FirstSpirit™

Unlock Your Content

FirstSpirit™ EnterpriseSearch
FirstSpirit™ Version 4.2R4

Version	**2.0.3**
State	**RELEASED**
Date	**2012-04-20**
Department	Product management
Author/ Authors	D. Marro, A. Knoor
Copyright	2012 e-Spirit AG
File name	EXAL50EN_FirstSpirit_EnterpriseSearch

e-Spirit AG

Barcelonaweg 14
44269 Dortmund | Germany

T +49 231 . 477 77-0
F +49 231 . 477 77-499

info@e-Spirit.com
www.e-spirit.com

e-Spirit

Table of content

1 Introduction

This document describes the different integration options between the FirstSpirit content management system and the enterprise search engine technology EXALEAD CLOUDVIEW™ (hereafter referred to as "Exalead" for short).

Both products can generally be used independent of each other. However, in combination, both modern technologies can be connected to each other to produce a high level of functional utility for the website user, who would like to find specific information within FirstSpirit content in the easiest way possible.

In total, the following six aspects are described:

- Chapter 2: Architecture of the integration solution between FirstSpirit and Exalead
- Chapter 3: Description of the FirstSpirit runtime component for displaying Exalead search results on the web server ("Exalead Live" module)
- Chapter 4: Explanation of the options for assigning permissions to documents and therefore for providing personalized search
- Chapter 5: Incremental index updating with the help of the FirstSpirit "Exalead Content-Update" module
- Chapter 6: Use of the autocomplete feature
- Chapter 7: Migration notes for updating from Version 1.x to Version 2.0

Technically the FirstSpirit Exalead integration is therefore divided into two sub-modules: "Exalead Live" for searching and display of the results, and "Exalead Content-Update" for incremental index updating.

This documentation refers to Version 2.0 of the FirstSpirit Exalead Live and FirstSpirit Exalead Content-Update modules!

2 Architecture and Components

The following components are relevant in the specified integration scenario:

- FirstSpirit Server (Version >= 4.2.438), incl. the modules:
    - FirstSpirit Exalead Live
    - FirstSpirit Personalisation (optional)
    - FirstSpirit Exalead Content-Update (optional)
- Exalead CloudView Server (Version 5.1)
- Java Application Server (Servlet Spec. >= 2.4, recommended: Apache Tomcat)
- Java Runtime Environment 1.5 (Version 1.6 is required for use of the PushAPI)

Actual installation of FirstSpirit, Exalead and the application server is not described here. Please refer to the respective installation documents instead.

Several of the listed modules are optional and are therefore not necessary or useful in all application cases. Therefore, different development stages of integration are examined in the following chapters. The options and variants become increasingly complex from step to step; however, functionally they are based on each other.

2.1 Basic architecture

For example, the basic architecture explained here is suitable for internet sites which primarily want to make public, impersonalized information browsable.

The interaction of the individual components and the data flow between each other is schematically represented in Figure 2-1:

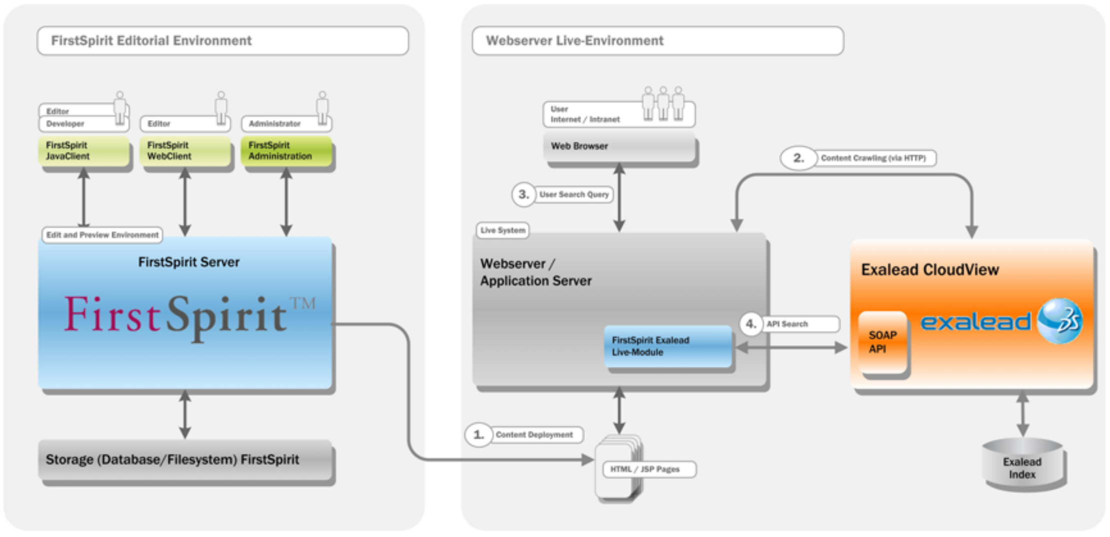


Figure 2-1: Basic architecture

Workflow description:

- FirstSpirit is used to enter and maintain the internet site content and to transform it into suitable HTML or JSP pages.
- The generated files are then transferred to the application server as part of a deployment process [Step 1].
- As soon as all pages have been transferred to the web server, the Exalead search engine can read in the files as part of a "Crawling" process, analyze them and include them in the index. This process takes place cyclically, e.g. after each deployment of new or amended files [Step 2]. The search index is now filled and can answer search queries.
- The visitor to the internet site can now use a search form to browse through the indexed content [Step 3]. The form passes the form query to the "FirstSpirit Exalead Live Module", which in turn passes the query to the Exalead server. Technically, the "FirstSpirit Exalead Live Module" contains a servlet, which uses the SOAP-API (Simple-Object-Access-Protocol) of the Exalead server.
- After the search query has been processed and answered by the Exalead server, the "FirstSpirit Exalead Live Module" clearly presents the search results on an appropriate page. Technically, the parameters for the output design are assigned via a JSP tag library.

This basic architecture enables application of the complete Exalead search options to FirstSpirit content, such as

- ✓ Indexing more than 300 formats, incl. Word, Excel, PowerPoint, PDF, RTF, OpenOffice, HTML, XML, Images, audio, video
- ✓ Multi-language support (Unicode)
- ✓ Natural language processing (spelling suggestions, phonetic matching)
- ✓ Automatic classification of search queries and automatic keyword generation
- ✓ Preview thumbnails of the hits
- ✓ Guided navigation through "drilldown" within the results lists

Limitations of the basic architecture:

- no personalized delivery of HTML pages on the basis of permissions
- no personalized delivery of binary documents on the basis of permissions
- no ad-hoc updating of the index in the event of new content

2.2 Architecture with page personalization

The basic approach of the previous chapter can be enhanced to include the aspect of personalization for web scenarios in which protected page content is also relevant (e.g. in the case of intranets).

In the following it is assumed that the "FirstSpirit Personalization" module is used to personalize the content on the application server side.

Architecturally, the layout is therefore as in Figure 2-2:

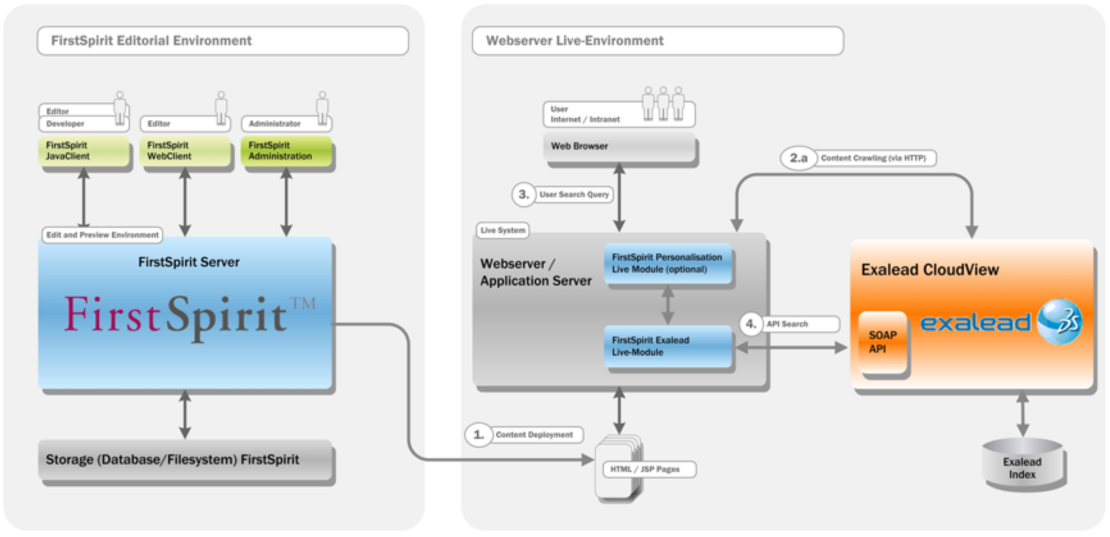

FirstSpirit Editorial Environment
Editor
Developer
Editor
Administrator
FirstSpirit JavaClient
FirstSpirit WebClient
FirstSpirit Administration
Edit and Preview Environment
FirstSpirit Server
FirstSpirit™
Storage (Database/Filesystem) FirstSpirit
Webserver Live-Environment
User Internet / Intranet
Web Browser
2.a Content Crawling (via HTTP)
3. User Search Query
Live System
Webserver / Application Server
FirstSpirit Personalisation Live Module (optional)
FirstSpirit Exalead Live-Module
4. API Search
Exalead CloudView
SOAP API
exalead
1. Content Deployment
HTML / JSP Pages
Exalead Index

Figure 2-2: Personalized search

To a large extent, the layout corresponds to that of the basic architecture; however, coupling of the "FirstSpirit Exalead Live module" and the "FirstSpirit Personalization module" is added.

Technically, the modules interact so that all search queries passed to the SOAP-API of Exalead are additionally supplemented with the relevant personalization information of the currently logged in visitor. Specifically, these are:

- Login name of the user
- List of the user's groups

This information is transparently copied from the personalization module by the "FirstSpirit Exalead Live Module".

The index continues to be developed completely via a crawling approach.

Suitable metatags, which define the personalization information per page, must be deposited within the HTML pages to enable correct filtering of the HTML pages. Details of this are given in Chapter 4.2. At this point it needs only be noted that the permissions information is injected into the HTML by suitable FirstSpirit templates.

This results in the following options as an enhancement of the basic architecture:

✓ very easy to implement support for page personalization by means of simple FirstSpirit template constructs

✓ Definition of permission on HTML pages directly by the FirstSpirit editor

- ✓ Personalized, user-specific filtered search results by coupling the user stores of FirstSpirit and Exalead

Limitations of this architecture:

- no personalized delivery of binary documents on the basis of permissions (personalization information cannot be injected there)
- no ad-hoc updating of the index in the event of new content (generally not possible with the crawling approach)

2.3 Architecture with document personalization (Push-API)

In order to realize the two aspects:

- Passing of personalization information to binary documents
- ad-hoc updating of the index,

the crawling approach used to date must be replaced by a second strategy: the Exalead Push-API.

When the Push-API is used, active documents from a source (FirstSpirit) are passed to the search engine (by means of the "Push command"). Apart from the pure document, meta attributes such as personalization information can also be passed at the same time.

The layout is shown in the diagram in Figure 2-3:

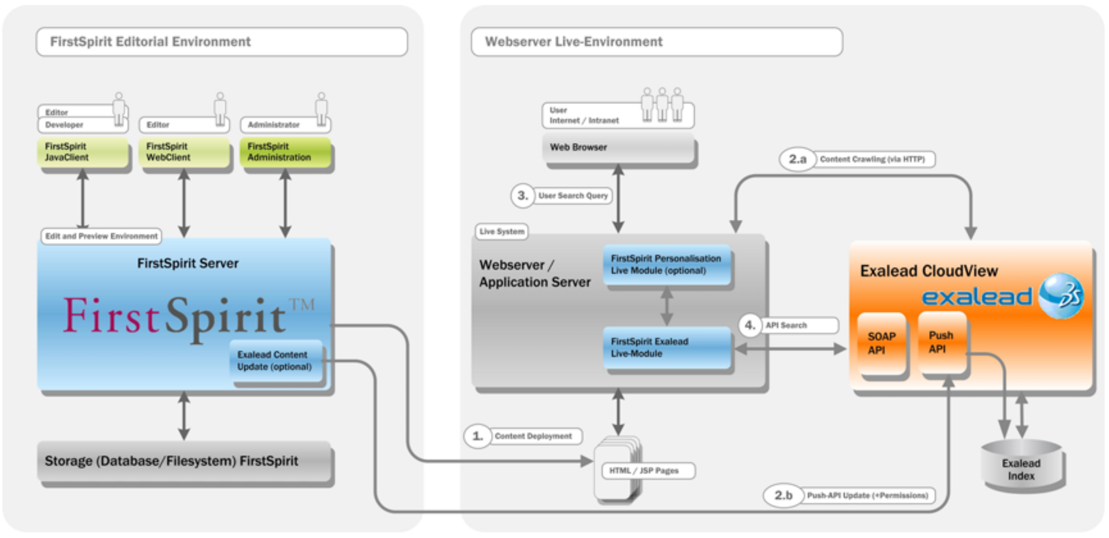

FirstSpirit Editorial Environment
Editor
Developer
Editor
Administrator
FirstSpirit JavaClient
FirstSpirit WebClient
FirstSpirit Administration
Edit and Preview Environment
FirstSpirit Server
FirstSpirit™
Exalead Content Update (optional)
Storage (Database/Filesystem) FirstSpirit
Webserver Live-Environment
User Internet / Intranet
Web Browser
3. User Search Query
2.a Content Crawling (via HTTP)
Live System
Webserver / Application Server
FirstSpirit Personalisation Live Module (optional)
FirstSpirit Exalead Live-Module
4. API Search
Exalead CloudView
exalead
SOAP API
Push API
1. Content Deployment
HTML / JSP Pages
Exalead Index
2.b Push-API Update (+Permissions)

Figure 2-3: Push-API architecture

The "Exalead Content-Update" has been added on the FirstSpirit server side. This is capable of directly passing binary documents and HTML pages to the Exalead server by means of Push-API.

