & only the reformulated query & other metadata is returned).

<u>returnReformulatedQuery</u>

This option indicates that the query that actually operated on the records is to be returned (following reformulation). The reformulated query will include a sequence of [tagType, tagValue] pairs indicating the presence & meaning of partial RSV's, query operand descriptions, & the various elements of collection, set, record, & term metadata.

<u>mData</u>

This is a sequence of [tagType, tagValue] pairs which specify the types of metadata to be accumulated by the server, & returned, either in the additionalSearchInfo field of the search response, or as metadata records (using record syntax RQRS) in Present responses. The details of this specification have not been worked out as yet, but they will be added to this document when they are finalized.

3.3.2 Client->Server Info

A parameter structure (ClientServerInfo) may be attached at either the query or the operand level. The parameters contained within the ClientServerInfo structure give advice to the server about how a particular query is to be interpreted, evaluated, & optionally, reformulated.

<u>reformClause</u>

When specified at the query level, the reformClause parameter allows the client to specify whether or not the server is allowed to reformulate the submitted query. Reformulation at the query level applies to expansion & modification of all terms (operands), operators, relevance feedback, & weights across the entire query.

When the reformClause parameter is specified at the operand level, of course, its scope is limited to changes to the terms & attributes of the associated operand only.

An optional reformMethod EXTERNAL structure provides for a private structure to define explicit dimensions in which the operands within the record may be expanded (e.g, morphological, thesaural, etc.).

<u>recallImportance</u>

RecallImportance (also optional) specifies a setting between 0 and 1 of how important every useful doc is. E.g., if 1, the server may try to add many related terms to the query, at a cost of precision. RecallImportance is only useful if the reformClause option is set.

<u>resultSetDesc</u>

ClientServerInfo may also include a resultSetDesc structure, containing one or more criteria to be used in filtering or truncating the Ranked Query result set. The two limiting criteria defined at this time are: numRecordsWanted & rsvThresholdValue.

NumRecordsWanted specifies that only the top ‘n’ relevant records are to be preserved in the final result set. It is up to the discretion of the search server as to whether it will always return this number of results, or whether it will return only the number of results which meet certain relevancy criteria, up to but not exceeding (and potentially less than) this limit.

RsvThresholdValue specifies that only records with an RSV score greater than or equal to the specified value are to placed in the result set. The actual result set may contain fewer records than requested (not all records exceeding rsvThresholdValue - with the actual number indicated by the resultCount parameter in the Search response) but may not contain more records.

3.4 Restriction Component of RLQ

As stated in earlier sections, a Type 102 query may comsist of a number of subquery expressions (called Need Statements), each of which may contain:

- a Ranked Query (RQ) expression;
- a Restriction expression;
- other Need Statement evaluation parameters.

Conceptually, the restriction expression defines the collection over which the Ranked Query is evaluated. No record that does not satisfy the restriction expression may appear in the set of candidate records input to the Ranked Query associated with this Need Statement. This specification says nothing about the order of evaluation supported by any particular implementation.

A Restriction expression consists of two components (both of which are optional):

1. a list of database names, all of which are to interpreted identically in one of the following ways:
    a. restrict the results of this Need Statement to exactly the databases listed here (a subset of those specified at the search request level or included by them in the case of group database names); or
    b. restrict the results of this Need Statement to the databases specified at the search request level **minus** those listed here; or
2. a standard RPN Type 1 query, allowing, among other possibilities, the use of existing result sets, docIds, and Boolean operators to effect a further restriction of the set of records specified by this Need Statement as eligible for searching.

3.5 Structured Operands & Attributes

Operands may be structured in the Type 102 query, allowing a number of different parameters & attributes to be attached to each operand. One useful parameter that can be attached to an operand is a weight, which specifies the value to be placed on the operand with respect to its importance in selecting records from the designated collection(s). The value of the weight is specified in IntUnit's, but for the Type 102 query the base range has been set between 0..1 (requiring the use of a scale factor in the IntUnit data type to allow encoding of integer valued weights).

