Z39_server

Version 18.01 and Later

CONFIDENTIAL INFORMATION

DISCLAIMER

TRADEMARKS

Document released: January 2012

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

Table of Contents

1 Overview

Note:
The terms Z39 and Z39.50 are used interchangeably throughout this document. Both terms refer to the Z39.50 NISO protocol for network retrieval of bibliographic data.

The `z39_server` program is a standard Z39.50 server (Z39.50 target). It communicates with remote Z39.50 clients, allowing them to search, scan, and retrieve records in your local databases.

The architecture is as follows:

1. `z39_server` gets a request from a Z39.50 client, performs the request in your local databases and returns the response to the calling Z39.50 client.

2. `z39_server` implements the following Z39.50 services:
   `Init, Search, Sort, Present, Scan, Close, Delete-Result-Set.`

1.1 Init Service

`z39_server` gets the `InitRequest` message from the Z39.50 client. If the message contains a user name and password, `z39_server` stores them (they can then be used for access to your local database; by default, the user name *`Z39`* and the password *`Z39`* are used).

`z39_server` establishes a connection with the client and sends an *`InitResponse`* message to the client. If an error occurs, the message contains an error indication.

1.2 Search Service

`z39_server` gets the *`SearchRequest`* message from the Z39.50 client. The message contains a database name, a Z39.50 search query and a result set name.

1. `z39_server` translates the Z39.50 query into an ALEPH CCL (Common Command Language) query, searches the database, and creates a result set with a given name.

2. `z39_server` sends a *`SearchResponse`* message to the client. The message contains the number of hits. If an error occurs, the message contains an error indication.

1.3 Sort Service

1. `z39_server` gets a *`SortRequest`* message from the Z39.50 client. The message contains a database name, a Z39.50 sort query and a result set name.

2. `z39_server` translates the Z39.50 query into ALEPH sort codes and performs a sort of the given set.

3. `z39_server` sends a *`SortResponse`* message to the client. The message contains a success/failure indication.

1.4 Present Service

1. `z39_server` gets a *`PresentRequest`* message from the Z39.50 client. The message contains a database name, a result set name, the number of the first record to retrieve, the total number of records to retrieve, an optional record format (USMARC, UNIMARC, MAB, XML, SUTRS, and OPAC) and an optional elements set name.

2. `z39_server` retrieves the requested records and sends a *`PresentResponse`* message to the client. The message contains the retrieved records in the requested format. If an error occurs, the message contains an error indication.

3. Note that, using OPAC record syntax, `z39_server` returns holdings/items information. The elements set name is ignored when the record syntax is OPAC.

1.5 Scan Service

`z39_server` gets a *`ScanRequest`* message from the Z39.50 client. The message contains a database name and a Z39.50 scan query.

`z39_server` translates a Z39.50 scan query into an ALEPH CCL query and scans the database.

`z39_server` sends a *`ScanResponse`* message to the client. The message contains the scan entries. If an error occurs, the message contains an error indication.

1.6 ES Update Service

`z39_server` gets an *`ExtendedServicesRequest`* message from the Z39.50 client. The message contains a database name, a record and an action (Insert, Replace or Delete).
`z39_server` updates the record in the database, and sends an *`ExtendedServicesResponse`* message to the client. The message contains relevant update information

Note:
All processing is done via pc_server. z39_server receives a request from a client. It translates the request from Z39.50 format into ALEPH format and transfers the translated request to pc_server. pc_server executes the request and sends the response to z39_server. z39_server translates the response from ALEPH format into Z39.50 format and sends it to the client.

2 z39_server Services

2.1 Search Service

The *Search* service supports the majority of Bath profile queries. The details of the conformance to the Bath profile are contained in the *z39_server_bath_conformance* document.

