YAZ

Introduction

This extension offers a PHP interface to the YAZ toolkit that implements the » Z39.50 Protocol for Information Retrieval. With this extension you can easily implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

The module hides most of the complexity of Z39.50 so it should be fairly easy to use. It supports persistent stateless connections very similar to those offered by the various RDB APIs that are available for PHP. This means that sessions are stateless but shared among users, thus saving the connect and initialize phase steps in most cases.

YAZ is available at » http://www.indexdata.dk/yaz/. You can find news information, example scripts, etc. for this extension at » http://www.indexdata.dk/phpyaz/.

Note
This extension has been moved to the » PECL repository and is no longer bundled with PHP as of PHP 5.0.0.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

Obtain YAZ (ANSI/NISO Z39.50 support) and install it. YAZ can be fetched in source or in various prebuilt packages from the » YAZ archive. Systems such as Debian GNU/Linux, Suse Linux, FreeBSD also has YAZ as part of their distribution.

For the PHP 4 series, the YAZ extension is bundled (but not YAZ itself) Build PHP with your favorite modules and add option *--with-yaz[=DIR]*.

Example #1 - YAZ compilation for PHP 4 on Unix

```
gunzip -c php-4.4.X.tar.gz|tar xf -
gunzip -c yaz-2.1.8.tar.gz|tar xf -
cd yaz-2.1.8
./configure --prefix=/usr
make
sudo make install
cd ../php-4.4.X.
./configure --with-yaz=/usr/bin
make
sudo make install
```

For PHP 5, the YAZ extension is in » PECL and is installed as a shared object/dll. If you have pear installed the easiest way to download, configure and install the YAZ extension is by using the *pecl* command.

Installation of YAZ on Linux

Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/yaz

Installation on Windows systems

The DLL for this PECL extension may be downloaded from either the » PHP Downloads page or from » http://pecl4win.php.net/ *php_yaz.dll* depends on *yaz.dll*. The *yaz.dll* is part of the Win32 ZIP from the PHP site. It is also part of the Windows YAZ install available from the » YAZ WIN32 area.

Warning

The PHP 5.0.5 Win32 zip includes a too old version of *yaz.dll* (version 1.9.1 < required version 2.0.13). If that's the case use the *yaz.dll* from a newer » YAZ WIN32 install.

On windows, don't forget to add the PHP directory to the PATH, so that the *yaz.dll* file can be found by the system.

If you are using YAZ as a shared extension, add (or uncomment) the following line in *php.ini* on Unix:

```
extension=yaz.so
```

And for Windows:

```
extension=php_yaz.dll
```

Warning

The IMAP, recode, YAZ and Cyrus extensions cannot be used in conjuction, because they share the same internal symbols.

Note

The above problem is solved in version 2.0 of YAZ.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

YAZ configuration options

Name	Default	Changeable	Changelog
yaz.max_links	"100"	PHP_INI_ALL	Available since PHP 4.3.0. Removed in PHP 5.0.0.
yaz.log_file	NULL	PHP_INI_ALL	Available since PHP 4.3.0. Removed in PHP 5.0.0.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Resource Types

This extension has no resource types defined.

Predefined Constants

This extension has no constants defined.

Examples

PHP/YAZ keeps track of connections with targets (Z-Associations). A resource represents a connection to a target.

The script below demonstrates the parallel searching feature of the API. When invoked with no arguments it prints a query form; else (arguments are supplied) it searches the targets as given in array *host*.

