mnoGoSearch

Introduction

These functions allow you to access the mnoGoSearch (former UdmSearch) free search engine. mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has a number of unique features, which makes it appropriate for a wide range of applications from search within your site to a specialized search system such as cooking recipes or newspaper search, FTP archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of the indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data is collected by the indexer for later use in a search process. The search is performed via Web interface. C, CGI, PHP and Perl search front ends are included.

More information about mnoGoSearch can be found at » http://www.mnogosearch.org/.

Note
This extension has been moved to the » PECL repository and is no longer bundled with PHP as of PHP 5.1.0.

Note
This extension is not available on Windows platforms.

Installing/Configuring

Requirements

Download mnoGosearch from » http://www.mnogosearch.org/ and install it on your system. You need at least version 3.1.10 of mnoGoSearch installed to use these functions.

Installation

In order to have these functions available, you must compile PHP with mnoGosearch support by using the *--with-mnogosearch* option. If you use this option without specifying the path to mnoGosearch, PHP will look for mnoGosearch under */usr/local/mnogosearch* path by default. If you installed mnoGosearch at a different location you should specify it: *--with-mnogosearch=DIR*.

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/mnogosearch.

Note
PHP contains built-in MySQL access library, which can be used to access MySQL. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic MySQL libraries. Thus, if you use mnoGoSearch with MySQL, during PHP configuration you have to indicate the directory of your MySQL installation, that was used during mnoGoSearch configuration, i.e. for example: *--with-mnogosearch --with-mysql=/usr*.

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

UDM_FIELD_URLID (integer)

UDM_FIELD_URL (integer)

UDM_FIELD_CONTENT (integer)

UDM_FIELD_TITLE (integer)

UDM_FIELD_KEYWORDS (integer)

UDM_FIELD_DESC (integer)

UDM_FIELD_DESCRIPTION (integer)

UDM_FIELD_TEXT (integer)

UDM_FIELD_SIZE (integer)

UDM_FIELD_RATING (integer)

UDM_FIELD_SCORE (integer)

UDM_FIELD_MODIFIED (integer)

UDM_FIELD_ORDER (integer)

UDM_FIELD_CRC (integer)

UDM_FIELD_CATEGORY (integer)

UDM_FIELD_LANG (integer)

UDM_FIELD_CHARSET (integer)

UDM_PARAM_PAGE_SIZE (integer)

UDM_PARAM_PAGE_NUM (integer)

UDM_PARAM_SEARCH_MODE (integer)

UDM_PARAM_CACHE_MODE (integer)

UDM_PARAM_TRACK_MODE (integer)

UDM_PARAM_PHRASE_MODE (integer)

UDM_PARAM_CHARSET (integer)

UDM_PARAM_LOCAL_CHARSET (integer)

UDM_PARAM_BROWSER_CHARSET (integer)

UDM_PARAM_STOPTABLE (integer)

UDM_PARAM_STOP_TABLE (integer)

UDM_PARAM_STOPFILE (integer)

UDM_PARAM_STOP_FILE (integer)

UDM_PARAM_WEIGHT_FACTOR (integer)

UDM_PARAM_WORD_MATCH (integer)

UDM_PARAM_MAX_WORD_LEN (integer)

UDM_PARAM_MAX_WORDLEN (integer)

UDM_PARAM_MIN_WORD_LEN (integer)

UDM_PARAM_MIN_WORDLEN (integer)

UDM_PARAM_ISPELL_PREFIXES (integer)

UDM_PARAM_ISPELL_PREFIX (integer)

UDM_PARAM_PREFIXES (integer)

UDM_PARAM_PREFIX (integer)

UDM_PARAM_CROSS_WORDS (integer)

UDM_PARAM_CROSSWORDS (integer)

UDM_PARAM_VARDIR (integer)

UDM_PARAM_DATADIR (integer)

UDM_PARAM_HLBEG (integer)

UDM_PARAM_HLEND (integer)

UDM_PARAM_SYNONYM (integer)

UDM_PARAM_SEARCHD (integer)

UDM_PARAM_QSTRING (integer)

UDM_PARAM_REMOTE_ADDR (integer)

UDM_LIMIT_CAT (integer)

UDM_LIMIT_URL (integer)

UDM_LIMIT_TAG (integer)

UDM_LIMIT_LANG (integer)

UDM_LIMIT_DATE (integer)

UDM_PARAM_FOUND (integer)

