NISO Z39.92-200X ISBN: xxxx-yyyy

Information Retrieval Service Description Specification

DRAFT STANDARD FOR TRIAL USE:
Period: November 1, 2005 – October 31, 2006

Abstract: This standard defines a method of describing Information Retrieval oriented electronic services, including but not limited to those services made available via the Z39.50, SRU/SRW, and OAI protocols. The standard addresses the need for machine readable descriptions of services in order to enable automatic discovery of and interaction with previously unknown systems. It specifies an abstract model for service description and a binding to XML for interchange.

A proposed American National Standard
developed by the
National Information Standards Organization

Published by the National Information Standards Organization

NISO Press, Bethesda, Maryland, U.S.A.

DRAFT STANDARD SUBJECT TO CHANGE

About NISO Standards

This is a Draft Standard for Trial Use (DSFTU). A DSFTU is issued when, in the opinion of the Standards Committee and the SDC, there is a need for field experience before proceeding with balloting, or when balloting of a draft standard fails to reach consensus and reveals the need for field experience.

This is a draft standard and subject to change.

NISO standards are developed by the Standards Committees of the National Information Standards Organization. The development process is a strenuous one that includes a rigorous peer review of proposed standards open to each NISO Voting Member and any other interested party. Final approval of the standard involves verification by the American National Standards Institute that its requirements for due process, consensus, and other approval criteria have been met by NISO. Once verified and approved, NISO Standards also become American National Standards.

This standard may be revised or withdrawn at any time. For current information on the status of this standard contact the NISO office or visit the NISO website at:
http://www.niso.org

Published by
NISO Press
4733 Bethesda Avenue, Suite 300
Bethesda, MD 20814
www.niso.org

ISSN: 1041-5635
ISBN: to be assigned

DRAFT STANDARD SUBJECT TO CHANGE

Contents

Figures

Foreword

(This foreword is not part of the *Information Retrieval Service Description Specification*, NISO Z39.92–200X. It is included for information only.)

Acknowledgements

Standards Committee BB wishes to gratefully acknowledge the assistance of the following individuals:

Ray Denenberg, Library of Congress; Matthew Dovey, University of Oxford; Thomas Habing, University of Illinois, Urbana-Champaign; Alan Kent, Royal Melbourne Institute for Technology; Ray Larson, University of California, Berkeley; Mark Needleman, Sirsi Corporation; Mike Taylor, Index Data.

About This Standard

This standard is the culmination of several related efforts to produce a means by which a software agent might discover the capabilities of a remote information retrieval service.

The Z39.50 standard for Information Retrieval defines a method of querying a database maintaining records that describe the capabilities of the server and the databases that it provides. This “Explain” database, named IR-Explain-1, has a specific attribute set (exp-1, 1.2.840.10003.3.2) and an ASN.1 based record syntax (Explain, 1.2.840.10003.5.100) through which the client may search for specific capabilities and retrieve information back in a well defined structure. Unfortunately, only a very few of the implementations of the Z39.50 standard include this service. It is thought in the Z39.50 community that this was due primarily to the complexity of the record syntax requiring specific ASN.1 development, as well as a lack of a specific requirement for system vendors to implement it.

In 2000, the ONE-2 project (OPAC Network in Europe) formulated an XML-based approach called “Explain Lite”. It was designed to provide only the essentials required for interaction with the service, without changing the Z39.50 model or protocol in any way. The Explain Lite schema contained such information as search and scan attributes, the names of databases available on the server, and the various record formats and element sets through which data could be retrieved. However, the distribution strategy was very dependent on Z39.50 and did not scale beyond a few databases on a system, as the server would send all of the information in a single XML record back as soon as any client accessed the system. The record could grow to be very large, as the structure was very flat and didn’t contain some essential information for interaction, such as the sorting capabilities of the server or a method of describing non-bibliographic search attribute sets. While Explain Lite was sufficient for the ONE-2 project, it did not address the requirements of the Z39.50 world in general.

In early 2002, a group from the Z39.50 community decided to redress some of the issues of Explain Lite and create an XML schema for describing Z39.50 databases. Initially called “XplainML” and then “Explain--”, the initiative was later renamed to ZeeRex, an acronym for “Z39.50 Explain Explained and Re-Engineered in XML”. The goal of the working group was to provide a method for describing a single database, rather than all of the databases on a server, as well as allowing harvesting and aggregation of the records to create registries. A second consideration was that a minimal set of elements should be usable for links between servers, termed “Friends and Neighbours”. This working group was led by Sebastian Hammer, Robert Sanderson, and Mike Taylor.

This work coincided with the formation of the ZNG initiative to bring Z39.50 into line with the current technology standards. By realigning the work as a method of describing information retrieval systems under the umbrella of the ZNG, ZeeRex also formed the basis of the service description method for the emerging SRW/U protocol. By August 2002, the schema was fairly stable and only minor modifications were made until version 2.0 in February 2004, which brought the schema into line with the release of the SRW/U stable version 1.1,

Standards Committee BB decided to adopt the ZeeRex specification as the basis for service descriptions in the Metasearch arena in April 2004, to be used in conjunction with the Z39.91-200X standard for collection description.

Note for this draft standard: This draft is considered version 0.9 of the standard. Since the definitions clause is missing, the text is not yet complete. The missing clause will be included into the second draft (to be published in spring 2006, half-way through the 12-month trial); this second draft will be considered version 1.0 of the standard. The second draft will also incorporate the feedback from implementers and other users up to that point.

Trademarks, Service Marks

Wherever used in this standard, all terms that are trademarks or service marks are and remain the property of their respective owners.

NISO Voting Members

At the time this standard was approved, NISO had the following voting members.

[to be added after ballot]

NISO Board of Directors

At the time NISO issue this DSFTU, the following individuals served on its Board of Directors:

Carl Grant, Chair
VTLS, Inc.

Robin Murray, Vice Chair and Chair-Elect
Fretwell-Downing Informatics

Jan Peterson, Immediate Past Chair
Infotrieve

Pat Stevens, Chair of SDC
Consultant

Douglas Cheney, Treasurer
Barnes & Noble, Inc.

Patricia R Harris, Executive Director/Secretary
NISO

Directors:

Nancy Davenport
Council on Library and Information Resources

Lorcan Dempsey
OCLC, Inc.

Daniel Greenstein
California Digital Library

James Neal
Columbia University

Oliver Pesch
EBSCO Information Services

Bruce Rosenblum
INERA, Inc.

Eric Swanson
John Wiley & Sons, Inc.

Committee BB Members

Grace Agnew
Rutgers University Libraries

Kristina Aston
National Archives of Canada

Patricia Brennan
Thomson Scientific

Mary Bushing
Library Consultant and Educator

Robina Clayphan
The British Library

Larry Dixson
Library of Congress

Emily Fayen
MuseGlobal Inc.

Juha Hakala (Chair)
Helsinki University Library

Sebastian Hammer
IndexData

Pete Johnston
UKOLN, University of Bath

Anne Karle-Zenith
Michigan State University

Vicki Miller
OCLC, Inc.

Julie Blume Nye
Fretwell-Downing Inc.

Judith Pearce
National Library of Australia

Robert Sanderson
University of Liverpool

Juliane Schneider
NYU School of Medicine

Sarah Shreeves
University of Illinois, Urbana-Champaign

Patricia Stevens
NISO

Chuck Thomas
Florida State University

Theo van Veen
National Library of the Netherlands

Terry Wilan
Talis Information Ltd.

Maja Zumer
National Library of Slovenia

Information Retrieval Service Description Specification

1 Purpose

The purpose of this specification is to allow the description of information retrieval oriented services in order to enable the aggregation of such descriptions into service registries and subsequently be used in a machine-to-machine fashion to enable automatic service discovery and interaction.

Unlike the World Wide Web, in which hyperlinks provide a means for discovery of other services, information retrieval protocols must either build the links into the protocol or have a database with a well known name of links to other services (the Explain proxy and ZeeRex approach). The centralization of links to services in a registry has been undertaken by various organizations in the past, but these have been done on an ad-hoc basis using internal databases for their descriptions, and hence lack any significant interoperability. This standard provides a single schema in which records may be exchanged and processed by client software, thus allowing for the development of standards based software agents that can be expected to work with other vendors' information retrieval systems.

