eZ Find Extension

Documentation

version 2.0.0

Table of Contents

1. Introduction

eZ Find is a search extension for eZ Publish, providing more functionality and better results than the default search in eZ Publish. This manual is a guide to installing, configuring and using eZ Find.

eZ Find enables site visitors to quickly and easily locate information on eZ Publish sites by providing relevant search results. High scalability and performance ensures that the eZ Find search engine can support enterprise-level sites.

eZ Find is a certified extension that integrates smoothly with all eZ Publish business solutions (eZ Publish On Demand, eZ Publish Now and eZ Publish Premium). With eZ Publish Premium, eZ Find can be further customized to meet specific requirements and site structures.

1.1. Target audience

This manual describes how to install, configure and use the eZ Find search extension, and thus is appropriate for system administrators and end users of the extension.

1.2. Conventions

- Code samples, functions, variable names, and so on are printed in `monospace font.`

- Filenames and paths are printed in *`monospace italic font.`*

- Commands are printed in **`monospace bold font`**.

- Elements of graphical user interfaces (such as buttons and field labels) are printed in **bold font**.

- Component names (such as an application) are capitalized, for example "Administration Interface".

- In sample URLs, replace "example.com" with the domain name of your site.

- The screenshots in this document might have been modified to fit the page or to illustrate a point, and therefore might not exactly match the display on your site.

1.3. More resources

For assistance with eZ Publish, refer to the following resources:

- **eZ Publish documentation**: eZ Find is an extension to eZ Publish. Where appropriate, there are links in this document to the online versions of the eZ Publish documentation, located at http://ez.no/doc.
- **eZ Publish forums**: The forums on the eZ Systems website are a valuable community-driven resource, where eZ Publish users provide assistance and support to each other. Accessing the forums is free. The forums are located at http://ez.no/community/forum.
- **Support from eZ Partners**: eZ's global network of partners provides professional assistance for all eZ products. To find a partner, contact sales@ez.no.
- **Other eZ solutions**: For information about other solutions provided by eZ Systems,

refer to http://ez.no/products/solutions.

- **Training and certification**: eZ Systems and eZ Partners offer training courses and certifications for eZ Publish. Contact sales@ez.no or visit http://ez.no/services/training for more information.
- **Sending feedback** about eZ Find : http://issues.ez.no/ezfind

1.4. Contacting eZ

For non-technical questions regarding eZ Publish or eZ Systems, please contact us:

- http://ez.no/company/contact

- info@ez.no

1.5. Copyright and trademarks

Copyright © 2008 eZ Systems AS. Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Other product and company names mentioned in this manual may be the trademarks of their respective owners. We use trademark names in an editorial fashion to the benefit of the trademark holder; therefore, these names are not marked with trademark symbols. All terms known to be trademarks have been appropriately capitalized. We cannot attest to the accuracy of this usage, and usage of a term in this book should not be regarded as affecting the validity of any trademark or servicemark.

2. How does eZ Find work?

eZ Find is an extension that provides enhanced search functionality for eZ Publish sites. It uses the Open Source enterprise search server Solr, which runs on the Lucene Java search library. While the default search function in eZ Publish stores the search index in the database, Solr uses its own highly optimized file for storing its index.

eZ Find uses Solr to handle indexing and searching with an eZ Publish-specific set of configuration files. While indexing, eZ Find uses this specific configuration to do index and query-time analysis of text and keywords. When users perform searches, eZ Find generates a Solr search query based on the user input that is submitted to the Solr back end. All communication with Solr is via Web services (using the REST model).

3. Installation

The extension requires the Java Runtime Environment (JRE) version 5. (JRE version 6 is unstable with eZ Find). This can be obtained from the Java download pages. Follow the installation instructions for JRE as described on the Java home pages.

After JRE is installed, download the eZ Find extension. The extension can be downloaded from the eZ website at http://ez.no/ezfind. Save the downloaded file to the root of your eZ Publish installation, then extract the files.

3.1. Very first things to do

You will need to add a table in the database used by your eZ Publish instance. You can do so as follows (from eZ Publish's root directory), in the case you are using MySQL:

```
mysql -u <user> -p <database_name> <
extension/ezfind/sql/ezfind_elevate_configuration.mysql.sql
```

The procedure is very similar in case you are using another RDMS. Please refer to the documentation reference for your DBMS if you are experiencing issues.

3.2. Starting Solr

First, navigate to the `extension/ezfind/java` directory.