Apart from the Bath profile queries, the *Search* Service supports the following attributes and any queries built from these attributes:

Use (1) = any attributes defined in `z39_server_<base>.conf`
Relation(2) = 3(equal)
Position(3) = 1(first in field), 3(any position in field)
Structure(4) = 1(phrase), 2(word), 6(word list)
Truncation(5) = 1(right truncation), 2(left truncation), 100(do not truncate)
Completeness(6) = 1(incomplete sub-field), 3(complete field)

2.1.1 Translation of a Z39.50 query into an ALEPH query

`z39_server` translates a Z39.50 query into an ALEPH CCL query according to the following rules:

- If Use (1) is not defined, set it to **`1016`**(Any).
- If Completeness(6) is 3(complete field) or Position(3) is 1(first in field) – translate Use attribute value to '`access/browse`' CCL code using `z39_server_<base>.conf`; otherwise – translate the Use attribute value to '`find`' CCL code using `z39_server_<base>.conf`.
- If Structure(4) is not defined – if the term is a single word, define Structure(4) to be 2(word), otherwise – if Completeness(6) is 3(complete field) or Position(3) is 1(first in field) – define it to be 1(phrase), otherwise define it to be 6(word list).
- If Structure(4) is 1(phrase) – search in **ALEPH** for adjacent words.
- If Structure(4) is 6(word list) – insert 'AND' between words of the term.
- If Structure(4) is 2(word) – the term must be a single word.
- If Truncation(5) is 1 (right truncation), and :
- if Structure(4) is not 6(word list)– add '?' to right end of the term;
- if Structure(4) is 6(word list)– add '?' to right end of each word of the term;
- If Truncation(5) is 2 (left truncation), and:
  - if Structure(4) is not 6(word list)– add '?' to left end of the term;
  - if Structure(4) is 6(word list)– add '?' to left end of each word of the term;

Examples:

Z39.50: Use(1)=4(title), term="history"
ALEPH: WTI=("history")

Z39.50: Use(1)=4(title), Completeness(6)=3(complete field), term="history"
ALEPH: TIT=("history")

Z39.50: Use(1)=4(title), Structure(4)=1(phrase), term="american history"
ALEPH: WTI=("american history"), search for adjacent words
Z39.50: Use(1)=4(title), Structure(4)=6(word list), term="american history"
ALEPH: WTI=("american" AND "history")

Z39.50: Use(1)=4(title), Truncation(5)=1(right truncation), term="histor"
ALEPH: WTI=("histor?")

2.2 Scan Service

The *Scan* service supports all Bath profile queries. The details of the conformance to the Bath profile are contained in the *z39_server_bath_conformance* document.

Apart from the Bath profile queries, the *Scan* Service supports the following attributes and any queries built from these attributes:

Use (1) = any attributes defined in `z39_server_<base>.conf`
Relation(2) = 3(equal)
Position(3) = not supported
Structure(4) = 1(phrase), 2(word)
Truncation(5) = not supported
Completeness(6) = 1(incomplete sub-field), 3(complete field)

2.2.1 Translation of a Z39.50 query into an ALEPH query

`z39_server` translates a Z39.50 query into an ALEPH CCL query according to the following rules:

- If Completeness(6) is undefined: if Structure(4) value is 2(word) – translate Use attribute value to '`find`' CCL code using `z39_server_<base>.conf`; otherwise – translate the Use attribute value to '`access/browse`' CCL code using `z39_server_<base>.conf`.
- If Completeness(6) is 1(incomplete sub-field) – translate Use attribute value to '`find`' CCL code using `z39_server_<base>.conf`; otherwise – translate the Use attribute value to '`access/browse`' CCL code using `z39_server_<base>.conf`.

Examples:
Z39.50: Use(1)=4(title), term="history"
ALEPH: TIT=("history")

Z39.50: Use(1)=4(title), Completeness(6)=1(incomplete sub-field), term="history"

ALEPH: WTI=(“history”)

Z39.50: Use(1)=4(title), Completeness(6)=3(complete field), term=”history”
ALEPH: TIT=(“history”)

Z39.50: Use(1)=4(title), Structure(4)=1(phrase), term=”history”
ALEPH: TIT=(“history”)

Z39.50: Use(1)=4(title), Structure(4)=2(word), term=”history”
ALEPH: WTI=(“history”)

2.3 Present Service

Records can be returned in the following formats: USMARC, UNIMARC, MAB, XML, SUTRS and OPAC. For each local database the records can be returned in XML, SUTRS and OPAC. If a local database contains USMARC record – the records can be returned in USMARC, if a local database is in UNIMARC – the records can be returned in UNIMARC, if a local database is in MAB - the records can be returned in MAB.

2.3.1 XML records

XML records can be returned according to Dublin Core DTD or can be returned according to Z39.50 Holdings Schema (v.4).

