NISO RP-2006-02

NISO Metasearch Initiative

Metasearch XML Gateway Implementers Guide

Version 1.0

A Recommended Practice of the National Information Standards Organization

Standards Committee BC / Task Group 3

August 7, 2006

Published by the National Information Standards Organization
Bethesda, MD

National Information Standards Organization

Contents

Foreword ii
1 Purpose and Audience 1
2 Overview 1
3 Levels of Implementation 2
4 Prerequisites 3
5 Decision Points 3
6 The MXG Protocol 4
6.1 MXG URL Request 4
6.1.1 Syntax 4
6.1.2 MXG Request Parameters 4
6.1.3 Parsed Examples 5
6.1.4 Result Set IDs 6
6.2 MXG XML Response 6
6.2.1 MXG Response Parameters 7
7 Compliance 12
7.1 URL Request Compliance 12
7.2 MXG Response Compliance 13
8 Advanced Interoperability (Levels 2 and 3) 13
8.1 Explain (Level 2 Compliance) 13
8.2 CQL (Level 3 Compliance) 14
Appendix A : Implementation Help 15
Appendix B : Resources 16
Appendix C : Glossary 18

Tables

Table 1: MXG request parameters 5
Table 2: MXG response header subelements 7
Table 3: MXG response result set elements 8
Table 4: MXG response record elements 9
Table 5: MXG response browser elements 9
Table 6: MXG response diagnostic elements 10

Figures

Figure 1: High level model of metasearch environment 1
Figure 2: MXG protocol model 2

Foreword

Metasearch—also called parallel search, federated search, broadcast search, and cross-database search—has become commonplace in the information community's vocabulary. All speak to a common theme of allowing search and retrieval to span multiple databases, sources, platforms, protocols, and vendors at once and integrate the results. Metasearch services rely on a variety of approaches to search and retrieval including open standards (such as NISO's Z39.50), proprietary API's, and screen scraping (extracting information from HTML responses). However, the absence of widely supported standards, best practices, and tools makes the metasearch environment less efficient for the metasearch service provider, the content provider, and ultimately the end-user.

NISO Metasearch Initiative

To move toward industry solutions, NISO sponsored a Metasearch Initiative to enable:

- metasearch service providers to offer more effective and responsive services
- content providers to deliver enhanced content and protect their intellectual property
- libraries to deliver services that distinguish their services from Google and other free web services.

The groundwork for NISO's Metasearch Initiative was laid in two important events:

- A two day strategy meeting in May 2003 defined the metasearch state-of-the-art and built consensus on ways to move forward.
- A Metasearch workshop in October 2003 informed librarians, content providers, and aggregators about metasearch.

Following these meetings, NISO established three Task Groups / Standards Committees to address the different Metasearch needs areas:

- **Access Management** (Standards Committee BA / Task Group 1)
- **Collection and Service Descriptions** (Standards Committee BB / Task Group 2)
- **Search and Retrieval** (Standards Committee BC / Task Group 3)

Search and Retrieval Task Group

The Search and Retrieval Task Group was charged with identifying and/or developing standards and best practices to improve interoperability between metasearch services and content providers. In particular, they were asked to investigate and report on:

- Current practices in metasearching search and retrieval
- Common metasearch vocabulary and terms
- Result set data elements
- XML interfaces
- Best practices for metasearch search and retrieval

This document represents one of the deliverables of the Search and Retrieve Task Group. It describes how to implement a Metasearch XML Gateway (MXG) that will allow a content provider's resource to be accessed by a Metasearch Service and included in the results of a metasearch.

Background on SRU

In the late 1990s, there was significant interest in making the Z39.50 standard more relevant to the Web environment. The result was the development of a new specification: SRU (Search / Retrieve via URL)

The specification retains such Z39.50 concepts as result sets, abstract record schemas, application level diagnostics, and "Explain", but differs from Z39.50 in the use of Web interfaces, XML, and the Common Query Language (CQL).

The Metasearch Initiative Search / Retrieve Task Group selected SRU as the foundation on which to build the NISO Metasearch XML Gateway (MXG). For further information about SRU and CQL, see the Resources Appendix.

Metasearch Search and Retrieve Task Group

Members of the Task Group included:

Katherine Kott, co-chair
Digital Library Federation

Sara Randall, co-chair
Endeavor Information Systems

Amira Aaron
Harvard University Library

Paul Cope
Auto-Graphics, Inc.

Ray Denenberg
Library of Congress

Dana Dietz
OCLC, Inc.

Matthew Dovey
University of Oxford

Susan Fariss
National Library of Medicine

Riccardo Ferrante
Smithsonian Institution Archives

Matthew Goldner
OCLC, Inc.

Cary Gordon
The Cherry Hill Co.

Renny Guida
Thomson Scientific

Sebastian Hammer
Index Data

Mary Jackson
Association of Research Libraries

Marc Krellenstein
Elsevier

Ralph LeVan
OCLC, Inc.

John Little
Duke University

Mike McKenna
California Digital Library

Ron Miller
The H.W. Wilson Company

William Mischo
University of Illinois, Urbana-Champaign

Peter Murray
OhioLink

Peter Noerr
MuseGlobal, Inc.

Audrey Novak
Yale University

Andrew Pace
North Carolina University

Oliver Pesch
EBSCO Information Services

Chris Roberts
Ex Libris, Inc.

Simona Rollinson
Follett Software Co.