To do this, the locally generated files (pages and documents) are read into the FirstSpirit server and are passed to the Exalead Push-API [Step 2.b]. Details of using the Push-API from FirstSpirit are given in Chapter 5.

2.3.1 Crawling vs. Push-API

The two versions of indexing, "Crawling" [Step 2.a] and Push-API [Step 2.b] do not necessarily exclude each other, but instead can also be used in combination. The cases in which this makes sense are examined in the following.

At first glance, sole use of the Push-API only brings advantages:

a) immediate index updating within the scope of a FirstSpirit deployment

b) Handover of metadata for the personalization of pages and documents

However, there is one condition under which a combination of crawling (for HTML pages only) and Push-API (for documents only) is necessary.

Whenever the HTML pages contain "indexing relevant" runtime logic, e.g. in the form of JSP, PHP or .NET code, which can only be run within the target application server, it is not possible to do without crawling via the application server.

Example:

A JSP page generated by FirstSpirit not only contains editorial FirstSpirit content, but also an inclusion logic, which integrates and displays message contributions from an external source during the runtime. The content of the messages should be taken into account in the indexing of the pages, so that these indexing terms also lead to a hit during the search.

In such situations, combined use of crawling and Push-API is necessary.

The integration architecture is in any case sufficiently flexible to allow such more complex scenarios to be adequately reproduced too.

3 FirstSpirit Exalead Live Module

3.1 General information

This chapter describes installation and configuration of the "FirstSpirit Exalead Live" module. It is assumed that both FirstSpirit and the Exalead server and a suitable Java Application server have already been set up and configured.

3.1.1 Character coding

The FirstSpirit Exalead Live Module uses UTF-8 throughout as character coding. To ensure uniform use of this character coding, it is necessary to provide several specific details concerning it.

Each HTML page should be assigned a corresponding Meta tag:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-
8">
```

In JSP pages, the page encoding must also be given:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
```

When using forms, ensure that the character coding to be used here is also given:

```
<form ... method="post" accept-charset="UTF-8">
```

Finally, it is also necessary to configure the Apache Tomcat server so that the right character coding is also used when hyperlinks are called. To do this, the `URIEncoding="UTF-8"` parameter must be added to the following section in the `server.xml` of the Apache Tomcat (in the `conf` directory):

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
<Connector port="8080" ... URIEncoding="UTF-8"/>
```

3.2 Quick install

The easiest and fastest way to install the "FirstSpirit Exalead Live" module is to copy the "search.war" archive into the Webapps directory of the server where it is automatically extracted if the server is running, or following a server reboot at the latest, and is created as a web application called "search".

Now the server name `myserver` in the `webapps/search/WEB-INF/web.xml` file must be replaced by the name of the Exalead server (if necessary adjust the port too).

The search can then be reached by calling `/search/index.jsp`:

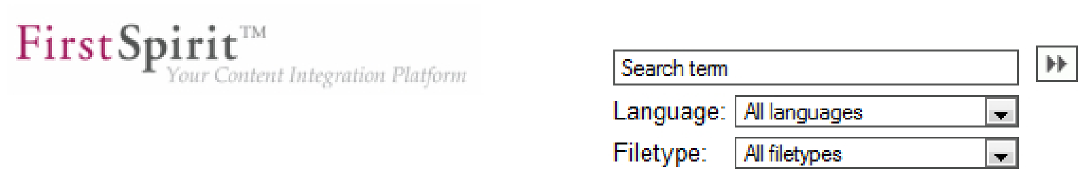


Figure 3-1: Search screen form

3.3 Integration in existing web application

The following files must be copied from the web application created in Chapter 3.2 into the existing web application to add the FirstSpirit Exalead Live module to an existing web application:

- `/WEB-INF`
  - `exalead.tld`
  - `web.xml` (only add entries not yet available)
  - `firstpersonalisation.tld` (if not yet available)
  - `firstpersonalisation.xml` (if not yet available)
- `/WEB-INF/lib`
  - Complete content
- `/WEB-INF/classes`
  - `log4j.properties` (if Log4J not configured elsewhere)

The necessary entries in the `web.xml` and the corresponding servlets are described in greater detail in the following. In addition, the changes to the entries in the `web.xml` necessary in relation to integration in an existing web application are also explained.

The tag library with which the search interface and results can be designed to your own wishes is described in Chapter 3.8 page 23.

3.4 web.xml – Servlets

The Exalead web application provides three servlets:

- *SearchServlet*
  This servlet forwards the search queries to the Exalead server (absolutely necessary).

- *SearchDownload*
  This servlet is responsible for calling results which cannot be reached directly from a user's browser (e.g. search hits in the file system) in the Exalead server and for delivering them to the browser via the web application.

- *ThumbnailServlet*
  This servlet outputs the preview images of the search hits, provided they exist.

- *AutocompleteServlet*
  This servlet forms the interface with Exalead's SuggestService, which carries out the autocompletion of the search terms entered.

For more detailed information on the servlets, please refer to Chapter 3.7 from page 19.

3.5 web.xml – Context Parameter

`exalead.server.host`

This parameter is used to name the server on which Exalead is installed.

`exalead.server.baseport`

This parameter is used to configure the baseport given on installing Exalead.

`getDocumentServlet`

This parameter is only required in conjunction with the `SearchDownload-Servlet` and contains the path to this servlet within the web application (name of the web application in the example: "`/search`").

3.6 Exemplary web.xml

The necessary parameters in the web.xml file are summarized again here with meaningful values. A web.xml file to which the Exalead integration elements have been added could look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
...
      <context-param>
            <param-name>exalead.server.host</param-name>
            <param-value>localhost</param-value>
      </context-param>
<context-param>
            <param-name>exalead.server.baseport</param-name>
            <param-value>10000</param-value>
      </context-param>
<context-param>
            <param-name>getDocumentServlet</param-name>
            <param-value>/search/do.searchDownload</param-value>
      </context-param>

      <servlet>
            <servlet-name>SearchServlet</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.SearchServlet
            </servlet-class>
            <init-param>
                  <param-name>userPrefix</param-name>
                  <param-value>user:</param-value>
            </init-param>
            <init-param>
                  <param-name>groupsPrefix</param-name>
                  <param-value>groups:</param-value>
            </init-param>
            <init-param>
                  <param-name>sendSecurityToken</param-name>
                  <param-value>false</param-value>
            </init-param>
      </servlet>

      <servlet>
            <servlet-name>SearchDownload</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.SearchDownload
            </servlet-class>
      </servlet>

      <servlet>
            <servlet-name>ThumbnailServlet</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.ThumbnailServlet
            </servlet-class>
      </servlet>

      <servlet>
            <servlet-name>AutocompleteServlet</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.AutocompleteServlet
            </servlet-class>

            <init-param>
                  <param-name>suggestServiceName</param-name>
                  <param-value>mysuggest</param-value>
            </init-param>
      </servlet>

      <servlet-mapping>
            <servlet-name>SearchServlet</servlet-name>
            <url-pattern>*.search</url-pattern>
      </servlet-mapping>

      <servlet-mapping>
            <servlet-name>SearchDownload</servlet-name>
            <url-pattern>*.searchDownload</url-pattern>
      </servlet-mapping>

      <servlet-mapping>
            <servlet-name>ThumbnailServlet</servlet-name>
            <url-pattern>*.searchThumbnail</url-pattern>
      </servlet-mapping>

      <servlet-mapping>
            <servlet-name>AutocompleteServlet</servlet-name>
            <url-pattern>*.autocomplete</url-pattern>
      </servlet-mapping>
...
</web-app>
```

3.7 Description of the servlets

As already mentioned in the previous chapter, the FirstSpirit Exalead Live module is supplied with three servlets, whose configuration is described in the following.

3.7.1 SearchServlet

The `SearchServlet` receives all search enquiries, suitably passes them on to the Exalead server, then receives the search results from it and then makes these results available to the JSP tags responsible for the display.

Configuration example:

```
<servlet>
      <servlet-name>SearchServlet</servlet-name>
      <servlet-class>
            de.espirit.ps.exalead.servlets.SearchServlet
      </servlet-class>
      <init-param>
            <param-name>userPrefix</param-name>
            <param-value>fs:user:</param-value>
      </init-param>
      <init-param>
            <param-name>groupsPrefix</param-name>
            <param-value>fs:group:</param-value>
      </init-param>
      <init-param>
            <param-name>sendSecurityToken</param-name>
            <param-value>false</param-value>
      </init-param>
</servlet>

<servlet-mapping>
      <servlet-name>SearchServlet</servlet-name>
      <url-pattern>*.search</url-pattern>
</servlet-mapping>
```

The `userPrefix` and `groupsPrefix` parameters give the prefixes of the user and group permissions for documents in the search index and therefore only play a role if personalized search results are to be displayed. These parameters are only evaluated if use of SecurityToken was activated for the search query in the third parameter `sendSecurityToken`.

If manual permissions are assigned for documents (see also Chapter 4.2, 5.3.2.13 and 5.3.2.14), the prefixes used there must correspond to the values of the `userPrefix` and `groupsPrefix` parameters. If these parameters are not configured in the `SearchServlet`, the default values `exalead:user:` and `exalead:group` are used.

Within the search page, the servlet is addressed using a form.

An exemplary call of the `SearchServlets` looks like this:

```
<form action="do.search">
      <input type="text" name="q" value=""/>
      <input type="hidden" name="b" value="0"/>
      <input type="hidden" name="l" value="de"/>
      <input type="hidden" name="hf" value="10"/>
      <input type="hidden" name="redirectUrl" value="search.jsp"/>
      <input type="hidden" name="errorUrl" value="error.jsp"/>
      <input type="submit" value="Search"/>
</form>
```

The parameters have the following meaning:

Parameter	Expected value
q	Input field for the search word (term)
b	Index of the first hit to be displayed in the results list
l	Language to be used to interpret the search term
hf	Number of hits to be displayed per results page
redirectUrl	Results page
errorUrl	Errors page

3.7.2 SearchDownload

The `SearchDownload` servlet is used to deliver search hits in the search results lists via the web application, as this cannot be directly accessed from the browser because, for example, they are located in a network file system. To do this, this servlet queries the Exalead server for the file and then forwards it to the browser.

Configuration example:

```
      <context-param>
            <param-name>getDocumentServlet</param-name>
            <param-value>/search/do.searchDownload</param-value>
      </context-param>

      <servlet>
            <servlet-name>SearchDownload</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.SearchDownload
            </servlet-class>
      </servlet>

      <servlet-mapping>
            <servlet-name>SearchDownload</servlet-name>
            <url-pattern>*.searchDownload</url-pattern>
      </servlet-mapping>
```

For automatic generation of links to the `SearchDownload` servlet, the `getDocumentServlet` context parameter must be used to inform the application of the address under which the `SearchDownload` servlet can be addressed.

A description of how the output of automatically generated download links can be controlled within the search results page is given in Chapter 3.12.3 from page 42.

3.7.3 ThumbnailServlet

The `ThumbnailServlet` delivers the thumbnails for the search result hits, provided they exist. If a thumbnail does not exist for the hit, a replacement image can also be given in interaction with the corresponding JSP tags.

```
      <servlet>
            <servlet-name>ThumbnailServlet</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.ThumbnailServlet
            </servlet-class>
      </servlet>

      <servlet-mapping>
            <servlet-name>ThumbnailServlet</servlet-name>
            <url-pattern>*.searchThumbnail</url-pattern>
      </servlet-mapping>
```

3.7.4 AutocompleteServlet

The `AutocompleteServlet` is required to enable linking to the SuggestService of Exalead, which allows autocompletion of the search terms entered via JavaScript.

The `suggestservicename` parameter is used to given the name of the SuggestService to be used for the autocomplete feature. Chapter 6 describes, step-by-step, how such a SuggestService is created and configured in Exalead.

```
      <servlet>
            <servlet-name>AutocompleteServlet</servlet-name>
            <servlet-class>
                  de.espirit.ps.exalead.servlets.AutocompleteServlet
            </servlet-class>

            <init-param>
                  <param-name>suggestServiceName</param-name>
                  <param-value>mysuggestservice</param-value>
            </init-param>
      </servlet>

      <servlet-mapping>
            <servlet-name>AutocompleteServlet</servlet-name>
            <url-pattern>*.autocomplete</url-pattern>
      </servlet-mapping>
```

3.8 JSP tags

With the FirstSpirit Exalead JSP tags it is possible to display the search results in virtually any layout. For example, a preview image can be displayed for each hit, hits in certain categories can be highlighted, refinements (drilldown) can be given for the search results, and much more.