Dublin Core
XML records contain Dublin Core elements and are built according to the Dublin Core DTD as recommended by the Bath profile. Here is the Dublin Core specification:

```
<!DOCTYPE dublin-core-simple [
<!-- Dublin Core Version 1.1 -->
<!-- Based on CIMI Guide to Best Practice 1999-08-12 -->
<!ELEMENT record-list (dc-record*)>
<!ELEMENT dc-record (title | creator | subject | description |
publisher | contributor
| date | type | format | identifier | source | language | relation |
coverage |rights)*>
<!ELEMENT title (#PCDATA) >
<!ELEMENT creator (#PCDATA) >
<!ELEMENT subject (#PCDATA) >
<!ELEMENT description (#PCDATA) >
<!ELEMENT publisher (#PCDATA) >
<!ELEMENT contributor (#PCDATA) >
<!ELEMENT date (#PCDATA) >
<!ELEMENT type (#PCDATA) >
<!ELEMENT format (#PCDATA) >
<!ELEMENT identifier (#PCDATA) >
<!ELEMENT source (#PCDATA) >
<!ELEMENT language (#PCDATA) >
<!ELEMENT relation (#PCDATA) >
<!ELEMENT coverage (#PCDATA) >
<!ELEMENT rights (#PCDATA) >
]>
```

The tab12 table supports Z39.50 and facilitates the translation from ALEPH internal format to XML. This table resides in the library's tab directory.

tab12 contains three columns:

col.1 - ALEPH internal field (for example: 245)

col.2 - subfields specification. It can include:

subfield(s) of the field (blank indicates the entire field).

OR

minus (-) sign followed by subfields to be stripped

OR

F followed by fixed field position, followed by hyphen (-) and the number of characters (for example: F35-03 means that the information starts in position 35 and its length is 3 characters).

col.3 - Dublin Core element.

Possible values: title, creator, subject, description, publisher, contributor, date, type, format, identifier, source, language, relation, coverage, rights

When a Z39.50 client sends a request to the ALEPH Z39.50 server for a record in XML format, the ALEPH Z39.50 server runs over the tab12 of the corresponding library and translates each specified ALEPH field into the corresponding Dublin Core element.

If a library contains records in USMARC in addition to Dublin Core elements defined by the tab12 table of the library, the XML record will contain the <type> element. Type is the translation of the LDR element of the original record.

The <type> element contains the following data:

If position 7 of LDR is "c", "s" or "i", the <type> element contains the "collection" string.

Otherwise, if position 6 of LDR is "a", "c", "d" or "t", the <type> element contains the "text" string.

If position 6 of LDR is "e", "f", "g" or "k" the <type> element contains the "image" string.

If position 6 of LDR is "i" or "j" the <type> element contains the "sound" string.

If positions 6 or 7 of LDR have other values, the <type> element is not added.

Holdings Schema

XML records can be returned with Holdings information according to Z39.50 Holdings Schema (v.4). The records are returned in XML format according to the element set names B1, B2 and C2 as defined in the Bath profile (v. 2.0)

Example of a retrieved record (B1):

```
<?xml version = "1.0" encoding = "UTF-8"?>
```

```
<holdingsStructure
xmlns="http://www.loc.gov/z3950/agency/defns/BathHoldingsLo
cationsOnly" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.loc.gov/z3950/agency/defns/BathHolding
sLocationsOnly
http://www.loc.gov/z3950/agency/defns/BathHoldingsLocationsOnly.xsd">
<bibItemInfo-1 targetItemId-3="000033902"></bibItemInfo-1>
<holdingsStatement-4>
<siteLocation-6 institutionOrSiteId-27="HUJI" locationName-28="Hebrew
University">
<subLocation-35 institutionOrSiteId-27="UEDUC" locationName-
28="Educatin Library"></subLocation-35>
</siteLocation-6>
</holdingsStatement-4>
</holdingsStructure>
```

B1 records configuration:

In the `tab_expand` of the local bibliographic library (for example, USM01), two expand programs must be under the Z39_H0L menu:

```
expand_doc_bib_loc_usm
expand_doc_sort_field          SBL##,a
```

The tab_location_name.<lng> table must reside in ./alephe/tab.

The table contains four columns:

Col.1: Code of the physical library, for example, USM50
Col.2: Location code (same as 852 $$a)
Col.3: ISL code (ISO 15511)
Col.4: Location name

Example:

```
! 1          2                      3                          4
!!!!!-!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!!!-!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
USM50 HUJI                                      Hebrew University
XYZ50 DNLM                                      National Library of
Medicine
```

B2 Records Configuration:

In the `tab_expand` of the local bibliographic library (for example, USM01), the following program must be under the Z39_H0L menu:

```
expand_doc_bib_loc_usm
```

The `tab_location_name.<lng>` table must reside in `./alephe/tab`.