Example #2 - Parallel searching using Yaz

```
<?php
$host=$_REQUEST[host];
$query=$_REQUEST[query];
$num_hosts = count($host);
if (empty($query) || count($host) == 0) {
   echo '<form method="get">
   <input type="checkbox"
   name="host[]" value="bagel.indexdata.dk/gils" />
       GILS test
   <input type="checkbox"
   name="host[]" value="localhost:9999/Default" />
       local test
   <input type="checkbox" checked="checked"
   name="host[]" value="z3950.loc.gov:7090/voyager" />
       Library of Congress
   <br />
   RPN Query:
   <input type="text" size="30" name="query" />
   <input type="submit" name="action" value="Search" />
   </form>
   ';
} else {
   echo 'You searched for ' . htmlspecialchars($query) . '<br />';
   for ($i = 0; $i < $num_hosts; $i++) {
       $id[] = yaz_connect($host[$i]);
       yaz_syntax($id[$i], "usmarc");
       yaz_range($id[$i], 1, 10);
       yaz_search($id[$i], "rpn", $query);
   }
   yaz_wait();
   for ($i = 0; $i < $num_hosts; $i++) {
       echo '<hr />' . $host[$i] . ':';
       $error = yaz_error($id[$i]);
       if (!empty($error)) {
           echo "Error: $error";
       } else {
           $hits = yaz_hits($id[$i]);
           echo "Result Count $hits";
       }
       echo '<dl>';
       for ($p = 1; $p <= 10; $p++) {
           $rec = yaz_record($id[$i], $p, "string");
           if (empty($rec)) continue;
```

```
            echo "<dt><b>$p</b></dt><dd>";
            echo nl2br($rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
?>
```

YAZ Functions

yaz_addinfo

yaz_addinfo -- Returns additional error information

Description

string **yaz_addinfo** (resource `$id`)

Returns additional error information for the last request on the server.

With some servers, this function may return the same string as yaz_error().

Parameters

`id`
The connection resource returned by yaz_connect().

Return Values

A string containing additional error information or an empty string if the last operation was successful or if no additional information was provided by the server.

See Also

- yaz_error()
- yaz_errno()

yaz_ccl_conf

yaz_ccl_conf -- Configure CCL parser

Description

void **yaz_ccl_conf** (resource `$id`, array `$config`)

This function configures the CCL query parser for a server with definitions of access points (CCL qualifiers) and their mapping to RPN.

To map a specific CCL query to RPN afterwards call the yaz_ccl_parse() function.

Parameters

`id`
The connection resource returned by yaz_connect().

`config`
An array of configuration. Each key of the array is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (=). Each pair is separated by white space. Additional information can be found on the » CCL page.

Return Values

No value is returned.

Examples

In the example below, the CCL parser is configured to support three CCL fields: *ti*, *au* and *isbn*. Each field is mapped to their BIB-1 equivalent. It is assumed that variable *$id* is the connection ID.

Example #3 - CCL configuration

```
<?php
$fields = array(
 "ti" => "1=4",
 "au"   => "1=1",
 "isbn" => "1=7"
);
yaz_ccl_conf($id, $fields);
?>
```

See Also

- yaz_ccl_parse()

yaz_ccl_parse

yaz_ccl_parse -- Invoke CCL Parser

Description

bool **yaz_ccl_parse** (resource `$id`, string `$query`, array `&$result`)

This function invokes a CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the yaz_search() function to perform a search.

To define a set of valid CCL fields call yaz_ccl_conf() prior to this function.

Parameters

`id`
The connection resource returned by yaz_connect().

`query`
The CCL FIND query.

`result`
If the function was executed successfully, this will be an array containing the valid RPN query under the key *rpn*. Upon failure, three indexes are set in this array to indicate the cause of failure:

- *errorcode* - the CCL error code (integer)
- *errorstring* - the CCL error string
- *errorpos* - approximate position in query of failure (integer is character position)

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #4 - CCL Parsing

We will try to search using CCL. In the example below, *$ccl* is a CCL query.

```
<?php

yaz_ccl_conf($id, $fields);  // see example for yaz_ccl_conf
```

```
if (!yaz_ccl_parse($id, $ccl, &$cclresult)) {
   echo 'Error: ' . $cclresult["errorstring"];
} else {
   $rpn = $cclresult["rpn"];
   yaz_search($id, "rpn", $rpn);
}
?>
```

yaz_close

yaz_close -- Close YAZ connection

Description

bool **yaz_close** (resource `$id`)

Closes the connection given by parameter *`id`*.