The following Exalead functions are not supported at the present time:

- Display of related terms
- Search within search results

The JSP tags available for this are briefly described in the following and are then dealt with in detail.

Global tags:

- <search:container>
  Provision of all data concerning the search results
  (see Chapter 3.10.1 page 30).
- <search:query>
  Output of the search term:
  (see Chapter 3.10.2 page 31).
- <search:nhits>
  Number of search results
  (see Chapter 3.10.3 page 31).
- <search:time_overall>
  Output of the search query duration
  (see Chapter 3.10.4 page 32).
- <search:sort_link> and <search:sort_linkparameter>
  Output of a like for sorting the search results by relevance, date or size
  (see Chapter 3.10.5 page 32).
- <search:suggestions_loop>
  Iteration over search word suggestions
  (see Chapter 3.10.7 page 34).
- <search:suggestions_link> and <search:suggestions_linkparameter>
  Output of the link for searching with the suggested search word
  (see Chapter 3.10.8 page 35).
- <search:suggestions_choice>
  Output of the suggested search word

(see Chapter 3.10.10 page 35).

Tags for refining the search results:

- <search:groups>
  Provision of all data concerning the refinement of the search results
  (see Chapter 3.11.1 page 36).
- <search:group>
  Provision of the information of a specific search category
  (see Chapter 3.11.2 page 36).
- <search:groups_categories_loop>
  Iteration over the entries of a specific search category
  (see Chapter 3.11.3 page 36).
- <search:groups_category_title>
  Output of the entry's title
  (see Chapter 3.11.4 page 37).
- <search:groups_category_count>
  Output of the number of hits within this entry
  (see Chapter 3.11.5 page 38).
- <search:groups_category_link> and <search:groups_category_linkparameter>
  Output of the link for limiting the search results to this entry
  (see Chapter 3.11.6 page 38).
- <search:reset_refinement>
  Provision of the information about search result limitations made
  (see Chapter 3.11.8 page 38).
- <search:hasRefinements>
  Checks whether limitations exist for a specific category
  (see Chapter 3.11.9 page 39).
- <search:refinements>
  Provision of the information on the restrictions of a specific category
  (see Chapter 3.11.10 page 40).
- <search:reset_link> and <search:reset_linkparameter>
  Link to reset the limitation of a specific category
  (see Chapter 3.11.11 page 40).

Tags for displaying the search results:

- <search:loop_hits>
  Iterates over the search results
  (see Chapter 3.12.1 page 42).
- <search:hits_id>
  Output of the consecutive number of the hit (starting at 0)
  (see Chapter 3.12.2 page 42).
- <search:hits_url>
  Output of the hit's URL
  (see Chapter 3.12.3 page 42).
- <search:hits_title>
  Output of the hit's heading
  (see Chapter 3.12.4 page 43).
- <search:hits_doctype>
  Output of the hit's document type
  (see Chapter 3.12.5 page 43).
- <search:hits_score>
  Output of the hit's ranking
  (see Chapter 3.12.6 page 44).
- <search:hits_summary>
  Output of the text extract of the hit
  (see Chapter 3.12.7 page 45).
- <search:hits_field>
  Output of other index fields of a hit
  (see Chapter 3.12.8 page 46).
- <search:hits_filesize>
  Output of the size of a hit
  (see Chapter 3.12.9 page 46).
- <search:hits_date>
  Output of the date of a hit
  (see Chapter 3.12.10 page 47).
- <search:hits_thumbnail>
  Enclosing tag for the area in which thumbnails are to be displayed
  (see Chapter 3.12.11 page 47).
- <search:hits_hasThumbnail> / <search:hits_hasNoThumbnail>
  Checks whether a preview image of the hit exists
  (see Chapter 3.12.12 page 47).
- <search:is_in_category> / <search:is_not_in_category>
  Checks whether the search result belongs to a specific category
  (see Chapter 3.12.14 page 48).

Tags for navigation:

- <search:navigation>
  Provision of the information for navigation
  (see Chapter 3.13.1 page 49).
- <search:navigation_page_first_link> and
  <search:navigation_page_first_linkparameter>
  Output of the link to the first results page
  (see Chapter 3.13.2 page 49).
- <search:navigation_page_previous_container>
  Checks whether the link to the previous results page is to be displayed or not
  (see Chapter 3.13.4 page 50).
- <search:navigation_page_previous_link> and
  <search:navigation_page_previous_linkparameter>
  Output of the link to the previous results page
  (see Chapter 3.13.5 page 50).
- <search:navigation_loop_pages>
  Iterates over the page numbers of the navigation to be displayed
  (see Chapter 3.13.7 page 50).
- <search:navigation_is_current_page>
  Checks whether the displayed page number corresponds to the currently displayed results page
  (see Chapter 3.13.8 page 51).
- <search:navigation_is_not_current_page>
  Checks whether the displayed page number does not correspond to the currently displayed results page
  (see Chapter 3.13.9 page 51).
- <search:navigation_page>
  Output the page number
  (see Chapter 3.13.10 page 51).
- <search:navigation_page_link> and
  <search:navigation_page_linkparameter>
  Output of the link of the results page corresponding to the displayed page number
  (see Chapter 3.13.11 page 51).
- <search:navigation_page_next_container>
  Checks whether the link to the next results page should be displayed
  (see Chapter 3.13.13 page 51).
- <search:navigation_page_next_link> and
  <search:navigation_page_next_linkparameter>
  Output of the link to the next results page
  (see Chapter 3.13.14 page 52).

- <search:navigation_page_last_container>
  Checks whether the link to the last results page can be displayed
  (see Chapter 3.13.16 page 52).
- <search:navigation_page_last>
  Output of the page number of the last results page
  (see Chapter 3.13.17 page 52).
- <search:navigation_page_last_link> and
  <search:navigation_page_last_linkparameter>
  Output of the link to the last results page
  (see Chapter 3.13.18 page 53).

A minimum search results page could therefore look like this, for example:

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@ taglib prefix="search" uri="/WEB-INF/exalead.tld" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Search results</title>
</head>
<body>
<form action="do.search">
<input type="text" name="q" value="<%= session.getAttribute("searchString") != null
? session.getAttribute("searchString") : "" %>"/>
<input type="hidden" name="b" value="0"/>
<input type="hidden" name="l" value="de"/>
<input type="hidden" name="hf" value="10"/>
<input type="hidden" name="redirectUrl" value="search.jsp"/>
<input type="hidden" name="errorUrl" value="search.jsp"/>
<input type="submit" value="Search"/>
</form>
<hr/>
<p>Display search results</p>
<search:container hasResults="true" redirectUrl="search.jsp" errorUrl="search.jsp"
hitsPerPage="10">
<table border="1">
<tr><th>Index</th><th>Title</th><th>URL</th><th>Summary</th></tr>
<search:loop_hits>
            <tr>
            <td><search:hits_id/></td>
            <td><a href="<search:hits_url completeDocUrl="true"/>">
```

```
                              <search:hits_title/>
            </a></td>
            <td><search:hits_url/></td>
            <td><search:hits_summary isLongForm="true" endline=" ...<br/>"
highlightStart="<strong>" highlightEnd="</strong>" /></td>
            </tr>
</search:loop_hits>
</table>
<br/>
<table border="1">
            <tr>
            <search:navigation surroundingPagesRadius="5">
            <td><a href="<search:navigation_page_first_link/>">
                              First page
            </a></td>
            <search:navigation_page_previous_container>
            <td><a href="<search:navigation_page_previous_link/>">
                              Previous page
            </a></td>
            </search:navigation_page_previous_container>
            <search:navigation_loop_pages>
            <td>
            <search:navigation_is_current_page>
                              <strong><search:navigation_page/></strong>
            </search:navigation_is_current_page>
            <search:navigation_is_not_current_page>
            <a href="<search:navigation_page_link/>">
            <search:navigation_page/></a>
            </search:navigation_is_not_current_page>
            </td>
            </search:navigation_loop_pages>
            <search:navigation_page_next_container>
            <td><a href="<search:navigation_page_next_link/>">
                              Next page
            </a></td>
            </search:navigation_page_next_container>
            <search:navigation_page_last_container>
            <td><a href="<search:navigation_page_last_link/>">
                              Last page (<search:navigation_page_last/>)
            </a></td>
            </search:navigation_page_last_container>
            </search:navigation>
            </tr>
</table>
```

```
</search:container>

<search:container hasResults="false" redirectUrl="index.jsp" errorUrl="index.jsp"
hitsPerPage="10">

<strong>There were no results matching the query....</strong><br/>

</search:container>

</body>

</html>
```

3.9 Tag prefix

The relevant tag library must be given in the JSP pages in order to be able to use FirstSpirit Exalead tags. This document uses the "search" prefix for the FirstSpirit Exalead tags.

Example of integration in JSP pages:

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ taglib uri="/WEB-INF/exalead.tld" prefix="search" %>
```

If the “search" prefix is changed to another value, this prefix must be used for the individual tags, i.e. “<**myPrefix**:ref>” instead of “<search:ref>".

3.10 Global tags

3.10.1 <search:container>

The <search:container> tag prepares the search results so that they can be evaluated and output by the other tags available (refinements, hits, navigation).

Attributes:

Attribute	Meaning	Mandatory parameters
hasResults	This attribute controls whether the content of the tags is to be displayed if search hits exist (hasResults=“true“) or whether the content is to be displayed if there are no search hits (hasResults=“false“).	Yes
redirectUrl	This attribute gives the address of the search results page. In general, this is the page on which you are currently located. It is necessary to specify this information so that the automatically generated links (e.g. for the navigation) link to the correct page.	Yes

errorUrl	This attribute gives the address of the page which is set in the event of an error. This information is also required for the automatically generated links.	Yes
hitsPerPage	This attribute is used to give the number of hits to be displayed per page. If such information was already specified in the SearchServlet form, the same value must be entered here.	no

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp" hitsPerPage="20">
      Display of the search results
</search:container>
<search:container hasResults="false" redirectUrl=" search.jsp"
errorUrl=" error.jsp" hitsPerPage="20">
      There were no results matching the query.
</search:container>
```

3.10.2 <search:query />

The <search:query/> tag outputs the current search query.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
Your search for '<search:query />' has returned <search:nhits />
results.
</search:container>
```

3.10.3 <search:nhits />

The <search:nhits/> tag outputs the number of hits found.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
Your search for '<search:query />' has returned <search:nhits />
results.
</search:container>
```

3.10.4 <search:time_overall />

The <search:time_overall /> tag outputs the duration of the search query.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
Your search for ‘<search:query />’ returned <search:nhits /> results
in <search:time_overall /> milliseconds.
</search:container>
```

3.10.5 <search:sort_link />

The <search:sort_link /> tag outputs automatically generated links for sorting by relevance, date or size. To do this, you give the link the sort criterion via an attribute. Switching between ascending and descending sorting takes place automatically and is adopted from the tag.

Attributes:

Attribute	Meaning	Mandatory parameters
sort	This attribute is used to give the sort criterion for the link to be output. Possible values are: score, date and size	Yes

Simple example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<a href="<search:sort_link sort="score"/>" >Sort by relevance</a>
<a href="<search:sort_link sort="date"/>" >Sort by date</a>
<a href="<search:sort_link sort="size"/>" >Sort by size</a>
</search:container>
```

Advanced example:

If you would like to hide the link to the current sort order and in addition, also indicate whether a click on a link allows sorting in ascending or descending order, you must get the current values from the session beforehand:

```
<%  final String currentSort = (String)
session.getAttribute(de.espirit.ps.exalead.resources.SearchRequestAr
guments.SESSION_SORT);
    final String currentOrder = (String)
session.getAttribute(de.espirit.ps.exalead.resources.SearchRequestAr
guments.SESSION_ORDER); %>
```

These values can then be used to more precisely specify output of the links:

```
<% if ("date".equals(currentSort) || "size".equals(currentSort)) {
%>

<!—- Linked output -->
<a href="<search:sort_link sort="score"/>" >By relevance</a>

<% } else { %>

<!-- Output without linking because no reversion of the sort order
is desired when displaying search results by relevance -->
<b>By relevance</b>

<% }
   if (!"date".equals(currentSort)) { %>

<!-- Output without displaying the sort order because results are
always sorted descending initially -->
<a href="<search:sort_link sort="date"/>" >By date</a>

<% } else { if ("1".equals(currentOrder)) { %>

<!-- If the results have been already sorted by date, the sort order
will be reversed by another click on the link -->
<b>By date</b> <a href="<search:sort_link sort="date"/>"
>downwards</a>


<% }  else { %>

<!-- If the results have been already sorted by date, the sort order
will be reversed by another click on the link -->
<b>By date</b> <a href="<search:sort_link sort="date"/>"
>upwards</a>

<% } }
   if (!"size".equals(currentSort)) { %>

<!-- Output without displaying the sort order because results are
always sorted descending initially -->
<a href="<search:sort_link sort="size"/>" >By size</a>

<% } else { if ("1".equals(currentOrder)) { %>

<!-- If the results have been already sorted by size, the sort order
will be reversed by another click on the link -->
<b>By size</b> <a href="<search:sort_link sort="size"/>"
>upwards</a>

<% } else { %>

<!-- If the results have been already sorted by size, the sort order
will be reversed by another click on the link -->
<b>By size</b> <a href="<search:sort_link sort="size"/>"
>downwards</a>

<% } } %>
```

3.10.6 <search:sort_linkparameter />

The function corresponds to that of the <search:sort_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

Example:

```
<a href="do.search?<search:sort_linkparameter sort="date"/>" >Sort
by date</a>
```

3.10.7 <search:suggestions_loop>

The <search:suggestions_loop> tag iterates over all search word suggestions and provides the respective suggestion with the corresponding search link.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
<% if ("true".equals(pageContext.getAttribute("hasSuggestions"))) {
%>
      Did you mean
      <search:suggestions_loop>
<a href="<search:suggestions_link/>">
      <search:suggestions_choice/>
</a>
      </search:suggestions_loop> ?
<% } %>
</search:container>
```

3.10.8 <search:suggestions_link />

The <search:suggestions_link/> tag shows the link for searching with the suggested search word.