C2 records configuration:

`tab_z30_sort` of the BIB library must contain the following line:

```
Z39-1                A 06 A 06
```

The `tab_location_name.<lng>` table must reside in `./alephe/tab`.

Note:
1. Maximum size of Holdings Records is 60,000 characters.
2. The records are returned in the UTF-8 character set.

2.3.2 SUTRS records

A SUTRS record is a list of lines. The line syntax is: `<title>:<content>` where `<title>` is the translation of the document code and `<content>` is the document line content. The document code is translated to `<title>` using the `tab01` table.

Example of a SUTRS record:

```
ME-Personal Name:    Smith, Tom
Title:    History of mumps.
Publication Area:    1987.
Owner:    MUMPS
```

Corresponding USMARC record:

```
001 000015762
005 20020716153629.0
008 020526s1987    cau     r     000 0 eng d
100 1  $a Smith, Tom
245 10 $a History of mumps.
260 $c 1987.
```

2.3.3 Element Sets

`z39_server` supports two special element sets, “F” and “X”.

When the F element set is requested:
If the F element set is defined in `./alephe/tab/z39_server/z39_server_elements`, the record is returned according to the lines defined in the table.
If the F element set is not defined in the `z39_server_elements` table, a full record is returned.

When the X element set is requested, a full record including Aleph alphanumeric fields (such as CAT, SYS, and so on) is returned. The X element set is recognized for USMARC format only. For all other formats, it is ignored.

You can define additional element sets using the `z39_server_elements` configuration file.

MARC records returned by `z39_server` always contain the 005 tag even if tag 005 is not defined in the element set configuration (in `./alephe/tab/z39_server/z39_server_elements`).

2.3.4 Internal caching

To improve performance, `z39_server` stores the records returned by `pc_server` in the internal cache. In `./aleph/proc/z39_server script` two environment variables are defined:

`z39_server_cache_size` – defines the size of the cache (for example, `z39_server_cache_size 100`). If `z39_server_cache_size` is 0 or undefined, the records returned by `pc_server` are not cached.

`z39_server_present_size` – defines how many records `z39_server` requests from `pc_server` in each interaction. (for example `z39_server_present_size 10`). The default is 1.

2.4 ES Update Service

The ES Update Service has been developed according to the Union Catalog Profile (http://lcweb.loc.gov/z3950/agency/profiles/ucp.doc) and partially conforms to the profile's requirements.

"The Z39.50 ES Update Service allows an origin to request that the target update a database, insert new records, replace or delete existing records." (from ANSI/NISO Z39.50-1995)

The ALEPH `z39_server` supports three Update actions: Record Insert, Record Replace and Record Delete. All the actions are performed immediately. Extended Services Database is not used.

For all Update actions, a z39 client must have the following permissions: `Z39-SERVER/OPAC, CATALOG/UPDATE and` CATALOG/REMOTE-UPDATE. (a z39 client user name/password are passed during *`Init Request`* in the `idAuthentication` parameter)

2.4.1 Record Insert

Purpose:
Inserts new record into an ALEPH database

ExtendedServicesRequest parameters (supplied by a z39 client):
database name (for example,. USM01)
record to insert - in USMARC format only

ExtendedServicesResponse parameters (supplied by the ALEPH Z39.50 server):

On error: error diagnostic (s)
On success: the new record from the ALEPH database and supplemental diagnostics. The supplemental diagnostics always include diagnostic 951 (Record insert reviewed and modified) or diagnostic 971 (Record insert accepted - warning suspect duplicate). There may also be additional warning diagnostics.
Addinfo of diagnostic 951 and diagnostic 971 contains the record ID and version ID of the new record, thus allowing subsequent update/deletion of the record.
The addinfo format is "`Record Id: <record id> Version Id: <version id>`" where
`<record id>` is the ALEPH system number (nine numbers) and
`<version id>` is the content of the ALEPH CAT field.