Note
This function will only close a non-persistent connection opened by setting the *persistent* option to **FALSE** with yaz_connect().

Parameters

`id`
: The connection resource returned by yaz_connect().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

See Also

- yaz_connect()

yaz_connect

yaz_connect -- Prepares for a connection to a Z39.50 server

Description

mixed **yaz_connect** (string `$zurl` [, mixed `$options`])

This function returns a connection resource on success, zero on failure.

yaz_connect() prepares for a connection to a Z39.50 server. This function is non-blocking and does not attempt to establish a connection - it merely prepares a connect to be performed later when yaz_wait() is called.

Note
The » YAZ proxy is a freely available Z39.50 proxy.

Parameters

`zurl`
: A string that takes the form *host[:port][/database]*. If port is omitted, port 210 is used. If database is omitted *Default* is used.

`options`
: If given as a string, it is treated as the Z39.50 V2 authentication string (OpenAuth). If given as an array, the contents of the array serves as options.

  user
  : Username for authentication.

  group
  : Group for authentication.

  password
  : Password for authentication.

  cookie
  : Cookie for session (YAZ proxy).

  proxy
  : Proxy for connection (YAZ proxy).

  persistent
  : A boolean. If **TRUE** the connection is persistent; If **FALSE** the connection is not persistent. By default connections are persistent.

Note
If you open a persistent connection, you won't be able to close it later with yaz_close().

piggyback
: A boolean. If **TRUE** piggyback is enabled for searches; If **FALSE** piggyback is disabled. By default piggyback is enabled. Enabling piggyback is more efficient and usually saves a network-round-trip for first time fetches of records. However, a few Z39.50 servers do not support piggyback or they ignore element set names. For those, piggyback should be disabled.

charset
: A string that specifies character set to be used in Z39.50 language and character set negotiation. Use strings such as: *ISO-8859-1*, *UTF-8*, *UTF-16*. Most Z39.50 servers do not support this feature (and thus, this is ignored). Many servers use the ISO-8859-1 encoding for queries and messages. MARC21/USMARC records are not affected by this setting.

preferredMessageSize
: An integer that specifies the maximum byte size of all records to be returned by a target during retrieval. See the » Z39.50 standard for more information.

Note
This option is supported in PECL YAZ 1.0.5 or later.

maximumRecordSize
: An integer that specifies the maximum byte size of a single record to be returned by a target during retrieval. This entity is referred to as Exceptional-record-size in the » Z39.50 standard.

Note
This option is supported in PECL YAZ 1.0.5 or later.

Return Values

A connection resource on success, **FALSE** on error.

ChangeLog

Version	Description
4.1.0	The parameter *options* was added.

See Also

- yaz_close()

yaz_database

yaz_database -- Specifies the databases within a session

Description

bool **yaz_database** (resource `$id`, string `$databases`)

This function allows you to change databases within a session by specifying one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to yaz_connect().

Parameters

`id`
: The connection resource returned by yaz_connect().

`databases`
: A string containing one or more databases. Multiple databases are separated by a plus sign +.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

yaz_element

yaz_element -- Specifies Element-Set Name for retrieval

Description

bool **yaz_element** (resource `$id`, string `$elementset`)

This function sets the element set name for retrieval.

Call this function before yaz_search() or yaz_present() to specify the element set name for records to be retrieved.

Note
If this function seems effectless, see the description of the *piggybacking* option in yaz_connect().

Parameters

`id`
: The connection resource returned by yaz_connect().

`elementset`
: Most servers support *F* (for full records) and *B* (for brief records).

Return Values

Returns **TRUE** on success or **FALSE** on failure.

yaz_errno

yaz_errno -- Returns error number

Description

int **yaz_errno** (resource `$id`)

Returns an error number for the server (last request) identified by *`id`*.

yaz_errno() should be called after network activity for each server - (after yaz_wait() returns) to determine the success or failure of the last operation (e.g. search).

Parameters

`id`
The connection resource returned by yaz_connect().