3.5.1 Proximity Qualifier

An optional Proximity qualifier (RQProximity) may be specified within a structured operand. The Proximity qualifier is used to indicate that all operands in this subtree must be satisfied (i.e, have non-zero partial RSV) within the same proximity unit. The 'Proximity Operator' structure is borrowed from the Type 1 query in order to encode the proximity unit (either a public or private type), the ordering, & the distance (in proximity units). Proximity is not specified as an operator in the Type 102 query because it would require two operators to specify a ranking operator in combination with proximity.

In Type 102, Proximity Unit specifies the scope of the Proximity definition, & Distance indicates how far away from each other the satisfaction of the operands can be: 0 indicates within the same unit, 1 indicates within adjacent units, etc. Note that RQProximity is a binding operator, with some of the properties of a Boolean AND. For instance, a Proximity Unit of the same word requires that that same word must match each of the operands of the subtree.

The value of the Proximity qualifier (RQProximity) itself is specified in a range between 0..1 (as an IntUnit scaled by some factor, such as 100). An RQProximity value of ‘0’ indicates that the Proximity qualifier is to be ignored in this expression, whereas a value of ‘1’ indicates that Proximity is very important & should be honored by the search engine to the greatest extent possible (i.e, a ‘1’ is not required to be treated as mandatory, and may not be treated as such by some search systems).

3.5.2 Client->Server Information

ClientServerInfo (at the operand level) contains the same set of parameters as clientServerInfo at the Type 102 query level. In the same way as the query-level structure, these parameters give reformulation advice (or suggestions) to the server; however, they apply only to reformulation of the associated operand, not to the entire query.

As a general rule, most RLQ-capable servers will attempt to perform expansion (reformulation) of the operands in the query (i.e, will attempt to enhance each term with lexical, morphological, phonetic, semantic, & other equivalents). In some cases, however, the client may want to specify that certain operands are either to be reformulated in a specific way, or are not to be reformulated at all.

The reformClause parameter is a simple boolean choice specifying whether or not reformulation is ‘allowed’. The recallImportance parameter allows for specification of a value (on a 0..1 scale), which indicates the relative importance of the recall of results (a higher value would encourage the server to expand the operand to a higher degree to obtain a greater recall) versus result precision (highest relevancy is based on the most literal match with the specified operand). An optional reformMethod EXTERNAL provides for specification of a reformulation method, used to indicate explicit dimensions in which this operand may be expanded (e.g, morphological, thesaural, etc.) or a well-known (and well-understood) industry methodology for expansion and reformulation of the query (e.g, ??).

ClientServerInfo may also include a resultSetDesc structure, containing one or more criteria to be used in filtering or truncating the Ranked Query result set. The two limiting criteria defined at this time are: numRecordsWanted & rsvThresholdValue. NumRecordsWanted specifies that only the top ‘n’ relevant records (based on satisfying clauses) are to be included in the results when this clause is evaluated. RsvThresholdValue specifies that only records with an RSV score greater than or equal to the specified value are to placed in the results obtained from evaluation of the clause.

3.5.3 Type 102 Attributes

Similarly to a Type 1 query, the operands in a Type 102 query may have attributes associated with them. Type 102 has defined a new attribute set (RLQ-1), which currently specifies four new attribute types, & re-uses one attribute type (Relation) from the Bib-1 attribute set.

The following attribute types are defined for RLQ-1:

1. Location in record (locationInRecord). This attribute type is used to convey information about the portion of the record within which the operand is to be evaluated. This attribute can be used to specify fields within records (e.g., evaluate operand in the Title field) or generic record components.
2. Semantic class of operand (semanticClass). This attribute type is used to convey information about the intended semantics of the operand (e.g., this operand represents a corporate, geographic, or personal name, a date, a monetary figure, etc.). The target may be able to use semantic class to improve query reformulation or alter the ranking function to be used (e.g., don't apply verb or adjective expansion to a name).
3. Content authority for operand (contentAuthority). This attribute type is used to specify the range of legal values that an operand may take (when the range of values is known). The range of values is specified through reference to a standard authority (e.g., “NISO Z39.53-1994 -- Codes for Representation of Languages”), a private but widely known authority (e.g., “Dow Jones Industry Codes”), or a list of values that is subject to private agreement.
4. Encoding used for operand (contentFormat). This attribute type specifies the syntax used to encode the operand (e.g., operand encoded using name normalization rule set X or chemical structure format Y, or operand is a regular expression type Z). This attribute is used when the structure of the operand is known but the range of values is not fixed. This attribute can be used in combination with contentAuthority (e.g, contentAuthority = Z39.53, contentFormat = fre - content language is modern French).
5. Relation. This attribute type is equivalent to the Bib-1 relation attribute, but the range of legal values is restricted to values 1 through 6 (less than, less than or equal, equal, greater than or equal, greater than, & not equal).

As stated above, the Type 102 query explicitly defines a small set of integer values for each of the valid RLQ-1 attribute types, & also defines how those attribute values are to be used in combination, & what behavior is expected when each combination is used.

By using a combination of several attributes from the RLQ-1 attribute set (e.g, Location, Semantic Class, Authority, Format), multiple dimensions can be communicated with more precision & less ambiguity. By combining sets of these attributes with the use of the ASN.1-encoded data type of the term, the client & server can more easily agree on the meaning & interpretation of these attributes, while requiring the use of a smaller quantity of total attribute values.

The use of any one of the attribute types is optional for any specific operand, so that, for instance, LocationInRecord could be used in combination with Semantic Class, or one or the other could be used alone or in combination with attributes in the other three type categories.

Since only a small set of (integer coded) values will be defined for RLQ-1 attributes, an extensibility mechanism is needed to add new values, which may only be supported by a subset of clients & servers. The Attribute Element structure of Z39.50 V3 allows a client to specify Type 102 attributes as either numeric (pre-defined), or string (dynamic) values. The string value can be used by Type 102 to provide the needed attribute extensibility.

3.6 Structured Operators

A number of new operators are defined for use with the Type 102 RL Query, including various ranking operators and an ‘operand relationship’ operator. The ranking operators are to be used across two or more operands to specify that the particular group of operands are to be relevancy ranked (using a particular methodology) in order to formulate this branch of the result set tree. The ranking of these operands is independent of the mechanisms used to formulate the branches underneath each operand.

A number of ranking operators can be used to combine operands (which may in turn be the results of nested operators). The currently defined ranking operators (rqIndep, rqAND, rqOR, rqANDNOT, & rqHeadRelation) are described in Sections 3.6.1-3.6.5.

3.6.1 rqIndep Operator

The rqIndep operator is used to indicate that the target operands are to be treated independently of each other. Increasing a particular operand’s weight will guarantee that the weight for the entire clause will not decrease.

3.6.2 rqAND Operator

The rqAND operator allows a client to specify a ranked AND which can recommend varying degrees to which its associated operands are desired to be present. The server may ignore the integer value, but if it does not, then the value of rqAND is a number between ‘0’ and ‘1’ giving the degree to which satisfaction of all operands should be emphasized. A value of ‘1’ indicates that all operands must be satisfied (i.e., all operands must have non-zero weight in order for the clause weight to be non-zero.). NOTE: Depending on the Ranking methodology used by the target server, a value of ‘1’ does not necessarily cause this operator to behave exactly like a Boolean AND.

3.6.3 rqOR Operator

This rqOR (server-dependent) operator allows a client to emphasize the presence of a single operand to varying degrees. A clause weight will be zero iff all operands are zero. The server may ignore the integer value, but if it does not, then the value of rqOR is a number between 0 and 1 giving the degree to which satisfaction of an operand is to be considered equivalent to the satisfaction of the other operands. A value of 1 indicates that all operands are equivalent.

3.6.4 rqANDNOT Operator

The rqANDNOT (server-dependent) operator emphasizes, but does not require, the presence of the first operand and the absence of all other operands. The server may ignore the integer value, but if it does not, then the value of rqANDNOT is a number between 0 and 1 giving the degree to which satisfaction and non-satisfaction (zero-valued) of the operands should be emphasized. A value of 1 indicates that the first operand must be satisfied (non-zero weight) and the second operand must have zero weight in order for the clause weight to be non-zero.