Robert Sanderson
University of Liverpool

Ezra Schwartz
ArtandTech.com

Jeff Steinman
Lexis Nexis Academic & Library Solutions

Patricia Stevens, NISO SDC Liaison
Consultant

Roy Tennant
University of California

Jenny Walker
Ex Libris, Inc.

David Yakimischak
formerly with JSTOR

Johan Zeeman
Research Libraries Group

Candy Zemon
Polaris Library Systems

Metasearch XML Gateway Implementers Guide

1 Purpose and Audience

The NISO Metasearch XML Gateway (MXG) is a low-barrier-to-entry method to expose content to metasearch services and more effectively interoperate with metasearching applications. The MXG protocol defines a simple message and response which allows a metasearch service to query a content database and receive a standardized XML response.

The MXG protocol is primarily directed to Content Providers who wish to expose their resources to one or more metasearch providers without expending substantial development resources. Metasearch Providers who will be using a Content Provider's MXG to access resources will also find this guide useful.

2 Overview

The NISO Metasearch XML Gateway (MXG) provides a mechanism for you, as a content provider, to expose your content and services to a Metasearch Service. Metasearch Services are a class of services that allow an end user to find content in multiple services with a single search. (For background information on metasearch, see the Resources Appendix.) Figure 1 provides a high-level model of the Metasearch environment.

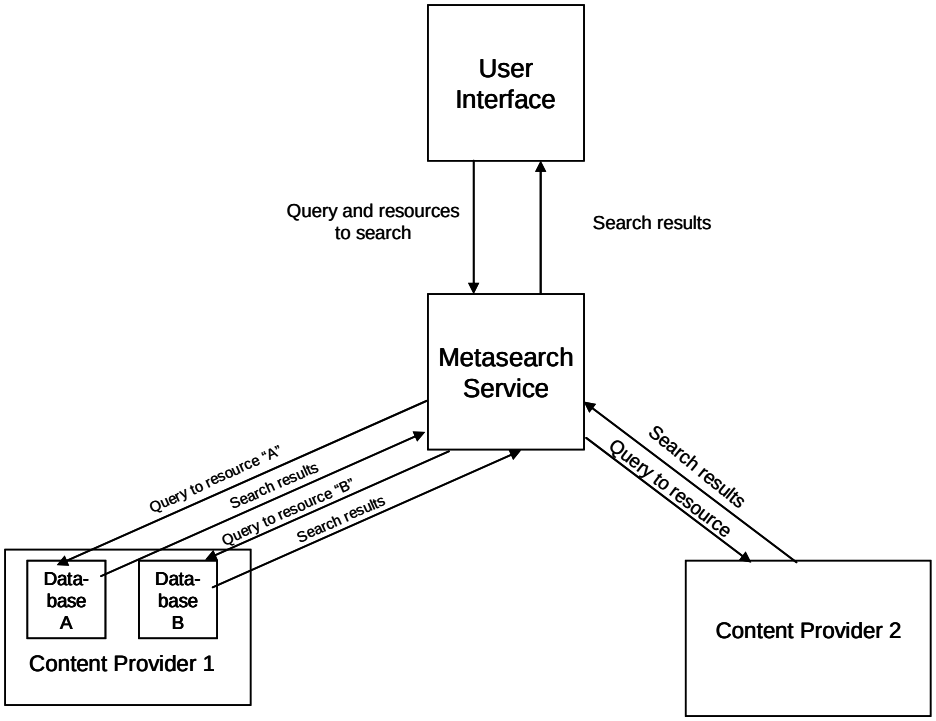


Figure 1: High level model of metasearch environment

The NISO Metasearch XML Gateway (MXG) protocol is a mechanism that can provide the query and the search results between the Metasearch Service and the resource as represented in this high level model. MXG a simple message / response based on the NISO-registered Search and Retrieve URL (SRU)[1] protocol. The Metasearch Provider sends individual queries for each resource that you host using MXG

[1] SRU utilizes the Z39.50 semantics to provide search and retrieval in a Web environment. See the Resources Appendix for more information on SRU

URLs via HTTP (Hypertext Transport Protocol). You, the Content Provider, return an MXG compliant XML formatted response to those queries. The Metasearch Service is responsible for parsing, aggregating, and displaying of the records retrieved from multiple sources—which may include records retrieved by methods other than the MXG gateway—to the end user.

The MXG protocol utilizes a standard client/server model where the Metasearch Service acts as the "client" and your database is the "server". (For basic information about the client/server model, see the Resources Appendix.) Figure 2 illustrates the MXG protocol.

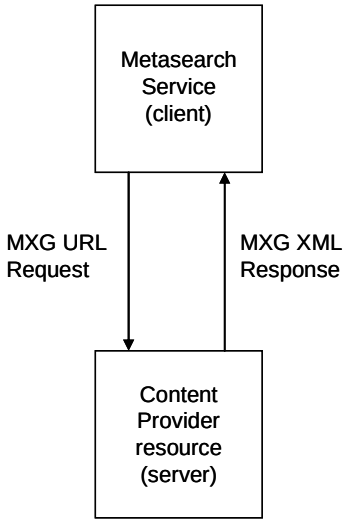


Figure 2: MXG protocol model

The steps involved in an MXG transaction are:

1. The Metasearch Service (MS) transmits an MXG URL request that incorporates the user's search query to a Content Provider's (CP) resource, usually a database.
2. The CP server receives and interprets the MXG URL and conducts the specified search query on its database.
3. The search query results are packaged into an MXG XML response, which is transmitted back to the MS.