Return Values

Returns an error code. The error code is either a Z39.50 diagnostic code (usually a Bib-1 diagnostic) or a client side error code which is generated by PHP/YAZ itself, such as "Connect failed", "Init Rejected", etc.

See Also

- yaz_error()
- yaz_addinfo()

yaz_error

yaz_error -- Returns error description

Description

string **yaz_error** (resource `$id`)

yaz_error() returns an English text message corresponding to the last error number as returned by yaz_errno().

Parameters

`id`
The connection resource returned by yaz_connect().

Return Values

Returns an error text message for server (last request), identified by parameter *`id`*. An empty string is returned if the last operation was successful.

See Also

- yaz_errno()
- yaz_addinfo()

yaz_es_result

yaz_es_result -- Inspects Extended Services Result

Description

array **yaz_es_result** (resource `$id`)

This function inspects the last returned Extended Service result from a server. An Extended Service is initiated by either **yaz_item_order()** or yaz_es().

Parameters

`id`
The connection resource returned by yaz_connect().

Return Values

Returns array with element *targetReference* for the reference for the extended service operation (generated and returned from the server).

See Also

- yaz_es()

yaz_es

yaz_es -- Prepares for an Extended Service Request

Description

void **yaz_es** (resource `$id`, string `$type`, array `$args`)

This function prepares for an Extended Service Request. Extended Services is family of various Z39.50 facilities, such as Record Update, Item Order, Database administration etc.

Note
Many Z39.50 Servers do not support Extended Services.

The yaz_es() creates an Extended Service Request packages and puts it into a queue of operations. Use yaz_wait() to send the request(s) to the server. After completion of yaz_wait() the result of the Extended Service operation should be expected with a call to yaz_es_result().

Parameters

`id`
: The connection resource returned by yaz_connect().

`type`
: A string which represents the type of the Extended Service: *itemorder* (Item Order), *create* (Create Database), *drop* (Drop Database), *commit* (Commit Operation), *update* (Update Record), *xmlupdate* (XML Update). Each type is specified in the following section.

`args`
: An array with extended service options plus package specific options. The options are identical to those offered in the C API of ZOOM C. Refer to the ZOOM » Extended Services.

Return Values

No value is returned.

Examples

Example #5 - Record Update

```
<?php
$con = yaz_connect("myhost/database");
$args = array (
   "record" => "<gils><title>some title</title></gils>",
   "syntax" => "xml",
   "action" => "specialUpdate"
);
yaz_es($con, "update", $args);
yaz_wait();
$result = yaz_es_result($id);
?>
```

See Also

- yaz_es_result()

yaz_get_option

yaz_get_option -- Returns value of option for connection

Description

string **yaz_get_option** (resource `$id`, string `$name`)

Returns the value of the option specified with *`name`*.

Parameters

`id`
: The connection resource returned by yaz_connect().

`name`
: The option name.

Return Values

Returns the value of the specified option or an empty string if the option wasn't set.

See Also

- The description of yaz_set_option() for available options

yaz_hits

yaz_hits -- Returns number of hits for last search

Description

int **yaz_hits** (resource `$id` [, array `&$searchresult`])

yaz_hits() returns the number of hits for the last search.

Parameters

`id`
The connection resource returned by yaz_connect().

`searchresult`
Result array for detailed search result information.

Return Values

Returns the number of hits for the last search or 0 if no search was performed.

The search result array (if supplied) holds information that is returned by a Z39.50 server in the SearchResult-1 format part of a search response. The SearchResult-1 format can be used to obtain information about hit counts for various parts of the query (subquery). In particular, it is possible to obtain hit counts for the individual search terms in a query. Information for first subquery is in $array[0], second subquery in $array[1], and so forth.

searchresult members

Element	Description
id	Sub query ID2 (string)
count	Result count / hits (integer)
subquery.term	Sub query term (string)
interpretation.term	Interpretated sub query term (string)
recommendation.term	Recommended sub query term (string)

Note

The SearchResult facility requires PECL YAZ 1.0.5 or later and YAZ 2.1.9 or later.