Example:

```
"RecordId: 000035257
 Version Id: $$aYOHANAN$$b99$$c20031110$$lUSM01$$h1055"
```

Workflow:
- The character conversion of the incoming record is performed (the character conversion is defined in the `in-find-char-conv` setting of `./alephe/tab/z39_server/z39_server_<base>.conf`)
- `check_doc` is performed with the check type `Z39-INSERT`

The `check_doc` errors are mapped to standard Z39.50 errors using the following configuration file:
`./alephe/tab/z39_server/z39_server_update_errors`

Note:
Not all check_docs errors are added to the z39_server response, only those listed in z39_server_update_errors file.

- The incoming record is added to the ALEPH database
- The new record from ALEPH database is returned to the z39 client in Extended Services Response. The record passes through character conversion. (the character conversion is defined in `out-record-char-conv` setting for `out-record-syntax` equal to USMARC in `./alephe/tab/z39_server/z39_server_<base>.conf`)

2.4.2 Record Replace

Purpose:
Replace record in an ALEPH database. The existing record is fully replaced by the supplied record.

ExtendedServicesRequest parameters (supplied by a z39 client):
database name (for example, USM01)
record to replace - in USMARC format only
recordId - ALEPH system number as supplied in the last Insert/Update response;
supplementalId - version ID as it was supplied in last Insert/Update response; should be equal to the value of last CAT field of the ALEPH document.

ExtendedServicesResponse parameters (supplied by ALEPH Z39.50 server):
On error: error diagnostic (s)
On success: the updated record from the ALEPH database and supplemental diagnostics. The supplemental diagnostics always include diagnostic 954 (Record replace or element update reviewed and modified).There may also be additional warning diagnostics.
Addinfo of diagnostic 954 contains record ID and version ID of the updated record (the format as in Record Insert)

Flow:
- The character conversion of the incoming record is performed (the character conversion is defined in `in-find-char-conv` setting of `./alephe/tab/z39_server/z39_server_<base>.conf`)
- the version of the record is checked: incoming version id must be equal to the value of last CAT field of the existing database record. (version ID from incoming supplemenatId or last CAT field of the incoming record is used)
- `check_doc` is performed with check type `Z39-REPLACE`
  The `check_doc` errors are mapped to standard Z39.50 errors using configuration file:
  `./alephe/tab/z39_server/z39_server_update_errors`

Note:
Not all `check_doc` errors are added to the `z39_server` response, only those listed in the `z39_server_update_errors` file.

- The existing record in the ALEPH database is fully replaced by the incoming record
- The updated record from the ALEPH database is returned to `z39 client` in Extended Services Response. The record passes through character conversion. (the character conversion is defined in `out-record-char-conv` setting for `out-record-syntax` equal to USMARC in `./alephe/tab/z39_server/z39_server_<base>.conf`)

2.4.3 Record Delete

Purpose:
Deletes an existing record from the ALEPH database.

ExtendedServicesRequest parameters (supplied by a z39 client):
database name (for example, USM01)
record to delete - in USMARC format only
recordId - ALEPH system number as it was supplied in last Insert/Update response;
supplementalId - version ID as it was supplied in the last Insert/Update response; should be equal to the value of last CAT field of the ALEPH document.

ExtendedServicesResponse parameters (supplied by ALEPH Z39.50 server):
On error: error diagnostic (s)
On success: diagnostic 958 (Record delete request reviewed and updated - record deleted) and optionally additional warning diagnostics.
Addinfo of diagnostic 958 contains record ID and version ID of the deleted record

(the format as in Record Insert)

Flow:

- The character conversion of the incoming record is performed
  (the character conversion is defined in `in-find-char-conv`
  setting of `./alephe/tab/z39_server/z39_server_<base>.conf`)
- the version of the record is checked: incoming version ID must be equal
  to the value of last CAT field of the existing database record. (version ID from
  incoming supplement ID or last CAT field of the incoming record is used)
- `check_doc` is performed with check type `Z39-DELETE`
  The `check_doc` errors are mapped to standard Z39.50 errors
  using configuration file:
  `./alephe/tab/z39_server/z39_server_update_errors`

Note:
Not all `check_doc` errors are added to `z39_server`
response, only those listed in the `z39_server_update_errors` file.

- The record is deleted from the ALEPH database.

3 z39_server configuration

z39_server makes extensive use of configuration files. All the configuration files of z39_server are in the $alephe_tab/z39_server directory.