3.6.5 rqHeadRelation Operator

The rqHeadRelation operator allows the specification of relationships between operands in a Ranked Query. When the rqHeadRelation operator is specified, the value of rqHeadRelation designates how the first operand is related to each of the other operands under this operator node. Operand weights (between 0 and 1, scaled) are used to describe the strength of the relationship.

Values currently envisioned are: synonym, antonym, ISA, homonym, homophone, thesauri classes, etc. By associating a root term with all of its related terms, the server can then take advantage of this knowledge in its ranking algorithms to provide more accurate term weightings and resultant record scorings, if it has the capability to do so.

The rqHeadRelation operator requires a value (or structure) to specify the desired relationship. One example of this would be a ‘semantic distance’ attribute, which would allow the client to specify degrees of relatedness between/among terms.

3.7 Relevance Feedback Information

The Type 102 query includes the ability to allow the client/user to specify Relevance Feedback information (FeedbackInfo) within an initial Type 102 query sent to the server, or as part of a resubmission of a reformulated Type 102 query. FeedbackInfo is a parameter structure attached (optionally) to each Need Statement within a Type 102 query.

It allows the user/client to specify a list of records (or record extracts or text segments) to be used by the server in finding records with either similar or dissimilar relevance characteristics. A signed IntUnit value (‘relevance’) in the range from [-1..0..1] (combined with a scale factor) is specified by the client to indicate the degree of desirability [0...1] or undesirability [-1...0] to assign to each specified Relevance Feedback item in the list.

If a Relevance Feedback item references a record, it is specified as an ‘opaque’ DocumentID, which was most likely obtained from a record previously received in a result set from the target Ranked Query server. The second choice for a Feedback item is a block of text, which could be extracted from a previously retrieved record, or from any other relevant document. The final choice for a Feedback item is an EXTERNAL, which could be used to carry non-textual relevance info (e.g, chemical structures, image fragments), or record fragment identifiers (such as ‘start opaque record fragment ID’ and ‘end opaque record fragment ID’), encoded based on a private scheme.

The conceptual model for the Relevance Feedback process is that the target will use this information along with other user/client input, to reformulate the query and retrieve a new set. The target will return the reformulated query (if requested), along with information about what terms were added, what re-weighting was done, etc.

If the client then chooses to resubmit the reformulated query (either with or without additional client modifications), it sends it to the server as if it were a new query. When it resubmits the query, the client again has the choice of requesting whether or not to reformulate the query, & also whether or not the query is to be submitted for search processing following any further reformulation.

3.8 Query Type 102 Result Demographic Data

As part of a Type 102 RLQ search, the client may want to obtain search result demographics data pertaining to the collection, result set, record w/in the result set, & the query terms as they relate to ‘hits’ within the result set. Collection level metadata can be returned in an additionalSearchInfo structure in the search response. All other levels of metadata, however, may only be retrieved using standard Z39.50 Present requests.

The lists below are intended to capture the possible data elements (all optional) which may be returned for subsequent use by a client or a user, or both. The detail definitions of these elements & their respective data types have not yet been finalized. Conceptually, however, the requesting client specifies what metadata elements it desires the server to maintain & return in relation to the search.

The metadata desired is specified in the SearchOutputRequest structure of the Type 102 query. The key options which may be specified are: perform the search & build a result set (if set to 'no', the search is not executed & only the reformulated query and/or other metadata is returned; return the reformulated query (yes or no); & a sequence of [tagType, tagValue] fields indicating the presence and meaning of sub-query RSV's, query annotations, & the various elements of collection, set, record, & term metadata.