Note

Very few Z39.50 implementations support the SearchResult facility.

yaz_itemorder

yaz_itemorder -- Prepares for Z39.50 Item Order with an ILL-Request package

Description

void **yaz_itemorder** (resource `$id`, array `$args`)

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See » this and the » specification.

Parameters

id
: The connection resource returned by yaz_connect().

args
: Must be an associative array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN. The ILL-Request parameters are:

```
protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-pe
rson
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-in
stitution transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier service-date-time,this,date
service-date-time,this,time service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution
responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-instit
ution delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
```

billing-address,postal-address,name-of-person-or-institution,name-of-institu
tion billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service search-type,need-before-date
search-type,expiry-date search-type,expiry-flag place-on-hold
client-id,client-name client-id,client-status client-id,client-identifier
item-id,item-type item-id,call-number item-id,author item-id,title
item-id,sub-title item-id,sponsoring-body item-id,place-of-publication
item-id,publisher item-id,series-title-number item-id,volume-issue
item-id,edition item-id,publication-date
item-id,publication-date-of-component item-id,author-of-article
item-id,title-of-article item-id,pagination item-id,ISBN item-id,ISSN
item-id,additional-no-letters item-id,verification-reference-source
copyright-complicance retry-flag forward-flag requester-note forward-note
There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package: package-name user-id contact-name contact-phone contact-email itemorder-item

Return Values

No value is returned.

yaz_present

yaz_present -- Prepares for retrieval (Z39.50 present)

Description

bool **yaz_present** (resource `$id`)

This function prepares for retrieval of records after a successful search.

The yaz_range() function should be called prior to this function to specify the range of records to be retrieved.

Parameters

`id`
The connection resource returned by yaz_connect().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

yaz_range

yaz_range -- Specifies a range of records to retrieve

Description

void **yaz_range** (resource `$id`, int `$start`, int `$number`)

Specifies a range of records to retrieve.

This function should be called before yaz_search() or yaz_present().

Parameters

`id`
: The connection resource returned by yaz_connect().

`start`
: Specifies the position of the first record to be retrieved. The records numbers goes from 1 to yaz_hits().

`number`
: Specifies the number of records to be retrieved.

Return Values

No value is returned.

yaz_record

yaz_record -- Returns a record

Description

string **yaz_record** (resource `$id`, int `$pos`, string `$type`)

The yaz_record() function inspects a record in the current result set at the position specified by parameter *`pos`*.

Parameters

`id`
The connection resource returned by yaz_connect().

`pos`
The record position. Records positions in a result set are numbered 1, 2, ... $hits where $hits is the count returned by yaz_hits().

`type`
The *`type`* specifies the form of the returned record.

Note
It is the application which is responsible for actually ensuring that the records are returned from the Z39.50/SRW server in the proper format. The type given only specifies a conversion to take place on the client side (in PHP/YAZ).

Besides conversion of the transfer record to a string/array, PHP/YAZ it is also possible to perform a character set conversion of the record. Especially for USMARC/MARC21 that is recommended since these are typically returned in the character set MARC-8 that is not supported by browsers, etc. To specify a conversion, add*; charset= from, to* where *from* is the original character set of the record and *to* is the resulting character set (as seen by PHP).

string
The record is returned as a string for simple display. In this mode, all MARC records are converted to a line-by-line format since ISO2709 is hardly readable. XML records and SUTRS are returned in their original format. GRS-1 are returned in a (ugly) line-by-line format. This format is suitable if records are to be displayed in a quick way - for debugging - or because it is not feasible to perform proper display.

xml
The record is returned as an XML string if possible. In this mode, all MARC records are converted to » MARCXML. XML records and SUTRS are returned in their original format. GRS-1 is not supported. This format is similar to *`string`* except that MARC records are converted to MARCXML This format is suitable if records

are processed by an XML parser or XSLT processor afterwards.