2 Scope

The scope of the specification is specific to services, so what is meant by a "service" in this context must first be defined. The relevant aspects of the service to be described must also be defined.

A **service**, in terms of this standard, is an electronic means by which a collection (a logical grouping) of digital objects is made available. This, therefore, does not include such services as a physical reading room in a library, as it is not electronic even if digital objects are available there. It also does not include electronic methods of requesting physical objects, such as the inter-library loan protocol.

This definition excludes "transactional" services which do not make a collection available. For example, a service which takes two numbers as input, adds them together and returns the result is not in scope for this description as it does not have a collection of digital objects to serve.

The standard defines the subset of all services as those that support the retrieval of a digital object, or record.

The standard further reduces this set of services to those which support a standard protocol for information retrieval, for example OAI, Z39.50, or SRW/U. The scope of the *Information Retrieval Service Description Specification* does not extend to attempt to cover all possible means of interacting with a collection. In particular, proprietary CGI scripts or SOAP-based protocols are out of scope, although they do provide electronic access to a collection.

The scope of this description is the means of interacting with a single collection, or logical grouping of items. Some services, for example Z39.50, allow the client to interact with multiple collections using the same network connection or on the same host. From the beginning, the approach has been taken that these must be separate records, as most users are interested only in their particular information need, not what else is available on the same server.

In terms of the features to be described, this standard is concerned with the description of the service's capabilities in terms of information retrieval, not the collection of objects that the service makes available. That is to say that the protocol is of interest, not what can be retrieved via the protocol. The collection description may be described using the Z39. 91-200X standard and linked from the record.

Secondly, this standard is concerned with the **semantics** of the protocol, not the syntax. As described above, the client and server must interact using a protocol which allows for retrieval of digital objects. The definition of those protocols is out of scope; the standard does not attempt to describe the underlying conversations between client and server. There are already languages for describing syntax such as WSDL or ASN.1. Inclusion of such information in this standard would not be useful to the recipient as the client would require an implementation of the syntax protocol to make use of it.

Given a client side implementation of the protocol, the scope of the standard in terms of syntax vs. semantics can thus be defined as the information necessary to use the protocol to successfully access the collection exposed by the service. This information includes such aspects as query types supported, record syntaxes and schemas supported for retrieval, and so forth.

Finally, although the discussion has been in terms of building registries, the specification does not mandate a model for such registry implementations. This standard is only concerned with the interchange model and schema, not how the information so encoded will be distributed. However, the appendices include profiles developed for Z39.50 and SRW in order to demonstrate the applicability of the schema.

3 References

This standard references the following documents:

Heaney, Michael. An Analytical Model of Collections and their Catalogues.
http://www.ukoln.ac.uk/metadata/rslp/model/amcc-v31.pdf

NISO Z39.91-200X, Collection Description Specification

RFC 1766, Tags for the Identification of Languages
http://www.ietf.org/rfc/rfc1766.txt?number=1766

ISO 8601, Data elements and interchange formats – Information interchange – Representation of dates and times

4 Definitions

The following terms, as used in this standard, have the meanings indicated.

Term	Definition
[to be added]	

5 Abstract Model for Services

In order to appropriately describe an information retrieval service, the standard needs an abstract model for such a service which can be generally applied. This model must include not only the information retrieval protocol aspects capable of being implemented by a service, but also must reference other entities such as the Collection the service makes available and the Agents which administer it. Although the description of these other entities is out of scope, the standard must be aware of their existence in order to permit the relations between them to be appropriately described. Once this context has been established, a high level model for information retrieval protocols must also be identified.

5.1 Collection/Service Model

The model described by Michael Heaney in his study, *An Analytical Model of Collections and their Catalogues*, forms the basis of the much simpler model adopted for the purposes of this standard. However Heaney's description does not explicitly refer to services as such; instead it takes the more general view of a Location being a place, either physical or electronic, in which the collection is located.

The derivation discussed below also takes into consideration work done by the JISC's Information Environment Services Registry in the United Kingdom and is known to be compatible with it.

Four entities have been derived from Heaney's model: Agent, Collection, Item, and Service:

- An **Agent** is either a personal or organizational entity capable of action. Heaney divides this into four entities with different roles—Administrator, Collector, Owner, and Producer—however the attributes of each remain the same and they can be treated as a single type of entity with different relations. An Agent may have a variety of relations including administering Services or Collections; creating Items; collecting Items; describing Services, Collections, or Items; and so forth. It is not within the scope of the standard to define all such relations, nor provide a means of describing the relevant attributes of agents.
- A **Collection** is a logical grouping of one or more Items and/or Collections. This is termed, following the Dublin Core model, a Part, and thus Collections have one or more Parts, each of which must be an Item or a subsidiary Collection. Collections are made available by zero or more Services, as described in more detail below. It is recommended that Collections be described using the NISO Z39.91 -200X *Collection Description Specification*. A Collection in information retrieval terms can be thought of as a database.
- An **Item** in Heaney's model is an instantiation of Content, which he defines as an intellectual creation. This is consistent with IFLA's recommendation, *Functional Requirements for Bibliographic Records*, however, in terms of the current model the standard is primarily concerned with what can be retrieved via the Service, so the two entities have been reduced to one which encompasses both aspects. Items are, therefore, retrieved from their parent Collection via a Service. There are many standards for describing or creating items, and those available for retrieval must be included in the final service description. An Item in information retrieval terms is often called a record or item.
- A **Service** is a subtype of Heaney's Location—an electronic Location at which the Collection is available. This forms the basis of the requirement that the Service support retrieval of Items from a Collection. A Service makes available exactly one Collection. It is this entity which the ZeeRex specification aims to describe.

For example (see Figure 1), a collection may consist of two sub-collections, and be made available by two different services. One of the sub-collections, consisting of one or more items not shown, is made available by a third service. The other sub collection consists of two further sub-sub-collections, one of which is also made available by a service.

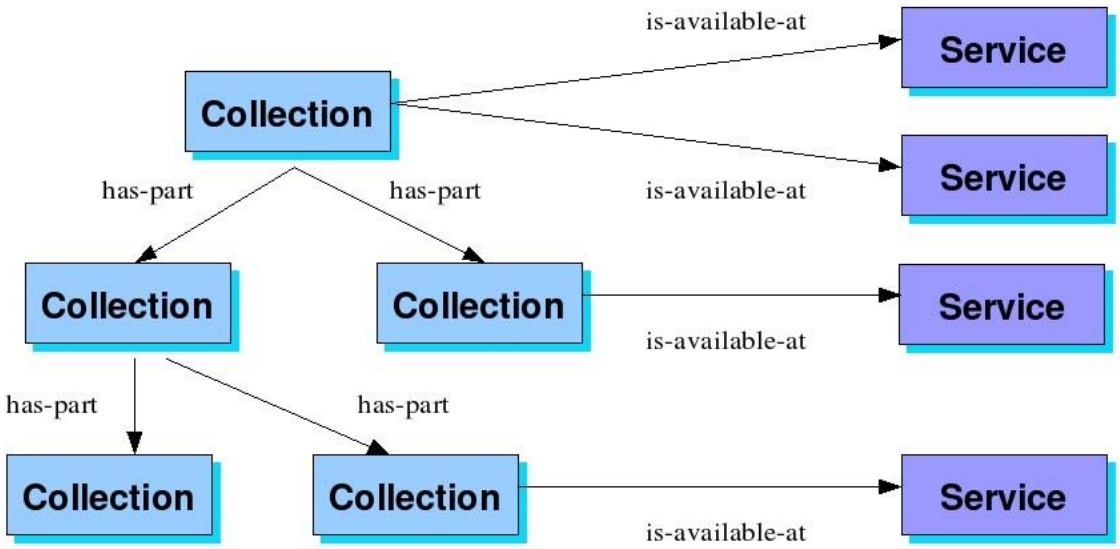

Figure 1: Collection/Service Model Example

5.2 Required Information Retrieval Service Attributes

Now that the context in which the service description resides has been defined and described, it is necessary to define the attributes of a service which require description. The distinction between protocol syntax and semantics has already been described, so the aspects which are required to fulfill the purpose of the specification must be fleshed out; to enable discovery of and interaction with the services.