3 Levels of Implementation

Three levels of implementation are defined for MXG. Each level requires increasing compliance with specifications of the SRU protocol; only the third level is fully compliant SRU. Full (Level 3) compliance is **strongly** encouraged for all implementers as interoperability and functionality increase with each level of implementation.

- **Level 1 defines a standard URL which will accommodate any query syntax or language.**
  With Level 1 compliance, the Metasearch Service will have to convert its users' queries to the Content Provider's native search language. The amount of customization required depends on how proprietary the search language is.
- **Level 2 extends Level 1 by adding the requirement that servers provide an SRU `Explain` record to define the capabilities of the server.**
  With Level 2 compliance, the Content Provider's server would provide an XML-formatted record that includes information about the resource, such as its host name and port and the database name, which is used as the context part of a URL.

Example:

The base URL for an MXG search against this resource:

```
host=oclc.org
port=80
database=search/WorldCat
```

would be:

```
http://oclc.org/search/WorldCat
```

Optionally, a Level 2 compliant server also includes human-readable information about the database(s) on the server such as its name, a database description, information about the indexes available for searching the database, and the schemas that can be used to display returned records. This server / database description utilizes the SRU `Explain` operation. See the Resources Appendix for background information on SRU `Explain`.

Section 8.1 in Advanced Interoperability further defines the use of the `Explain` function in Level 2 of MXG.

- **Level 3 extends Level 2 by adding the requirement that servers support a standard query grammar: CQL (Common Query Language).**
  "CQL tries to combine simplicity and intuitiveness of expression for simple, every day queries, with the richness of more expressive languages to accommodate complex concepts when necessary."[2] Support for CQL eliminates the need for any customized search interface, which could make the content more widely available to services that don't have the resources to write custom interfaces. See the Resources Appendix for background information on CQL.

  Section 8.2 in Advanced Interoperability further defines the use of CQL in Level 3 of the MXG.

4 Prerequisites

There are very few prerequisites for implementing MXG. Obviously you will need to have an electronic content resource that is Internet accessible. This resource must have:

- a Web-addressable Uniform Resource Identifier (URI),
- a server that can receive and parse a URL request into its components parts: the base URL and parameter names and values,
- an existing search interface, and
- the ability to output search results in XML.

If you do not currently output search results in XML format but do have some type of API to a web interface for displaying records, then implementing the XML MXG response should be fairly easy. For further information on XML, see the Resources Appendix.

5 Decision Points

Prior to implementing the MXG, you will need to make two decisions:

1. **What level of compliance will you implement?**
   See the Levels of Implementation section above for the description of each compliance level. While Level 1 requires the least amount of standardized protocol use and is the easiest for the Content Provider to implement, it requires the Metasearch Provider to have or create a custom interface to your search language. Any changes to your search language could necessitate coordinated changes by the Metasearch Provider to ensure that your content continues to be retrieved accurately. The

[2] Common Query Language, CQL Version 1.1 13th February 2004. <http://www.loc.gov/standards/sru/cql/index.html>

need for this customization may limit the number of metasearch providers that access your content.

If implementing the gateway at Level 1, a good practice would be to provide technical information on your website about your search language and API or, at a minimum, identify a contact person for Metasearch Providers who are interested in accessing your content.

Levels 2 and 3 assume that you already have in place additional features of the SRU specification. Level 2 requires you to have implemented the SRU `Explain` operation and Level 3 requires support for the Common Query Language (CQL). You will need to add this functionality, if you aren't currently using `Explain` or CQL, to support Levels 2 and 3 of MXG. By implementing these higher levels, you will make your resources more easily accessible to a greater number and variety of Metasearch Providers. Additionally, you can make changes to your own search language and update its "translation" to CQL without involving any Metasearch Providers. You won't have to coordinate your software change schedules with the Metasearch Providers and will have greater control over the accurate translation of search queries.

2. **What XML schema will you utilize for records that are returned?**
   A minimum of one XML Schema is required, although multiple schemas may be supported for different Metasearch Providers or different communities of users. Any schema is allowable, even a custom created one as long as it uses standardized XML mark-up and is used consistently.

   A standard schema, such as *Dublin Core*, is one possible choice, although some content may require a more sophisticated schema. Your choice of schema should be based on the content's attributes and the user community. For example, highly bibliographic content may find the *MODS (Metadata Object Description Standard)* schema useful, while an e-learning community may like the *LOM (Learning Object Metadata)* schema. For further information on XML schemas, see the Resources Appendix.

6 The MXG Protocol

The MXG protocol consists of a **Request** made by the Metasearch Service (MS) to the Content Provider's (CP) resource and a **Response** from the CP to the MS. The **Request** is in the form of a simple SRU URL and the **Response** is an instance of an SRU/SRW `searchRetrieveResponse`. For background information on SRU/SRW, see the Resources Appendix.

6.1 *MXG URL Request*

6.1.1 Syntax

The MXG URL consists of an SRU base URL and a search part, separated by a question mark, in the form:

```
http://host/context?<version>&query=<query>&<optionalParms>
```

where "host" is replaced with the host domain name and optional port number of the Content Provider's server and "context" is replaced with the path and name of the resource database. The items following the question mark are MXG URL Request parameters, which are of the form `name=value` and are separated by an ampersand (`&`). Section 6.1.2 describes the possible parameters.