raw
The record is returned as a string in its original form. This type is suitable for MARC, XML and SUTRS. It does not work for GRS-1. MARC records are returned as a ISO2709 string. XML and SUTRS are returned as strings.

syntax
The syntax of the record is returned as a string, i.e. *USmarc*, *GRS-1*, *XML*, etc.

database
The name of database associated with record at the position is returned as a string.

array
The record is returned as an array that reflects the GRS-1 structure. This type is suitable for MARC and GRS-1. XML, SUTRS are not supported and if the actual record is XML or SUTRS an empty string will be returned. The array returned consists of a list corresponding to each leaf/internal node of GRS-1. Each list item consists a sub list with first element *path* and *data* (if data is available). The path which is a string holds a list of each tree component (of the structured GRS-1 record) from root to leaf. Each component is a tag type, tag value pair of the form *(type, value* String tags normally has a corresponding tag type 3. MARC can also be returned as an array (they are converted to GRS-1 internally).

Return Values

Returns the record at position *pos* or an empty string if no record exists at the given position.

If no database record exists at the given position an empty string is returned.

Examples

Example #6 - Array for GRS-1 record

Consider this GRS-1 record:

```
(4,52)Robert M. Pirsig
(4,70)
     (4,90)
           (2,7)Transworld Publishers, ltd.
```

This record has two nodes at root level. First element at root level is (4,52) [tag type 4, tag value 52], and has data *Robert M. Pirsig*. Second element at root level (4,70) has a subtree with a single element (4,90). (4,90) has yet another sub tree (2,7) with data *Transworld Publishers, ltd.*.

If this record is present at position $p, then

```
<?php
```

```
$ar = yaz_record($id, $p, "array");
print_r($ar);

?>
```

will output:

```
Array
(
   [0] => Array
       (
           [0] => (4,52)
           [1] => Robert M. Pirsig
       )
   [1] => Array
       (
           [0] => (4,70)
       )
   [2] => Array
       (
           [0] => (4,70)(4,90)
       )
   [3] => Array
       (
           [0] => (4,70)(4,90)(2,7)
           [1] => Transworld Publishers, ltd.
       )
)
```

Example #7 - Working with MARCXML

The following PHP snippet returns a MARC21/USMARC record as MARCXML. The original record is returned in marc-8 (unknown to most XML parsers), so we convert it to UTF-8 (which all XML parsers must support).

```
<?php
$rec = yaz_record($id, $p, "xml; charset=marc-8,utf-8");
?>
```

The record *$rec* can be processed with the Sablotron XSLT processor as follows:

```
<?php

$xslfile = 'display.xsl';
$processor = xslt_create();
$parms = array('/_xml' => $rec);
$res = xslt_process($processor, 'arg:/_xml', $xslfile, NULL, $parms);
xslt_free($processor);
$res = preg_replace("'</?html[^>]*>'", '', $res);
echo $res;

?>
```

For PHP 5 the XSL extension must be used instead of Sablotron XSLT.

yaz_scan_result

yaz_scan_result -- Returns Scan Response result

Description

array **yaz_scan_result** (resource `$id` [, array `&$result`])

yaz_scan_result() returns terms and associated information as received from the server in the last performed yaz_scan().

Parameters

`id`
The connection resource returned by yaz_connect().

`result`
If given, this array will be modified to hold additional information taken from the Scan Response:

- *number* - Number of entries returned
- *stepsize* - Step size
- *position* - Position of term
- *status* - Scan status

Return Values

Returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where the first item is the term, and the second item is the result-count.

yaz_scan

yaz_scan -- Prepares for a scan

Description

void **yaz_scan** (resource `$id`, string `$type`, string `$startterm` [, array `$flags`])

This function prepares for a Z39.50 Scan Request on the specified connection.

To actually transfer the Scan Request to the server and receive the Scan Response, yaz_wait() must be called. Upon completion of yaz_wait() call yaz_error() and yaz_scan_result() to handle the response.

