ALEPH SRU/SRW Server

Version 18.01 and Later

CONFIDENTIAL INFORMATION

DISCLAIMER

TRADEMARKS

Document released: August 2009

Web address: http://www.exlibrisgroup.com

Table of Contents

1 General

1.1 SRU Server

SRU (Search/Retrieve via URL http://www.loc.gov/standards/sru/) is a standard search protocol for Internet search queries, utilizing CQL (Common Query Language), a standard query syntax for representing queries.

An SRU request is a HTTP URL. It consists of a base URL and a search part, separated by a question mark. The search part consists of parameters separated by an ampersand, each with structure "key=value". For example, consider the SRU request:

```
http://localhost:5661/usm01?version=1.1&operation=searchRetrieve&query=dinosaur
```

The base URL is `http://localhost:5661/usm01` and the search part is `version=1.1& operation=searchRetrieve&query=dinosaur`.

The response to an SRU request is an XML document, for example:

```
<record>
  <recordSchema>
     info:srw/schema/1/dc-v1.1
  </recordSchema>
  <recordPacking>xml</recordPacking>
  <recordData>
       <srw_dc:dc>
             <dc:title>This is a Sample Record</dc:title>
       </srw_dc:dc>
  </recordData>
  <recordPosition>1</recordPosition>
  <extraRecordData>
     <rel:rank>0.965</rel:rank>
  </extraRecordData>
</record>
```

1.2 SRW Server

SRW is a variation of SRU. Messages are conveyed from the client to the server, not by a URL, but instead using XML over HTTP via the W3C recommendation SOAP, which specifies how to wrap an XML message within an XML envelope. The SRW specification tries to adhere to the Web Services Interoperability recommendations. The incremental benefits of SRW are better extension support, authentication, and web service features.

A typical SRW request might be:

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <SRW:searchRetrieveRequest xmlns:SRW="http://www.loc.gov/zing/srw/">
      <SRW:version>1.1</SRW:version>
      <SRW:query>(dc.author exact "jones" and dc.title >="smith")</SRW:query>
      <SRW:startRecord>1</SRW:startRecord>
      <SRW:maximumRecords>10</SRW:maximumRecords>
      <SRW:recordSchema>info:srw/schema/1/mods-v3.0</SRW:recordsSchema>
    </SRW:searchRetrieveRequest>
  </SOAP:Body>
</SOAP:Envelope>
```

A response would be in the form:

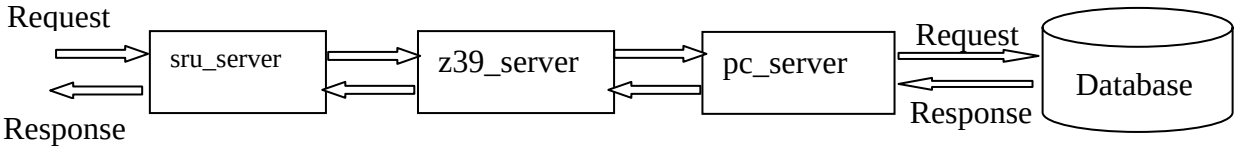
```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP:Body>
    <SRW:searchRetrieveResponse xmlns:SRW="http://www.loc.gov/zing/srw/"
         xmlns:DIAG="http://www.loc.gov/zing/srw/diagnostics/">
      <SRW:version>1.1</SRW:version>
      <SRW:numberOfRecords>2</SRW:numberOfRecords>
      <SRW:resultSetId>8c527d60-c3b4-4cec-a1de-1ff80a5932df</SRW:resultSetId>
      <SRW:resultSetIdleTime>600</SRW:resultSetIdleTime>
      <SRW:records>
        <SRW:record>
          <SRW:recordSchema>info:srw/schema/1/mods-v3.0</SRW:recordSchema>
          <SRW:recordPacking>string</SRW:recordPacking>
          <SRW:recordData>
            &lt;?xml version=&quot;1.0&quot; encoding=&quot;UTF-8&quot;?&gt;
            &lt;mods xmlns:xlink=&quot;http://www.w3.org/TR/xlink&quot;
               xmlns:xsi=&quot;http://www.w3.org/2001/XMLSchema-instance&quot;
               ...
          </SRW:recordData>
          <SRW:recordPosition>1</SRW:recordPosition>
        </SRW:record>
      </SRW:records>
      <SRW:diagnostics>
            <DIAG:diagnostic>
                  <DIAG:uri>info:srw/diagnostic/1/59</DIAG:uri>
                  <DIAG:message>Result set created with valid
                   partial results available</DIAG:message>
             </DIAG:diagnostic>
      </SRW:diagnostics>
    </SRW:searchRetrieveResponse>
  </SOAP:Body>
</SOAP:Envelope>
```

2 Architecture

In ALEPH, a single server implements both the SRU and the SRW servers. The server is named `sru_sever`, and is managed, as with all other ALEPH servers, using the Server and Daemon Management utility (Util W). As with all other ALEPH servers, the `sru_server` creates log files in `$LOGDIR`.

Note : Util W cannot be used to monitor the `sru_server`.

The SRU\SRW server acts as a translation layer, passing the translated SRU requests on to the Z39.50 server. The server architecture may be demonstrated by the following diagram.



ALEPH's `sru_server` supports the following SRU\SRW services:

- Explain Request
- SearchRetrieve

Note:
As is evident from this diagram, the SRU\SRW server requires a license for an additional Z39.50 user.

3 Supported Actions

3.1 Explain Request

3.1.1 Request Parameters

The SRU Explain Request has the following format:

```
http://<host>:<port>/<base>?<request parameters>
```

- `<host>` is the `sru_server` host
- `<port>` is the `sru_server` port
- `<base>` is the ALEPH base code
- `<request parameters>` are parameters as listed in the table below.

For example:

```
http://il-aleph02:9000/usm01?version=1.1&operation=explain
```

The request parameters are specified in the following table:

Parameter Name	Mandatory/ Optional	Support	Description
version	Mandatory	Supported	SRU\SRW protocol version
recordPacking	Optional	Supported	A string to determine how the record should be formatted in the response. Defined values are `string` and `xml`. The default is `xml`

Parameter Name	Mandatory/ Optional	Support	Description
stylesheet	Optional	Supported	A URL for an xml style sheet. the `sru_server` adds to the response a reference to the stylesheet, thus enabling the client application to see the response rendered according to the rules of the supplied style sheet.
extraRequestData	Optional	Not Supported	Provides additional profile specific information.
operation	Mandatory	Supported	The string `explain`

3.1.2 Request Processing

On receiving the Explain Request, the `sru_server` returns the explain response.

3.1.3 Response Parameters

Parameter name	Mandatory /Optional	Supported	Description
version	Mandatory	Supported	SRU/SRW protocol version (1.1)
record	Mandatory	Supported	Single explain record
extraResponseData	Optional	Not Supported	Additional information

3.1.4 Example of Explain Response