Start Solr by running the command:

```
java -Dezfind -jar start.jar
```

(Make sure the user executing the Java program has write access to the folders `extension/ezfind/java/solr/data` and `extension/ezfind/java/logs`).

This will start the Solr application in the bundled servlet container (Jetty). This application must be running for the indexing and search operations to work. If you are working directly on the web server, you can verify that Solr is running by accessing the URL http://localhost:8983/solr/admin/, or replace "localhost" in the URL with the IP address of the server.

The "-Dezfind" option makes it easier to identify the eZ Find Java process.

If the Java process dies during indexing or searching, the memory limits ("heap space") need to be increased. Specify the heap space as startup parameters, for example:

```
java –Dezfind –Xms512M –Xmx512M –jar start.jar
```

This will allocate a heap space of 512 MB.

3.2.1. High performance, large sites

For large volumes and traffic (search takes about 30% of ez.no page requests), you need to setup Solr in simalar ways as the database engine (64 bit OS, 64 bit application/service).

Just like MySQL needs to scale up its InnoDB pool size beyond 2GB, so does eZ Find/Solr, though usually it is fine with less RAM

It is all about Java VM parameters

If you don't have a 64 bit OS for your server, try the following:

```
java –server –Xmx600m –Xms600m –XX:+UseParallelGC –XX:
+AggressiveOpts –XX:NewRatio=5 –jar start.jar
```

If you do have a 64 bit OS, use a 64 bit Sun JRE (1.5 for Linux, 1.6 for Solaris/Windows/Mac OS X)

```
java –server –d64 –Xmx600m –Xms600m –XX:+UseParallelGC –XX:
+AggressiveOpts –XX:NewRatio=5 –jar start.jar
```

3.2.2. Starting Solr as a service

eZ Find provides scripts for running eZ Find as a service on Linux platforms. The supported platforms are Debian and RedHat based distributions.

4.1.1.1 Redhat based systems

Copy the file *extension/ezfind/bin/scripts/redhat/solr* to your /etc/init.d folder. Edit the file, and configure the required variables:

SOLR_HOME

Set this variable to the java folder of the ezfind extension you want to run as a service.
Example: *SOLR_HOME=/var/www/ezpublish/extension/ezfind/java*
JAVA_HOME

If your java executable is not located in /usr/bin/java or /usr/local/bin/java, and the JAVA_HOME environnement variable is not set, you can specify the path to your java folder here.
Example: JAVA_HOME=/path/to/javafolder

4.1.1.2 Debian based systems

Copy the file *extension/ezfind/bin/scripts/debian/solr* to your */etc/init.d* folder.

You can then use update-rc.d to set solr as auto-start in the required run levels:
$ update-rc.d solr defaults

Finally, edit the file, and configure the required variables:

SOLR_HOME

Set this variable to the java folder of the ezfind extension you want to run as a service.
Example: SOLR_HOME=*/var/www/ezpublish/extension/ezfind/java*.

4. Configuration

4.1   Indexing multiple sites

eZ Find can index multiple eZ Publish installations using the same instance of Solr. This is enabled by default. All indexed content objects are associated with a unique installation key. The installation key is used to identify the installation where the content originates. The installation key is stored in the table `ezsite_data`, with the name `ezfind_site_id`.

To specify that content from one installation should be included in the search index on other installations, use the option `IndexPubliclyAvailable` set in *extension/ezfind/settings/ezfind.ini.append.php*. Only content that is accessible to anonymous users will indexed on other eZ Publish installations.

```
IndexPubliclyAvailable=enabled
```

This value is set to `enabled` by default.

4.2   Searching multiple sites

As described above, it is possible to search for content on several eZ Publish installations simultaneously. To show results from multiple installations, check the configuration option `SearchOtherInstallations` in *extension/ezfind/settings/ezfind.ini.append.php*.

```
[SiteSettings]
SearchOtherInstallations=enabled
```

4.3   Enabling eZ Find

To enable the extension, open *settings/override/site.ini.append.php*, and add the following parameter in the `[ExtensionSettings]` block:

```
ActiveExtensions[]=ezfind
```

To use the correct templates for the `ezwebin` (Website Interface) extension, the eZ Find extensions must be enabled before the `ezwebin` extension.