UDM_PARAM_NUM_ROWS (integer)

UDM_PARAM_WORDINFO (integer)

UDM_PARAM_WORD_INFO (integer)

UDM_PARAM_SEARCHTIME (integer)

UDM_PARAM_SEARCH_TIME (integer)

UDM_PARAM_FIRST_DOC (integer)

UDM_PARAM_LAST_DOC (integer)

UDM_MODE_ALL (integer)

UDM_MODE_ANY (integer)

UDM_MODE_BOOL (integer)

UDM_MODE_PHRASE (integer)

UDM_CACHE_ENABLED (integer)

UDM_CACHE_DISABLED (integer)

UDM_TRACK_ENABLED (integer)

UDM_TRACK_DISABLED (integer)

UDM_PHRASE_ENABLED (integer)

UDM_PHRASE_DISABLED (integer)

UDM_CROSS_WORDS_ENABLED (integer)

UDM_CROSSWORDS_ENABLED (integer)

UDM_CROSS_WORDS_DISABLED (integer)

UDM_CROSSWORDS_DISABLED (integer)

UDM_PREFIXES_ENABLED (integer)

UDM_PREFIX_ENABLED (integer)

UDM_ISPELL_PREFIXES_ENABLED (integer)

UDM_ISPELL_PREFIX_ENABLED (integer)

UDM_PREFIXES_DISABLED (integer)

UDM_PREFIX_DISABLED (integer)

UDM_ISPELL_PREFIXES_DISABLED (integer)

UDM_ISPELL_PREFIX_DISABLED (integer)

UDM_ISPELL_TYPE_AFFIX (integer)

UDM_ISPELL_TYPE_SPELL (integer)

UDM_ISPELL_TYPE_DB (integer)

UDM_ISPELL_TYPE_SERVER (integer)

UDM_MATCH_WORD (integer)

UDM_MATCH_BEGIN (integer)

UDM_MATCH_SUBSTR (integer)

UDM_MATCH_END (integer)

mnoGoSearch Functions

udm_add_search_limit

udm_add_search_limit -- Add various search limits

Description

bool **udm_add_search_limit** (resource `$agent`, int `$var`, string `$val`)

udm_add_search_limit() adds search restrictions.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

`var`
Defines the parameter, indicating limits. Possible *`var`* values:

- **UDM_LIMIT_URL** - defines document URL limitations to limit the search through subsection of the database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://www.example.___/catalog may stand for http://www.example.com/catalog and http://www.example.net/catalog.
- **UDM_LIMIT_TAG** - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- **UDM_LIMIT_LANG** - defines document language limitations.
- **UDM_LIMIT_CAT** - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then it's id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So it's id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the URL.
- **UDM_LIMIT_DATE** - defines limitation by date the document was modified. Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

Example #1

```
<?php
udm_add_search_limit($udm, UDM_LIMIT_DATE, "&lt;908012006");
?>
```

If > character is used, then the search will be restricted to those documents having a modification date greater than entered, if <, then smaller.

val
Defines the value of the current parameter.

Return Values

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

udm_alloc_agent_array

udm_alloc_agent_array -- Allocate mnoGoSearch session

Description

resource **udm_alloc_agent_array** (array `$databases`)

udm_alloc_agent_array() will create an agent with multiple database connections.

Parameters

`databases`
: The array *`databases`* must contain one database URL per element, analog to the first parameter of udm_alloc_agent().

Return Values

Returns a resource link identifier on success, or **FALSE** on failure.

See Also

- udm_alloc_agent()

udm_alloc_agent

udm_alloc_agent -- Allocate mnoGoSearch session

Description

resource **udm_alloc_agent** (string `$dbaddr` [, string `$dbmode`])

Allocate a mnoGoSearch session.

Parameters

`dbaddr`
: *`dbaddr`* - URL-style database description, with options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format for *`dbaddr`*: *DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/*. Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, and ibase. Actually, it does not matter for native libraries support, but ODBC users should specify one of the supported values. If your database type is not supported, you may use *unknown* instead.

`dbmode`
: *`dbmode`* - You may select the SQL database mode of words storage. Possible values of *`dbmode`* are: *single*, *multi*, *crc*, or *crc-multi*. When *single* is specified, all words are stored in the same table. If *multi* is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster, but requires more tables in the database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorithm instead of words. This mode requires less disk space and it is faster comparing with "single" and "multi" modes. *crc-multi* uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like in "multi" mode.