For example,

```
http://alcme.oclc.org/search/ORPubs?version=1.1&query=book
```

is a compliant URL where the host is `alcme.oclc.org` at the default port 80, the path and database name is `search/ORPubs`, and the request has two parameters: `version` and `query`.

6.1.2 MXG Request Parameters

The MXG request contains two mandatory parameters, <version> and <query> and two optional parameters: `startRecord` and `maximumRecords`. Table 1 describes these parameters.

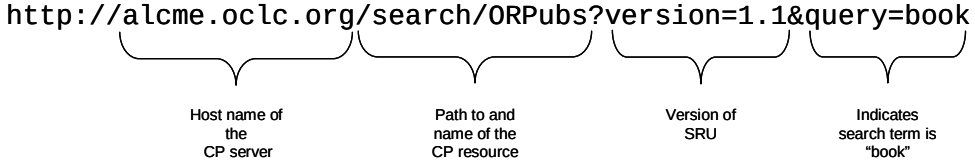
Table 1: MXG request parameters

Element	Value	Requirement	Note
<*version*>	1.1	mandatory	Specifies the version of SRU being utilized, which is currently version 1.1.
<*query*>	a string within double-quotes OR a single word/token (without blanks)	mandatory	Double-quotes utilized as part of the search need to be escaped with a preceding backslash.
Optional Parameters			
startRecord	a positive integer between 1 and the maximum number of records retrieved Recommended default = 1	optional	This value, if used, is obtained from the original search query where the searcher may specify which record from the result set should be the first record returned. The value `1` specifies the first record. If omitted, the server may choose any value, but the recommended default value is `1`.
maximumRecords	a positive integer.	optional	This value, if used, may be obtained from the original search query where the searcher may specify the maximum number of records to be returned. Or the Metasearch Service may choose to provide a default value to limit the number of records retrieved. The server may return fewer records if the search results include fewer records than the maximum specified. If omitted, the server may choose any value, including zero.

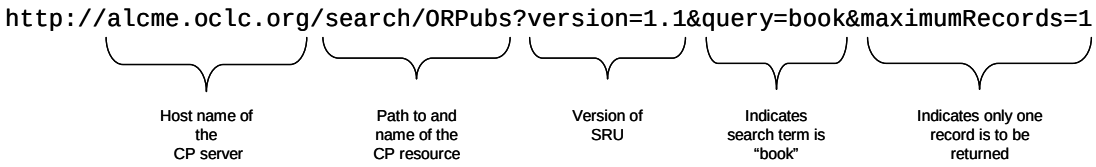
6.1.3 Parsed Examples

The following are examples of MXG URL Requests that illustrate the use of the various parameters. Each example is parsed to identify the meaning of the different parameters within the URL request.

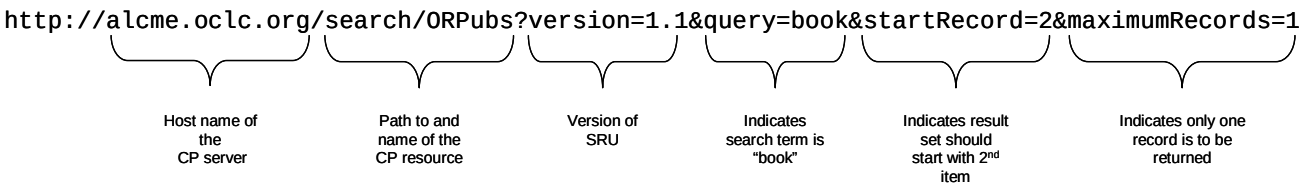
- Example 1: Send a single term search. Do not specify how many records should be returned.



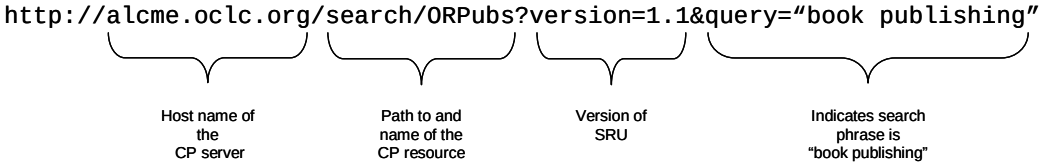
- Example 2: Ask for a single record to be returned:



- Example 3: Ask for the next record in the result set to be retrieved:



- Example 4: Send a more complex query, a search phrase:



6.1.4 Result Set IDs

The CP server may optionally provide a `resultSetId` in its response. This is a name that has been provided so that a result set may be referenced after the query response. The MS client uses this Id to keep track of the search without rerunning the query each time. `resultSetId`s are typically used to retrieve more records from an existing result set. The MS client does this by submitting a new URL with the `resultSetId` in the query.

The syntax for this query is:

```
query=cql.resultSetId=<resultSetId>
```

where `<resultSetId>` is replaced with the value of the `resultSetId` element provided in the MXG XML Response. (See section 6.2.1.2 for a description of providing the `resultSetId` in a Response record.)

An example of a URL with a result set reference that asks for the second record of the result set is:

```
http://alcme.oclc.org/search/ORPubs?version=1.1&query=cql.resultSetId=
abc123&startRecord=2&maximumRecords=1
```

The alternative to using a `resultSetId` is to reissue the original query with a new `startRecord` parameter. While this is simpler for the client than tracking a `resultSetId`, it may have unexpected consequences if the database has been updated between the searches. MS clients are encouraged to use a `resultSetId` if it is provided.