4.4 More configuration options

4.4.1 SearchHandler

The default search handler can be configured by default in ezfind.ini or specified with the dedicated ezfind template search function

Setting	Description
standard	The Solr standard handler is called with all syntax supported, searching is done against all searcheable fields
simplestandard	the Solr standard handler is called with all all syntax supported, searching is done against the aggregated field ezf_df_text
ezpublish	the recommended handler (Solr dismax based) for typical user searches using keywords without boolean or other operators (prefixes + and – are supported)
heuristic	depending on the presence of special characters indicating boolean, wildcard or fuzzy expressions, either the simplestandard or dismax handler is called

Default:

   DefaultSearchHandler=heuristic

4.4.2 Index time boosting

Optionally, you can specify boost factors during indexing. When the backend calculates the relevancy scores, the boost factors are taken into account

See the inline comments in ezfind.ini for more

4.4.3 DelayedIndexing

Indexing content in solr can be a time consuming operation, and can impact publishing time depending on the solr index site.

It is possible to delay content indexing by enabling *[SearchSettings].DelayedIndexing* in *site.ini* (global override). The indexing operations will be queued for deferred handling.

To have the objects actually indexed, you need to enable two cronjobs: ezfindexcontent, and ezfoptimizeindex. Refer to the official eZ publish documentation [1] to find out how to configure eZ publish cronjobs.

4.4.3.1 Cronjob: ezfindexcontent

This cronjob has to be executed frequently. Objects published or modified between two executions of ezfindexcontent won't be returned or up-to-date in search results. Running it every five minutes, or even every minute, is a good option.

Suggested frequency: frequent (every X minutes)

---

1 http://ez.no/doc/ez_publish/technical_manual/4_0/features/cronjobs/running_cronjobs

Usage: php runcronjobs.php -s <siteaccess> ezfindexcontent

4.4.3.2 Cronjob: ezfoptimizeindex

This cronjob will optimize the solr index so that solr handles search queries faster. It doesn't have to be executed very frequently, as optimizing is a heavy operation.

Suggested frequency: infrequent (once or twice a day, or every X hours if content publishing is really frequent)

Usage: php runcronjobs.php -s <siteaccess> ezfoptimizeindex

4.4.3.3 OptimizeOnCommit

If *DelayedIndexing* is enabled, *OptimizeOnCommit* should be disabled in order to avoid useless optimization calls on commit during content indexing.

This setting can be found in *ezfind.ini*.

5. Indexing

5.1.   Updating the search index

To add content to the search engine index, run the *updatesearchindexsolr.php* script using the administration siteaccess as the specified siteaccess. For example:

```
php extension/ezfind/bin/php/updatesearchindexsolr.php -s <admin
siteaccess> --php-exec=php --conc=2
```

The indexing process typically indexes at a rate of 5 to 15 objects per second. However, if you are using external filters to convert binary files to plain text, this may add more processing time.

The `--php-exec` parameter must specify the path to the PHP executable used. This parameter is used to start sub-processes in the indexing operation. This is done to prevent internal memory and reference problems.

The `--conc` parameter specifies how many concurrent processes should be used to index the content. This number should be equal to the number of processor cores on the server.

An optional parameter is `--clean` which will delete all existing entries (and also the spell check index terms in the default configuration)

6. Customization

6.1.   Facets

Facets[1] were introduced in eZ Find 2.0, and enable you to create browse-based search functionality. In other words, the search results can be further narrowed after the original search has been made. This is sometimes called "drilling down" into the results. Facets

---

1 http://www.searchtools.com/info/faceted-metadata.html

refer to the characteristics of the content objects in the results, such as their class, author, and translation. The default templates provide some simple facet examples, although eZ Find is capable of providing much more functionality.

To use facets, you must customize the search templates. To make this easier, the template operators `facetParameters` and `filterParameters` have been added, which provide default parameters to implement facet functionality. Facet-related parameters are described in more detailed in the next section about the eZ Find fetch function.

6.2. Template fetch functions

New in eZ Find 2.0 are dedicated template fetch functions:

`fetch(ezfind, search, hash(<parameters>))` that returns eZ Find search results exposing the powerful features of the backend Solr;

`fetch(ezfind, moreLikeThis, hash(<paramerters>))` that will find related content with heuristic techniques;

`fetch(ezfind, rawSolrRequest, hash(<parameters>))` which allows for "raw" Solr requests (not for normal use, but for example to search "foreign" Solr or Lucene indexes).