5.2.1 Location and Interaction

In order for a client to interact with a service, it needs to know where the service is located. Any service must have a host name, a port number, and then perhaps a subsidiary location. For example, a Z39.50 service might be available at z3950.cheshire3.org, on port 2100, with a database name of “ir-explain---1”. On the other hand, HTTP based protocols would likely have a path in place of the database name, such as SRU or OAI.

The service must be capable of interaction via an information retrieval protocol. This protocol thus must be identified, including a version number. The request may be carried over a transport layer, such as HTTP or HTTPS and there may be transport specific metadata such as if an HTTP based protocol is available via POST and/or GET.

Once the client knows the protocol to use, and the location of the service, it may be required to send authentication or authorization information. This may be public knowledge, and hence included in the record, or may be the subject of an agreement between the service provider and client. Several types of authentication are in common use, however this is an area in which much research is ongoing and any binding should be capable of future extension.

5.2.2 Discovery

Once the client has been able to successfully connect to the service, there must be a means by which it can select or discover Items. This can be expressed in terms of access points. Each type of search or browse available via the service is treated as a point of access to the data, and hence needs to be described.

Search, browse, and sort are identified as the three basic operations of any information retrieval system, and which of these operations are supported by the access point must be recorded.

In order for the client to make use of the access point, the description must include the query language-specific identifiers for it. For example, attribute combinations in Z39.50's default query syntax or index names in CQL. The access point should also have an identifier by which it can be uniquely distinguished from other access points in the record, irrespective of query language.

The access points may also have different configurations in terms of default settings or additional features that are supported, and these should be recorded as well. For example, an access point for a numeric field may default to an exact equality comparison, whereas an access point for a full text field may default to a keyword search. An access point with a controlled vocabulary should be able to be described with a reference to the vocabulary.

The sort operation may also occur via named flag, rather than by a particular point in the data. A sort only field should therefore be distinguishable from one which can be searched or browsed.

5.2.3 Retrieval

Once the records have been selected, the service must allow them to be retrieved. The model adopted is that the same Item (or Content, in Heaney's model) may be available in different representations. For example a bibliographic record may be available in several different XML schemas, as well as in MARC21 format. There may also be further distinctions as to which fields in the record are present, for example to create a full or brief display.

Systems may also sort based on arbitrary fields of a record. It is necessary to know which record schemas are supported by the service for this usage, such that the client is able to format their sort requests appropriately. For example, a service might accept an XPath expression to select an arbitrary XML element to sort upon.

5.2.4 Feature Support and Values

Information retrieval systems also have default (changeable) and set (unchangeable) values. For example, the default number of records that will be retrieved at once, as opposed to the maximum number of records that the server will return in one response. Such settings are not index specific, but apply generally across the service.

Most services will also not implement all of the features available in the protocol or query language that they utilize. For example, many Z39.50 servers do not support proximity searching. There should be a means by which the capabilities of the server can be described, or links to service profiles supported.

5.2.5 Relations and Descriptions

The service has links to other entities, potentially described in different documents, as discussed above. In particular, the service is administered, owned, and hosted by one or more Agents, and there may be a fourth agent who should be contacted about the service. There should be a link to the description of the collection that the service makes available, but also space for free text to enable basic discovery in the absence of a more formal description. There should also be links to other services which make the same collection available, normally via different protocols.

The service will also have various attributes which are appropriate to be described in free text, such as a title and a description. In the same manner as agents and collections, these fields should be able to be either included directly, linked against external resources, or incorporate other schemas developed specifically for the purpose. In particular, the useful attributes of a service to describe are a title, description, the history of the service, any restrictions on usage of the service, the extent to which the service makes the collection available, and the languages in which the service makes the items in the collection available.

For environmental survey purposes and potentially providing debugging information, the service description should also include the implementation of the information retrieval protocol used. This would enable the client vendor to contact the server vendor, rather than just the administrator of a service which uses the vendor's system who may be unable to fix any software problems.

5.2.6 Metadata

The expectation is that these records will be used in registries of services. In order to maintain the records in a potentially distributed environment, there are a few metadata fields which have been identified as important to include in the record itself.

As the record may become out of date in a registry, for example if the service changes and the record is not modified, the description should maintain the date it was last modified by the original author, the date it was aggregated into the registry, and from where it was harvested. This allows the client to determine if it should attempt to discover a more recent version of the service description, and if so, where to look for it.

6 XML Binding

The established interchange format for the records is XML. The XML DTD is included in Appendix A.

The namespace for these records is: `http://explain.z3950.org/dtd/2.1/`

XPath conventions will be used in the following discussion of the schema; as such, `@attr` refers to an attribute named “attr” and `/path/to/element` refers to an element named “element”.

The top level element is `/explain`. It has one attribute, `@id`. This attribute should be used to establish the identity of the record within the registry, and then used for retrieval purposes. As aggregated records may have the same identifier as that from a different source, the registry may modify the value of this attribute upon aggregation. This attribute is optional.

Example:

```
<explain id="iesr-http://www.loc.gov:7090/voyager">
```

The structure of the XML is derived from the six sections identified in the required attributes section above, however only the first section is mandatory.

The sections are:

1. **serverInfo**: The basic details required to start a network connection to the described server. This includes such things as the hostname, port, database name, and any authentication required.
2. **databaseInfo**: This section contains full text information describing the database, its history, who is responsible for it, and so forth.
3. **metaInfo**: The metaInfo section contains information about the record, such as when it was created and when and where it was aggregated from.
4. **indexInfo**: The details regarding how one can interact with the server are recorded here. This information is recorded in the form of “indexes” and sort keywords.
5. One of the following, depending on the protocol:
    - **recordInfo**: This section contains elements describing the various record syntaxes and any elementsets that may be available for those record syntaxes under Z39.50.
    - **schemaInfo**: This alternative section describes the XML schemas used in SRW or similar protocols for retrieval and sorting.
6. **configInfo**: The data in this section concerns the configuration of the server, such as which protocol features it supports and any default values.

6.1 /explain/serverInfo

The /explain/serverInfo element contains the information necessary to start a connection to the described server.

/explain/serverInfo has five attributes:

- @protocol – This attribute is used to record the protocol that should be used to connect to the server. This attribute is optional and can contain any text. The following values are expected:
  1. "Z39.50" – This is the default value, if the attribute is not supplied.
  2. "SRW" – Only SRW (SOAP-based) is supported.
  3. "SRU" – Only SRU (URL-based) is supported.
  4. "SRW/U" – Both SRW and SRU are supported at the same endpoint.
  5. "OAI" – The Open Archives Initiative Protocol for Metadata Harvesting.
  6. "OpenSearch" – Amazon's OpenSearch protocol.
- @version – This attribute should record the highest version of the protocol supported by the service. This attribute is optional and defaults to the most recent version of the named protocol.
- @transport – For protocols carried over another transport protocol, this attribute should record the name of the transport protocol. This attribute is optional, and defaults to "http" for protocols where the distinction should be made. If more than one transport protocol is available, then they should be space separated. Any value is allowed, but the following are expected:
  1. "http" – The protocol is transported over simple http, the default value.
  2. "https" – The protocol is transported over secure http.
- @method – For http based protocols, this attribute should record which methods are available for interaction with the service. If multiple methods are available, then they should be space separated. It has the following possible values:
  1. "GET" – The service is available via GET, the default value.
  2. "POST" – The service is available via POST.

Example:

```
<serverInfo protocol="SRU" version="1.1" transport="http" method="GET">
```

6.1.1 /explain/serverInfo/host

The /explain/serverInfo/host element contains the address of the server which hosts the service. This address should be in a name which will resolve to the correct IP address. Only if the server does not have a symbolic name should the numeric IP address be given.

Example:

```
<host>srw.cheshire3.org</host>
```

6.1.2 /explain/serverInfo/port

The port on the server to connect to, in numeric form.