3.10.9 <search:suggestions_linkparameter />

The function corresponds to that of the <search:suggestions_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.10.10 <search:suggestions_choice />

The <search:suggestions_choice/> tag shows the suggested search word.

3.11 Tags for refining the search results

3.11.1 <search:groups>

The <search:groups> tag defines the area for the display of the available search categories and the search refinements made

3.11.2 <search:group>

The <search:group> tag is called within the <search:groups> tag and contains the available entries of a specific search category, which the tag is notified of by means of an attribute. If this search category does not contain any entries the tag hides its complete contents.

Attributes:

Attribute	**Meaning**	**Mandatory parameters**
groupID	Name of the search category whose entries are to be displayed. If no entries are available the content is hidden.	Yes

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
      <search:groups>
            <search:group groupID="Categorie1">
                  Display of entries of Categorie1
            </search:group/>
            <search:group groupID="Categorie2">
                  Display of entries of Categorie2
            </search:group/>
      </search:groups>
</search:container>
```

3.11.3 <search:groups_categories_loop>

The <search:groups_categories_loop> tag iterates over all entries of a search category and makes the displayable information of these entries available.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
      <search:groups>
            <search:group groupID="Categorie1">
                  <search:groups_categories_loop>
<a href="<search:groups_category_link/>">
<search:groups_category_title/>
</a>
(<search:groups_category_count/>)
<a href="<search:groups_category_link action="exclude"/>">
exclude
</a>
                  </search:groups_categories_loop>
            </search:group/>
      </search:groups>
</search:container>
```

Within this tag, the PageContext variable "level" can be used to define the degree of indentation of the category.

3.11.4 <search:groups_category_title />

This tag outputs the title of the entry.

Attributes:

Attribute	Meaning	Mandatory parameters
trimAt	Characters or character strings, which give the place in the title from which this is to be displayed. Example: The unshortened title is text#html. With trimAt="#", only html is given as the title.	No

Instead of the tag, the title can also be output within the group_categories_loop tag using the "title" PageContext variable.

For an example, see Chapter 3.11.3 page 36.

3.11.5 <search:groups_category_count />

This tag outputs the number of search hits for this entry.

Instead of the tag, the number of search hits can also be output within the group_categories_loop tag using the "count" PageContext variable.

For an example, see Chapter 3.11.3 page 36.

3.11.6 <search:groups_category_link />

This tag outputs the link for limiting the search results to this entry or to exclude the search results for this entry.

Attributes:

Attribute	Meaning	Mandatory parameters
action	If action="include" (default value), the link for limiting the search results to this entry is output. If action="exclude" the link for excluding the search results of this entry is output.	No

For an example, see Chapter 3.11.3 page 36.

3.11.7 <search:groups_category_linkparameter />

The function corresponds to that of the <search:groups_category_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.11.8 <search:reset_refinement>

The <search:reset_refinements> tag defines the area for the display of the search refinements made. The content of the tag is hidden if no refinements have been made.

If refinements have been made and the content of the tag is shown, the pagecontext variable "resetUrl“ can be used to get a link for removing all the refinements made. Alternatively, the Pagecontext variable "resetUrlParameter" can be used to only query the parameters of the reset link.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
      <search:groups>
            <search:reset_refinement>
<a href="<%= resetUrl %>">Reset all refinements</a>
            </search:reset_refinement>
      </search:groups>
</search:container>
```

3.11.9 <search:hasRefinements>

The <search:hasRefinements> tag is called within the <search:reset_refinement> tag and contains the refinements to a specific search category made, which the tag was notified of by means of an attribute. If no refinements have been made for this search category, the tag hides its complete contents.

Attributes:

Attribute	**Meaning**	**Mandatory parameters**
groupID	Name of the search category whose refinements are to be displayed. If no refinements exist, the content is hidden.	Yes

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
      <search:groups>
            <search:reset_refinement>
            <search:hasRefinements groupID="Categorie1">
                        Display refinements of Categorie1
                  </search:hasRefinements />
            <search:hasRefinements groupID="Categorie2">
                        Display refinements of Categorie2
                  </search:hasRefinements />
            </search:reset_refinement>
      </search:groups>
</search:container>
```

3.11.10 <search:refinements>

The <search:refinements> tag iterates over all refinements made to a search category.

Attributes:

Attribute	Meaning	Mandatory parameters
trimAt	Characters or character strings, which give the place in the title from which this is to be displayed. Example: The unshortened title is text#html. With trimAt="#", only html is given as the title.	No

Within the tag, the pagecontext variables "isExcluded" and "isIncluded" can be used to check whether the refinement involves a limitation or whether it is an exclusion rule. The "title" variable returns the names of the refinement. The PageContext variable "level" can be used to define the degree of indentation of the category.

Example:

```
<search:container hasResults="true" redirectUrl="search.jsp"
errorUrl="error.jsp">
      <search:groups>
            <search:reset_refinement>
            <search:hasRefinements groupID="Categorie1">
                  <search:refinements>
<%= isExcluded ? "NOT " : "" %><%= title %>
<a href="<search:reset_link/>">delete</a>
                  </search:refinements>
                  </search:hasRefinements />
            </search:reset_refinement>
      </search:groups>
</search:container>
```

3.11.11 <search:reset_link />

The <search:reset_link /> tag returns the link for resetting the refinement.

For an example, see Chapter 3.11.10 page 40.

3.11.12 <search:reset_linkparameter />

The function corresponds to that of the <search:reset_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.12 Tags for displaying the search results

3.12.1 <search:loop_hits>

The <search:loop_hits> tag iterates over all search hits of the currently displayed results page and makes the displayable information of these search hits available.

3.12.2 <search:hits_id />

This tag gives the current index of the search hit within the hit list. The first search hit has the ID 0.

Example:

```
<search:loop_hits>
      ID of the search result: <search:hits_id/>
</search:loop_hits>
```

3.12.3 <search:hits_url />

This tag returns the URL of the search hit.

Attributes:

Attribute	**Meaning**	**Mandatory parameters**
completeDocUrl	Set to “false” (default value), the tag returns the URL of the document, as it is saved in the index. However, these links cannot be effortlessly opened in the case of files which cannot be reached by http. completeDocUrl=“true“ is used to extend links to documents which cannot be reached by http to include the path of the DownloadServlet, so that these documents can be delivered.	No

replaceRule	Replacement rule for replacing part of the URL with another value, for example, another server. Syntax: "<old string>,<new string>"	No

Example:

```
<search:loop_hits>
      URL of the document: <a href=”<search:hits_url
completeDocUrl="true"
replaceRule=”myIndexServer,myWebsiteServer”/>”>
<search:hits_url replaceRule=”myIndexServer,myWebsiteServer”/>
</a>
</search:loop_hits>
```

3.12.4 <search:hits_title />

This tag returns the title of the search hit.

Attributes:

Attribute	**Meaning**	**Mandatory parameters**
highlightStart / highlightEnd	These attributes can be used to define the way in which the search word is highlighted within the title.	No

Example:

```
<search:loop_hits>
      Title: <search:hits_title highlightStart="<strong>"
highlightEnd="</strong>" />
</search:loop_hits>
```

3.12.5 <search:hit_doctype />

This tag returns the document type of the search hit in the "doctype" PageContext variable.

Attributes:

Attribute	Meaning	Mandatory parameters
completeDoctype	If true is given the complete document type is returned. If false is given, only the part after the first '/' is returned.	No

Example:

```
<search:loop_hits>
      Typ:
       <search:hits_doctype completeDoctype="false">
          <% if ("html".equals(doctype)) { %>[HTML]<% }
             else if ("pdf".equals(doctype)) { %>[PDF]<% }
             else { %>[unknown]<% } %>
       </search:hits_doctype>
</search:loop_hits>
```

3.12.6 <search:hits_score />

This tag returns the score (number of points) of the search hit.

Example:

```
<search:loop_hits>
      Type: <search:hits_score />
</search:loop_hits>
```

3.12.7 <search:hits_summary />

This tag returns an extract of the search hit.

Attributes:

Attribute	Meaning	Mandatory parameters
isLongForm	This attribute is used to control whether the first line of the extract only (isLongForm = false) or whether the complete extract (isLongForm = true) is to be shown. The default value is false.	No
endline	This attribute can be used to define a text snippet, which is appended to the end of each line of the extract (e.g. "...<br/>", to end each line with three dots and a line break).	No
highlightStart / highlightEnd	These attributes can be used to define the way in which the search word is highlighted within the text extract.	No

Example:

```
<search:loop_hits>
      <search:hits_summary isLongForm="true" endline=" ...<br/>"
highlightStart="<strong>" highlightEnd="</strong>" />
</search:loop_hits>
```

3.12.8 <search:hits_field />

This tag can be used to read out additional index fields (e.g. metadata from FirstSpirit, see Chapter 5.2.3, or metadata, which was given in the HTML header of a page). The fields which can be read out must be defined beforehand using Exalead.

Attributes:

Attribute	Meaning	Mandatory parameters
Key	Name of the index field to be output.	Yes

Example:

```
<search:loop_hits>
      <search:hits_field key="myCustomAttribute">
            <%= value %>
      </search:hits_field>
</search:loop_hits>
```

3.12.9 <search:hits_filesize />

This tag returns the size of the document.

Attributes:

Attribute	Meaning	Mandatory parameters
type	Units in which the size is to be output. Possible values: b, kb, mb, gb. If this attribute is omitted, a suitable unit is automatically selected.	No

Example:

```
<search:loop_hits>
      File size: <search:hits_filesize type=”kb” />
</search:loop_hits>
```

3.12.10 <search:hits_date />

This tag returns the date on which the document was created.

Attributes:

Attribute	Meaning	Mandatory parameters
format	The output format of the date, e.g. dd.MM.yy	Yes

Example:

```
<search:loop_hits>
      Date: <search:hits_date format="dd.MM.yy" />
</search:loop_hits>
```

3.12.11 <search:hits_thumbnail>

If thumbnails are to be displayed in a web page for the search hit, the corresponding area must be enclosed by this tag.

Example:

```
<search:loop_hits>
      <search:hits_thumbnail>
            <search:hits_hasThumbnail>
                  <img src="/exalead/do.searchThumbnail?index=<%=
index %>" alt="" border="0" />
            </search:hits_hasThumbnail>
            <search:hits_hasNoThumbnail>
                  <img src="images/no_thumbnail.gif" alt=""
border="0" />
            </search:hits_hasNoThumbnail></a>
      </search:hits_thumbnail>
</search:loop_hits>
```

3.12.12 <search:hits_hasThumbnail>

This tag checks whether a preview image exists for a search hit. If this is the case, the content of the tag is shown, otherwise it is hidden.

For an example, see Chapter 3.12.11 page 47.

3.12.13 <search:hits_hasNoThumbnail>

This tag checks whether a preview image exists for a search hit. If this is **not** the case, the content of the tag is shown, otherwise it is hidden.

For an example, see Chapter 3.12.11 page 47.

3.12.14 <search:is_in_category>

This tag can be used to check whether a search hit belongs to a specific category. If this is the case, the contents of the tag is shown.

Attributes:

Attribute	Meaning	Mandatory parameters
fullPath	The absolute path to the category.	Yes

Example:

```
<search:loop_hits>
      <search:is_in_category fullPath="Top/FirstSpirit/common">This
hit does not belong to the categorie FirstSpirit/common
      </search:is_in_category>
      <search:is_not_in_category
fullPath="Top/FirstSpirit/common">This hit does not belong to the
categorie  FirstSpirit/common
      </search:is_not_in_category>
</search:loop_hits>
```

3.12.15 <search:is_not_in_category>

This tag can be used to check whether a search hit belongs to a specific category. The content of the tag is shown if this is **not** the case.

Attributes:

Attribute	Meaning	Mandatory parameters
fullPath	The absolute path to the category.	Yes

For an example, see Chapter 3.12.14 page 48.