6.2.1. Fetch function search

The available parameters are:

Name	Type	Description	Required
query	String	Search query string	No

offset	Integer	Result offset	No
limit	Integer	Result count limit	No
sort_by	Array	Sort definition	No
facet	Array	Facet query definition	No
filter	Mixed	Search filter, independent from ranking	No
class_id	Mixed	Class ID limitation	No
subtree_array	Array	List of subtree limitations	No
section_id	Integer	Section filter	No
ignore_visibility	Boolean	Visibility filter	No
limitation	Array	Override of the current user's access array to the 'read' function of the 'content' module.	No
as_objects	Boolean	*Not implemented yet*	No
spell_check	Array	Configure the spellcheck behaviour	No
query_handler	String	Which search handler to use	No

`query`

The `query` parameter can contain one or multiple search terms. The query is used to rank and limit the search results.

For information about standard Solr query syntax, see:

http://wiki.apache.org/solr/SolrQuerySyntax and
http://lucene.apache.org/java/docs/queryparsersyntax.html

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems' ) )
```

Returns:

Search result with documents containing the words "ez" and "systems".

Example:

```
fetch( ezfind, search, hash( query, '"eZ Systems"' ) )
```

Returns:

Search result with documents containing the term "ez systems".

`offset`

Search result offset. The default value is "0".

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems',
                        offset, 20 ) )
```

Returns:

eZ Systems

Search result containing the words "ez" and "systems", starting from the 20th result.

`limit`

Search result count limitation. The default value is "10".

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems',
                             offset, 0,
                             limit, 25 ) )
```

Returns:

Search result with the first 25 results that contain the words "ez" and "systems".

`sort_by`

The `sort_by` parameter is used to define the sort order of the search result. It supports the following options:

Key	Description
relevance	Default option. Sorts the result by Solr internal relevancy calculations[1].
score	Alias to "relevance"
<class attribute>	Content class attribute, following the syntax "<class_identifier>/<class_attribute>[/<sub_structure]"
modified	Modified time
published	Published time
author	Author name
class_name	Content class name
class_id	Content class identifier or content class ID
name	Content object name
path	Node location path
section_id	Section ID

All sort keys can be used to sort in ascending ("asc") or descending ("desc") order. It is also possible to specify multiple sort options in the same fetch function.

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems',
                        sort_by, hash( class_name, asc,
                                    published, desc ) ) )
```

Returns:

All documents containing the words "ez" and "systems", sorted by the content class name in ascending order, then by the published time in descending order.

---

1   http://lists.tartarus.org/pipermail/xapian-discuss/2004-November/000571.html

`<class_attribute>`

Sorting can be done based on a content class attribute field, specified by its ID number or identifier. If the content class attribute datatype extends `ezfSolrDocumentFieldBase`, the sub-structure can be used as well.

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems',
                             sort_by, hash( 'article/title', 'asc'
) ) )
```

Returns:

All documents containing the words "ez" and "systems", sorted by the article title in ascending order.

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems',
                             sort_by,
hash( 'article/options/opt1', 'asc' ) ) )
```

Returns:

All documents containing the words "ez" and "systems", sorted by the "opt1" part of the article options.

Example:

```
fetch( ezfind, search, hash( query, 'eZ Systems',
                             sort_by, hash( 234, 'asc' ) ) )
```

Returns:

All documents containing the words "ez" and "systems", sorted by the content class attribute with an ID of "234", in ascending order.

`facet`

The `facet` parameter is used to define the facet query that should be performed. The results include information about the facets and facet groups relevant to the current search and are returned in addition to the normal query results. It is possible to perform multiple facet queries in one fetch request.

The following facet options are available:

Option	Description
field	The object characteristic that will serve as the facet. This can be a field, specified using the syntax "<class_identifier>/<class_attribute>[/<sub_structure]" The sub-structure is only available for complex datatypes. To enable "<sub_structure>" support, the datatype must contain distinct sub-items (such as the alternative image text for images) and these sub-items must be indexed. Other supported characteristics are: ● author – content object author

	• class – content class • translation – translation
query	Facet query[1]. The facet queries are used to specify facets for the sub-selection of content object attributes.
prefix	Limits the facet fields to only list facet groups where the field value starts with the prefix.
sort	Sort by "count" or "alpha". "alpha" will sort the facet results alpha-numerically by field value.
limit	Maximum number of facet groups to return. The default value is "20".
offset	Offset. The default value is "0".
mincount	Returns only facet groups with more results than the specified minimum count. The default value "0".
missing	If set to "true", the results will also include facet groups with no results. The default value is "false".
date.start	Start date for facet. This must be specified using a strict dateTime syntax.
date.end	End date for facet. This must be specified using a strict dateTime syntax.
date.gap	Size of date range.