Note
`dbaddr` and *`dbmode`* must match those used during indexing.

Return Values

Returns a mnogosearch agent identifier on success, **FALSE** on failure. This function creates a session with database parameters.

Notes

Note
In fact this function does not open a connection to the database and thus does not check the entered login and password. Establishing a connection to the database and login/password verification is done by udm_find().

udm_api_version

udm_api_version -- Get mnoGoSearch API version

Description

int **udm_api_version** (void)

Gets the mnoGoSearch API version.

This function allows the user to identify which API functions are available, e.g. udm_get_doc_count() function is only available in mnoGoSearch 3.1.11 or later.

Return Values

udm_api_version() returns the mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return *30110*.

Examples

Example #2 - udm_api_version() example

```
<?php
if (udm_api_version() >= 30111) {
   echo  "Total number of URLs in database: " . udm_get_doc_count($udm) .
"<br />\n";
}
?>
```

udm_cat_list

udm_cat_list -- Get all the categories on the same level with the current one

Description

array **udm_cat_list** (resource `$agent`, string `$category`)

Gets all the categories on the same level with the current one.

The function can be useful for developing categories tree browser.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

`category`

Return Values

Returns an array listing all categories of the same level as the current *`category`* in the categories tree.

The returned array consists of pairs. Elements with even index numbers contain the category paths, odd elements contain the corresponding category names.

```
$array[0] will contain '020300'
 $array[1] will contain 'Audi'
 $array[2] will contain '020301'
 $array[3] will contain 'BMW'
 $array[4] will contain '020302'
 $array[5] will contain 'Opel'
 ...
etc.
```

Examples

Following is an example of displaying links of the current level in format:

```
Audi
 BMW
 Opel
 ...
```

Example #3 - udm_cat_list() example

```
<?php
$cat_list_arr = udm_cat_list($udm_agent, $cat);
$cat_list = '';
for ($i=0; $i<count($cat_list_arr); $i+=2) {
   $path = $cat_list_arr[$i];
   $name = $cat_list_arr[$i+1];
   $cat_list .= "<a href=\"$_SERVER[PHP_SELF]?cat=$path\">$name</a><br />";
}
?>
```

See Also

- udm_cat_path()

udm_cat_path

udm_cat_path -- Get the path to the current category

Description

array **udm_cat_path** (resource `$agent`, string `$category`)

Returns an array describing the path in the categories tree from the tree root to the current one, specified by *`category`*.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

`category`

Return Values

The returned array consists of pairs. Elements with even index numbers contain the category paths, odd elements contain the corresponding category names.

For example, the call *$array=udm_cat_path($agent, '02031D');* may return the following array:

```
$array[0] will contain ''
$array[1] will contain 'Root'
$array[2] will contain '02'
$array[3] will contain 'Sport'
$array[4] will contain '0203'
$array[5] will contain 'Auto'
$array[4] will contain '02031D'
$array[5] will contain 'Ferrari'
```

Examples

Example #4 - Specifying path to the current category in the following format: '> Root > Sport > Auto > Ferrari'

```
<?php
 $cat_path_arr = udm_cat_path($udm_agent, $cat);
 $cat_path = '';
 for ($i=0; $i<count($cat_path_arr); $i+=2) {
   $path = $cat_path_arr[$i];
   $name = $cat_path_arr[$i+1];
```

```
    $cat_path .= " > <a href=\"$_SERVER[PHP_SELF]?cat=$path\">$name</a> ";
  }
?>
```

See Also

- udm_cat_list()

udm_check_charset

udm_check_charset -- Check if the given charset is known to mnogosearch

Description

bool **udm_check_charset** (resource `$agent`, string `$charset`)

Warning
This function is currently not documented; only its argument list is available.

udm_check_stored

udm_check_stored -- Check connection to stored

Description

int **udm_check_stored** (resource `$agent`, int `$link`, string `$doc_id`)

Warning
This function is currently not documented; only its argument list is available.

udm_clear_search_limits

udm_clear_search_limits -- Clear all mnoGoSearch search restrictions

Description

bool **udm_clear_search_limits** (resource `$agent`)

udm_clear_search_limits() resets defined search limitations.

Parameters

`agent`
: A link to Agent, received after call to udm_alloc_agent().

Return Values

Returns **TRUE**.