Below is the list of z39_server configuration files:

- z39_server.conf – general z39_server parameters
- z39_server_<base>.conf – translation of Z39.50 attributes to **ALEPH** codes
- z39_server_elements – elements sets definition
- z39_server_update_errors – translation of check_doc error codes to z39.50 ES error codes

3.1 z39_server.conf

This file contains general settings for z39_server. Each setting must be defined in a separate line. The file can contain comments (lines starting with #).

The following setting is mandatory:

pc_server address

Syntax:
hostname <host>:<port>
where **<host>** is pc_server host and **<port>** is pc_server port;

Examples:
hostname localhost:6515
This means that pc_server runs on a local host, on port 6515

hostname ram42:6515
This means that pc_server runs on host ram42 on port 6515

The following setting is optional:

log records indication

Syntax:
marclog **<marc log file name>**
If it is defined, the records returned by z39_server in *PresentResponse* are also saved in the $TMPDIR/**<marc log file name>** logfile. This is useful for debugging purposes.

Example:
marclog my.log
This means that the records are stored in a file called $TMPDIR/my.log

3.2 z39_server_<base>.conf

For each base handled by `z39_server`, a `z39_server_<base>.conf` file must be defined, where **<base>** is the base name (for example, `z39_server_USM01.conf` is used for USM01). The file contains different setting definitions for working with a given base. Each setting must be defined in a separate line. The file can contain comments (lines starting with #)

The file can contain the following groups of settings:

- General settings
- Present settings
- Settings for search/scan in Word index (Word translations)
- Setting for searc/scan in Access index (Phrase translations)

3.2.1 General settings

Syntax

`<in-find-char-conv> <char-conv>`
Character conversion for find request; Character conversion for incoming record in ES Update request

`<in-scan-char-conv> <char-conv>`
Character conversion for scan request

`<out-scan-char-conv> <char-conv>`
Character conversion for scan response

`<connection-language> <language-code>`
For Z39 server searches to support a language (a three-letter uppercase language code).

`<opac-record-size> <num>`
`<num>` is number of kilobytes. If the total size of an OPAC record is greater than `<num>`, the OPAC record returned by `z39_server` will include only MARC data without any holdings data.

`<unioncatalog>`
If this setting is enabled, the OPAC record has a special form

`<nosort>`
If this setting is enabled, sort is disabled for this base

`<noscan>`
If this setting is enabled, scan is disabled for this base

`<real-base> <base name>`
Optional configuration, enabling the setting of a tab_base.lng search base that is different from the externally known base. For example, if two externally known bases (e.g. HIS1 and HI2) differ only in their search and retrieve modes, but search the same tab_base.lng base (e.g. HIST), then two separate

alephe/tab/z39_server/ z39_server_<*base*>.conf files will be created (e.g. z39_server_HIS1.conf and z39_server_HIS2.conf). Both files will include a line such as:

real-base HIST

Note that you should define settings to the real base name in all relevant configuration tables (z39_server_elements, tab_base.eng, etc.).

Examples:

```
in-find-char-conv 8859_1_TO_UTF
in-scan-char-conv 8859_1_TO_UTF
out-scan-char-conv UTF_8859_1
```

3.2.2 Present settings

`<out-record-syntax> <syntax>`

Record syntax must be defined for each syntax z39_server returns.

If you want to allow the retrieval of records in SUTRS and OPAC (for example), you must define two corresponding `out-record-syntax` lines.

Possible values are: SUTRS, OPAC, XML and GRS-1.

If the library contains records in the following syntaxes, the following values are also possible: UNIMARC, INTERMARC, CCF, USMARC, UKMARC, NORMARC, LIBRISMARC, DANMARC, FINMARC, CANMARC, SBN, PICAMARC, AUSMARC, IBERMARC, CATMARC, MALMARC and MAB.

Note:
If a Z39 client does not specify a preferred record syntax, then the syntax specified in the first out-record-syntax line is returned.

`<out-record-format> <format>`

This setting defines which "physical" record formats can be returned from a given base. It must be defined for each syntax. The setting is optional.

The default values are:

if `out-record-syntax` is USMARC – out-record-format is USMARC
if `out-record-syntax` is DANMARC – out-record-format is DANMARC
if `out-record-syntax` is MAB - out-record-format is MAB
if `out-record-syntax` is XML - out-record-format is XML
if `out-record-syntax` is SUTRS - out-record-format is SUTRS
if `out-record-syntax` is GRS-1 - out-record-format is GRS-1
if `out-record-syntax` is OPAC - out-record-format is OPAC

For all other values of out-record-syntax, the default value of out-record-format is USMARC

`<out-record-char-conv> <char-conv>`
Character conversion for the record; When `out-record-syntax` is USMARC the setting defines also character conversion for outgoing record in ES Update Response.