```
<?xml version="1.0" ?>
<zs:explainResponse xmlns:zs="http://www.loc.gov/zing/srw/">
 <zs:version>1.1</zs:version>
 <zs:record>
  <zs:recordSchema>http://explain.z3950.org/dtd/2.0/</zs:recordSchema>
  <zs:recordPacking>xml</zs:recordPacking>
  <zs:recordData>
   <explain>
    <serverInfo>
     <host>il-aleph02</host>
      <port>9000</port>
       <database>usm01</database>
    </serverInfo>
   </explain>
  </zs:recordData>
 </zs:record>
</zs:explainResponse>
```

3.2 SearchRetrieve Request

3.2.1 Request Parameters

The SRU SearchRetrieve request has the following format:
`http://<host>:<port>/<base>?<request parameters>`

- `<host>` is the `sru_server` host
- `<port>` is the `sru_server` port
- `<base>` is the Aleph base code
- `<request parameters>` are parameters, as listed in the table below.

For example:
`http://il-aleph02:9000/usm01?version=1.1&operation=searchRetrieve&query=history&maximumRecords=1`

The `<request parameters>` are specified in the following table:

Parameter name	Mandatory /Optional	Supported	Description
version	Mandatory	Supported	SRU/SRW protocol version
query	Mandatory	Supported	Contains a query expressed in CQL to be processed by the server. See section 3.3 *CQL Query* for more information on CQL queries supported by the ALEPH `sru_server`
startRecord	Optional	Supported	The position within the sequence of the matched records of the first record to be returned. The first position in the sequence is 1. The value supplied must be greater than 0. Default value if not supplied is 1.
maximumRecords	Optional	Supported	The number of records requested to be returned. The value must be 0 or greater.
recordPacking	Optional	Supported	A string to determine how the record should be formatted in the response. Defined values are `string` and `xml`. The default is `xml`.
recordSchema	Optional	Supported	The schema requested for the records to be returned. Supported values: `marcxml`
recordXPath	Optional	Not Supported	An XPath expression, to be applied to the records before they are returned.

Parameter name	Mandatory /Optional	Supported	Description
resultSetTTL	Optional	Not Supported	The number of seconds for which the client requests that the created result set should be maintained.
sortKeys	Optional	Not Supported	Contains a sequence of sort keys to be applied to the results.
stylesheet	Optional	Supported	A URL for an xml style sheet. The `sru_server` adds to the response a reference to the stylesheet, thus enabling the client application to see the response rendered according to the rules of the supplied style sheet.
extraRequestData	Optional	Not Supported	Provides additional profile specific information.
operation	Mandatory	Supported	The string `searchRequest`

3.2.2 Request Processing

- The `sru_server` translates the search query from CQL (see section 3.3 *CQL Query*) to Z39.50 query language and sends it to the `z39_server`.
- The `z39_server` processes the request and sends the response to the `sru_server`.
- The `sru_server` packs the response into a SRU\SRW format and returns it to the SRU\SRW client.

3.2.3 Response Parameters

Parameter name	Mandatory\ Optional	Supported	Description
Version	Mandatory	Supported	SRU/SRW protocol version (1.1)
numberOfRecords	Mandatory	Supported	The number of records matched by the query. If the query fails, this is 0.
resultSetId	Optional	Not Supported	The identifier for a result set that was created through the execution of the query.
resultSetIdleTime	Optional	Not Supported	The number of seconds after which the created result set is deleted.

Parameter name	Mandatory\ Optional	Supported	Description
Records	Optional	Supported	A sequence of records matched by the query or the surrogate diagnostics. See * below for details.
nextRecordPosition	Optional	Not Supported	The next position within the result set following the final returned record.
diagnostics	Optional	Supported	A sequence of non surrogate diagnostics generated during execution.
extraResponseData	Optional	Not Supported	Additional information.
echoedSearch RetrieveRequest	Optional	Not Supported	The request parameters echoed back to the client in a simple XML form.

* The ALEPH `sru_server` can be configured to return marcxml records according to:

- http://www.loc.gov/standards/marcxml/schema/MARC21slim.xsd
- DC records according to the Bath profile DTD http://www.collectionscanada.ca/bath/tp-bath2.28-e.htm

3.2.4 Example of SearchRetrieve Response