The following elements of collection-level metadata are eligible to be returned in a predefined structure within the additionalSearchInfo parameter of the search response:

- Number of records in collection (numRcdsInColl)
- Number of unique terms in collection (numUniqueTermsInColl). NOTE: must be profiled to either include or exclude stop/noise words.
- Total number of term occurrences in the collection (numTotalTermsInColl)
- Total number of records in the collection in which each query term appears (numRcdsByQueryTerm - structure consisting of a list of [term,#] pairs)
- Total number of occurrences of each query term in the collection (numTotalOccByQueryTerm - structure consisting of a list of [term,#] pairs)

The following result set level elements can be retrieved by the Origin by issuing a special Present request against the RLQ result set:

- Number of records in set (numRcdsInSet - based on RSV, size, & count thresholds)
- Total number of query term occurrences in result set (numQueryTermsInSet)
- Number of times each query term occurs in result set (numSetOccByQueryTerm - structure consisting of a list of [terms,#] pairs)

NOTE: As a temporary provision for experimental development & testing, this special Present request will be issued against 'Record 0' of the result set, which will be modeled as containing result-set level metadata. The requirement to retrieve result set level metadata will be brought before the ZIG, & the Advanced Search sub-group will honor the 'official' solution as it is recommended & concurred by the ZIG.

The following elements are eligible to be returned at the record level (via a standard record level Present of one or more records):

- Record length in words (lengthRcdInWords - must be profiled to either include or exclude stop/noise words)
- Total number of query term occurrences in record (numQueryTermsInRcd)
- Relevance score of record (score - already in TagSet-M)
- Relevance rank of record (rank - already in TagSet-M)
- Number of times each query term occurs in record (numRcdOccByTerm - structure consisting of [term,#] pairs)
- Position of each term occurrence in record (PosOfTermsInRcd - structure consisting of [term,relative word position #] pairs)