Example:

```
<port>80</port>
```

6.1.3 /explain/serverInfo/database

The /explain/serverInfo/database element should contain the name of the Z39.50 database which the record describes, or for other protocols, such as SRW and SRU, this element may be used to contain the remainder of the URL to the service's endpoint without any preceding "/" character.

Examples:

```
<database>IR-Explain-1</database>
<database>cgi-bin/sru.cgi</database>
```

6.1.4 /explain/serverInfo/authentication

The last element in /explain/serverInfo is the optional authentication element. If this element is present, but empty, then it implies that authentication is required to connect to the server, however there is no publicly available login. Otherwise it will contain one of the following two options: open, or one or more elements from: user, group, and password. These elements are described below.

/explain/serverInfo/authentication has one attribute: @type. This may contain any value, but should be a profiled identifier for the type of authentication required. Expected values include:

1. "basic" – The service requires basic http authentication.
2. "password" – The service's authentication mechanism requires a user and/or password.
3. "token" – The service's authentication mechanism requires an opaque token.
4. "shibboleth" – The service requires Shibboleth-based authorization.
5. "domain" – The service will authenticate the user based on their domain.

More values may be profiled by the community as required.

Example:

```
<authentication type="token"/>
```

6.1.4.1 /explain/serverInfo/authentication/open

This element contains the token to give in order to authenticate with the service.

Example:

```
<open>07baf99d-f89e-4323-b2bf-7de43aa85e33</open>
```

6.1.4.2 /explain/serverInfo/authentication/user|group|password

These elements are used to record the values to supply to authenticate with a user/password based system. Some systems also require the user to identify which group or domain they are part of.

Example:

```
<user>azaroth</user>
<group>uk/liverpool</group>
<password>squirrelfish</password>
```

6.2 /explain/databaseInfo

This section contains the full text descriptions of various aspects of the database. Unless otherwise specified, the elements in this section are repeatable and may have the following attributes:

1. @lang – This is the language of the text within the element. This should be set to allow clients to present the appropriate text to the user based on language preferences. This should be recorded using the two letter code for the language, as defined in RFC 1766.

2. `@primary` – This attribute, if set to `"true"`, denotes that this should be considered the default text to provide if no particular language is requested. For example, the record author might set this on the English version of a text which is also represented in the record in Spanish and German.

6.2.1 /explain/databaseInfo/title

This element contains the name or title of the service to be presented to a user.

Example:

```
<title lang="en" primary="true">DSpace OAI Repository</title>
```

6.2.2 /explain/databaseInfo/description

This element should contain a full text description of why the database might be interesting. It can contain anything which is not covered by the other elements available in `/explain/databaseInfo`. It might contain a summary of the full collection description or if there is a collection description available, may not be present or may contain information relevant to this particular interface to the collection.

Example:

```
<description>An interface to the bibliographic holdings of the
Science Fiction Foundation at the University of
Liverpool</description>
```

6.2.3 /explain/databaseInfo/history

This element may contain a description of the history of the service. This might include references to the sponsors of the development of the service or significant events such as when it was first launched.

It has one additional attribute `@lastUpdate`. This attribute should contain the time at which the service was last updated. For example, the time at which new records were added or an indexing process was run. The value should be a time and date in ISO 8601 format (YYYY-MM-DD hh:mm:ss).

Example:

```
<history lastUpdate="2005-07-20 15:00:00">This service was created in
2002 to serve the information needs of the higher education
community. It is sponsored by the JISC and the NSF.</history>
```

6.2.4 /explain/databaseInfo/extent

This element may contain a description of the extent to which the service provides access to the collection. For example, an interface to an email list would be considered complete if it searched all of the emails sent to the list, or if not all emails are available, then that situation could be noted in this field.

It has one additional attribute: `@numberOfRecords`. This attribute should contain the number of records that the service makes available, in numeric form.

Example:

```
<extent numberOfRecords="5128">All records in the collection are
exposed through this interface.</extent>
```

6.2.5 /explain/databaseInfo/restrictions

This element may contain a free text description of any usage restrictions on the service or the material available through the service. It could include copyright or trademark statements, times at which the service is unavailable, and so forth. Profiles may instead specify a rights language to use in place of this free text.

Example:

```
<restrictions> All material is copyrighted by the University of
California.</restrictions>
```

6.2.6 /explain/databaseInfo/langUsage

This element may contain a description of the languages in which the records are made available by the service.

It has one additional attribute: `@codes`. This attribute should contain the two letter language codes for the record's languages (as specified in RFC-1766) separated by spaces.

Example:

```
<langUsage codes="en ru">The records are available in English and
Russian.</langUsage>
```

6.2.7 /explain/databaseInfo/agents

The `agents` section is a wrapper around one or more `agent` elements. It does not have `@lang` or `@primary` attributes and is not repeatable.

Each `agent` element can describe a personal or organizational agent associated with the service. The content can be anything, but it is recommended to be of the form `"name, email"`. The element has two attributes:

- `@type` – This may contain the name of the role the agent plays with respect to the service. It is optional and defaults to `"contact"`. The value for the attribute can be profiled, but expected roles include:
  1. `"administrator"` – An agent that performs administrative tasks.
  2. `"contact"` – The agent to contact in the first instance with correspondence concerning the service.
  3. `"creator"` – The agent that initially created or conceived of the service.
  4. `"owner"` – The owner of the service.
  5. `"sponsor"` – An agent that provides funding for the operation of the service.
  6. `"vendor"` – An agent that sells access to the service.
- `@identifier` – This attribute may contain the identifier for the agent in an external database of agents, from which more information can be obtained. The nature of the identifier and external database is not within the scope of this standard.

Example:

```
<agents>
  <agent type="creator">Andy Sawyer</agent>
  <agent type="contact">Robert Sanderson, azaroth@liv.ac.uk</agent>
</agents>
```

6.2.8 /explain/databaseInfo/implementation

The `implementation` section contains information about the underlying software which implements the service. It does not have the `@primary` or `@lang` attributes and is not repeatable. The element has two attributes:

- `@identifer` – This should contain an identifier for the software application. A URI is recommended, in particular the URL to the homepage for the software if one exists.
- `@version` – This attribute should contain the version number of the application.

`explain/databaseInfo/implementation` can contain zero or one `agents` elements and zero or more `title` elements.

The `agents` section is as described above, but should relate to the application rather than the service. For example, “contact” would reference the agent to contact for information about the application, and “vendor” would be the person or organization that sells or makes the software available.

The `title` element in this section should be a human readable name for the software.

Example:

```
<implementation identifier="http://www.cheshire3.org" version="1.0">
  <agents>
    <agent type="vendor">University of Liverpool</agent>
    <agent type="contact">Robert Sanderson, azaroth@liv.ac.uk</agent>
  </agents>
  <title>Cheshire3 SRW Server</title>
</implementation>
```

6.2.9 /explain/databaseInfo/links

The `links` element is a wrapper around one or more `link` elements. It does not have the `@lang` or `@primary` attributes and is not repeatable.

`/explain/databaseInfo/links/link` contains a reference to a related resource, collection, or service. It should contain a URL for the resource. It has one attribute, `@type`, which records the type of resource pointed at. This may contain any value, but the following types are expected:

1. `"collectionDescription"` – A link to the description of the collection that the service makes available.
2. `"www"` – URL to a native web based interface to the collection.
3. `"z39.50"` – URL to a z39.50 interface to the collection.
4. `"srw"` – URL to an SRW interface to the collection.
5. `"sru"` – URL to an SRU interface to the collection.
6. `"oai"` – URL to an OAI interface to the collection.
7. `"rss"` – URL to an RSS interface to the collection.
8. `"icon"` – URL to a graphic which can be used as an icon for this service.

Example:

```
<links>
  <link type="srw">http://srw.cheshire3.org/services/spy</link>
  <link type="z39.50">z3950s://cheshire3.org:2100/spy</link>
</links>
```

6.3 /explain/metaInfo

This section of the record is short and contains elements describing information about the record itself, essential for maintaining and using an aggregation of service description records.