Below are examples and more detailed descriptions of the different facet options.

Example:

```
fetch( ezfind, search, hash( 'query', 'Cabriolet',
                             'facet', array( hash( 'field',
'car/model',
                                              'limit', 20 ) )
) )
```

Returns:

Normal result set of 10 documents containing the word "cabriolet", and a facet list with 20 groups of car models (this is specific to "model" attributes of objects of the "car" class) also containing the word "cabriolet".

Example:

```
fetch( ezfind, search,
      hash( 'query', 'Cabriolet',
            'facet', array( hash( 'field', 'car/make',
                                  'limit', 25 ),
                            hash( 'field', 'car/size',
                                  'missing', true(),
                                  'limit', 25 ) ) ) )
```

---

1 http://wiki.apache.org/solr/SimpleFacetParameters#head-529bb9b985632b36cbd46a37bde9753772e47cdd

Returns:

Normal result set with a list of 10 documents containing the word "cabriolet", a facet list with 25 groups of car makes (this is specific to "make" attributes of objects of the "car" class) and a facet list with 25 groups of car sizes (this is specific to "size" attributes of objects of the "car" class), both containing the word "cabriolet". The "car/size" results will also list elements with 0 matching elements.

Example:

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             facet, array( hash( query, 'path:2' ),
                           hash( query, 'path:5' ) ) ) )
```

Returns:

Normal result set with a list of 10 documents containing the word "cabriolet", and a facet list with groups in the subtree with a parent node with an ID of "2", and a facet list with groups in the subtree with a parent node with an ID of "5".

For more information about facets, see:
http://wiki.apache.org/solr/SimpleFacetParameters

`filter`

The filter is used when creating faceted search templates for drill-down navigation. Filters are used to limit the search result set without altering the relevancy sort order. Facet results contain filter definitions that can be used directly. Custom filter definitions can also be created.

A filter is specified by
`<class_identifier>/<class_attribute>[/<sub_structure]:<value>`. The filter option may be a string or list of strings.

Example:

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             filter, 'car/in_stock:1' ) )
```

Returns:

Result with documents containing the words "ez" and "systems", having the content object attribute "car/in_stock" with a value of "1".

Example:

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             filter, array( 'car/in_stock:1',
                            'car/make:Alfa Romeo',
                            'car/model:8C' ) ) )
```

Returns:

Result with documents containing the words "ez" and "systems", having the content object

attribute "car/in_stock" with a value of "1", the "car/make" attribute with the value "alfa romeo" and the "car/model" attribute with the value "8c".

Example:

```
fetch( ezfind, search,
        hash( query, 'eZ Systems',
                filter, 'car/make:( Audi OR Volvo )' ) )
```

Returns:

Result with documents containing the words "ez" and "systems", having the content object attribute "car/make" with the value "audi" or "volvo".

Example:

```
fetch( ezfind, search,
        hash( query, 'eZ Systems',
                filter, array( 'path:2',
                                'contentclass_id:1' ) )
```

Returns:

Result with documents containing the words "ez" and "systems", in the subtree below the node with an ID of "2", for objects of the content class with an ID of "1".

Example:

```
fetch( ezfind, search,
        hash( query, 'eZ Systems',
                filter, array( 'or',
                                array( 'and',
                                        'article/body:hello',
                                        'article/rating:[1 TO 10]'
                                    ),
                                array( 'and',
                                        'article/body:goodbye',
                                        'article/rating:[10 TO 20]'
                                    )
                            )
            )
```

Returns:

Result with articles containing the words "ez" and "systems", either having both 'hello' present in the body and a rating comprized between 1 and 10, either both 'goodbye' in the body and a rating comprized between 10 and 20.

```
class_id
```

This parameter is used to limit the search result to specific content classes, using either their identifiers or ID numbers. This can also be achieved by using the filter functionality. `class_id` may be either a single value or a list of values.

Example:

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             class_id, 1 ) )
```

Returns:

Result with documents containing the words "ez" and "systems", for objects of the content class with an ID of "1".

Example:

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             class_id, array( 'folder', 'article' ) ) )
```

Returns:

Result with documents containing the words "ez" and "systems", for objects of the Folder and Article content classes.

