Overview:
- Motivations
- Explain Operation
- SearchRetrieve Operation
- Scan Operation
- CQL
- Implementations
- NISO Metasearch Initiative
SRW/U: Introduction

SRW is an Information Retrieval web service based protocol.

Motivations:
- Create an easy to implement protocol with the power of Z39.50
- Need for a generic IR web service
- Use existing off the shelf solutions where possible
- Re-evaluate Z39.50 “a good idea at the time” baggage
- Avoid library-centric preconception

Information:
http://www.loc.gov/srw/
http://www.loc.gov/cql/
http://explain.z3950.org/

Rob Sanderson (azaroth@liv.ac.uk)
SRW is the SOAP binding. (POST of XML request and response)
SRU is the REST binding. (GET parameters for XML response)

**Current Version:** 1.1

**Related Specifications:**
- CQL (Common Query Language)
- ZeeRex (Z39.50 Explain Explained and Re-Engineered in XML)

**Three Operations Available:**
- Explain
- SearchRetrieve
- Scan
SRW/U: Explain Operation

Purpose: Retrieve Service Description

Request Parameters:
- Version: '1.1'
- RecordPacking: 'string' or 'xml'
- Stylesheet: URI
- ExtraRequestData: Namespaced XML

Response Parameters:
- Version: '1.1'
- Record: Record (ZeeRex schema)
- Diagnostics: Sequence of Diagnostic
- ExtraResponseData: Namespaced XML
- EchoedExplainRequest: Request
Version (all messages):
Request: version of the request and that the client wants the response to be less than, or preferably equal to, that version.
Response: version of the response.

Stylesheet (all requests):
URL of xml stylesheet for the response, for SRU thin clients.

Extra{Request|Response}Data (all messages):
Space for extensions. Extensions must be requested, and requests may be silently ignored. Carried as embedded, namespaced XML in SRW, or on the URL with an 'x-' prefix for SRU requests.

Echoed*Request (all responses):
Thin clients may not even know what they just asked for.
SRW/U: Explain Record

ZeeRex Service Description Schema

Six sections to a record:

- **ServerInfo:** Host, port, database path
- **DatabaseInfo:** Title, description, access rights, etc.
- **MetaInfo:** Updates, aggregation
- **IndexInfo:** CQL contextSets and indexes
- **SchemaInfo:** Available record schemas
- **ConfigInfo:** Settings, supports and defaults

Can be (and is already) used to automatically configure a client, in the way that Z39.50's Explain service was supposed to work. Even simple XSLT transformations can be used to generate a usable HTML interface.

**Website:** [http://explain.z3950.org/](http://explain.z3950.org/)
SRW/U: XSLT Thin Client

Legend of the Five Rings Database

A database containing complete descriptions of cards from the LSR CCG.

### Search

<table>
<thead>
<tr>
<th>Index</th>
<th>Relation</th>
<th>Term</th>
<th>Boolean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anywhere</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Name</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Type</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Text</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Keyword</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Card Flavor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artist</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Set</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Legality</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rarity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Location of Card</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number at Location</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Force</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHI</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Experience Level</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focus Value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Province Strength</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Personal Honor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Honor Requirement</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Honor Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Record Schema: Dublin Core

Number of Records: 1

Record Position: 1

### Browse

Browse: Any

Response: No Max

Maximum To: 

Browse:

---

XML code for the Legend of the Five Rings Database:
Purpose: Find, Sort and Display Records

Request Parameters:
- **Version**: '1.1'
- **Query**: CQL Query (string)
- **RecordSchema**: URI Schema Identifier or alias
- **RecordPacking**: 'string' or 'xml'
- **RecordXPath**: XPath Expression (string)
- **StartRecord**: Positive Integer
- **MaximumRecords**: Non Negative Integer
- **ResultSetTTL**: Non Negative Integer
- **SortKeys**: Sort Expression (string)
- **Stylesheet**: URI
- **ExtraRequestData**: Namespaced XML data
**RecordPacking:**
Either embed the record or return as an encoded string. Raw XML is good for thin clients, but can be problematic in SOAP.

**RecordXPath:**
An XPath expression to select nodes from the record to return. Useful for paging through long records (EAD, X3D, SVG), or selecting a known section (title/author display). Returned in an XML encoded form.

**ResultSetTTL:**
Servers need to know if the client wants a result set to be maintained or not. This parameter allows the client to request the set be maintained for a certain time.
State:
Z39.50 has session state due to a persistent connection. HTTP does not have a persistent connection between requests.

State is optional in SRW as it is only necessary for persistent result sets. These result sets might persist between connections in Z39.50 as well. Not every SRW service will require persistent result sets, eg an SRW interface to Google.

Result Sets:
A result set is an ordered list of records. Each result set has a unique id and a time to live. It is named by the server and cannot change once created. For example, if a result set is sorted, it will receive a new identifier.
SortKeys:
One or more keys by which to sort the result set. Each key has the following structure:

- Path: XPath expression
- Schema: URI Identifier for a record schema
- Ascending: Boolean, sort upwards or downwards
- CaseSensitive: Boolean, treat case separately
- MissingValue: Action to take if the path is not present

Some schemas might include 'utility' paths which do not resolve to a particular element, but instead allow sorting by a dynamic feature such as relevance score.
SRW/U: SearchRetrieve Response

Purpose: Find and Display Records

Response Parameters:
- Version: '1.1'
- NumberOfRecords: Non Negative Integer
- ResultSetId: String
- ResultSetIdleTime: Non Negative Integer
- Records: Sequence of Record
- NextRecordPosition: Non Negative Integer
- Diagnostics: Sequence of Diagnostic
- ExtraResponseData: Namespaced XML
- EchoedSearchRetrieveRequest: Request + XQuery
Records are transferred in SRW in a simple structure:

- **RecordSchema**: URI identifier for the schema
- **RecordPacking**: 'string' or 'xml'
- **RecordData**: The actual record
- **RecordPosition**: Record's position in the resultSet
- **ExtraRecordData**: Space for further metadata

Like Z39.50, SRW does not assume a record structure in the database, but all records must be transferred in an XML format, either raw or string encoded. A list of registered schema identifiers is available, but any may be used.
We need to know which XML schema to use for the response. Can't use namespace or schema location. Identified by a unique URI, some of which are from 'info'.