6.3.1 /explain/metaInfo/dateModified

This element contains the time and date at which the author of the record last modified it. This should be updated every time the record is changed by the owner. (An aggregator changing the authoritative attribute or the following two elements does not constitute a change that should be recorded in this element.) The date should be in ISO 8601 format.

Example:

```
<dateModified>2005-07-20 13:30:00</dateModified>
```

6.3.2 /explain/metaInfo/dateAggregated

This element should be present if the record was harvested from another source into its current location. It should contain the time and date at which the aggregation took place.

Example:

```
<dateAggregated>2005-08-01 19:27:05</dateModified>
```

6.3.3 /explain/metaInfo/aggregatedFrom

This element should contain sufficient information for a third party to retrieve the original, authoritative record. The contents should be in the form of a URL. For records retrieved from Z39.50 services, the z39.50r specification should be used.

Example:

```
<aggregatedFrom>z39.50r://gondolin.hist.liv.ac.uk:210/IR-Explain---
1?id=ghlau-1;esn=F;rs=XML</aggregatedFrom>
```

6.4 /explain/indexInfo

This element is where information concerning the access points to the content is provided. These access points are represented as semantic indexes, where each index described should have different content to all other indexes.

It must contain at least one `index` element, and may contain any number of `set` or `sortKeyword` elements.

6.4.1 /explain/indexInfo/set

The `set` element is used to declare the support of a particular grouping mechanism for access point specification. For example, Z39.50 queries have sets of attributes and CQL has context sets, both of which serve to disambiguate and divide access points into contextual groupings.

This element has two attributes:

- `"name"` – This declares a short name for the set. For systems that support CQL, this is the name which should be used in the query to distinguish the context set. For Z39.50, this name is not sent to the server, but serves as a mapping mechanism for the index specifications.
- `"identifier"` – The globally unique identifier for this particular grouping. For CQL, this is the URI assigned as the identifier for the context set. For Z39.50, this should be the OID assigned to the attribute set.

The `set` element may contain one or more `title` elements. These elements contain a human readable name for the set, as opposed to the short form used in the query in the @name attribute.

Examples:

```
<set name="bib2" identifier="1.2.840.10003.3.18"/>
<set name="dc" identifier="info:srw/cql-context-set/1/dc-v1.1"/>
<set name="rec" identifier="info:srw/cql-context-set/2/rec-1.1">
  <title>Record Metadata Context Set</title>
</set>
```

6.4.2 /explain/indexInfo/index

An `index` represents a single type of search, scan, or non-keyword sort that can be performed.

The element has 4 attributes, the first three being `@search`, `@scan`, and `@sort`. These are true/false flags and record whether this particular operation is allowed on the index described. If the flag is not present, then the implication is that the creator of the record did not know whether the function was available or not, such as might be the case for a remotely discovered server. The last attribute, `@id`, is similar in function to the same attribute on `/explain`, but is used to uniquely identify the index within the record rather than the record within an aggregation.

Indexes may contain one or more `title` elements, one or more `map` elements and zero or one `configInfo` elements.

6.4.2.1 /explain/indexInfo/index/title

This element should contain the human readable title for the index to be presented to a user by client software.

Example:

```
<title>Author</title>
```

6.4.2.2 /explain/indexInfo/index/map

This element contains the protocol level information required to interact with the index. If it appears multiple times in one index, then the maps described are multiple ways of interacting with the same index. For example, a Z39.50 server may expose the same index with both BIB-1 and BIB-2 attribute combinations.

This element may have two attributes:

- `@primary` – This attribute has the same semantics as when used on other repeatable elements. If there is a choice, unless the client has a reason not to, it is desired that this map be used. It may contain "true" or "false".
- `@lang` – This attribute contains the name of the query language for which the map applies. It may contain any text, but expected values are:

  1. `"CQL"` – Common Query Language
  2. `"RPN"` – Reverse Polish Notation – Type1 Z39.50 query

  For CQL or other named maps, it may contain one `name` element.

  For RPN maps, it may contain one or more `attr` elements.

6.4.2.3 /explain/indexInfo/index/map/name

The `name` element contains the name of the CQL index, as defined by the context set of which it is a part. It has one attribute, `@set`. This should contain the name of the context set the index comes from, as defined in the appropriate `/explain/indexInfo/set` element.

Example:

```
<map lang="CQL"><name set="dc">title</name></map>
```

6.4.2.4 /explain/indexInfo/index/map/attr

The `attr` element is used to record a single RPN attribute. Multiple elements are then used together to build up an attribute combination.

The element may have two attributes:

- `@type` – This should contain the numeric type of the attribute. For example Bib-1 USE attributes have type 1.
- `@set` – This should contain the name of the attribute set the attribute comes from, as defined in the appropriate /explain/indexInfo/set element. If not supplied, this defaults to the Bib-1 attribute set.

The contents of the `attr` element should be the value to supply with the attribute. This may be either numeric data or a string.

Example:

```
<map>
  <attr type="1">1003</attr>
</map>
<map>
  <attr type="1" set="xd">3</attr>
  <attr type="2" set="bib2">3</attr>
  <attr type="12" set="bib2">aut</attr>
</map>
```

6.4.2.5 /explain/indexInfo/index/configInfo

Each index can have its own `configInfo` section, described in detail in section 6.6. This section allows description of defaults, settings, and capabilities of the index, as opposed to for the server as a whole. It may be useful, for example, to say that the server only supports the "within" operator on a certain index, rather than on every index. Equally, a default relation may be specified for an index which is different to the default relation for the server.

6.4.3 /explain/indexInfo/sortKeyword

This element can contain a non-indexed field by which the records can be sorted.

Example:

```
<sortKeyword>relevanceScore</sortKeyword>
```

7 Retrieval

In order to not clutter up records with redundant information, this standard has two parallel sections for recording the formats in which records can be obtained from the service. Many modern IR protocols make their records available in XML (e.g., SRU, SRW, and OAI). Z39.50 however can transfer records in any format. In the first scenario, a `schemaInfo` section would be used. In the latter, a `recordInfo` section would be used.

These are described in sections 7.1 and 7.2.

7.1 /explain/recordInfo

This element may contain one or more `recordSyntax` elements and has no attributes. This element should be used for services that make records available in formats other than just XML.

7.1.1 /explain/recordInfo/recordSyntax

This element is used to specify a format in which the record may be retrieved. For example, records may be available in plain text, PDF, XML and as JPEG images.

It has two attributes:

- `@name` – The name of the format.
- `@identifier` – An external identifier for the format. For Z39.50 this should be the assigned OID.

It may contain any number of `elementSet` elements. If it is empty, then it is assumed that only the default complete record is available in that format.

Example:

```
<recordSyntax name="XML" identifier="1.2.840.10003.5.109.10"/>
```


7.1.1.1 /explain/recordInfo/recordSyntax/elementSet

This element is used to record an identifier for the set of fields used in alternate ways of retrieving the record in the same `recordSyntax`. For example, a record in XML might conform to a schema, or a record in GRS-1 might conform to a tag-set. Equally, these `elementSets` might be used to differentiate between complete records and summary records.

`elementSet` has the same two attributes as `recordSyntax` (`@name` and `@identifier`) with the same semantics.

The element may be empty or contain one or more `title` elements. The title should be a human readable name for the element set.

Example:

```
<elementSet name="F">
  <title lang="en">Full XML Record</title>
</elementSet>
```

7.2 /explain/schemaInfo

This element should be present only if `recordInfo` is not present. It is used to list the XML schemas available.

It should contain one or more `schema` elements.

7.2.1 /explain/schemaInfo/schema

This element is used to record the support for an XML schema. It has five attributes:

- `@name` – A short name for the schema, in the same way as other name attributes.
- `@identifier` – An external identifier for the schema.
- `@retrieve` – Either "true" or "false", if the schema is usable for retrieval.
- `@sort` – Either "true" or "false", if the schema is usable for sorting.
- `@location` – A URL to the schema definition.

It may contain one or more `title` elements which should contain a human readable name for the schema.

Example:

```
<schema name="dc" identifier="info:srw/schema/1/dc-v1.1"
retrieve="true" sort="true"
location="http://www.loc.gov/z3950/agency/zing/srw/dc-schema.xsd">
  <title>Simple Dublin Core</title>
</schema>
```

7.3 /explain/configInfo

This section contains configuration information about how the server is set up. It has three possible tags within it, each being repeatable as many times as required.