6.2 *MXG XML Response*

The MXG XML record returned by the CP server is an instance of an SRU `searchRetrieveResponse`.

`SRW Types` is the schema that defines the SRU/SRW `searchRetrieveResponse`. If you are familiar and comfortable with tools that automatically generate XML from schemas, this schema is available online. (http://www.loc.gov/standards/sru/xml-files/). There is also a Web Service Description Language (WSDL) definition of the SOAP version of SRU (http://www.loc.gov/standards/sru/xml-files/srw-bindings.wsdl). The discussion in this section of the guide assumes that you are hand-constructing the XML response, rather than using the schema.

6.2.1 MXG Response Parameters

An MXG XML record is made up of some mandatory general header elements (section 6.2.1.1) followed by the relevant elements related to the entire result set (section 6.2.1.2) and relevant items about the records in the set (section 6.2.1.3). Optionally, the XML record may also contain elements to support a "context-free" browser (section 6.2.1.4) and elements to support diagnostic messages (section 6.2.1.5).

Many of the elements in the `searchRetrieveResponse` are optional for an MXG-compliant response. However, Metasearch Services should be prepared to handle responses which include these optional elements, either by italicizing the optional element parameters or by ignoring them. In either case, the search should not fail because optional elements were included in the response.

6.2.1.1 Header Elements

The header information in the MXG Response takes the form:

```
<?xml version="1.0" ?>
<searchRetrieveResponse xmlns="http://www.loc.gov/zing/srw/">
    <version>1.1</version>
    <numberOfRecords>30</numberOfRecords>
```

The first line of the response is a declaration that this is an XML record; it is mandatory.

The second line is the actual `searchRetrieveResponse`. It includes a namespace[3] attribute that specifies the default namespace for this element and all subelements.

The first subelement indicates the version of SRU. It is followed by the `numberOfRecords` subelement. Table 2 describes these subelement parameters.

Table 2: MXG response header subelements

Element	Value	Requirement	Note
<version>	1.1	mandatory	Specifies the version of SRU being utilized, which is currently version 1.1.
<numberOfRecords>	a non-negative integer	mandatory	Specifies the count of the number or records that satisfies the query. If the query fails this will be 0.

6.2.1.2 Result Set Elements

The next group of elements in the MXG XML Response describes the result set and takes the form:

```
<resultSetId>717zar</resultSetId>
<resultSetIdleTime>30</resultSetIdleTime>
```

If the CP server generates result sets that can be referenced after the query is complete, then this is where it will specify the identifier for the result set and indicate how long the result set will remain available. Table 3 describes the parameters of these elements.

[3] A namespace provides context for identifiers. The same identifier can have different meanings in different namespaces. XML namespaces are defined in the W3C Recommendation, *Namespaces in XML 1.1*, available from: http://www.w3.org/TR/xml-names11/.

Table 3: MXG response result set elements

Element	Value	Requirement	Note
<resultSetId>	a string	optional	An identifier for the result set. It is created at the time of execution of the query. It can contain anything that is valid in XML content. (Avoid angle brackets, quotes, apostrophes, and ampersands.)
<resultSetIdleTime>	a positive integer	optional	The number of seconds from last use after which the created result set will be deleted. If omitted, it means that the server is not making any promises about how long the result set will be available.

Section 6.1.4 describes how the `resultSetId` is used in the MXG URL Request.

The `<resultSetIdleTime>` is essentially a countdown timer that is started each time the result set is used. When it reaches zero, the result set can be thrown away by the CP server. If the MS client wants to prevent the `resultSetId` from expiring, it can send a request with `maximumRecords=0`, which will restart the timer. A result set idle time is not a guarantee; it is a promise of best effort. The server is always permitted, as necessary, to throw result sets away arbitrarily. If a result set that no longer exists is later referenced, then the CP server should issue a diagnostic. (See section 6.2.1.5 for more information on diagnostics.)

6.2.1.3 Record Elements

The next group of elements in the MXG XML Response describes the records and takes the form:

```
<records>
    <record>
        <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
        <recordPacking>xml</recordPacking>
        <recordData>
            <srw_dc:dc xmlns="http://www.w3.org/TR/xhtml1/strict"
                xmlns:dc="http://purl.org/dc/elements/1.1/"
                xmlns:srw_dc="info:srw/schema/1/dc-v1.1">
                <dc:identifier>rrl1234</dc:identifier>
                <dc:title>Dog and Cat</dc:title>
                </srw_dc:dc>
            </recordData>
        <recordPosition>1</recordPosition>
        </record>
    </records>
```

Table 4 describes the parameters of these elements.

Table 4: MXG response record elements

Element	Value	Requirement	Note
`<records>`	N/A	mandatory	A wrapper element for the collection of individual <record> elements.
`<record>`	N/A	mandatory	A wrapper for each individual record. It may repeat any number of times.
`<recordSchema>`	a string	mandatory	The identifier of the XML schema used for encoding the record.[4]
`<recordPacking>`	xml	mandatory	Specifies how the content in the `<recordData>` element is structured. For MXG, the only valid value is `xml`.[5]
`<recordData>`	a string (encoded according to the specified schema)	mandatory	Contains the actual record's data.
`<recordPosition>`	a positive integer	optional	Indicates the position of the record in the current result set. For example, if the query uses startRecord=10 and maximumRecords=5, the <recordPosition> results would be numbered 10-15.