Parameters

id
: The connection resource returned by yaz_connect().

type
: Currently only type *rpn* is supported.

startterm
: Starting term point for the scan. The form in which the starting term is specified is given by parameter *type*. The syntax this parameter is similar to the RPN query as described in yaz_search(). It consists of zero or more *@attr* -operator specifications, then followed by exactly one token.

flags
: This optional parameter specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags array: *number* (number of terms requested), *position* (preferred position of term) and *stepSize* (preferred step size).

Return Values

No value is returned.

Examples

Example #8 - PHP function that scans titles

```
<?php
function scan_titles($id, $startterm)
{
```

```
 yaz_scan($id, "rpn", "@attr 1=4 " . $startterm);
 yaz_wait();
 $errno = yaz_errno($id);
 if ($errno == 0) {
   $ar = yaz_scan_result($id, &$options);
   echo 'Scan ok; ';
   foreach ($options as $key => $val) {
     echo "$key = $val  ";
   }
   echo '<br /><table>';
   while (list($key, list($k, $term, $tcount)) = each($ar)) {
     if (empty($k)) continue;
     echo "<tr><td>$term</td><td>$tcount</td></tr>";
   }
   echo '</table>';
 } else {
   echo "Scan failed. Error: " . yaz_error($id) . "<br />";
 }
}
?>
```

yaz_schema

yaz_schema -- Specifies schema for retrieval

Description

void **yaz_schema** (resource `$id`, string `$schema`)

yaz_schema() specifies the schema for retrieval.

This function should be called before yaz_search() or yaz_present().

Parameters

`id`
: The connection resource returned by yaz_connect().

`schema`
: Must be specified as an OID (Object Identifier) in a raw dot-notation (like *1.2.840.10003.13.4*) or as one of the known registered schemas: *GILS-schema*, *Holdings*, *Zthes*, ...

Return Values

No value is returned.

yaz_search

yaz_search -- Prepares for a search

Description

bool **yaz_search** (resource `$id`, string `$type`, string `$query`)

yaz_search() prepares for a search on the given connection.

Like yaz_connect() this function is non-blocking and only prepares for a search to be executed later when yaz_wait() is called.

Parameters

`id`
The connection resource returned by yaz_connect().

`type`
This parameter represents the query type - only *"rpn"* is supported now in which case the third argument specifies a Type-1 query in prefix query notation.

`query`
The RPN query is a textual representation of the Type-1 query as defined by the Z39.50 standard. However, in the text representation as used by YAZ a prefix notation is used, that is the operator precedes the operands. The query string is a sequence of tokens where white space is ignored unless surrounded by double quotes. Tokens beginning with an at-character (@) are considered operators, otherwise they are treated as search terms.

RPN Operators

Construct	**Description**
@*and* query1 query2	intersection of query1 and query2
@*or* query1 query2	union of query1 and query2
@*not* query1 query2	query1 and not query2
@*set* name	result set reference
@*attrset* set query	specifies attribute-set for query. This construction is only allowed once - in the beginning of the whole query
@*attr* [set] type=value query	applies attribute to query. The type and value are integers specifying the

	attribute-type and attribute-value respectively. The set, if given, specifies the attribute-set.

You can find information about attributes at the » Z39.50 Maintenance Agency site.

Note

If you would like to use a more friendly notation, use the CCL parser - functions yaz_ccl_conf() and yaz_ccl_parse().

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #9 - Query Examples

You can search for simple terms, like this:
```
computer
```
which matches documents where "computer" occur. No attributes are specified.

The query
```
"knuth donald"
```
matches documents where "knuth donald" occur (provided that the server supports phrase search).

This query applies two attributes for the same phrase.
```
@attr 1=1003 @attr 4=1 "knuth donald"
```
First attribute is type 1 (Bib-1 use), attribute value is 1003 (Author). Second attribute has is type 4 (structure), value 1 (phrase), so this should match documents where Donald Knuth is author.

The query
```
@and @or a b @not @or c d e
```
would in infix notation look like *(a or b) and ((c or d) not e)*.