```
<?xml version="1.0" ?>
zs:searchRetrieveResponse xmlns:zs="http://www.loc.gov/zing/srw/">
zs:version>1.1</zs:version>
zs:numberOfRecords>1000</zs:numberOfRecords>
zs:records>
zs:record>
zs:recordSchema>info:srw/schema/1/marcxml-v1.1</zs:recordSchema>
zs:recordPacking>xml</zs:recordPacking>
zs:recordData>
record xmlns="http://www.loc.gov/MARC21/slim">
leader>00692nam a22002051 4500</leader>
controlfield tag="001">000000002-7</controlfield>
controlfield tag="005">20020418155342.1</controlfield>
controlfield tag="008">700716s1964 mauach b 000 0 eng</controlfield>
datafield tag="010" ind1="" ind2="">
subfield code="a">68001304</subfield>
 </datafield>
datafield tag="035" ind1="0" ind2="">
subfield code="a">(OCoLC)00430285</subfield>
 </datafield>
datafield tag="040" ind1="" ind2="">
subfield code="a">DLC</subfield>
subfield code="c">DLC</subfield>
subfield code="d">m.c.</subfield>
subfield code="d">FLL</subfield>
 </datafield>
```

```
datafield tag="050" ind1="0" ind2="">
subfield code="a">LD2151</subfield>
subfield code="b">.S7 1964</subfield>
 </datafield>
datafield tag="245" ind1="0" ind2="4">
subfield code="a">The Story of Harvard, a short history.</subfield>
 </datafield>
datafield tag="260" ind1="" ind2="">
subfield code="a">Cambridge,</subfield>
subfield code="b">University Information Center</subfield>
subfield code="c">[1964]</subfield>
 </datafield>
datafield tag="300" ind1="" ind2="">
subfield code="a">33 p.</subfield>
subfield code="b">illus., facsim., ports.</subfield>
subfield code="c">17 cm.</subfield>
 </datafield>
datafield tag="504" ind1="" ind2="">
subfield code="a">"Some books of Harvard history": p. 33.</subfield>
 </datafield>
datafield tag="610" ind1="2" ind2="0">
subfield code="a">Harvard University</subfield>
subfield code="x">History.</subfield>
 </datafield>
datafield tag="710" ind1="2" ind2="">
subfield code="a">Harvard University.</subfield>
subfield code="b">Information Center.</subfield>
 </datafield>
datafield tag="956" ind1="" ind2="">
subfield code="a">CONV</subfield>
subfield code="b">00</subfield>
subfield code="c">20040307</subfield>
subfield code="l">USM01</subfield>
subfield code="h">1200</subfield>
 </datafield>
datafield tag="956" ind1="" ind2="">
subfield code="a">CONV</subfield>
subfield code="b">00</subfield>
subfield code="c">20040307</subfield>
subfield code="l">USM01</subfield>
subfield code="h">1402</subfield>
 </datafield>
 </record>
 </zs:recordData>
zs:recordPosition>1</zs:recordPosition>
 </zs:record>
 </zs:records>
 </zs:searchRetrieveResponse>
```

3.3 CQL Query

Common Query Language (CQL) is a formal language for representing queries to information retrieval systems such as web indexes, bibliographic catalogs, and museum collection information. The design objective is that queries should be human readable and writable, and that the language be intuitive while maintaining the expressiveness of more complex languages. A SRU\SRW search statement is expressed in CQL syntax.

A CQL query is made up of a single search clause or multiple search clauses connected by boolean operators. Each search clause consists of the following components:

- Index
- Relation
- Search Term

For example: `title = cat`
In this example, the index is `title`, the relation is = and the search term is `cat`.

It is also possible to include only the search term, for example: `cat`.
In this example the search term is `cat`, and no index or relation is expressed in the query.

The index name may include a prefix, which defines the context set. For example:
`dc.title = cat`
In this example the index is `title`. The index belongs to the context set `dc`, the relation is = and the search term is `cat`.

The same index name (for example, `title`) belonging to different context sets or non belonging to any context set may be treated differently by the SRU\SRW server.

A CQL query may also be made up of multiple search clauses connected by boolean operators, for example: `dc.title = history or dc.title = art`

A CQL query may contain special characters, for example:

- Truncation sign , for example: `title=histo*`
- ^ Anchor sign, for example: `title=^history` – find all titles starting with `history`.

The ALEPH SRU\SRW server conforms to the CQL conformance level 1.

4 Configuration

The `sru_server` configuration is stored in the `alephe_tab/sru_server` directory. It consists of two files: `config.xml` and `pqf.properties`.

4.1 config.xml

The `config.xml` file is the main configuration file of the `sru_server`. It defines search bases, the location of the backend `z39_server`, the format of responses, and so on.

4.1.1 config.xml Structure

The root element of `config.xml` is `<proxy>` element. It contains the following elements:

- "<target> - Mandatory. This element must be defined for every ALEPH base which is exposed through the sru_server. See section 4.1.1.1 <target> Element Structure for an explanation of this element.
- <log>. Optional. This element defines the content of the log file. See part 4.1.1.2 log Element for an explanation of this element"

4.1.1.1 <target> Element Structure

This element has 2 attributes:

- **name** – Unique id of the target
- **database** – Must be equal to the ALEPH base code and to base of the `z39_server`

The element includes the following sub-elements:

- **<url>** – Mandatory, single occurrence
- **<syntax>** – Optional, repeatable
- **<explain>** – Mandatory, single occurrence
- **<cql2rpn>** – Mandatory, single occurrence

Each element is detailed in the following chapters.

For example:

```
<target name="server1" database="usm01">
    <url>il-aleph02:9991</url>
    <syntax type="xml" marcxml="1" identifier = "info:srw/schema/1/marcxml-
    1.1">
          <name>marcxml</name>
    </syntax>
    <explain>
          <serverInfo>
                <host>il-aleph02</host>
                <port>9000</port>
                <database>usm01</database>
          </serverInfo>
    </explain>
    <cql2rpn>pqf.properties</cql2rpn>
</target>
```

4.1.1.1.1 url Element

The element contains the host and port of the `z39_server`.

4.1.1.1.2 syntax Element

Each base exposed by the `sru_server` can be configured to use the DCXML structured responses or the MARCXML structured responses.

- **DC XML** – If the DCXML structured responses are required, then no `<syntax>` element should be defined. In this case the `sru_server` accepts `searchRetrieve` requests without the `recordSchema` parameter and returns the DC XML structured responses. An error is returned if the `recordSchema` is specified in the request.

- **MARCXML –** If MARCXML structured responses are required then the `<syntax>` element should be defined as follows:

```
        <syntax type="xml" marcxml="1"
               identifier="info:srw/schema/1/marcxml-v1.1">
               <name>marcxml</name>
        </syntax>
```

The `sru_server` accepts `searchRetrieve` requests without the `recordSchema` parameter or with `recordSchema` equal to `marcxml`. It returns MARCXML structured responses.
When `marcxml` transformation is defined (`marxcml=1`), `backendcharset` attribute should be defined and set to "UTF-8".

```
<syntax type="xml" marcxml="1"
      identifier="info:srw/schema/1/marcxml-v1.1"
backendcharset="utf-8">
      <name>marcxml</name>
    </syntax>
```

4.1.1.1.3 explain Element

This element contains information that is returned in the explain response. The element must contain a `<serverInfo>` element with the basic information about the server, in the following format:

```
  <serverInfo>
       <host>host</host>
       <port>port</port>
      <database>database</database>
 </serverInfo>
```

- **Host** – `sru_server` host
- **Port** – `sru_server` port
- **Database** – The exposed base code. The database must be equal to the database code in the database attribute of the `<target>` element.

Additional information may be added to the `<explain>` element according to the http://explain.z3950.org/ specification.

4.1.1.1.4 cql2rpn Element

This element contains the name of the file that contains translation of the CQL language used by the SRU\SRW protocol to the RPN language used by the Z39.50

protocol. The file is used when CQL search query received by the `sru_server` is translated to a RPN query and passed to the `z39_server`.

The ALEPH file for CQL to RPN translation is named `pqf.properties` and is stored in the `alephe_tab/sru_server` directory. That is why the content of this element **must** be equal to `pqf.properties`. For a description of the `pqf.properties` file see section 4.2 *pqf.properties*.

4.1.1.2 log Element

This element defines the contents of the log file created by the user. It contains one or more log formats separated by spaces. For example:

```
<log>clients-requests client-apdu server-apdu</log>
```

Possible values are:

- **client-apdu** – Messages received by the `sru_server` are logged.
- **server-apdu** – Messages sent by the `sru_server` to the `z39_server` are logged.
- **clients-requests** – Information about the requests received by the `sru_server` is logged. This is the default value.
- **servers-requests** – Information about the requests sent to the `z39_server` is logged.
- **client-ip** – The IP addresses of clients connecting to the `sru_server` are logged.

4.2 pqf.properties

The `pqf.properties` file, stored in the `alephe/tab/sru_server` directory, defines the translation of CQL queries received by the `sru_server` to Z39.50 RPN queries sent to the z39_server. Lines starting with # are ignored.

The file consists of the following parts:

- Definition of context sets supported by the `sru_server`
- Mapping Z39.50 Attributes to CQL Patterns

4.2.1 Context Sets Definitions

Context sets define indexes, relations, and other CQL expression components. The components of a CQL query may take special meanings, depending on the context set they are related to, as explained in section 3.3 *CQL Query*.

For each context set, a line should be added with the following syntax:

set.<prefix> = <set specification>
For example:
set.cql = info:srw/cql-context-set/1/cql-v1.1
set.dc = info:srw/cql-context-set/1/dc-v1.1
set.bath = http://zing.z3950.org/cql/bath/2.0/

Note: set.cql must be defined; each set prefix used in the translations below must belong to one of the defined context sets.

4.2.2 Mapping Z39.50 Attributes To CQL Patterns

A Z39.50 query consists of a term and optional attributes. There are 6 attributes types:

Attribute Name	Attribute Code
Use	1
Relation	2
Position	3
Structure	4
Truncation	5
Completeness	6

Each component of the CQL query (such as index and relation) is translated to one or more Z39.50 attribute.

The translation line has the following syntax:

```
CQL pattern = Z39.50 Attribute Pair
```

The following CQL patterns can be translated:

- Index
- Relation
- Position of anchor sign (^) in the term
- Position of wildcard character(*) in the term

Each attribute pair has form `type = value`

- Type is one of 6 Z39.50 attribute types: 1, 2, 3, 4, 5 or 6
- Value is attribute value

If the Z39.50 attribute pair is not defined the CQL pattern (for example, CQL index) is recognized, but no Z39.50 attributes are added to the Z39.50 query.
For example:

```
index.rec.id          =     1=12
index.dc.title        =     1=4
index.dc.subject      =     1=21

relation.eq           =     2=3
relation.scr          =     2=3

truncation.right      =     5=1
truncation.left       =     5=2
```

4.2.2.1 Translation of CQL Index to Z39.50 Attributes

A line should be defined for each recognized CQL index. The syntax is:

```
index.<set>.<name> = <z39.50 attributes list>
```

- **Set** – The context set defined in the context sets definition part described above.
- **Name** – The CQL index name.

For example:

```
index.dc.title =  1=4
```

In this example, if the `sru_server` receives the query `dc.title=history` it is translated to the Z39.50 term `history` with the Z39.50 `Use` attribute set to 4 (title).

Note:

- The `index.cql.serverChoice` denotes the default attribute pair that is used when no index has been defined in the CQL query. Therefore, it must be defined. For example, consider a setup such as:

  ```
  index.cql.serverChoice = 1=1016.
  ```

  In this setup, if the `sru_server` receives a query such as:

  ```
  query=history
  ```

  the query is translated to the Z39.50 term `history`, with the Z39.50 `Use` attribute set to 1016 (any).

- If the CQL index should not be translated to a `Use` attribute then the Z39.50 attribute pair part should be left empty.

  For example, consider a setup such as
  ```
  index.dc.title = .
  ```

  In this setup, if the `sru_server` receives a query such as:
  ```
  dc.title=history
  ```

  it is translated to the Z39.50 term `history`. Z39.50 `Use` attribute is not set.

4.2.2.2 Translation of CQL Relation to Z39.50 Attributes

CQL relations supported by the `sru_server` are translated to Z39.50 attributes by using the relation keyword. The syntax is as follows:

```
relation.<relation name> = <z39.50 attributes list>
```

For example:

- `relation.<  =  2=1`
  Relation < (less than) is translated to relation attribute 1.
- `relation.le  =  2=2`
  Relation <= (less than or equal) is translated to relation attribute 2.
- `relation.eq  =  2=3`
  Relation = (equal) is translated to relation attribute 3.

- `relation.ge = 2=4`
  Relation >= (greater than or equal) is translated to the relation attribute 4
- `relation.> = 2=5`
  Relation > (greater than) is translated to the relation attribute 5

Note:

- A special pattern `relation.*` can be used to translate unmatched relations. For example, if there is a definition *`relation.* = 2=3`*, then any unmatched relation will be translated to relation attribute 3.

- A special pattern `relation.scr` must be defined. This relation sets the Z39.50 relation attributes that are used for queries which do not contain a relation. It is recommended to define it as `relation.scr = 2=3`.
- A structure attribute may be defined using *`relation`* pattern or using *`structure`* pattern. A line such as the following must be defined:

  ```
  structure.* = [<Z39.50 attribute list>]
  ```

  For example:

  ```
  structure.* = 4=1
  ```

- Relation modifiers can be translated using the `relationModifier.<mod>` keyword. The syntax is `relationModifier. <mod> = <z359.50 attributes list>.`

4.2.2.3 Translation of CQL Anchor Sign to Z39.50 Attributes

In CQL the anchor sign (^) defines to which side (left, right, or both) the search term should be anchored. If ^ does not occur in the expression, then it is not anchored to any side.

The ^ sign is translated to Z39.50 attributes using the *`position`* keyword with the following syntax:

```
position.<anchor position> = <Z39.50 attributes list>
```

The <`anchor position`> can have the following values**:**

- first
- any

The anchor position may be translated with the Z39.50 Position (3) attribute, for example:

- `position.first = 3=1`
- `position.any = 3=3`

The `position.*` line may be used to define the default anchor for all unmatched anchors.

4.2.2.4 Translation of CQL Wildcard Sign to Z39.50 Attributes

The CQL wildcard sign (*) is translated to Z39.50 attributes using the truncation keyword. The syntax is:

```
truncation.<wildcard position> = <Z39.50 attributes list>
```

The <wildcard position> can have one of the following values:

- left
- right
- none

The wildcard (*) may be translated with the Z39.50 Truncation (5) attribute, for example:

- `truncaion.right = 5=1`
- `truncaion.left = 5=2`
- Possible translation for `none` is `truncaion.none = 5=100`

4.2.2.5 Adding Z39.50 Attributes to All Queries

There is an option to add Z39.50 attributes to all queries sent by the `sru_server` to the `z39_server`. The syntax is:

```
always = <Z39.50 attributes list>.
```

For example, if `always = 6 = 1` is defined, the completeness attribute (6) with the value incomplete field (1) is added to all z39.50 queries.

4.2.2.6 pqf.properties Example

```
#context sets definition ##############################
set.cql         = info:srw/cql-context-set/1/cql-v1.1
set.dc          = info:srw/cql-context-set/1/dc-indexes/v1.0/
set.rec         = info:srw/cql-context-set/2/rec-1.0

# index translation ###################################
index.cql.serverChoice = 1=1016
index.rec.id           = 1=12
index.dc.title         = 1=4
index.dc.subject       = 1=21
index.dc.creator       = 1=1003

#relation translation #################################
relation.eq            = 2=3
relation.scr           = 2=3

#truncation translation ###############################
truncation.right       = 5=1
truncation.left        = 5=2

#default values #######################################
position.*     =
structure.*    =
```