`<out-record-fix> <fix>`
Fix routine from tab_fix

`<out-record-expand> <expand>`
Expand routine from tab_expand

Examples:

```
out-record-syntax USMARC
out-record-format USMARC
out-record-char-conv UTF_TO_8859_1
out-record-fix FIX1
out-record-expand EXP1

out-record-syntax UNIMARC
out-record-format USMARC
out-record-char-conv UTF_TO_8859_1
out-record-fix FIX2
out-record-expand EXP2

out-record-syntax XML

out-record-syntax OPAC
```

3.2.3 Translation of Z39.50 Use Attributes into CCL Codes

Syntax:

`<register>  <CCL code>  <z39 Use attribute>`

`<register>` is a‘word’ or a ‘phrase’
‘word’ register– contains ‘find’ CCL codes
‘phrase’ register – contains ‘access’ CCL codes

`<CCL code>` is any CCL code defined in ALEPH

`<z39 Use attribute>` is any Z39.50 use attribute

You must define a translation line for each required Z39.50 to CCL translation. These settings define all Z39.50 to CCL translations handled by `z39_server`.

Examples:

‘`word wti 4`’ – translates Z39.50 Use attribute `4` (title) into the ALEPH CCL code ‘`wti`’.

‘`phrase tit 4`’ – translates Z39.50 Use attribute `4` (title) into the ALEPH CCL code ‘`tit`’.

It is also possible to map the Z39.50 use attribute to multiple CCL codes, causing the search to be on multiple indexes with a Boolean OR. For example:

word (wau,wti) 1016

This definition will cause a query with the use attribute 1016 to be a Boolean OR in the WAU and WTI indexes.

3.2.4 Translation of Z39.50 Use Attributes / Sort Keywords into ALEPH Sort Codes

Syntax:
`sort <ALEPH Sort code> <Z39 Use attribute value> <sortkey>`
When `z39_server` gets a *`SortRequest`* message, it translates the supplied Z39.50 Use attribute or sort key into ALEPH sort code. After that, the sort in ALEPH is made according to this ALEPH sort code. The translation is done according to these settings.

You must define a translation line for each required Z39.50 to ALEPH translation. These settings define all Z39.50 to ALEPH translations handled by `z39_server`.

Example
'`sort 02 1 Author`' - translates Z39.50 Use attribute 1(personal name) and keyword 'Author' into the ALEPH sort code '`02`'.

3.3 z39_server_elements

This file appears in the regular ALEPH format. It contains a header according to which each line is filled. Lines starting with '!' are comments. Each line contains element specification. The line has the following structure.

Column 1 – base name
Column 2 – element set name (for example, "B"). It is case-sensitive, so "B" and "b" are different element set names
Column 3 – Format – as defined in the FMT tag of the record. The field can contain the '#' wild card. The element is added to the created record if the content of the FMT tag of the database record matches this Format specification.
Column 4 – record field. The field can contain the '#' wild card.
Column 5 - subfield list (for example, abc)

The `z39_server_elements` table lets you define several elements sets for every base. Each element set may contain several elements. Each element can be valid for any record or for a record with a specific FMT value only.

Example

Column1	Column2	Column3	Column4	Column5
USM01	B	##	100##	
USM01	B	##	245##	ab
USM01	C	##	245##	
USM01	C	BK	100##	
UNI01	B	##	100##	
UNI01	B	##	200##	
UNI01	B	##	700##	

This example defines two element sets for records returned from USM01 –"B" and "C" and one elements sets for UNI01 - "B".

If a Present Request contains base "USM01" and the elements set name "B", the returned record will contain tags 100 and 245 only. The 245 tag of the returned record will contain subfields a and b only.

If a Present Request contains base "USM01" and elements set name "C":

If the FMT tag of the record is equal to "BK" the returned record will contain tags 100 and 245, otherwise it will contain tag 245 only.

If a Present Request contains base UNI01 and elements set name "B" the returned record will contain tags 200 and 700 only.

Note
Following any change in the z39_server_elements table, pc_server **must** be restarted.