3.13 Tags for the navigation

3.13.1 <search:navigation>

The <search:navigation> tag defines the area for displaying the navigation through the search results pages.

Attributes:

Attribute	Meaning	Mandatory parameters
surroundingPagesRadius	Defines the page radius of the navigation around the current page. If no value or a values smaller than 5 is given, a radius of 5 is used.	No

Example:

```
<search:navigation surroundingPagesRadius="5">
      ... navigation bar ...
</search:navigation>
```

3.13.2 <search:navigation_page_first_link />

This tag generates the link to the first results page.

Example:

```
<search:navigation surroundingPagesRadius="5">
      <a href="<search:navigation_page_first_link />">First page</a>
</search:navigation>
```

3.13.3 <search:navigation_page_first_linkparameter />

The function corresponds to that of the <search:navigation_page_first_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.13.4 <search:navigation_page_previous_container>

This tag checks whether the current search results page is the first page and only outputs the content of the tag if this is not the case.

Example:

```
<search:navigation surroundingPagesRadius="5">
      <search:navigation_page_previous_container>
            <a href="<search:navigation_page_previous_link/>">
                  Previous page
            </a>
      </search:navigation_page_previous_container>
</search:navigation>
```

3.13.5 <search:navigation_page_previous_link />

This tag generates the link to the previous page in the results page navigation.

For an example, see Chapter 3.13.4 page 50.

3.13.6 <search:navigation_page_previous_linkparameter />

The function corresponds to that of the <search:navigation_page_previous_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.13.7 <search:navigation_loop_pages>

This tag iterates over the search results pages of the navigation, depending on the set pages radius.

Example:

```
<search:navigation surroundingPagesRadius="5">
      <search:navigation_loop_pages>
            <search:navigation_is_current_page>
                  <strong><search:navigation_page/></strong>
            </search:navigation_is_current_page>
            <search:navigation_is_not_current_page>
                  <a href="<search:navigation_page_link/>">
                        <search:navigation_page/>
                  </a>
            </search:navigation_is_not_current_page>
      </search:navigation_loop_pages>
</search:navigation>
```

3.13.8 <search:navigation_is_current_page>

This tag checks whether the page from the iteration is the currently displayed search results page. If this is the case, the contents of the tag is shown.

For an example, see Chapter 3.13.7 page 50.

3.13.9 <search:navigation_is_not_current_page>

This tag checks whether the page from the iteration is **not** the currently displayed search results page. If this is the case, the contents of the tag is shown.

For an example, see Chapter 3.13.7 page 50.

3.13.10 <search:navigation_page />

This tag outputs the page number of the search results page.

For an example, see Chapter 3.13.7 page 50.

3.13.11 <search:navigation_page_link />

This tag generates the link to the search results page.

For an example, see Chapter 3.13.7 page 50.

3.13.12 <search:navigation_page_linkparameter />

The function corresponds to that of the <search:navigation_page_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.13.13 <search:navigation_page_next_container>

This tag checks whether the current search results page is the last page and only outputs the content of the tag is this is not the case.

Example:

```
<search:navigation surroundingPagesRadius="5">
      <search:navigation_page_next_container>
            <a href="<search:navigation_page_next_link/>">
                  Next page
            </a>
      </search:navigation_page_next_container>
</search:navigation>
```

3.13.14 <search:navigation_page_next_link />

This tag generates the link to the next page in the results page navigation.

For an example, see Chapter 3.13.13 page 51.

3.13.15 <search:navigation_page_next_linkparameter />

The function corresponds to that of the <search:navigation_page_next_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.13.16 <search:navigation_page_last_container>

This tag checks whether the page number of the last page is known or not. If there is a very large quantity of hits, it is possible that the number of hits and therefore the number of results pages too, can only be estimated. If this is the case, the contents of this tag is hidden.

Example:

```
<search:navigation surroundingPagesRadius="5">
      <search:navigation_page_last_container>
            <a href="<search:navigation_page_last_link/>">
            Last page(<search:navigation_page_last/>)
            </a>
      </search:navigation_page_last_container>
</search:navigation>
```

3.13.17 <search:navigation_page_last />

This tag gives the page number of the last search results page.

For an example, see Chapter 3.13.16 page 52.

3.13.18 <search:navigation_page_last_link />

This tag generates the link to the last search results page.

For an example, see Chapter 3.13.16 page 52.

3.13.19 <search:navigation_page_last_linkparameter />

The function corresponds to that of the <search:navigation_page_last_link/> tag, with the single difference that this tag only generates the URL parameters for the required servlet call.

3.14 Implicit search

3.14.1 <search:loop_tophits>

The <search:loop_tophits> tag is used to display the top N search results for a specific predetermined search word or for the current search query; however, independent of any search refinements.

Attributes:

Attribute	**Meaning**	**Mandatory parameters**
query	Specification of the search word (term). If nothing is specified, the current search word of the normal search is used.	No
filetype	Limitation to search results of a specific file type.	No
numberOfHits	Maximum number of displayed hits. Default value: 5	No

Example:

```
<search:loop_tophits query="internet" filetype="pdf"
numberOfHits="3">
      <a href="<search:hits_url completeDocUrl="true"/>">
            <search:hits_title/>
      </a><br/>
</search:loop_tophits>
```

The same tags can be used within the <search:loop_tophits> tag as within the <search:loop_hits> tag, with the exception of the display of thumbnails.

4 Personalized Search Results

In order to achieve personalized display of the search results, Exalead gives users the opportunity to read out user and group information from documents and to display these documents in the list of the search results to authorized users only.

This chapter deals with the possibilities of assigning permissions to documents. In this context, it also explains use of the Push-API, which allows documents to be individually transferred into the index.

In order for personalized search results to be displayed, the SearchServlet must send a corresponding SecurityToken with each query. This must be activated in the web.xml, as by default the search queries are performed without a SecurityToken. See Chapter 3.7.1 page 19.

In the case of search queries without a SecurityToken, all documents found are returned as the search result, regardless of the assigned permissions. However, this is only the case as long as the Exalead Search API is not protected and can therefore be used without a SecurityToken. To change this behavior, the Exalead Search API can be configured so that use of SecurityTokens is forced (Figure 4-1 Protecting the search API).

4.1 Personalized search in HTML pages

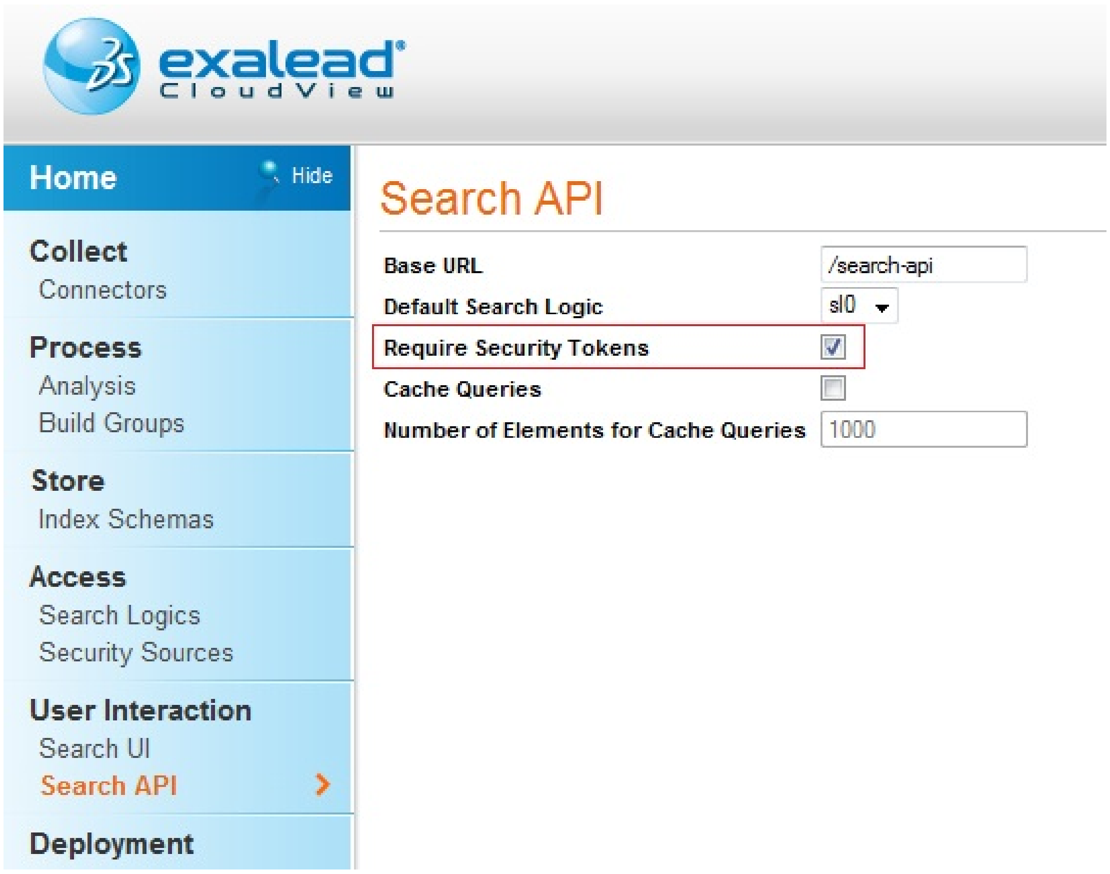
An easy way to assign permissions to HTML pages is to specify meta information, which can be read out by Exalead and used to determine the user and group guidelines.

4.2 Preparation of the HTML files / templates

The groups or users are entered in the HTML file as a MetaTag in the following form:

```
<meta name="security"
content="exalead:group:AnyGroup,exalead:user:Jane Doe" />
```

Any name can be chosen for the MetaTag (in the example: security). The `content` attribute is used to assigned the permissions for this document. This should always consist of a prefix and the user or group name. **Important:** The prefix for groups (here `exalead:group:`) should differ from the prefix for user names (here `exalead:user:`).



Search API

Base URL	/search-api
Default Search Logic	sl0
Require Security Tokens	✓
Cache Queries	☐
Number of Elements for Cache Queries	1000

Figure 4-1 Protecting the search API

Apart from one exception, any prefixes can be chosen (see also Chapter 3.7.1).

The MetaTag may only occur once in the document. Several group and user permissions can be given within the MetaTag as a comma-separated list:

```
<meta name="security"
content="exalead:group:editor,exalead:group:guest,exalead:group:Admi
n,exalead:user:John Doe,exalead:user:Jane Doe" />
```

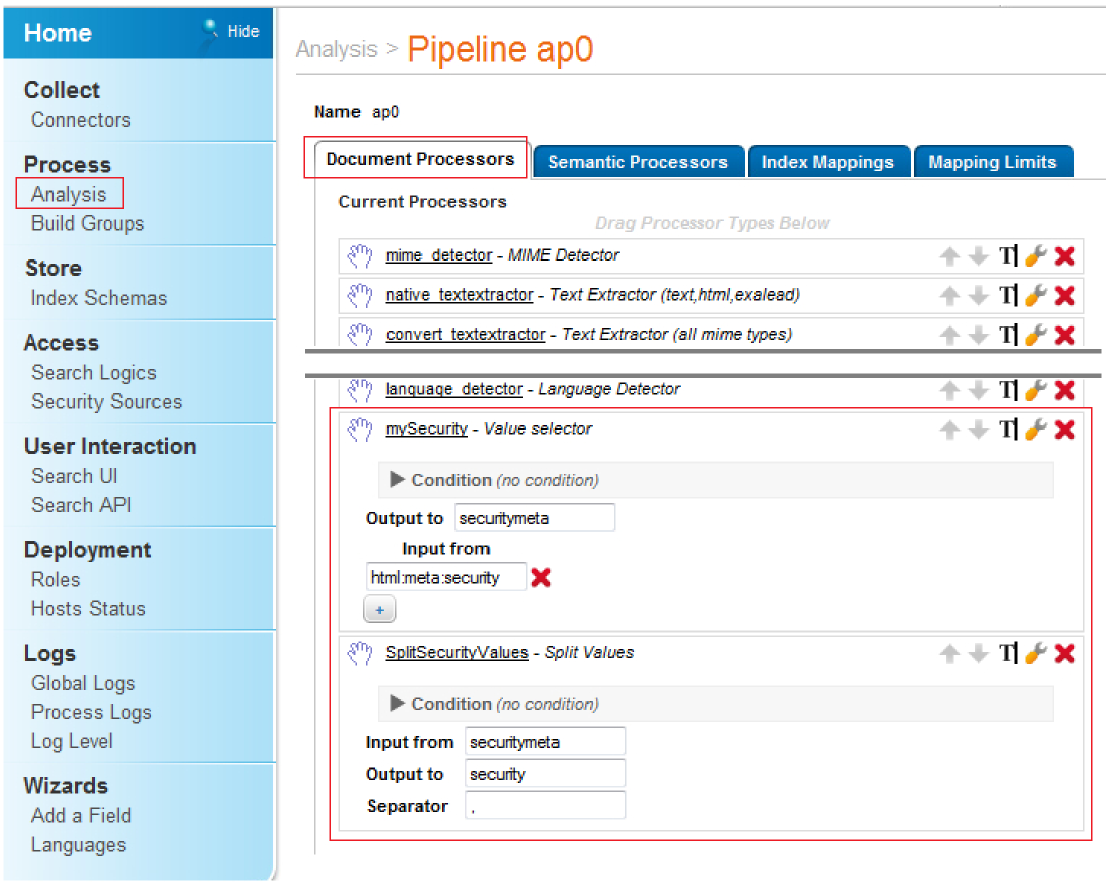
Pages, which are to be displayed to every user in the search results must be explicitly additionally assigned to the `fs_public` group:

```
<meta name="security" content="
exalead:group:editor,exalead:group:Admin,exalead:group:fs_public" />
```

As a reminder: These permissions are only evaluated if use of a SecurityToken is activated (for details of configuration of use of the SecurityToken see Chapter 3.7.1 "SearchServlet").