Appendix A: ASN.1 for Ranked List Query

```
Q102  DEFINITIONS ::=
BEGIN
--
EXPORTS RankedQuery;
--
IMPORTS AttributeSetId, AttributesPlusTerm, ProximityOperator, IntUnit,
StringOrNumeric, OtherInformation, DatabaseName, ResultSetId, RPNQuery
FROM Z39-50-APDU
HumanString
FROM RecordSyntax-explain;
--
RankedQuery ::= SEQUENCE {
  needList            [1]   IMPLICIT SEQUENCE OF NeedStatement,
                          -- The actual content of the query, stating
                          -- the information need(s) of the user.
  combineNeedLists     [2]   SEQUENCE {
                          -- Evaluation of each need statement will
                          -- result in a list of records and RSVs. This
                          -- will take those sets and form the final
                          -- result set. default is server-choice.
combinePreference [1] CHOICE {
    useAlgorithm [1] IMPLICIT NULL,
-- server must use the specified
-- algorithm or return an error
-- diagnostic and fail the search
    recommended  [2] IMPLICIT NULL,
-- client recommends server use
-- specified algorithm
    serverChoice [3] IMPLICIT NULL },
-- server is free to use whatever
-- algorithm is most appropriate.

 combineAlgorithm  [2] CHOICE {
                              addWeight   [1] IMPLICIT NULL,
                                        -- server weighted addition.
                              ext        [2] IMPLICIT EXTERNAL}}
                                OPTIONAL,
  attributeSet        [3]   IMPLICIT AttributeSetId,
                          -- default attribute set for terms outside
-- the RestrictSet
  searchOutputRequest  [4]   IMPLICIT SearchOutputRequest OPTIONAL,
                          -- The information to be accumulated and
                          -- possibly returned by the server for this
                          -- seach request.
  clientServerInfo     [5]   IMPLICIT ClientServerInfo OPTIONAL,
```

```
                              -- Advice/Info from the client about how this
                              -- query should be reformulated and search to
                              -- be done.
  serverClientInfo     [6]   IMPLICIT ServerClientInfo OPTIONAL
                              -- Advice/Info from the server about query.
-- (Ignored by the server on resubmission of a
-- reformulated query).
                     }


NeedStatement ::= SEQUENCE {
  restrictSet  [1]   RestrictSet  OPTIONAL,
  feedbackInfo [2]   FeedbackInfo  OPTIONAL,
  rQuery       [3]   IMPLICIT OperandPlusWeight OPTIONAL, -- actual ranked query
  weight       [4]   IMPLICIT IntUnit OPTIONAL
                       -- relative importance of this need statement as a 0
                       -- to 1 scaled number
                          }

RestrictSet  ::= SEQUENCE {
  databaseNames [1]  CHOICE {
                      dbExclude [1] IMPLICIT SEQUENCE OF DatabaseName,
  -- List of database names to be excluded from the
-- list in the search request when evaluating this
-- need statement
                      dbOnly    [2] IMPLICIT SEQUENCE OF DatabaseName
                       -- List of database names (a subset of the databases
                       -- specified in the search request) - the only
        -- databases to be included in the search collection
-- for this need statement
                           } OPTIONAL,
  query [2]  IMPLICIT RPNQuery
                      -- Definition of set of docs on which rQuery for this
-- Need Statement is to be performed (within the
-- databases specified in SearchRequest and qualified
                       -- by databaseNames).
                       }

FeedbackInfo      ::= SEQUENCE OF SEQUENCE {
   documentId [1]CHOICE {
     localDocid[1]   IMPLICIT OCTET STRING,
        -- Opaque document identifier local to a server
 feedbackText[2]   IMPLICIT HumanString,
 -- Relevance feedback text extracted from a document
 -- or fabricated by the user
 otherFeedbackInfo [3] EXTERNAL
```

```
 -- Non-textual feedback info (e.g,chemical structures)
 -- or record fragment identifiers (e.g, start/end
 -- fragment), encoded using a private scheme.
},
   relevance [2]   IntUnit
                  -- Relevance judgement data (-1..0..1 inclusive)
                   -- for the document identified by documentId
-- (-1 is most undesirable, 1 is most desirable, )
     }

OperandPlusWeight ::= SEQUENCE {
  operand          [1] CHOICE      {
                   attrTerm          AttributesPlusTerm, -- base term
                   sOperand    [1]   StructuredOperand
                          -- recursive definition.
                          -- NOTE: A Boolean query could be specified here in the future.
                                   },
  weight           [2] IMPLICIT IntUnit OPTIONAL,
                   -- Relative importance of term, on 0 to 1 scale.
                   -- Default 1.
  clientServerInfo   [3] IMPLICIT ClientServerInfo OPTIONAL,
                   -- Advice/Info from the client about how this
                   -- particular operand reformulated.  This
                   -- over-rides the query ClientServerInfo
  serverClientInfo   [4] IMPLICIT ServerClientInfo OPTIONAL
                   -- Advice/Info from the server about this
                   -- particular operand. Allows server to annotate
                   -- the 102 Query structure with additional
                   -- information. Examples would include partial
                   -- RSV's being returned after a search, or
                   -- possibly collection information. (Ignored by the
  -- server on resubmission of a reformulated query).
                     }

StructuredOperand ::= SEQUENCE {
  rqOperator      [1] IMPLICIT RQOperator,