```
subtree_array
```

This parameter is a list of node IDs that specifies which subtrees should be included in the search. The same functionality can be achieved with the filter functionality.

Example:

```
fetch( ezfind, search,
       hash( query, 'eZ Systems',
             subtree_array, array( 23, 42 ) ) )
```

Returns:

Result with documents containing the words "ez" and "systems", from the subtrees with parent nodes with IDs of "23" and "42".

```
boost_functions
```

This parameter is used to pass query-time boosts. This parameter is an associative array, with two keys : 'fields' and 'functions'.

- 'fields' accepts either an associative array, either an array, placing a boost factor on given fields (see 'Example 1' and 'Example 1 bis').
- 'functions' accepts an array, containing an expression. The latter is not be interpreted, and must therefore comply with Solr's Functin Query syntax (http://wiki.apache.org/solr/FunctionQuery). See Example 2.

Both can be provided simultaneously.

Example 1:

```
fetch( ezfind, search,
       hash( 'query', 'eZ Systems',
             'boost_functions',
```

```
                              hash( 'fields',
                                      array( 'article/title:2' )
                              )
              )
```

Returns:

Result with documents containing the words "ez" and "systems". The articles for which the search words were found in the 'title' will be returned first.

Example 1 bis:

```
fetch( ezfind, search,
        hash( 'query', 'eZ Systems',
              'boost_functions',
                      hash( 'fields',
                              hash( 'article/title', 2 )
                              )
              )
```

Returns:

Identical to Example 1.

Example 2:

```
fetch( ezfind, search,
        hash( 'query', 'eZ Systems',
              'boost_functions',
                      hash( 'functions',
                              array( 'ord(meta_modified_dt)^2' )
                              )
              )
```

Returns:

Result with documents containing the words "ez" and "systems". The most recent documents will be returned first, the result score being influenced by the formula : `'ord(meta_modified_dt)^2'`

`section_id`

This parameter is a integer specifying which section should be searched. The same functionality can be achieved with the filter functionality.

Example:

```
fetch( ezfind, search,
        hash( query, 'eZ Systems',
              section_id, 3 ) )
```

Returns:

Result with documents containing the words "ez" and "systems", in the Media section (3).

ignore_visibility

This parameter is a boolean specifying whether or not hidden nodes should be returned in the search results.

Example:

```
fetch( ezfind, search,
      hash( query, 'eZ Systems',
            ignore_visibility, true() ) )
```

Returns:

Result with documents containing the words "ez" and "systems", even if they are hidden.

limitation

This parameter is an associative array overriding the current user's access rights to the 'read' function of the 'content' module. It's format must exactly match the return format of the eZUser::hasAccessTo() method :

Array elements : 'accessWord', 'yes' - access allowed

'no' - access denied

'limited' - access array describing access included

'policies',   array containing the policy limitations

'accessList',   array describing missing access rights

Example:

```
fetch( ezfind, search,
      hash( query, 'eZ Systems',
            limitation, hash( 'accessWord', 'yes' ) ) )
```

Returns:

Result with documents containing the words "ez" and "systems". All results will be returned, regardless of the current user's access rights to the 'read' function of the 'content' module.

spell_check

This parameter is an array configuring the spell check behaviour of the search. The first parameter is a boolean value, enabling or not spellcheck. The second one is taken into account if the first one is set to true, and is optional. It contains the identifier of the dictionnary to be used. Only 'default' is supported for now.

The spellcheck behaviour can also be controlled from ezfind.ini.

Example:

```
fetch( ezfind, search,
```

```
hash( query, 'eZy Sistems',

     spell_check, array( true(), 'default' ) ) )
```

Returns:

Result with documents containing the words "ezy" and "sistems", and a spellcheck suggestion if Solr considered these two words as uncorrectly spelled. The spellcheck feedback is placed under $search_results.SearchExtras.spellcheck and $search_results.SearchExtras.spellcheck_collation, $search_results being the result of the fetch function call.

query_handler

This parameter is a string defining which Solr search handler should be used. The possible values are :

- heuristic (default)

- simplestandard

- standard

- ezpublish

Here are the specificities of each handler :

- standard: the Solr standard handler is called with all syntax supported, searching is done against all searcheable fields

- simplestandard: the Solr standard handler is called with all syntax supported, searching is done against the aggregated field ezf_df_text

- ezpublish: the recommended handler (Solr dismax based) for typical user searches using keywords without boolean or other operators except for + (required) and – (excluding)

- heuristic: depending on the presence of special characters indicating boolean, wildcard or fuzzy expressions, either the standard or dismax handler is called.

The default behaviour can be controlled from ezfind.ini.

Example:

```
fetch( ezfind, search,

     hash( query, 'eZ AND Publish',

          query_handler, 'standard' ) )
```

Returns:

Result with documents containing both words "ez" and "systems", using the standard handler.

6.2.2.   Fetch function `moreLikeThis`

With the "More Like This" functionality and template function, you can use the eZ Find backend to provide similar objects/pages for a given object.

Instead of providing keywords, the `moreLikeThis` function accepts a node id, object id, url or a blob of text. After analyzing the text, the Solr engine will create a search with keywords chosen with heuristis that depend on the indexed documents as well as the object fed to it.

Besides the special query parameters specified below, facets, filters and sorting parameters as in the regular search function are supported as well.

Parameter	Values/Description
query_type	string with possible values ('nid' | 'oid' | 'text' | 'url') nid: node id oid: object id text: a blob of text url: a url, accessible from the Solr backend
query	the value, depending on query type

In some cases, you may want to provide only part of an object for getting similar objects. The url can be useful for links to external pages, or which you want to show internal objects that resemble the content behind the url

6.3.   Customizing result templates

Customizing result templates for eZ Find is similar to customizing templates for the regular search in eZ Publish, except that eZ Find offers some additional options.

The eZ Find fetch function returns some extra information compared to the default search in eZ Publish, including:

- Relevancy ranking
- Language information
- Complete URLs to external results

The default search in eZ Publish returns a list of `eZContentObjectTreeNode` objects. To provide extra information, eZ Find returns a list of `eZFindResultTree` objects. The `eZFindResultTree` class extends `eZContentObjectTreeNode` and contains the following extra attributes:

- `is_local_installation`: a boolean value indicating whether the result item is from the installation where the search was performed
- `name`: the language-dependent name of the result item
- `global_url_alias`: URL to the result item, including the protocol and domain name
- `published`: published timestamp

- `language_code`: result item language code

- `highlight`: text extracts that include the search terms

- `score_percent`: a relative value that indicates how well the result item matches the search terms

These values are accessible as regular attributes in the templates.

The default eZ Find result templates are stored in *extension/ezfind/design/standard/templates/content.*

Custom templates for the Website Interface are provided with eZ Find. An example of the result item is shown below (*extension/ezfind/design/ezwebin/templates/node/view/ezfind_line.tp l*). $node is an instance of eZFindResultTree.

ezfind_line.tpl code:

```
<div class="content-view-line">

    <div class="class-article float-break">


        <div class="attribute-title">
            <h2 style="margin-top: 0.5em; margin-bottom: 0.25em"><a
href="{$node.global_url_alias}">{$node.name|wash}</a></h2>
        </div>


        {if is_set( $node.data_map.image )}
            {if $node.data_map.image.has_content}
                <div class="attribute-image">
                    {attribute_view_gui image_class=small href=concat( '"',
$node.global_url_alias, '"' ) attribute=$node.data_map.image}
                </div>
            {/if}
        {/if}