4.3 Setting up the document processors

After the HTML files have been adjusted accordingly, two document processors are now set up in Exalead, which on the one hand read out the individual values of the comma-separated list given in the MetaTag and on the other hand assign these values to the document as security information (see Figure 4-2 Configuring the document processors).



Name ap0

mime_detector - MIME Detector
native_textextractor - Text Extractor (text,html,exalead)
convert_textextractor - Text Extractor (all mime types)
language_detector - Language Detector
mySecurity - Value selector
Output to: securitymeta
Input from: html:meta:security
SplitSecurityValues - Split Values
Input from: securitymeta
Output to: security
Separator: ,

Figure 4-2 Configuring the document processors

To do this, take the following steps:

1. Create a new document processor of the type "Value selector" ("Chunk Operations" category). Any value can be chosen for "Output to". The value for "Input from" must be `html:meta:` followed by the name of the MetaTag from the HTML page.

2. Create a new document processor of the type "Split Values" ("Text Operations" category). The value for "Input from" must correspond to the value for "Output to" from the value selector. Here the value for "Output to" must be `security`. The separator must correspond to the separator in the

MetaTag of the HTML page.

After completing these steps, the new document processors created are now applied in future indexing of the documents; which read out the users and groups from the designated security MetaTag and assign them to the document as user and group guidelines.

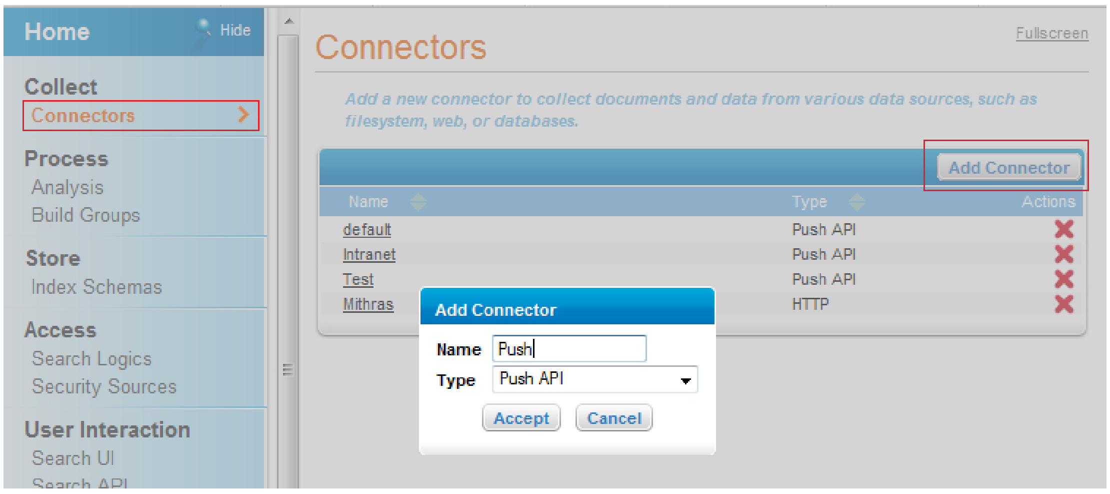
5 Incremental update via Push-API

Exalead's Push-API provides an interface which can be used to directly write ("push") documents in the index of the search engine. As a result, changes to documents can be made immediately known to the index, without having to wait for the next crawling occasion.

The steps necessary to configure the Push-API are explained in the following. The FirstSpirit sub-module "Exalead Content-Update", which is based on the Push-API, is then described.

5.1 Setting up a Push Connector

First, a new Push Connector must be set up on the Exalead server (see Figure 5-1 Setting up a Push Connector). Any name can be chosen for the connector.



Add a new connector to collect documents and data from various data sources, such as filesystem, web, or databases.

Name	Type	Actions
default	Push API	✗
Intranet	Push API	✗
Test	Push API	✗
Mithras	HTTP	✗

Add Connector

Name	Push
Type	Push API

Figure 5-1 Setting up a Push Connector

This connector should now be protected with a login and password. To do this, the API console (can be accessed under `http://<exalead-server-host>:<baseport>+1`) must be opened and the "Manage" area in it selected:

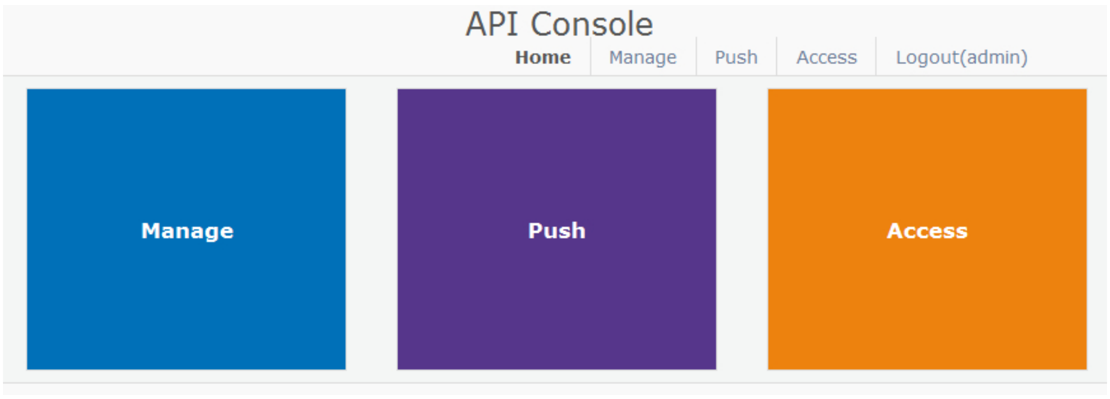
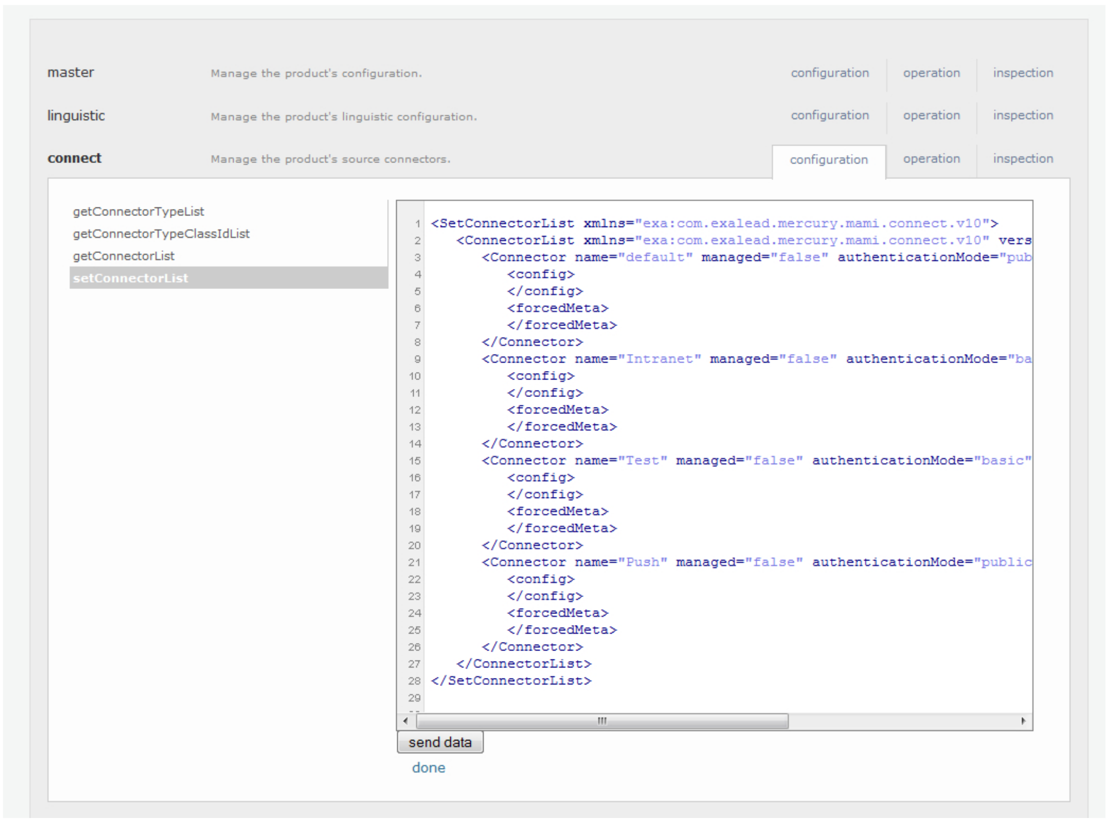


Figure 5-2 API Console

In this area, select the "connect" menu item and in it the "setConnectorList submenu item:



```
<SetConnectorList xmlns="exa:com.exalead.mercury.mami.connect.v10">
   <ConnectorList xmlns="exa:com.exalead.mercury.mami.connect.v10" vers
      <Connector name="default" managed="false" authenticationMode="pub
         <config>
         </config>
         <forcedMeta>
         </forcedMeta>
      </Connector>
      <Connector name="Intranet" managed="false" authenticationMode="ba
         <config>
         </config>
         <forcedMeta>
         </forcedMeta>
      </Connector>
      <Connector name="Test" managed="false" authenticationMode="basic"
         <config>
         </config>
         <forcedMeta>
         </forcedMeta>
      </Connector>
      <Connector name="Push" managed="false" authenticationMode="public
         <config>
         </config>
         <forcedMeta>
         </forcedMeta>
      </Connector>
   </ConnectorList>
</SetConnectorList>
```

Figure 5-3 setConnectorList

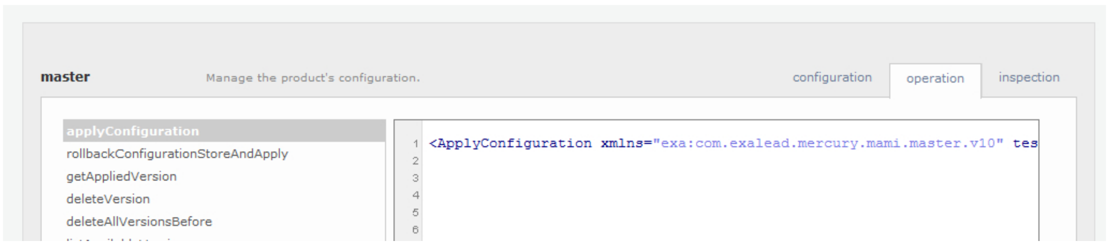
the XML text displayed there contains a section, which refers to the new Push Connector created previously (recognizable by the name assigned to it).

```
<Connector name="Push" managed="false" authenticationMode="public">
    <config>
    </config>
    <forcedMeta>
    </forcedMeta>
</Connector>
```

In this section the value of the `authenticationMode` attribute must now be changed to `basic` and a login and password must be assigned:

```
<Connector name="Push" managed="false" authenticationMode="basic"
login="test" password="test">
    <config>
    </config>
    <forcedMeta>
    </forcedMeta>
</Connector>
```

Then select "send data" to save the changes. To adopt the saved changes you must now call "applyConfiguration". This call is located within the "master" menu item. Here too the "send data" button must be pressed again.



```
<ApplyConfiguration xmlns="exa:com.exalead.mercury.mami.master.v10" tes
```

Figure 5-4 applyConfiguration

5.2 FirstSpirit Module: Exalead Content-Update

5.2.1 Use

The FirstSpirit "Exalead Content-Update" module is used to write content generated in FirstSpirit in the search index by using the Push-API immediately after they have been generated, so that this content is not indexed with the next crawler run, but instead can be indexed immediately.

If permissions assigned via metadata exist, they are adopted and written in the index together with the respective document. This requires installation of the PermissionService in FirstSpirit (see module documentation for *FirstSpirit Personalisation*, Chapter 1.3.2).

In addition, FirstSpirit Personalization is required to enable personalized search queries to be set on the web server side, which use the permissions from the metadata of the PermissionService. Here it must be noted that the group names of the PermissionService must correspond to the group names in FirstSpirit Personalization.