See Also

- udm_add_search_limit()

udm_close_stored

udm_close_stored -- Close connection to stored

Description

int **udm_close_stored** (resource `$agent`, int `$link`)

Warning
This function is currently not documented; only its argument list is available.

udm_crc32

udm_crc32 -- Return CRC32 checksum of given string

Description

int **udm_crc32** (resource `$agent`, string `$str`)

Warning
This function is currently not documented; only its argument list is available.

udm_errno

udm_errno -- Get mnoGoSearch error number

Description

int **udm_errno** (resource `$agent`)

Receiving numeric agent error code.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

Return Values

Returns the mnoGoSearch error number, zero if no error.

udm_error

udm_error -- Get mnoGoSearch error message

Description

string **udm_error** (resource `$agent`)

Gets the agent error message.

Parameters

`agent`
: A link to Agent, received after call to udm_alloc_agent().

Return Values

udm_error() returns mnoGoSearch error message, empty string if no error.

udm_find

udm_find -- Perform search

Description

resource **udm_find** (resource `$agent`, string `$query`)

Performs a search.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

Parameters

`agent`
: A link to Agent, received after call to udm_alloc_agent().

`query`
: mnoGoSearch understands the following boolean operators: & - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc". | - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc". ~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing! () - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

Return Values

Returns a result link identifier on success, or **FALSE** on failure.

udm_free_agent

udm_free_agent -- Free mnoGoSearch session

Description

int **udm_free_agent** (resource `$agent`)

Freeing up memory allocated for agent session.

Parameters

`agent`
: A link to Agent, received after call to udm_alloc_agent().

Return Values

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

udm_free_ispell_data

udm_free_ispell_data -- Free memory allocated for ispell data

Description

bool **udm_free_ispell_data** (int `$agent`)

Frees the memory allocated for ispell data.

Parameters

`agent`
: A link to Agent, received after call to udm_alloc_agent().

Return Values

udm_free_ispell_data() always returns **TRUE**.

Notes

Note
This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res

udm_free_res -- Free mnoGoSearch result

Description

bool **udm_free_res** (resource `$res`)

Freeing up memory allocated for results.

Parameters

`res`
: A link to a result identifier, received after call to udm_find().

Return Values

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

udm_get_doc_count

udm_get_doc_count -- Get total number of documents in database

Description

int **udm_get_doc_count** (resource `$agent`)

udm_get_doc_count() returns the number of documents in the database.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

Return Values

Returns the number of document.

Notes

Note
This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field

udm_get_res_field -- Fetch a result field

Description

string **udm_get_res_field** (resource `$res`, int `$row`, int `$field`)

Fetch a mnoGoSearch result field.

Parameters

`res`
`res` - a link to result identifier, received after call to udm_find().

`row`
`row` - the number of the link on the current page. May have values from 0 to *`UDM_PARAM_NUM_ROWS-1`*.

`field`
`field` - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_CATEGORY - document category field. Use udm_cat_path() to get full path to current category from the categories tree root. (This parameter is available only in PHP 4.0.6 or later).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.
- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

Return Values

udm_get_res_field() returns result field value on success, **FALSE** on error.

udm_get_res_param

udm_get_res_param -- Get mnoGoSearch result parameters

Description

string **udm_get_res_param** (resource `$res`, int `$param`)

Gets the mnoGoSearch result parameters.

Parameters

`res`
: *`res`* - a link to result identifier, received after call to udm_find().

`param`
: *`param`* - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

Return Values

udm_get_res_param() returns result parameter value on success, **FALSE** on error.

udm_hash32

udm_hash32 -- Return Hash32 checksum of gived string

Description

int **udm_hash32** (resource `$agent`, string `$str`)

udm_hash32() will take a string *`str`* and return a quite unique 32-bit hash number from it.

Parameters

`agent`
: A link to Agent, received after call to udm_alloc_agent().

`str`
: The input string.

Return Values

Returns a 32-bit hash number.

See Also

- udm_alloc_agent()

udm_load_ispell_data

udm_load_ispell_data -- Load ispell data

Description

bool **udm_load_ispell_data** (resource `$agent`, int `$var`, string `$val1`, string `$val2`, int `$flag`)

udm_load_ispell_data() loads ispell data.

After using this function to free memory allocated for ispell data, please use udm_free_ispell_data(), even if you use UDM_ISPELL_TYPE_SERVER mode.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

`var`
Indicates the source for ispell data. May have the following values:

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *`val1`* and *`val2`* are ignored and should be left blank. *`flag`* should be equal to *1*.