The following elements have one attribute, `@type`, which specifies the sort of configuration option. It may have any value, but the following table describes commonly expected ones.

They may contain either text, or a map element, as defined above.

Type	*Description*
numberOfRecords	The default number of records that a server will return at once
contextSet	The default context set
index	The default index
relation	The default relation
sortSchema	The default schema used in sorting, in short name form
retrieveSchema	The default schema used for retrieved records
resultSetTTL	Default number of seconds that a result set will be maintained for
stylesheet	Default stylesheet URL, or if stylesheets are supported
recordPacking	Default record packing returned (string or xml)
numberOfTerms	Default number of terms to be returned in scan
maximumRecords	The maximum number of records that a server will return at once
proximity	Does the server support proximity (Empty)
resultSets	Does the server support result sets (Empty)
relation	A relation supported by the server or index
relationModifier	A relation modifier supported by the server or index
booleanModifier	A relation modifier supported by the server or index
sort	Does the server support sort
sortModifier	A supported parameter for sort (ascending, missingValue, caseSensitive)
maskingCharacter	A masking character supported (* or ?)
anchoring	Is anchoring supported? (^ character)
emptyTerm	Are empty terms supported (Empty)
proximityModifier	A proximity modifier supported by the server or index (relation, distance, unit, ordering)
extension	An extension supported by the server Represented as two space separated words: the first is the identifier for the extension and the second is the individual element name from the extension. If there is only one word, then it is the extension id and all elements from within are supported.
profile	The URI identifier of a supported profile
recordXPath	Is XPath retrieval supported?
scan	Is the Scan operation supported?

Type	*Description*
version	An older version of the protocol supported, as opposed to the one listed in serverInfo/@version

7.3.1 `/explain/configInfo/default`

This element can contain a default value, one which can be overridden in a request such as the number of records to be returned in a single response.

Example:

```
<default type="numberOfRecords">50</default>
```

7.3.2 `/explain/configInfo/setting`

This element can contain an unmodifiable value for the server. For example, the maximum number of records that the server will return in a single response.

Example:

```
<setting type="maximumRecords">100</setting>
```

7.3.3 `/explain/configInfo/supports`

This final element describes a feature which is supported by the server, for example a CQL relation modifier or an optional feature.

Examples:

```
<supports type="relationModifier">partial</supports>
<supports type="resultSets"/>
```

Appendix A
(normative)
ZeeRex Definitions

A.1 XML DTD

```
<!-- ZeeRex DTD, version 2.1/2005-08-04 -->
<!-- Maintainer: Robert Sanderson, azaroth@liv.ac.uk -->

<!ELEMENT explain (serverInfo, databaseInfo?, metaInfo?, indexInfo?,
(recordInfo|schemaInfo)?, configInfo?)>
<!ATTLIST explain
                  id CDATA #IMPLIED>

<!-- Server Info -->


<!ELEMENT serverInfo (host, port, database, authentication?)>
<!ATTLIST serverInfo
                     protocol CDATA #IMPLIED
                     version CDATA #IMPLIED
                     transport CDATA #IMPLIED
                     method CDATA #IMPLIED>

<!ELEMENT host (#PCDATA)>
<!ELEMENT port (#PCDATA)>
<!ELEMENT database (#PCDATA)>


<!ELEMENT authentication (open | (user?, group?, password?))>
<!ATTLIST authentication
                         type CDATA #IMPLIED
                         required (true|false) "true">
<!ELEMENT open (#PCDATA)>
<!ELEMENT user (#PCDATA)>
<!ELEMENT group (#PCDATA)>
<!ELEMENT password (#PCDATA)>

<!-- Database Info -->

<!ELEMENT databaseInfo (title*, description*, (extent | history | langUsage
| restrictions)*, agents?, links?, implementation?)>


<!-- Note that title is used in more than just databaseInfo -->
<!ELEMENT title (#PCDATA)>
<!ATTLIST title
               primary (true|false) #IMPLIED
               lang CDATA #IMPLIED>
```



```
<!ELEMENT description (#PCDATA)>
<!ATTLIST description
                      primary (true|false) #IMPLIED
                      lang CDATA #IMPLIED>

<!ELEMENT extent (#PCDATA)>
<!ATTLIST extent
                 primary (true|false) #IMPLIED
                 lang CDATA #IMPLIED>

<!ELEMENT history (#PCDATA)>
<!ATTLIST history
                  primary (true|false) #IMPLIED
                  lang CDATA #IMPLIED>

<!ELEMENT langUsage (#PCDATA)>
<!ATTLIST langUsage
                    codes CDATA #IMPLIED
                    primary (true|false) #IMPLIED
                    lang CDATA #IMPLIED>

<!ELEMENT restrictions (#PCDATA)>
<!ATTLIST restrictions
                       primary (true|false) #IMPLIED
                       lang CDATA #IMPLIED>
<!ELEMENT agents (agent+)>

<!ELEMENT agent (#PCDATA)>
<!ATTLIST agent
                 type CDATA #IMPLIED
                 identifier CDATA #IMPLIED>


<!ELEMENT implementation (agents?, title*)>
<!ATTLIST implementation
               identifier CDATA #IMPLIED
               version CDATA #IMPLIED>

<!ELEMENT links (link+)>
<!ELEMENT link (#PCDATA)>
<!ATTLIST link
               type CDATA #IMPLIED>


<!-- Meta Info -->

<!ELEMENT metaInfo (dateModified, (aggregatedFrom, dateAggregated)?)>


<!ELEMENT dateModified (#PCDATA)>
<!ELEMENT aggregatedFrom (#PCDATA)>
<!ELEMENT dateAggregated (#PCDATA)>


<!-- Index Info -->
```

```
<!ELEMENT indexInfo ((set | index | sortKeyword)+)>

<!ELEMENT set (title*)>
<!ATTLIST set
                  name CDATA #REQUIRED
                  identifier CDATA #REQUIRED>

<!ELEMENT index (title*, map+, configInfo?)>
<!ATTLIST index
                id CDATA #IMPLIED
                search (true|false) #IMPLIED
                scan (true|false) #IMPLIED
                sort (true|false) #IMPLIED>

<!ELEMENT sortKeyword (#PCDATA)>

<!ELEMENT map ((attr+)|name)>
<!ATTLIST map
              primary (true|false) "false">

<!ELEMENT name (#PCDATA)>
<!ATTLIST name
               set CDATA #IMPLIED>

<!ELEMENT attr (#PCDATA)>
<!ATTLIST attr
               type CDATA #REQUIRED
               set CDATA #IMPLIED>


<!-- Record Info and Schema Info -->

<!ELEMENT recordInfo (recordSyntax+)>
<!ELEMENT recordSyntax (elementSet+)>
<!ATTLIST recordSyntax
                      name CDATA #IMPLIED
                      identifier CDATA #IMPLIED>

<!ELEMENT elementSet (title*)>
<!ATTLIST elementSet
                      name CDATA #REQUIRED
                      identifier CDATA #IMPLIED>

<!ELEMENT schemaInfo (schema+)>
<!ELEMENT schema (title*)>
<!ATTLIST schema
                identifier CDATA #REQUIRED
                name CDATA #REQUIRED
                location CDATA #IMPLIED
                sort (true|false) "false"
                retrieve (true|false) "true">

<!-- Config Info -->
```

```
<!ELEMENT configInfo ((default|setting|supports)*)>
<!ELEMENT default (#PCDATA|map)>
<!ATTLIST default
                  type CDATA #REQUIRED>
<!ELEMENT setting (#PCDATA|map)>
<!ATTLIST setting
                  type CDATA #REQUIRED>
<!ELEMENT supports (#PCDATA|map)>
<!ATTLIST supports
                   type CDATA #REQUIRED>
```