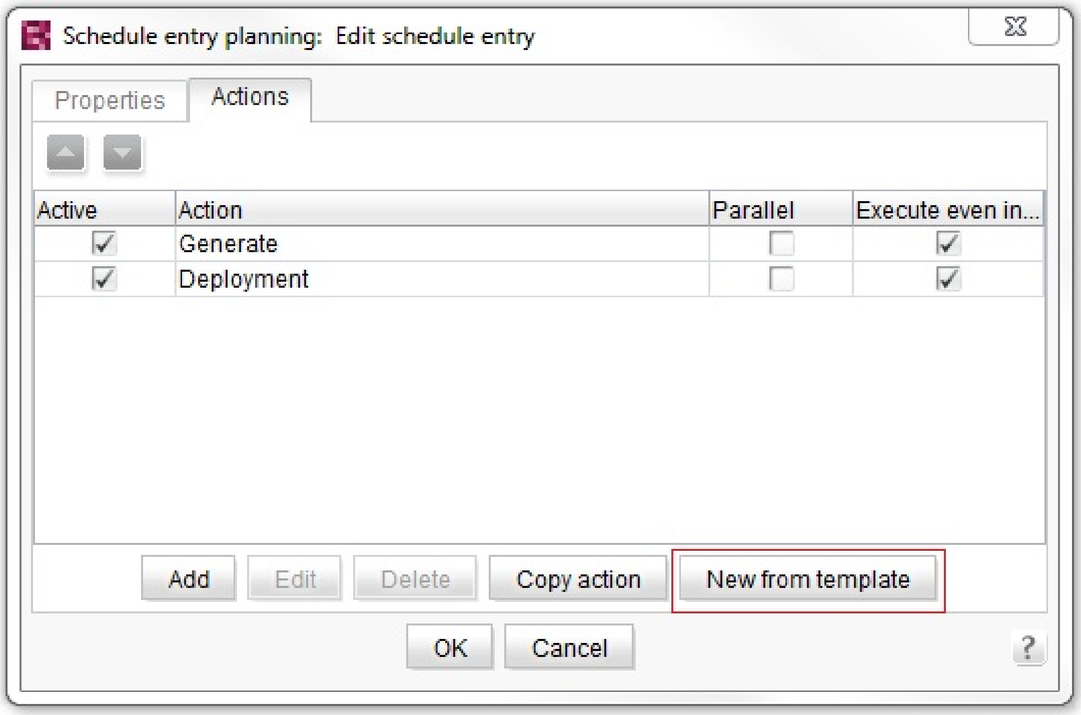
5.2.2 Installation

To install the Exalead Content-Update module, it is only necessary to load the corresponding FSM file onto the FirstSpirit server. The installation is carried out as described in the *FirstSpirit Manual for Administrators*.

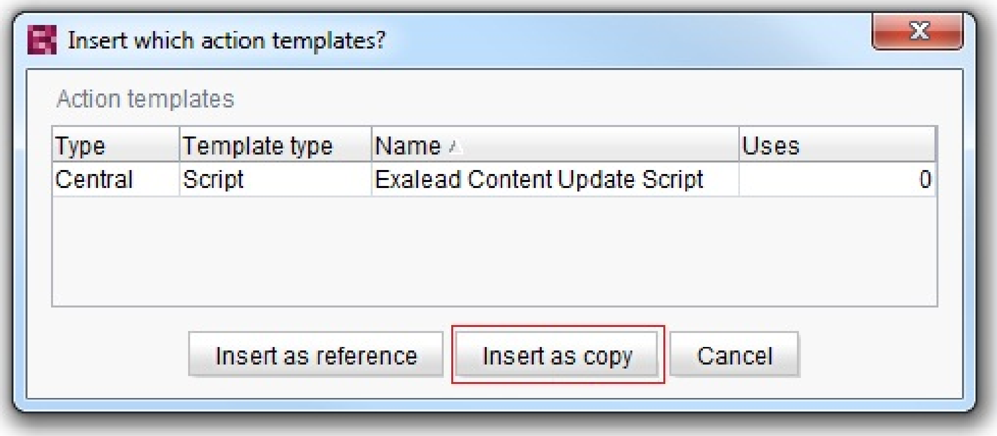
When the module is installed, an action template is automatically created, which can be used within a schedule to generate the action for calling the Push-API helper class.

The action generated in this way contains a script, which initiates transfer of the generated content into the search index. This script must be called within a schedule following a generation.

Note: Within the generation action it is necessary to ensure that use of the ACL database has been enabled there.


Schedule entry planning: Edit schedule entry

Active	Action	Parallel	Execute even in...
✓	Generate	☐	✓
✓	Deployment	☐	✓


Insert which action templates?

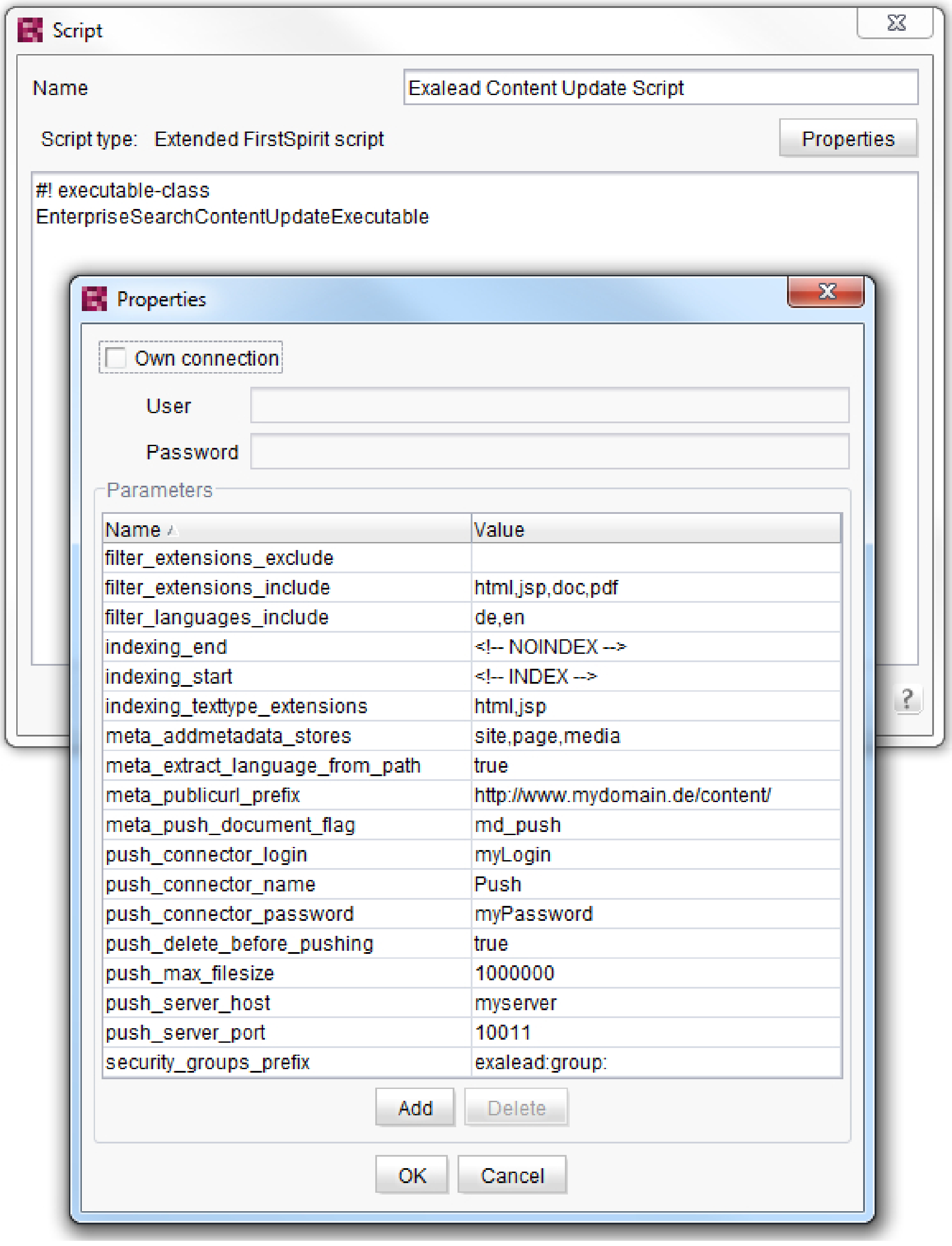
Action templates

Type	Template type	Name	Uses
Central	Script	Exalead Content Update Script	0

Figure 5-5: Adding the action template

5.2.3 Configuration

The script is configured using the script properties, as can be seen in Figure 5-6.



Name Exalead Content Update Script

Script type: Extended FirstSpirit script

```
#! executable-class
EnterpriseSearchContentUpdateExecutable
```

☐ Own connection

User

Password

Parameters

Name	Value
filter_extensions_exclude	
filter_extensions_include	html,jsp,doc,pdf
filter_languages_include	de,en
indexing_end	<!-- NOINDEX -->
indexing_start	<!-- INDEX -->
indexing_texttype_extensions	html,jsp
meta_addmetadata_stores	site,page,media
meta_extract_language_from_path	true
meta_publicurl_prefix	http://www.mydomain.de/content/
meta_push_document_flag	md_push
push_connector_login	myLogin
push_connector_name	Push
push_connector_password	myPassword
push_delete_before_pushing	true
push_max_filesize	1000000
push_server_host	myserver
push_server_port	10011
security_groups_prefix	exalead:group:

Figure 5-6: Script properties

`filter_extensions_exclude`

This parameter accepts a comma-separate list of file types which are not to be written in the search index. This parameter is only evaluated if `filter_extensions_include` is empty.

If both `filter_extensions_include` and `filter_extensions_exclude` are empty, all generated files (incl. referenced pictures, stylesheets, etc.) are written in the search index.

`filter_extensions_include`

This parameter accepts a comma-separated list of file types. If this list is not empty, only files which correspond to this list are written into the search index. If the list is empty, in addition, the `filter_extensions_exclude` parameter is evaluated.

`filter_languages_include`

Use of this parameter only makes sense in conjunction with use of `meta_extract_language_from_path`. By giving a comma-separated list of language abbreviations, only the pages and documents whose language is included in this list are written in the index. Documents to which no language is assigned are selected with the abbreviation 'xx'.

Examples:

- No language details given: All documents are written in the index.
- `de`: Only documents in language 'de' are written in the index.
- `de,en`: Documents in languages 'de' and 'en' are written in the index.
- `xx`: Only language-independent documents are written in the index.
- `de,xx`: Documents in language 'de' and language-independent documents are written in the index.

`indexing_end`

This parameter sets the character string, which is used to label the start of a section, which is not to be indexed. The default value is `<!-- NOINDEX -->`

`indexing_start`

This parameter sets the character string, which is used to label the end of a section, which is not to be indexed. The default value is `<!-- INDEX -->`

`indexing_texttype_extensions`

Comma-separated list of file extensions of documents saved in text format. Areas excluded from the indexing can only be defined for such documents. The default value is: `html,jsp`

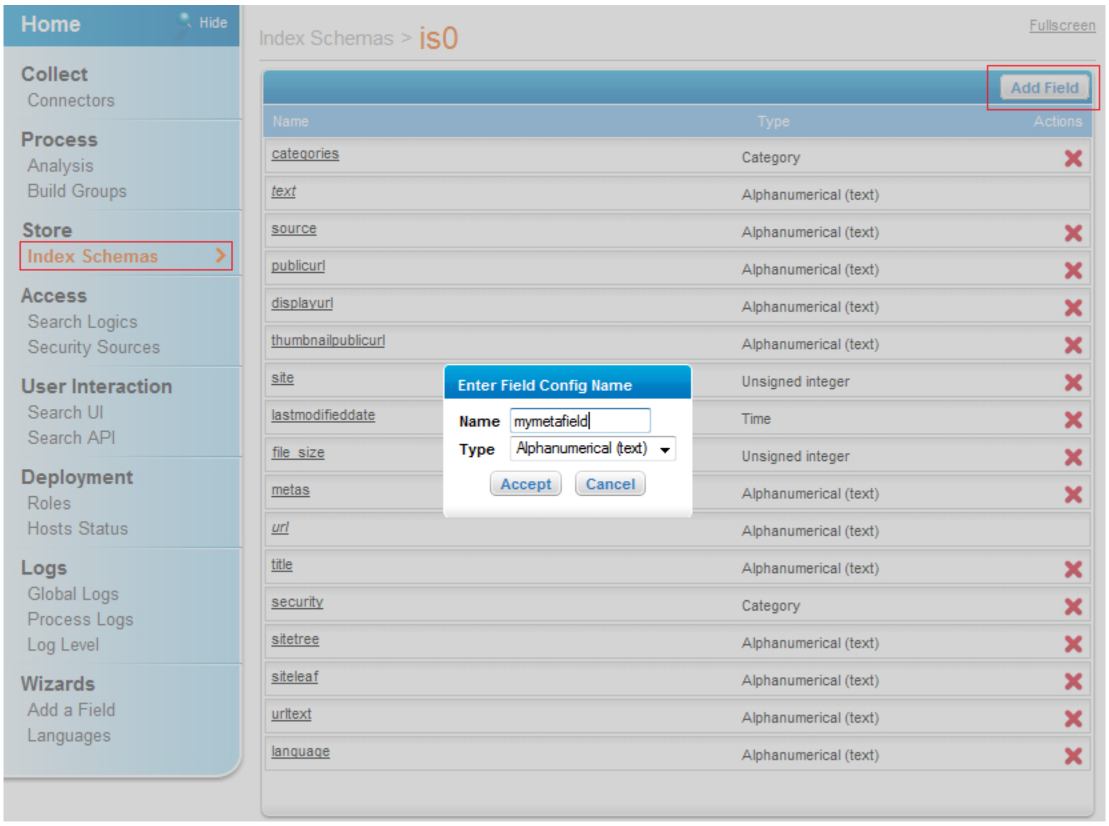
`meta_addmetadata_stores`

This parameter can be used to control whether, in addition to the content of documents, additional metadata is to be written in the index. The value of the parameter – a comma-separated list of the `site`, `media` and `page` values – gives the Store within FirstSpirit, from which the metadata for the document is to be drawn.