Note
`flag` indicates that after loading ispell data from defined source it should be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to udm_load_ispell_data(), and there is no sense to sort data after every call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value *1*. In this mode in case of error, e.g. if ispell tables are absent, the function will return **FALSE** and code and error message will be accessible through udm_error() and udm_errno().

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *`val1`* defines double letter language code for which affixes are loaded, and *`val2`* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return **FALSE**, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *`flag`* parameter description in UDM_ISPELL_TYPE_DB.

Example #5 - udm_load_ispell_data() example

```
<?php
if ((! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'en',
'/opt/ispell/en.aff', 0)) ||
   (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru',
'/opt/ispell/ru.aff', 0)) ||
   (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'en',
'/opt/ispell/en.dict', 0)) ||
   (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'ru',
'/opt/ispell/ru.dict', 1))) {
   exit;
}
?>
```

Note

flag is equal to *1* only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return **FALSE**, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

```
<?php
if ((! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'en',
'/opt/ispell/en.aff', 0)) ||
  (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_AFFIX, 'ru',
'/opt/ispell/ru.aff', 0)) ||
  (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'en',
'/opt/ispell/en.dict', 0)) ||
  (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SPELL, 'ru',
'/opt/ispell/ru.dict', 1))) {
exit;
}
?>
```

Note

flag is equal to *1* only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2* ` is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted. Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client (search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data). udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by udm_find() function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

Note
This function is available in mnoGoSearch 3.1.12 or later.

  Example:

```
<?php
if (!udm_load_ispell_data($udm, UDM_ISPELL_TYPE_SERVER, '', '', 1)) {
   echo "Error loading ispell data from server<br />\n";
   exit;
}
?>
```

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is **TRUE** for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

val1

val2

flag

Return Values

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

Examples

Example #6 - udm_load_ispell_data() **example**

```
<?php
if (! udm_load_ispell_data($udm, UDM_ISPELL_TYPE_DB, '', '', 1)) {
 printf("Error #%d: '%s'\n", udm_errno($udm), udm_error($udm));
 exit;
}
?>
```

udm_open_stored

udm_open_stored -- Open connection to stored

Description

int **udm_open_stored** (resource `$agent`, string `$storedaddr`)

Warning
This function is currently not documented; only its argument list is available.

udm_set_agent_param

udm_set_agent_param -- Set mnoGoSearch agent session parameters

Description

bool **udm_set_agent_param** (resource `$agent`, int `$var`, string `$val`)

Defines mnoGoSearch session parameters.

Parameters

`agent`
A link to Agent, received after call to udm_alloc_agent().

`var`
The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See udm_find() for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current Unix timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well. Examples of phrase search: "Arizona

desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...
- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the PHP script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let's imagine that we have these weights: `URLWeight 1 BodyWeight 2 TitleWeight 4 KeywordWeight 8 DescWeight 16` As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 aggregate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight. This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration: `E is a factor for weight 1 (URL Weight bit) D is a factor for weight 2 (BodyWeight bit) C is a factor for weight 4 (TitleWeight bit) B is a factor for weight 8 (KeywordWeight bit) A is a factor for weight 16 (DescWeight bit)` Examples:
  UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.
  UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.
  UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.
  UDM_PARAM_WEIGHT_FACTOR=F9421 will search through: `Description with factor 15 (F hex) Keywords with factor 9 Title with factor 4 Body with factor 2 URL with factor 1` If UDM_PARAM_WEIGHT_FACTOR variable is omitted, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.
- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values: UDM_MATCH_BEGIN - word beginning match; UDM_MATCH_END - word ending match; UDM_MATCH_WORD - whole word match; UDM_MATCH_SUBSTR - word substring match.
- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.
- UDM_PARAM_ISPELL_PREFIXES - Possible values:

UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site's spelling checking purposes. In order to enable ispell, you have to load ispell data with udm_load_ispell_data().

- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED. The crosswords feature allows to assign words between and also to a document this link leads to. It works in SQL database mode and is not supported in built-in database and Cachemode.
- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default */var* directory of mnoGoSearch installation is used. Can have only string values.

val

ChangeLog

Version	Description
4.1.0	**UDM_PARAM_VARDIR** was added.

Return Values

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

Notes

Note
Crosswords are supported only in mnoGoSearch 3.1.11 or later.