        <div class="attribute-short">
            {$node.highlight}
        </div>
```

```
        <div class="attribute-short">
            <i>{$node.score_percent}% - <a
href="{$node.global_url_alias}">{$node.global_url_alias|shorten(70, '...',
'middle')|wash}</a> - {$node.object.published|l10n(shortdatetime)}</i>
        </div>
    </div>
</div>
```

6.4. Elevation, or "Sponsored results"

The "Elevation" feature allows you to make sure a given content object is placed as first search result when searching for specific words. Let us assume that you would like your company's description (which is an Article object) be part the first results when searching for "services", in any language.

Here is how you would proceed :

Navigate to eZ Publish's administration interface, and hit the "eZ Find" tab. The URL is */ezfind/elevate/.* In the "Elevate an object" box, fill the "Search query" text field with "services", and hit the "Elevate Object" button.

This will let you browse for the object you want to Elevate, and end up on the following screen, where you will choose the language to apply elevation to, and confirm the configuration :

Click on the "Elevate" button. It is now time to synchronise your local Elevate configuration

with Solr's. Do so by a simple click on the "Synchronise" button from the confirmation screen :
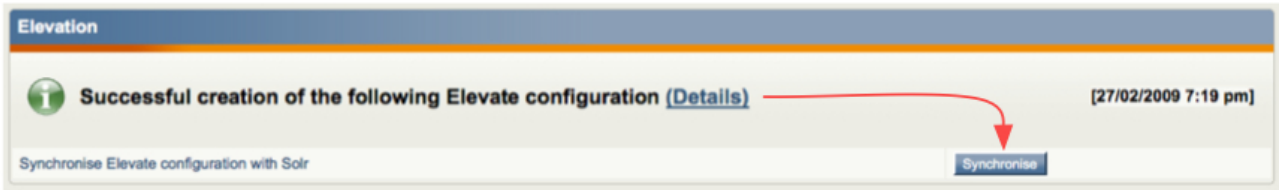
As a shortcut for elevating a given object, you can use the left-click menu on nodes' icons, like that :
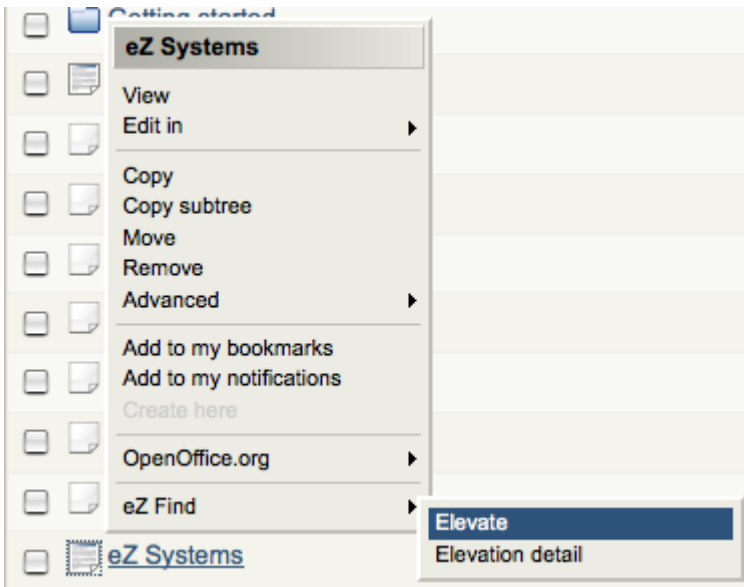
Note the "Elevation detail" menu entry, which leads you to the detail of existing elevations for this object.

And here we are, with the before and after-elevation comparison :

Illustration 1: Before Elevation

Illustration 2: After Elevation

7. Using eZ Find

7.1. Basic search

To perform a search, enter one or more search terms in the search field. To execute the search, press Enter on your keyboard or click the **Search** button.

After the search is executed, the search result page is displayed. The search result page lists the content objects that matched the search term, with some information about each object. The total number of content objects matching the search terms is displayed under the search term field. The search time shows how long the internal search engine took to find the results.

Each result contains a small description of the content object. Results can also contain a title, image, summary, relevance, location and creation date. The summary includes one or more excerpts where the search terms occur. The relevance indicates how well the content object fits the search terms and conditions. The title, image and location link to the content object. Image preview is not provided for content objects located on other eZ Publish installations.

7.2. Search term options

7.2.1. Phrase search

To search for a phrase, surround the phrase with quotation marks (for example, *"Grenland in Telemark").*

You can combine searching for a phrase with searching for individual terns. For example, entering '"Grenland in Telemark" boat' will return all documents containing the phrase "Grenland in Telemark" and the word "boat".

7.2.2. Exclude or require terms

You can exclude documents that contain specific words from the search results by prefixing the word with a minus symbol. For example, "Telemark -Grenland":

... will return results that contain the work "Telemark" but do not contain the word "Grenland".

Similarly, prefix a search term with a plus symbol to specify that all documents must have the term in order to be included in the search results.

7.2.3. Searching for multiple terms

When multiple search terms are specified, a certain number (depending on the number of terms entered) must match the content object in order for the content object to be included in the search results. For example, when two terms are specified, at least one of them must match for the content object to be included in the results. When three or four terms are specified, at least two must match. For more than four terms, 30% of the terms must be present for a match.

These (heuristic) rules reduce the returned results to a smaller but usually more meaningful set.

7.3. Advanced search

Advanced search is accessed by clicking the **Advanced search** link on the search result page.

The **Advanced search** interface enables you to further limit search conditions.

The **Search the exact phrase** field provides the same functionality as the phrase search described earlier.

7.3.1. Content class limitation

To limit the search to a specific content class, select the class from the dropdown list. Click the **Update attributes** button to see the attributes associated with the selected content class.

If a class attribute is selected, the search terms must occur in the selected attribute.

7.3.2. Other limitations

You can also limit search results based on the section where the content is published and the published date.