6.2.1.4 Browser Elements

"Context-free" browser clients are "stateless", meaning that there is no memory of previous messages between the client and server. The optional `<echoedSearchRetrieveRequest>` element is provided to support such context-free, browser-based clients and its presence is strongly recommended. This element set takes the form:

```
<echoedSearchRetrieveRequest>
    <version>1.1</version>
    <query>cql.any = &quot;dog&quot;</query>
    </echoedSearchRetrieveRequest>
```

Table 5 describes the parameters of these elements.

Table 5: MXG response browser elements

Element	Value	Requirement	Note
<echoedSearch RetrieveRequest>	N/A	optional	A wrapper element that indicates the request parameter should be echoed back to the client in a simple XML form.

[4] There is no restriction on the schema used. If using a standard schema, it should have its own specified ID. If using a customized schema, you will have to create your own ID.

[5] For more information on the <recordPacking> element, see http://www.loc.gov/standards/sru/sru-spec.html#packing.

Element	Value	Requirement	Note
<version>	1.1	mandatory	Should be identical to the <version> value provided in the MXG request.[6]
<query>	a string	mandatory	Should be identical to the <query> value provided in the MXG request.
<startRecord>	a positive integer	optional	If the corresponding parameter was included in the MXG request, then it is mandatory to include it here.
<maximumRecords>	a positive integer	optional	If the corresponding parameter was included in the MXG request, then it is mandatory to include it here.

6.2.1.5 Diagnostic Elements

By using diagnostics, if a search fails, the server can explain to the MS (and ultimately the searcher) why the search failed. Without this information, it is possible that a user could be told incorrectly that there were no matching records when in fact a system error occurred.

A response may include any number of diagnostic messages. The presence of diagnostics does not necessarily indicate a failure of the request; the diagnostic message may be purely for information purposes. This element set takes the form:

```
<diagnostics>
    <diagnostic xmlns="http://www.loc.gov/zing/srw/diagnostic/">
        <uri>info:srw/diagnostic/1/51</uri>
        <details>66ntqk</details>
        </diagnostic>
    </diagnostics>
```

Table 6 describes the parameters of these elements. A detailed list of available SRU diagnostics is available at: http://www.loc.gov/standards/sru/diagnostics-list.html.

Table 6: MXG response diagnostic elements

Element	Value	Requirement	Note
<diagnostics>	N/A	optional	A wrapper element for the collection of individual <diagnostic> elements.
<diagnostic>	N/A	mandatory	A wrapper for each individual diagnostic. It may repeat any number of times.
<uri>	URI string	mandatory	Contains the unique identifier for the diagnostic. URIs that begin with the string info:srw/diagnostic/1/ are from the standard SRU diagnostic list.[7]

[6] The values from the original request may be modified by the CP server if necessary to support XML rules which are different from the URL, e.g. translating special characters used in the URL into XML-supported characters. In the example at the beginning of this section, the quotes around the search term “dog” were replaced with the " XML entity.

[7] The CP server can optionally define its own diagnostics, but it is strongly recommended that the standard list of SRU diagnostics be used. Requests for additions to the current SRU list can be made through the SRU public listserv (http://www.loc.gov/standards/sru/listserv.html).

Element	Value	Requirement	Note
<details>	a string	optional	Contains extra information associated with the diagnostic. In the example above, the <uri> element indicates the diagnostic for "resultset does not exist" and the <details> element identifies the relevant resultSetId.
<message>	a string	optional	Contains a human readable message. This is a limited capability as the message element can only occur once, so the message can only be in a single language. It is believed that the diagnostic URI and accompanying details should be sufficient to formulate language appropriate messages.

6.2.1.6 Parsed Example of an MXG XML Response

The following is a complete XML MXG response example. It is parsed to indicate the different parts of the XML response record that correspond to the sections above. This example includes optional result set information and the record data is returned as a Dublin Core[8] record.

```
<?xml version="1.0" ?>
<searchRetrieveResponse xmlns="http://www.loc.gov/zing/srw/">
   <version>1.1</version>
   <numberOfRecords>30</numberOfRecords>
   <resultSetId>717zar</resultSetId>
   <resultSetIdleTime>30</resultSetIdleTime>
   <records>
      <record>
         <recordSchema>info:srw/schema/1/dc-v1.1</recordSchema>
         <recordPacking>xml</recordPacking>
         <recordData>
            <srw_dc:dc xmlns="http://www.w3.org/TR/xhtml1/strict"
               xmlns:dc="http://purl.org/dc/elements/1.1/"
               xmlns:srw_dc="info:srw/schema/1/dc-v1.1">
               <dc:identifier>rrl1234</dc:identifier>
               <dc:title>Dog and Cat</dc:title>
               </srw_dc:dc>
            </recordData>
         <recordPosition>1</recordPosition>
         </record>
      </records>
   <nextRecordPosition>1</nextRecordPosition>
   <echoedSearchRetrieveRequest>
      <version>1.1</version>
      <query>cql.any = &amp;quot;dog&amp;quot;</query>
      </echoedSearchRetrieveRequest>
<diagnostics>
      <diagnostic xmlns="http://www.loc.gov/zing/srw/diagnostic/">
         <uri>info:srw/diagnostic/1/51</uri>
         <details>66ntqk</details>
         </diagnostic>
      </diagnostics>
</searchRetrieveResponse>
```

Header

Result Set elements

Record elements

Browser elements

Diagnostic elements

7 Compliance

This section summarizes the requirements for Level 1 MXG compliance. Information on Level 2 and 3 implementations is found in Section 8.