Another, more complex, one:
```
@attrset gils @and @attr 1=4 art @attr 1=2000 company
```
The query as a whole uses the GILS attributeset. The query matches documents where *art* occur in the title (GILS,BIB-1) and in which *company* occur as Distributor (GILS).

yaz_set_option

yaz_set_option -- Sets one or more options for connection

Description

void **yaz_set_option** (resource `$id`, string `$name`, string `$value`)

void **yaz_set_option** (resource `$id`, array `$options`)

Sets one or more options on the given connection.

Parameters

`id`
The connection resource returned by yaz_connect().

`name` or *`options`*
May be either a string or an array. If given as a string, this will be the name of the option to set. You'll need to give it's *`value`*. If given as an array, this will be an associative array (option name => option value).

PHP/YAZ Connection Options

Name	Description
implementationName	implementation name of server
implementationVersion	implementation version of server
implementationId	implementation ID of server
schema	schema for retrieval. By default, no schema is used. Setting this option is equivalent to using function yaz_schema()
preferredRecordSyntax	record syntax for retrieval. By default, no syntax is used. Setting this option is equivalent to using function yaz_syntax()
start	offset for first record to be retrieved via yaz_search() or yaz_present(). First record is numbered has a start value of 0. Second record has start value 1. Setting this option in combination with option *count* has the same effect as calling yaz_range() except that records are numbered from 1 in yaz_range()

count	maximum number of records to be retrieved via yaz_search() or yaz_present().
elementSetName	element-set-name for retrieval. Setting this option is equivalent to calling yaz_element().

value
: The new value of the option. Use this only if the previous argument is a string.

Return Values

No value is returned.

yaz_sort

yaz_sort -- Sets sorting criteria

Description

void **yaz_sort** (resource `$id`, string `$criteria`)

This function sets sorting criteria and enables Z39.50 Sort.

Call this function *before* yaz_search(). Using this function alone does not have any effect. When used in conjunction with yaz_search(), a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present (yaz_present().

Parameters

`id`
: The connection resource returned by yaz_connect().

`criteria`
: A string that takes the form *field1 flags1 field2 flags2* where field1 specifies the primary attributes for sort, field2 seconds, etc.. The field specifies either a numerical attribute combinations consisting of type=value pairs separated by comma (e.g. *1=4,2=1*) ; or the field may specify a plain string criteria (e.g. *title*. The flags is a sequence of the following characters which may not be separated by any white space. **Sort Flags**

  a
  : Sort ascending

  d
  : Sort descending

  i
  : Case insensitive sorting

  s
  : Case sensitive sorting

Return Values

No value is returned.

Examples

Example #10 - Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you would use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you would use:

```
1=4 ia 1=1003 sa
```

yaz_syntax

yaz_syntax -- Specifies the preferred record syntax for retrieval

Description

void **yaz_syntax** (resource `$id`, string `$syntax`)

yaz_syntax() specifies the preferred record syntax for retrieval

This function should be called before yaz_search() or yaz_present().

Parameters

`id`
The connection resource returned by yaz_connect().

`syntax`
The syntax must be specified as an OID (Object Identifier) in a raw dot-notation (like *1.2.840.10003.5.10*) or as one of the known registered record syntaxes (sutrs, usmarc, grs1, xml, etc.).

Return Values

No value is returned.

yaz_wait

yaz_wait -- Wait for Z39.50 requests to complete

Description

mixed **yaz_wait** ([array `&$options`])

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions yaz_connect(), yaz_search(), yaz_present(), yaz_scan() and yaz_itemorder().

yaz_wait() returns when all servers have either completed all requests or aborted (in case of errors).

Parameters

`options`
: An associative array of options:

timeout
: Sets timeout in seconds. If a server has not responded within the timeout it is considered dead and yaz_wait() returns. The default value for timeout is 15 seconds.

event
: A boolean.

Return Values

Returns **TRUE** on success or **FALSE** on failure. In event mode, returns resource or **FALSE** in case of an error.