                  -- gives relationship between operands and information
                  -- about combining them.
  rqOperandList   [2] IMPLICIT SEQUENCE OF OperandPlusWeight,
                  -- ordered list of operands
  rqProximity        [3] CHOICE {
                  -- All operands in this subtree must be satisfied
                   -- (have non-zero partial RSV) in the same context.
                   -- 'unit' gives the scope of the context definition,
                   -- and 'distance' gives how far away from each other
                   -- the satisfaction can be (0 indicates within the
```

```
                -- same unit, 1 indicates in adjacent units, and so
                -- on) Note that rqProximity is a binding operator,
                -- with some of the properties of a Boolean AND.  A
                -- context of the same word implies that same word must
                -- match each of the operands of the subtree.
                    prox  [1] IMPLICIT ProximityOperator,
                            -- same syntax as in type-101 query,
                            -- but semantics expanded as above.
                    ext  [2] IMPLICIT EXTERNAL} OPTIONAL}

SearchOutputRequest ::= SEQUENCE {
        -- Describes output of query.  All except doSearch
        -- will be given in AdditionalSearchInfo of SearchResponse.
  doSearch          [1] IMPLICIT BOOLEAN,
                        -- True means perform the search
                        -- and establish a result set.
  returnReformulatedQuery   [2] IMPLICIT BOOLEAN,
                        -- True means return the query that
                        -- actually operated on the docs after
                        -- reform. This will be returned as a
                        -- RQInfo EXTERNAL within
                        -- AdditionalSearchInfo,
                        -- and will have Tagvalue from tagset-RQM
  mData                 [3] IMPLICIT SEQUENCE OF SEQUENCE{
                        tagType  [1]   IMPLICIT INTEGER OPTIONAL,
                                    -- default tagset-RQM
                        tagValue [2]    StringOrNumeric
                            -- value (probably from
                            -- tagset-RQM) indicating what sorts
                            -- of metadata should be returned.
                            -- Note that depending on tag, some
                            -- of this metadata may be returned
                            -- within AdditionalSearchInfo of
                            -- SearchResponse, and some within
                            -- a Present of the document using
                            -- record syntax of RQRS
                            -- (RankedRecordSyntax).
                          } OPTIONAL}

ClientServerInfo ::= SEQUENCE {
  reformClause      [1] IMPLICIT BOOLEAN,
                 -- allow reformulation this operand/query. If false
                 -- and the server does not support a term or
                 -- attribute, then this part of the search fails.
  recallImportance  [2] IMPLICIT IntUnit OPTIONAL,
                 -- setting between 0 and 1 of how every useful doc
```

```
                    -- is. Eg, 1, the server may try to add many
                    -- related terms to the query, at a cost of
                    -- precision.  Only used if reformClause is true.
  reformMethod      [3] IMPLICIT EXTERNAL OPTIONAL,
                    -- client/server agreed upon specification of what
                    -- reformulation methods to be done. We will define
                    -- a methodology for this later. Only used if
                    -- reformClause is true.
  resultSetDesc     [4] IMPLICIT SEQUENCE {
                    -- describes size and properties of result set
                    numRecordsWanted  [1] IMPLICIT INTEGER OPTIONAL,
                                -- max records in the result set.
                    rsvThresholdValue [2] IMPLICIT IntUnit OPTIONAL
                                -- If included, all docs in
                                -- result set must have RSV >=
                                -- rsvThresholdValue. This should
                                -- be between 0 and 1.
                          } OPTIONAL}

ServerClientInfo ::= SEQUENCE {
    -- Not completely worked out; must wait until Advanced Search group
    -- works out specifics of how the reformulated (marked-up) query & its
    -- associated meta-data is to be returned to the client.
  intUnit     [1]   IMPLICIT IntUnit OPTIONAL, -- Scaled integer
  humanString [2]   IMPLICIT HumanString OPTIONAL,
  metaData          OtherInformation OPTIONAL
                  -- Data (eg collection frequency) attached to an
                  -- operand.  We may flesh this out some after
                  -- operational experience.
                                    }

RQOperator ::=  SEQUENCE {
 operator      [1] CHOICE {
             rqIndep         [1] IMPLICIT NULL,
                  -- The operands are independent of each other.
                  -- Increasing a particular operand's RSV will
                  -- guarantee the clause RSV will not decrease.
             rqAND           [2] IMPLICIT IntUnit,
                  -- Ranked AND.  This server-dependent operator
                  -- emphasizes, but does not require, the presence of
                  -- all operands.  The server may ignore the integer
                  -- value, but if it does not, then the value of rqAND
                  -- is a number between 0 and 1 giving the degree to
                  -- which satisfaction of all operands should be
                  -- emphasized.  1 indicates all operands must be