'info' was developed in the OAI/OpenURL world. SRW info URIs have the form:
  info:srw/<type>/<owner>/<identifier>

eg: info:srw/cql-context-set/1/cql-v1.1

Identifiers used for record schemas, context sets, diagnostics, extensions and profiles. Can be info, but many are also regular http URIs.

eg: http://srw.cheshire3.org/contextSets/ccg/1.1/

(Yes a URI Identifier is kind of like an ATM Machine)
SRW/U: Scan Operation

Purpose: Retrieve Ordered List of Searchable Terms

Request Parameters:
- Version '1.1'
- ScanClause CQL Search Clause (string)
- ResponsePosition Non Negative Integer
- MaximumTerms Non Negative Integer
- Stylesheet URI
- ExtraRequestData Namespaced XML

Response Parameters:
- Version '1.1'
- Terms Sequence of Term
- Diagnostics Sequence of Diagnostic
- ExtraResponseData Namespaced XML
- EchoedScanRequest Request
Scan Term Structure:

- **Value**: String
- **NumberOfRecords**: Non Negative Integer
- **DisplayTerm**: String
- **WhereInList**: 'first', 'last', 'only' or 'inner'
- **ExtraTermData**: Namespaced XML

Scan in SRW is almost identical to the same operation in Z39.50. Even ScanClause is similar, although other options were discussed for carrying this information to determine the index and position.
Sometimes things go wrong.

**Diagnostic Structure:**
- **Uri**  
  URI diagnostic code
- **Details**  
  Machine readable or debugging string
- **Message**  
  Human readable string

Both fatal and non-fatal, surrogate and non-surrogate. URI is new in 1.1, and allows user communities to define their own profiled diagnostics.

Mappings available both to and from Z39.50's diagnostics.
Goals: Simple and Intuitive, also Powerful and Expressive

**Primary CQL Components:**
- Term: “fish”
- Index: dc.title
- Relation: =
- Boolean: and

**Extended CQL Components:**
- Relation Modifiers: cql.word
- Boolean Modifiers: distance>3
- Context Sets: dc.
- Prefixes: >dc=“info:srw/cql-context-set/...”
Nothing explains a query language like a few examples:

- “duck”
- title = “duck”
- title = “duck” and author = “sanderson”
- dc.title any “duck fish”
- cql.resultSetId=“a8fljqqq” or cql.resultSetId=“889flkcc”
- dc.date within/cql.isoDate “2004-04-06 2004-04-23”
- title any/relevant/rel.algorithm=CORI and/rel.mergeSum ...
- (a or b) and (c or d)
- cat prox/distance<5/unit=word/ordered hat
- dc.title any “^cat ^dog hat*”
- bib1.1003 =/bib1.2=3/bib1.3=6/bib1.5=100 “fish”
- prox and/or  <dc:title>and or</dc:title>  and or
SRW/CQL: Extensability

CQL Context Sets:
- Indexes
- Relations
- Relation/Boolean Modifiers

Operations:
- ExtraRequestData/ExtraResponseData
- SOAP/HTTP Headers
- New Operations

Individual Entries:
- ExtraRecordData/ExtraTermData
- Diagnostics
- Record Schemas
Standards Body: NISO (Or ISO as profile of Z39.50)
Maintenance Agency: Library of Congress
Proposed Editorial Board:
  Managing Editor: Ray Denenberg
  Senior Editor: Rob Sanderson
  Technical Editor: Matthew Dovey
  Plus about 8 others from Australia, US and Europe.

All additions for future versions will first require implementation as extensions to prevent creeping bloat.

Extensions to SRW and CQL do NOT have to be registered.
SRW/CQL: Implementations

Cheshire:
- Python based gateway to Z server
- Native CQL/SRW in Cheshire3

IndexData:
- Native SRW/CQL support in YAZ 2.0
- Gateway to the Library of Congress bibliographic database

OCLC:
- Axis based gateway to Z servers
- DSpace/SRW implementation

Others:
- British Library and TEL
- Ex-Libris Metalib
- BibSys
- ...
IndexData:
- Gateway to the Library of Congress bibliographic database
- Open Source (next version of YAZ toolkit)
- Provides Z39.50 to Z39.50 and SRW to Z39.50
- Load balancing, timeouts and bandwidth throttling
- Improved session management
- Protection from 'aggressive' clients
- These features in web servers already
- Provides additional record transformations from MARC

These issues are all important in a MetaSearch environment as well, where many clients will be cross-searching the database.
NISO MetaSearch Initiative:
  - NISO organising 3 task groups:
    - Access Management
    - Collection Description
    - Search and Retrieve
  - Content and gateway providers
  - First face to face meeting follows DLF
  - SRW specifically named as search protocol to investigate
  - All but one requirement met by SRW
  - Not filled: multiple resource access by single request
  - Less of an issue than for Z39.50
What To Take Away With You:

- SRW is an IR web service similar to Z39.50
- Easy enough to implement with off the shelf/OSS tools
- Multiple open source implementations
- Very customizable without losing focus
- Three operations, either SOAP or URL form
- NISO MetaSearch group looking at SRW
- [http://www.loc.gov/srw/](http://www.loc.gov/srw/)
- azaroth@liv.ac.uk