7.1 *URL Request Compliance*

A compliant Level 1 MXG server must:

1. Require the `version` parameter with a value of `1.1`.
2. Accept the `query` parameter in the form:

   `query=<word>`

   or

   `query="<phrase>"`

[8] For more on Dublin Core, visit the website: http://www.dublincore.org/.

where <word> is any string of characters with no embedded blanks (‘ ‘) and a <phrase> is any string of characters except the unescaped quote (‘”’) and backslash (‘\’). Quotes and backslashes can be escaped in a <phrase> by preceding them with a backslash (‘\’)

An example of a compliant <word> query is:

```
query=book
```

An example of a compliant <phrase> query is:

```
query=”find cat w/1 house”
```

NOTE: This "cat w/1 house" query is only valid when the default CQL index has been defined as mxg.notCQL. This provides the mechanism that allows non-CQL queries to be passed in a context that normally expects a CQL query. `mxg.notCQL` is the default index for Level 1 implementations of MXG. Level 2 and 3 implementations are expected to state explicitly what their default index is.

3. Accept `startRecord,` and `maximumRecords` parameters.
   While it is optional for the MS client to include the `startRecord` and `maximumRecords` parameters, if this information is included in the request, a compliant MXG server must accept and respond to them. The server may ignore any other parameters in the request, but it is strongly recommended that diagnostic messages be issued for each ignored parameter.

7.2 *MXG Response Compliance*

A compliant Level 1 MXG server must:

1. Provide a well-formed XML response whose root element is:
   `<searchRetrieveResponse>`.
2. Have a valid XML record according to the schema at
   http://www.loc.gov/standards/sru/xml-files/srw-types.xsd

8 Advanced Interoperability (Levels 2 and 3)

The MXG protocol described in section 6 defines the requirements for Level 1, which is a non-conformant subset of the SRU specification. The features missing from MXG that are necessary for SRU conformance are support for an `Explain` record (Level 2 MXG) and rich CQL support (Level 3 MXG), which are defined in sections 8.1 and 8.2, respectively.

8.1 *Explain (Level 2 Compliance)*

In Level 1 MXG, the MS has no automated method to learn about the capabilities of the CP’s resource. By utilizing the SRU `Explain` record (required for Level 2 compliance), a CP can provide information about its capabilities through an automated request on demand. The `Explain` record contains a list of the indexes that can be used in a search and a list of the record schemas that can be used when requesting database records. It also provides a human readable description of the database and contact information and can describe default behaviors.

The `Explain` operation consists of: 1) a request from the client that contains the `operation=explain` parameter; and 2) a response from the server that includes the `Explain` record describing the server's capabilities.

Specific information on the syntax and parameters of an `Explain` request or response is available at: http://www.loc.gov/standards/sru/explain/.

8.2 *CQL (Level 3 Compliance)*

Common Query Language (CQL), the requirement for Level 3 compliance, is a standardized formal language for representing queries to information retrieval systems. "CQL tries to combine simplicity and intuitiveness of expression for simple, every day queries, with the richness of more expressive languages to accommodate complex concepts when necessary."[9] Use of CQL, a feature of SRU/SRW, supports high functionality, highly interoperable searches.

Level 1 MXG does not define a standard query grammar. At this level, the Metasearch Service is responsible for converting its query grammar into the query grammar of the content provider's resource. From a CQL perspective, this is considered CQL Level 0, a trivial subset of CQL has been specified that will support passing non-CQL queries in the CQL context.

To be Level 3 MXG compliant, the CP server would have to accept standard CQL query syntax in an MXG URL request. Supporting a standard query grammar reduces the complexity and customization required for the MS client to interoperate with the CP server and allows a greater diversity of metasearch services to access the server's content. It is strongly recommended that Content Providers support CQL queries.

MXG Level 3 requires CQL Level 1 compliance, defined as:

1. Support for CQL Level 0
2. Ability to parse both:
    a) search clauses consisting of "index relation searchTerm"; and
    b) queries where search terms are combined with Boolean logic, e.g. "term1 AND term2"
3. Support for at least one of (a) and (b)

Although CQL is not required for Level 1 MXG, all of the examples in section 6 use standard CQL queries. More information about CQL can be found at: http://www.loc.gov/standards/sru/cql/.

[9] Common Query Language, CQL Version 1.1 13th February 2004. <http://www.loc.gov/standards/sru/cql/>

Appendix A: Implementation Help

A.1. NISO MXG Email ListServ

NISO maintains an MXG email list where questions can be asked and implementation experiences can be shared. To subscribe, send an email message to: MXG-subscribe@list.niso.org. Use the word "subscribe" in the subject line of the message. You will automatically receive a confirmation message. You must respond to the confirmation message in order to successfully subscribe. Once subscribed, you will receive an automated "Welcome!" message.

A.2. SRU Community

There is an active SRW implementer community. They have a mailing list where questions can be asked. At the SRU web site (http://www.loc.gov/standards/sru/) there are pointers to both Open Source and commercial implementations of clients and servers, as well as complete documentation on the standard and the mailing list.

The SRU website currently provides access to a test server at: http://alcme.oclc.org/srw/SRUServerTester.html. This server requires an `Explain` record and is therefore suitable only for Level 2 and 3 implementations.