                  -- satisfied (ie, all operands must have non-zero RSV
```

```
            -- in order for clause RSV to be non-zero.)
        rqOR            [3] IMPLICIT IntUnit,
            -- Ranked OR. This server-dependent operator
            -- emphasizes the presence of a single operand.
            -- A clause RSV will be zero iff all operands are
            -- zero. The server may ignore the integer
            -- value, but if it does not, then the value of rqOR
            -- is a number between 0 and 1 giving the degree to
            -- which satisfaction of an operand is to be
            -- considered equivalent to the satisfaction of the
            -- other operands.  1 indicates all operands are
            -- equivalent.
        rqANDNOT        [4] IMPLICIT IntUnit,
            -- Ranked ANDNOT. This server-dependent operator
            -- emphasizes, but does not require, the presence of
            -- the first operand and the absence of all other
            -- operands.  The server may ignore the integer
            -- value, but if it does not, then the value of
            -- rqANDNOT is a number between 0 and 1 giving the
            -- degree to which satisfaction and non-satisfaction
            -- (zero-valued) of the operands should be emphasized.
            -- 1 indicates the first operand must be satisfied
            -- (non-zero RSV) and the second operand must have
            -- zero RSV in order for clause RSV to be non-zero.
        rqHeadRelation   [5] IMPLICIT SEQUENCE {
            -- The first operand is related to each operand
            -- (independently) in a fashion described by
            -- RQHeadRelation.  The operand weights (between 0
            -- and 1) describe the strength of the relationship.
            -- Examples might be synonyms, antonyms, ISA, thesauri
            -- classes, etc.
                tagType  [1]   IMPLICIT INTEGER OPTIONAL,
                        -- default tagset-RQRelation
                tagValue [2]     StringOrNumeric
                -- value (probably from tagset-RQRelation)
                -- indicating the relationship between the
                -- first operand of the StructuredOperand
                -- and the remaining operands.  The server
                -- will use this relationship information
                -- to determine a RSV for this clause.
              },
        other           [6] IMPLICIT EXTERNAL
            -- Other operators, not guaranteed to be understood
            -- by client.
          },
allowedReform [2] IMPLICIT BOOLEAN
```

```
                    -- allowed to replace this operator. If false, and the
                    -- server does not support the operator, then this part
                    -- of the search fails.
        }
END

--  Tagset-RQRelation.  Tag values used within rqHeadRelation to indicate the
--                    relationship between the first operand (the "head"),
--                    and the each of the other operands ("sub-operands").
--                    For the experimental stage, tag values between
--                    0 and 99 are things we expect most systems will
--                    deal with (ie will be in the final tagset, even if
--                    not required to support), 100-999 will be shared by a
--                    class of systems, and 1000+ for a single system.
--                    All definitions may (and probably will) change
--                    as we get operational experience.  Values 1,2,3
--                    are the only ones currently required to be recognized.
--        0     Reserved
--        1     Term variants
--         10    Morphological variants (stemming, truncation, etc)
--         11    Common mispellings
--         12    Phonetic variants
--        2     Related words
--         20    Synonyms
--         21    Co-ocurring words
--        3     Class relationships
--         30    ISA relationship (Sub operand "is a" head operand).
--         31    Thesaurus class (Sub operand is member of a thesaurus
--              class headed by the head operand).

--  Tagset-RQM        Tag values used for Metadata within the RankedQuery
--                   Framework.
--                    For the experimental stage, tag values between
--                    0 and 99 are things we expect most systems will
--                    deal with (ie will be in the final tagset, even if
--                    not required to support), 100-999 will be shared by a
--                    class of systems, and 1000+ for a single system.
--                    All definitions may (and probably will) change
--                    as we get operational experience.  Values 1,2,3
--                    are the only ones currently required to be recognized.
--                    After operational experience, things like
--                    collection frequency may be added.
--                    This may eventually be subsumed within Tagset-M,
--                    but it's easier to keep disjoint for the initial
--                    implementation phase.
--        0     Reserved
```

```
--      1       RankedQuery
--      2       RankedQuery with ServerClientInfo of each operand being
--              an IntUnit, interpreted as the partial RSV for that operand
--      3       RankedQuery with ServerClientInfo of each operand being
--              a HumanString, interpreted as a server explanation of
--              the operand.
--      4       RankedQuery with ServerClientInfo of each operand being
--              an EXTERNAL Diagnostic Record.
--      5       HighlightingInfo (not yet developed, included as example).
```