3.4 z39_server_update_errors

This file appears in the regular ALEPH format. It contains a header according to which each line is filled. Lines starting with '!' are comments.
Each line contains two columns:

Column 1 - `check_doc` error code (from `./alephe/error_eng/check_doc`)
Column 2 – z39 error code (from `./alephe/tab/z39_gate/z39_es_error_list` or from `./alephe/tab/z39_gate/z39_target_error_list`)

The file defines translation of `check_doc` error codes to Z39.50 ES error codes.
Each `z39 server` Extended Services Response contains one main diagnostic which is defined by the program and optional additional diagnostics. The "additional" diagnostics are configurable: each `check_doc` error/warning which occurs in `z39_server_update_errors` file is translated into corresponding "additional" diagnostic.

4 User Authentication

InitRequest has an optional parameter: idAuthentication. This parameter contains user and password. If the idAuthentication parameter in *InitRequest* is not empty, z39_server sends the received user and password to pc_server. If idAuthentication is empty, z39_server sends user = "Z39" and password = "Z39" to pc_server.

pc_server authenticates the user as a regular ALEPH user. The user must have Z39.50 user permissions for the requested library. You can define Z39.50 user permissions via the Staff Privileges window as for a regular ALEPH user.

To access the Staff Privileges window, right-click the key icon  on the Operations bar of an ALEPH GUI (at the bottom right of the screen) and select Staff Privileges/summary of access.

The Staff Privileges function allows you to grant access to a library to all users or to specified users only.

To grant access to all users, assign to the "Z39" user *z39.50 users permissions* to the library. Z39.50 clients connecting to your z39_server must send *empty idAuthentication* in *Init Request*.

To grant access to specified users, assign to specified users *z39.50 users permissions* for the library. Z39.50 clients connecting to your z39_server must send this user name and password in *idAuthentication* in *Init Request*.

5 Running z39_server

You run the `z39_server` program using UTIL.

1. From the command prompt, enter any library (for example, **`dlib USM01`**).
2. Enter **`UTIL, W, 3, 4`**. You are asked for a port number:

   ```
   Enter server port number or 0 to QUIT [9929]
   ```

3. Type a port number and press ENTER. The `z39_server` program runs on this port number in the background.

The log file is in `$LOGDIR/z39_server_<port>.log`, where **`<port>`** is the port you have given.

Notes

- To stop the `z39_server` process use util W.
- Before running `z39_server` check that `pc_server` runs on the host and port defined in `$alephe_tab/z39_server/z39_server.conf`.
- Before running `z39_server`, check that all appropriate users (including default user Z39) have *Z39.50 users* permission for access to desired libraries. Check this via the Staff Privileges window. To access the Staff Privileges window, right-click the key icon  on the Operations bar of an ALEPH GUI (at the bottom right of the screen) and select Staff Privileges/summary of access.

6 Defining a New Base for z39_server

To define a new base for `z39_server`:

1. From the command prompt, enter any library environment (for example, **`dlib usm01`**).
2. Enter **`UTIL, N, 2`**. The following menu appears:

```
Z39 server configuration
------------------------
0. Exit Procedure
1. Add Base
2. Edit Base
3. Delete Base
4. Display Base
5. Edit Server Conf (z39_server.conf)
6. Edit Element Sets Definition (z39_server_elements)
7. Help
```

3. Use the menu options to edit the `z39_server` configuration.

   - To define a new base, choose option '*`1. Add base`*'. Then enter a base name. The name is translated into uppercase and the file
   - `$alephe_tab/z39_server/z39_server_<base>.conf` can be opened in **vi**. Edit this file.
   - Use option 5 to edit `$alephe_tab/z39_server/z39_server.conf`.
   - Use option 6 to edit `$alephe_tab/z39_server/z39_server_elements.`

4. Rerun `z39_server` after any changes in configuration files.

7 Checking New Bases

There are two ways of checking new bases:

- Using `z39_gate`.
- Using `yaz_client`.

Using z39_gate
Define this base as the target for `z39_gate` and check it using `z39_gate_client`(UTIL/N/3). For example, USM01 has been defined as the base for `z39_server` on the local host and `z39_server` is run on port 9919. `USM01_Z39` can now be defined as the target for `z39_gate` on the local host with database `USM01` and hostname `localhost:9919`. After that, you can check target `USM01_Z39` using `z39_gate_client` (see also the `z39_gate` document and the `z39_gate_client` document).

Using yaz_client (UTIL/N/4)
See the *YAZ Client* document.