Appendix B: Resources

Metasearching Basics

Elliott, Susan A. *Metasearch and Usability: Toward a Seamless Interface to Library Resources*. Anchorage, AK: University of Alaska, August 2004. http://www.lib.uaa.alaska.edu/tundra/msuse1.pdf

Hane, Paula J. *The Truth About Federated Searching*. Information Today, October 2003. http://www.infotoday.com/it/oct03/hane1.shtml

Sadeh, T. *The Challenge of Metasearching.* New Library World, v. 105, no. 1198/1199, p. 104-112, 2004. http://www.exlibrisgroup.com/resources/metalib/The_Challenge_of_Metasearching.pdf

Sadeh, Tamar. *Google Scholar Versus Metasearch Systems*. High Energy Physics Libraries Webzine, 11/01/2006. http://library.cern.ch/HEPLW/12/papers/1/

Tennant, Roy. *The Right Solution: Federated Search Tools*. Library Journal, June 15, 2003. http://www.libraryjournal.com/article/CA302427.html

UC Berkeley, Teaching Library Internet Workshops. *Meta-Search Engines*. UC Berkeley Library, c2005. http://www.lib.berkeley.edu/TeachingLib/Guides/Internet/MetaSearch.html

Client/Server Basics

comp.client-server newsgroup. *Client/Server Frequently Asked Questions*. Aug 17, 1998. http://www.faqs.org/faqs/client-server-faq/

Sadoski, Darleen. *Client/Server Software Architectures--An Overview*. Carnegie Mellon University, Software Engineering Institute, August 2, 1997. http://www.sei.cmu.edu/str/descriptions/clientserver_body.html

The Client Server Architecture. Web Developers Notes. http://www.webdevelopersnotes.com/basics/client_server_architecture.php3

SRU and CQL

Common Query Language. SRU Editorial Board, February 2004. http://www.loc.gov/standards/sru/cql/

Morgan, Eric Lease. *An Introduction to the Search/Retrieve URL Service (SRU).* Ariadne, Issue 40, July 2004. http://www.ariadne.ac.uk/issue40/morgan/

SRU Search/Retrieve via URL. SRU Editorial Board, February 2004. (http://www.loc.gov/standards/sru/

SRW: Search/Retrieve Web Service. SRU Editorial Board, February 2004. http://www.loc.gov/standards/sru/srw/

XML Basics

Extensible Markup Language (XML). W3C. http://www.w3.org/XML/

W3Schools. *XML Tutorial*. Refsnes Data. http://www.w3schools.com/xml/default.asp

XMLFiles.com. *XML Basics – An Introduction to XML*. Jupitermedia Corporation. http://www.xmlfiles.com/xml/

XML Schemas

IEEE Learning Technology Standards Committee. IEEE Standard for Learning Technology—Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata. IEEE Standard No:1484.12.3-2005.

Powell, Andy and Johnston, Pete. Guidelines for implementing Dublin Core in XML. Dublin Core Management Initiative, 04-02-2003. http://dublincore.org/documents/dc-xml-guidelines/index.shtml

Metadata Object Description Schema (MODS). Library of Congress. http://www.loc.gov/standards/mods/

SRU searchRetrieveResponseType Schema (srw-types.xsd). http://www.loc.gov/standards/sru/xml-files/srw-types.xsd

W3C XML Schema WG. *XML Schema*. http://www.w3.org/XML/Schema

Appendix C: Glossary

Term	Definition
client	In a client/server computing architecture, the computer that makes requests to the "server" and then processes the resulting response information. See also *client/server* and *server*.
client/server	A computing architecture that separates application functionality between two computers on a network; a client makes a request to a server, which performs the necessary services to process the request and returns the requested information to the client; the client may then do further post-processing of the data received from the server. For example, in a Web application, the Web browser on the end user's computer acts as the "client" and interacts with various servers for different purposes; This is the most typical kind of client/server application where the end user is directly involved, however, in many machine-to-machine interactions where no end user is present the "client" may also be a server, i.e. in some interactions, it takes the client role and in other interactions it takes the "server" role. In the metasearch environment as illustrated in Figure 1, the end user is a client to the Metasearch Service (MS) server; the MS then acts as a client to the Content Provider's resource server. See also *client* and *server*.
CQL (Common Query Language)	A standard query syntax for representing queries.
HTTP (Hypertext Transport Protocol)	A request/response protocol used on the Internet for transferring information between clients and servers.
metasearch	Search and retrieval that spans multiple databases, sources, platforms, and protocols and presents aggregated results to the searcher.
server	In a client/server computing architecture, the computer that responds to requests from the "client" by performing a requested task or function to provide the requested service. See also *client* and *client/server*.
SRU (Search/Retrieve via URL)	A standard search protocol for Internet search queries, utilizing a URL request that contains CQL (Common Query Language).
SRW (Search/Retrieve Web Service)	A web service for search and retrieval that utilizes CQL (Common Query Language).
URI (Uniform Resource Identifier)	A formatted string that serves as an identifier for a resource, usually on the Internet.
URL (Uniform Resource Locator)	The address of a resource available on the Internet. A URL is a specific type of URI.

Term	Definition
XML (eXtensible Markup Language)	A standardized markup language developed by the World Wide Web Consortium that describes a class of data objects and the behavior of computer programs which process them.
Z39.50	A client/server protocol for information retrieval from remote computers. The name refers to the standard that defines it: ANSI/NISO Z39.50.