A.2 XML Schema

```
<xs:schema xmlns:xs='http://www.w3.org/2001/XMLSchema'>
 <xs:annotation>
  -- This schema was automatically generated by Syntext Dtd2Schema --
  -- conversion tool (from file: zeerex-2.1.dtd) --
  -- Copyright (C) 2002, 2003 Syntext Inc. See http://www.syntext.com for
updates. --
 </xs:annotation>
 <xs:element name='agent'>
  <xs:complexType mixed='true'>
  <xs:attribute name='type'/>
  <xs:attribute name='identifier'/>
  </xs:complexType>
 </xs:element>
 <xs:element name='agents'>
  <xs:complexType>
   <xs:sequence>
     <xs:element ref='agent' maxOccurs='unbounded'/>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:element name='aggregatedFrom'>
  <xs:complexType mixed='true'>
  </xs:complexType>
 </xs:element>
 <xs:element name='attr'>
  <xs:complexType mixed='true'>
  <xs:attribute name='type' use='required'/>
  <xs:attribute name='set'/>
  </xs:complexType>
 </xs:element>
 <xs:element name='authentication'>
  <xs:complexType>
   <xs:choice>
     <xs:element ref='open'/>
     <xs:sequence>
       <xs:element ref='user' minOccurs='0'/>
       <xs:element ref='group' minOccurs='0'/>
       <xs:element ref='password' minOccurs='0'/>
     </xs:sequence>
```

```
   </xs:choice>
  <xs:attribute name='type'/>
  <xs:attribute name='required' default='true'>
   <xs:simpleType>
    <xs:restriction base='xs:string'>
     <xs:enumeration value='true'/>
     <xs:enumeration value='false'/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name='configInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='default'/>
      <xs:element ref='setting'/>
      <xs:element ref='supports'/>
    </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='database'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='databaseInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='title' minOccurs='0' maxOccurs='unbounded'/>
    <xs:element ref='description' minOccurs='0' maxOccurs='unbounded'/>
    <xs:choice minOccurs='0' maxOccurs='unbounded'>
      <xs:element ref='extent'/>
      <xs:element ref='history'/>
      <xs:element ref='langUsage'/>
      <xs:element ref='restrictions'/>
      <xs:element ref='agents'/>
      <xs:element ref='links'/>
      <xs:element ref='implementation'/>
    </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='dateAggregated'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='dateModified'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='default'>
 <xs:complexType name='defaultType' mixed='true'>
```

```
 <xs:attribute name='type' use='required'/>
 <xs:sequence>
   <xs:element ref="map"/>
 </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='description'>
 <xs:complexType mixed='true'>
 <xs:attribute name='primary'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name='lang'/>
 </xs:complexType>
</xs:element>
<xs:element name='elementSet'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='title' minOccurs='0' maxOccurs='unbounded'/>
  </xs:sequence>
 <xs:attribute name='name' use='required'/>
 <xs:attribute name='identifier'/>
 </xs:complexType>
</xs:element>
<xs:element name='explain'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='serverInfo'/>
    <xs:element ref='databaseInfo' minOccurs='0'/>
    <xs:element ref='metaInfo' minOccurs='0'/>
    <xs:element ref='indexInfo' minOccurs='0'/>
    <xs:choice minOccurs='0'>
      <xs:element ref='recordInfo'/>
      <xs:element ref='schemaInfo'/>
    </xs:choice>
    <xs:element ref='configInfo' minOccurs='0'/>
  </xs:sequence>
 <xs:attribute name='id'/>
 </xs:complexType>
</xs:element>
<xs:element name='extent'>
 <xs:complexType mixed='true'>
 <xs:attribute name='primary'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
```

```
 <xs:attribute name='lang'/>
 </xs:complexType>
</xs:element>
<xs:element name='group'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='history'>
 <xs:complexType mixed='true'>
 <xs:attribute name='primary'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name='lang'/>
 </xs:complexType>
</xs:element>
<xs:element name='host'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='implementation'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='title' minOccurs='0' maxOccurs='unbounded'/>
  </xs:sequence>
 <xs:attribute name='identifier'/>
 <xs:attribute name='version'/>
 </xs:complexType>
</xs:element>
<xs:element name='index'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='title' minOccurs='0' maxOccurs='unbounded'/>
    <xs:element ref='map' maxOccurs='unbounded'/>
    <xs:element ref='configInfo' minOccurs='0'/>
  </xs:sequence>
 <xs:attribute name='id'/>
 <xs:attribute name='search'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name='scan'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
```

```
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name='sort'>
   <xs:simpleType>
    <xs:restriction base='xs:string'>
     <xs:enumeration value='true'/>
     <xs:enumeration value='false'/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name='indexInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:choice maxOccurs='unbounded'>
      <xs:element ref='set'/>
      <xs:element ref='index'/>
      <xs:element ref='sortKeyword'/>
    </xs:choice>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='langUsage'>
 <xs:complexType mixed='true'>
 <xs:attribute name='codes'/>
 <xs:attribute name='primary'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name='lang'/>
 </xs:complexType>
</xs:element>
<xs:element name='link'>
 <xs:complexType mixed='true'>
 <xs:attribute name='type'/>
 </xs:complexType>
</xs:element>
<xs:element name='links'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='agents' minOccurs='0'/>
    <xs:element ref='link' maxOccurs='unbounded'/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='map'>
 <xs:complexType>
  <xs:choice>
```

```
      <xs:element ref='attr' maxOccurs='unbounded'/>
     <xs:element ref='name'/>
   </xs:choice>
  <xs:attribute name='primary' default='false'>
   <xs:simpleType>
    <xs:restriction base='xs:string'>
     <xs:enumeration value='true'/>
     <xs:enumeration value='false'/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
 </xs:complexType>
</xs:element>
<xs:element name='metaInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='dateModified'/>
    <xs:sequence minOccurs='0'>
      <xs:element ref='aggregatedFrom'/>
      <xs:element ref='dateAggregated'/>
    </xs:sequence>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='name'>
 <xs:complexType mixed='true'>
 <xs:attribute name='set'/>
 </xs:complexType>
</xs:element>
<xs:element name='open'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='password'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='port'>
 <xs:complexType mixed='true'>
 </xs:complexType>
</xs:element>
<xs:element name='recordInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='recordSyntax' maxOccurs='unbounded'/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='recordSyntax'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='elementSet' maxOccurs='unbounded'/>
  </xs:sequence>
  <xs:attribute name='name'/>
```

```
 <xs:attribute name='identifier'/>
 </xs:complexType>
</xs:element>
<xs:element name='restrictions'>
 <xs:complexType mixed='true'>
 <xs:attribute name='primary'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name='lang'/>
 </xs:complexType>
</xs:element>
<xs:element name='schema'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='title' minOccurs='0' maxOccurs='unbounded'/>
  </xs:sequence>
 <xs:attribute name='identifier' use='required'/>
 <xs:attribute name='name' use='required'/>
 <xs:attribute name='location'/>
 <xs:attribute name='sort' default='false'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 <xs:attribute name='retrieve' default='true'>
  <xs:simpleType>
   <xs:restriction base='xs:string'>
    <xs:enumeration value='true'/>
    <xs:enumeration value='false'/>
   </xs:restriction>
  </xs:simpleType>
 </xs:attribute>
 </xs:complexType>
</xs:element>
<xs:element name='schemaInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='schema' maxOccurs='unbounded'/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
<xs:element name='serverInfo'>
 <xs:complexType>
  <xs:sequence>
    <xs:element ref='host'/>
    <xs:element ref='port'/>
```

```
    <xs:element ref='database'/>
    <xs:element ref='authentication' minOccurs='0'/>
   </xs:sequence>
  <xs:attribute name='protocol'/>
  <xs:attribute name='version'/>
  <xs:attribute name='transport'/>
  <xs:attribute name='method'/>
  </xs:complexType>
</xs:element>
<xs:element name='set'>
  <xs:complexType>
   <xs:sequence>
     <xs:element ref='title' minOccurs='0' maxOccurs='unbounded'/>
   </xs:sequence>
  <xs:attribute name='name' use='required'/>
  <xs:attribute name='identifier' use='required'/>
  </xs:complexType>
</xs:element>
<xs:element name='setting'>
  <xs:complexType mixed='true'>
  <xs:attribute name='type' use='required'/>
  <xs:sequence>
    <xs:element ref="map"/>
  </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name='sortKeyword'>
  <xs:complexType mixed='true'>
  </xs:complexType>
</xs:element>
<xs:element name='supports'>
  <xs:complexType mixed='true'>
  <xs:attribute name='type' use='required'/>
  <xs:sequence>
    <xs:element ref="map"/>
  </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name='title'>
  <xs:complexType mixed='true'>
  <xs:attribute name='primary'>
   <xs:simpleType>
    <xs:restriction base='xs:string'>
     <xs:enumeration value='true'/>
     <xs:enumeration value='false'/>
    </xs:restriction>
   </xs:simpleType>
  </xs:attribute>
  <xs:attribute name='lang'/>
  </xs:complexType>
</xs:element>
<xs:element name='user'>
  <xs:complexType mixed='true'>
  </xs:complexType>
```



```
 </xs:element>
</xs:schema>
```

Appendix B
(informative)
ZeeRex Profile for CQL

(This appendix is not part of the *Information Retrieval Service Description Specification*, NISO Z39.92–200X. It is included for information only.)

B.1 Introduction

This profile addresses the need for searching aggregations of ZeeRex records via CQL.

The identifier for this profile is: **info:srw/profile/2/zeerex-1.1**

The maintainer of this profile is: Rob Sanderson, azaroth@liv.ac.uk

Conformance with this profile can be advertised in a ZeeRex record with:

```
<supports type="profile">info:srw/profile/2/zeerex-1.1</supports>
```

B.2 Profile

The ZeeRex profile uses indexes from five different context sets; cql (identifier: info:srw/cql-context-set/1/cql-v1.1), dublin core (dc, identifier: info:srw/cql-context-set/1/dc-v1.1), network (net, identifier: info:srw/cql-context-set/2/net-1.0), metadata (rec, identifier: info:srw/cql-context-set/2/rec-1.1) and ZeeRex (zeerex, identifier: info:srw/cql-context-set/2/zeerex-1.1).

For more information on these context sets, please see:
http://www.loc.gov/z3950/agency/zing/cql/context-sets.html

B.2.1 Indexes

Req'd	***Index Name***	***Description***	***XPath in ZeeRex***
	cql.anywhere	Anywhere in the record	/
Yes	dc.title	The human readable title for the database	databaseInfo/title
Yes	dc.description	A description of the database	databaseInfo/description
	dc.date	Time that the database was last updated	databaseInfo/history/@lastUpdate
	dc.creator	The maintainer for the database	databaseInfo/agents/agent[@type='creator']
	dc.language	The language of the records in the database described by the ZeeRex record	databaseInfo/langusage
Yes	net.host	The hostname (srw.cheshire3.org) or IP address (138.253.81.47) of the database/server, but each case will likely find only records which contain the host in that form. For example 138.253.81.47 will not necessarily find srw.cheshire3.org	serverInfo/host

Req'd	*Index Name*	*Description*	*XPath in ZeeRex*
Yes	net.port	The port on which the database/server can be reached	serverInfo/port
Yes	net.protocol	The protocol to use to interact with the database/server	serverInfo/@protocol
Yes	net.version	The version of the protocol	serverInfo/@version
Yes	net.path	The name (z39.50) or path (OAI, FTP, SRW/U, etc) to the service from the base URL	serverInfo/database
Yes	net.method	The HTTP method used to access the service (values: 'GET' 'POST' or 'GET POST')	serverInfo/@method
	zeerex.numberOfRecords	Number of records in the database	databaseInfo/extent/@numberOfRecords
	zeerex.set	The context set/attribute set for an index	indexInfo/set/@identifier
	zeerex.index	The name of an SRW/U index	indexInfo/index/map/name
	zeerex.attributeType	Type of an attribute in Z39.50	indexInfo/index/map/attr@type
	zeerex.attributeValue	Value of an attribute in Z39.50	indexInfo/index/map/attr
	zeerex.schema	Identifier for a record schema in SRW/U	schemaInfo/schema/@identifier
	zeerex.recordSyntax	Identifier for a record syntax in Z39.50. This should be specified as an OID (1.2.840.10003.5.10).	recordInfo/recordSyntax/@identifier
	zeerex.supports_*	Described in ZeeRex context set	/explain/configInfo/supports[@type=*]
Yes	rec.lastModificationDate	The date that the ZeeRex record was last modified (as opposed to the date that the database was last updated (dc.date))	metaInfo/dateModified
Yes	rec.authorityIndicator	Is the record known to be the authoritative description, or might there be a more correct record somewhere else? (values: 'true' or 'false')	explain/@authoritative

Servers MUST support the indexes marked with "Yes" in order to be in compliance with the ZeeRex profile. Servers should still implement as many of the other indexes as possible, however.

B.2.2 Relation Modifiers

Relation Modifier	*Description*
cql.uri	To distinguish a URI when used with zrx.set, zrx.schema
cql.oid	To distinguish an OID when used with zrx.set, zrx.recordSyntax
cql.isoDate	To distinguish an ISO8601 date with dc.date

It is hoped that servers will support the use of proximity to search for combinations of set, index, attributeType and attributeValue, but it is recognized that this may not be possible in all implementations. See the ZeeRex context set for examples of this use of proximity.

B.2.3 Retrieval

Servers must allow retrieval of records in the ZeeRex schema. Servers should allow retrieval in the simple Dublin Core schema.

Appendix C
(informative)
ZeeRex Profile for Z39.50

(This appendix is not part of the *Information Retrieval Service Description Specification*, NISO Z39.92–200X. It is included for information only.)

C.1 Introduction

ZeeRex databases are searched in accordance with the framework specified by the Z39.50 Attribute Architecture, using attributes drawn from three architecture-conformant attribute sets:

- The utility set (http://www.loc.gov/z3950/agency/attrarch/util.html)
- The cross-domain set (http://staff.oclc.org/~levan/docs/crossdomainattributeset.html)
- The ZeeRex attribute set (http://explain.z3950.org/search/attrs.html)

C.2 Access Points

The following access points may be supported. Those which are marked as “required” *must* be supported by a conformant ZeeRex server; the others are optional.

Attribute Set	***Access Point***	***Required***	***Name***	***Notes***
Utility	12	Yes	Anywhere in Record	This access point may be used in conjunction with the Utility attribute set's a`lwaysMatches` comparison attribute (type 8, value 1) to find all the records in a ZeeRex database. Any search-term may be used: the term itself is ignored in all-records searches.
Utility	1	-	Record Date/time	Searches for the date specified in the `<dateModified>` element. Most useful if supported with the utility set's inequality comparison attributes 4 (less than), 5 (less than or equal), 6 (greater than) and 7 (greater than or equal).
Cross-Domain	1	Yes	Title	Searches for the title of the whole database, not that of an index or element set.
Cross-Domain	3	-	Name	Searches for the name specified in the creator agent element.
Cross-Domain	4	-	Description	

Attribute Set	*Access Point*	*Required*	*Name*	*Notes*
Cross-Domain	10	-	Language	Searches for the language specified in the <langUsage> element—that is, the language of the described database rather than that of the ZeeRex record that describes it.
ZeeRex	1	Yes	explainAdmin	
ZeeRex	2	Yes	hostName	
ZeeRex	3	Yes	portNumber	
ZeeRex	4	Yes	databaseName	
ZeeRex	5	-	recordSyntax	
ZeeRex	6	Yes	protocol	
ZeeRex	7	-	attribute	See below.

For further information about these access points, consult the attribute sets from which they are taken.

C.3 Attribute Combinations

In order to search for databases supporting a particular combination of attributes, it is not sufficient to search for those attributes ANDed together. For example, a search for

```
attribute util:1=1 AND attribute util:12=creation
```

will not only find databases which support a utility-set search on creation date/time, it will also find databases which support both modification date/time and creation agent.

In order to search for multiple attributes used in combination, it's necessary to specify that they occur within the same <map> element. That may be done using the PROXIMITY operator with the following parameters:

- distance = 0 (meaning “the same element”)
- relation = Equal
- Unit = Element (meaning the <map> element)
- Exclusion = off