Examples:

- `site` = Metadata from the Site Store is used.
- `media,page` = Metadata from the Media Store and Page Store are used and if applicable are combined.

To evaluate the metadata, appropriate settings must be made on the Exalead server side:



Name	Type	Actions
categories	Category	✗
text	Alphanumerical (text)	
source	Alphanumerical (text)	✗
publicurl	Alphanumerical (text)	✗
displayurl	Alphanumerical (text)	✗
thumbnailpublicurl	Alphanumerical (text)	✗
site	Unsigned integer	✗
lastmodifieddate	Time	✗
file_size	Unsigned integer	✗
metas	Alphanumerical (text)	✗
url	Alphanumerical (text)	
title	Alphanumerical (text)	✗
security	Category	✗
sitetree	Alphanumerical (text)	✗
siteleaf	Alphanumerical (text)	✗
urltext	Alphanumerical (text)	✗
language	Alphanumerical (text)	✗

Name: mymetafield
Type: Alphanumerical (text)

Figure 5-7: Creating a new field in the index

First, a new field must be created in the index for each piece of metadata information to be evaluated (see Figure 5-7). Any name can be chosen for the field.

In the next step a new mapping must be created, which reads in the FirstSpirit metadata information and maps it in the index field just created (see Figure 5-8 Creating a mapping).

The name of the mapping must be exactly the same as the name of the metadata from the metadata template in FirstSpirit.

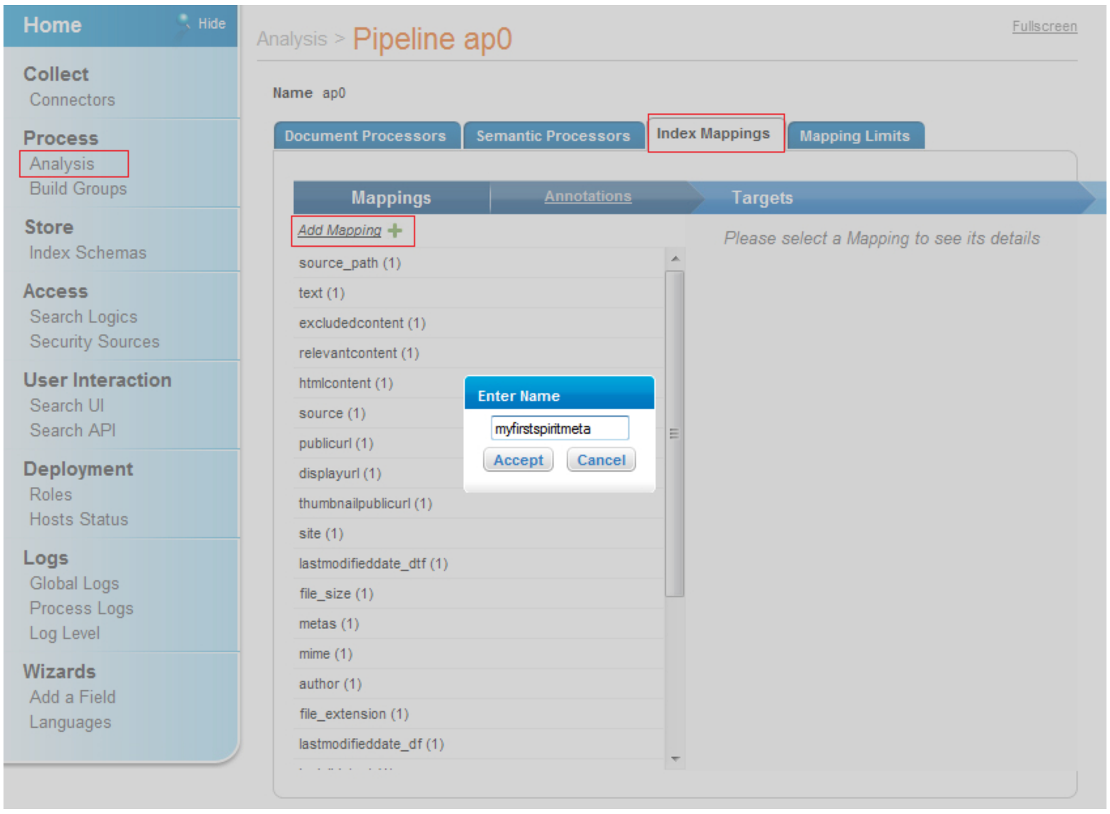
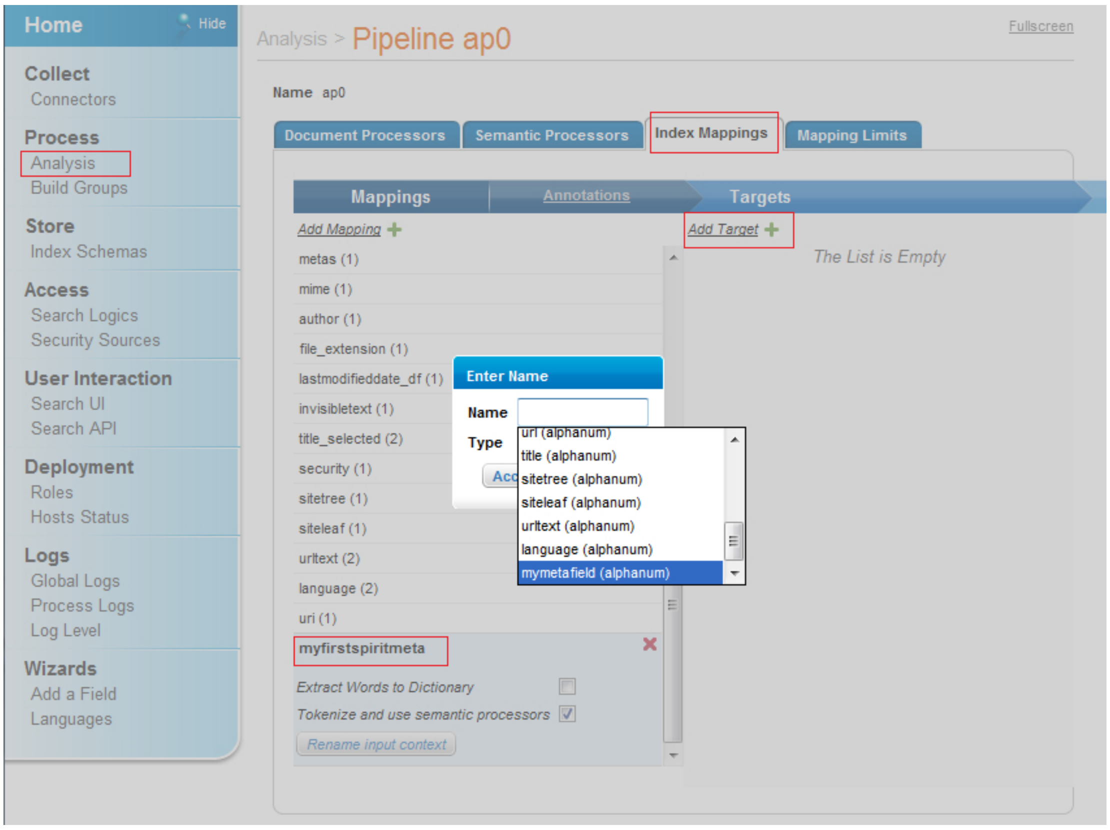


Figure 5-8 Creating a mapping

After creating the mapping, a target must be assigned to the mapping (see Figure 5-9 Adding a mapping target). Choose "Index Field" for the mapping type and the previously created index field as the name.

In order to be able to output the content of the new index field created with a subsequent search query (see also Chapter 3.12.8), they must be selected for display in the "Search Logics" area (see Figure 5-10 Display metadata information in the search result).



Name ap0

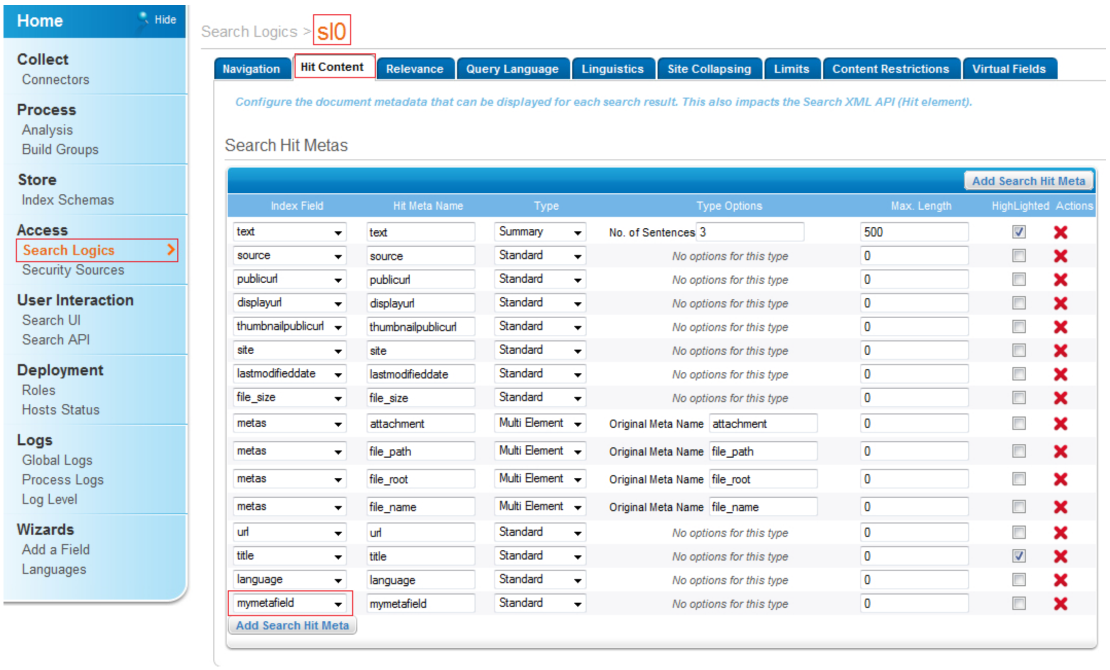
metas (1)
mime (1)
author (1)
file_extension (1)
lastmodifieddate_df (1)
invisibletext (1)
title_selected (2)
security (1)
sitetree (1)
siteleaf (1)
urltext (2)
language (2)
uri (1)
myfirstspiritmeta

Extract Words to Dictionary
Tokenize and use semantic processors ☑

The List is Empty

url (alphanum)
title (alphanum)
sitetree (alphanum)
siteleaf (alphanum)
urltext (alphanum)
language (alphanum)
mymetafield (alphanum)

Figure 5-9 Adding a mapping target



Configure the document metadata that can be displayed for each search result. This also impacts the Search XML API (Hit element).

Search Hit Metas

Index Field	Hit Meta Name	Type	Type Options	Max. Length	HighLighted	Actions
text	text	Summary	No. of Sentences 3	500	☑	✗
source	source	Standard	No options for this type	0	☐	✗
publicurl	publicurl	Standard	No options for this type	0	☐	✗
displayurl	displayurl	Standard	No options for this type	0	☐	✗
thumbnailpublicurl	thumbnailpublicurl	Standard	No options for this type	0	☐	✗
site	site	Standard	No options for this type	0	☐	✗
lastmodifieddate	lastmodifieddate	Standard	No options for this type	0	☐	✗
file_size	file_size	Standard	No options for this type	0	☐	✗
metas	attachment	Multi Element	Original Meta Name attachment	0	☐	✗
metas	file_path	Multi Element	Original Meta Name file_path	0	☐	✗
metas	file_root	Multi Element	Original Meta Name file_root	0	☐	✗
metas	file_name	Multi Element	Original Meta Name file_name	0	☐	✗
url	url	Standard	No options for this type	0	☐	✗
title	title	Standard	No options for this type	0	☑	✗
language	language	Standard	No options for this type	0	☐	✗
mymetafield	mymetafield	Standard	No options for this type	0	☐	✗

Figure 5-10 Display metadata information in the search result

Metadata from the following FirstSpirit input components is supported:

- `CMS_INPUT_TEXT`
- `CMS_INPUT_DATE`
- `CMS_INPUT_COMBOBOX`
- `CMS_INPUT_CHECKBOX`
- `CMS_INPUT_TOGGLE`

For `CMS_INPUT_TEXT` and `CMS_INPUT_DATE` the content of the input component is transferred to Exalead as a value. For `CMS_INPUT_COMBOBOX` and `CMS_INPUT_CHECKBOX` the label of the corresponding selection defined in the metadata template is transferred to Exalead as a value. In the case of `CMS_INPUT_TOGGLE`, `true` is transferred if the component is selected.

`meta_extract_language_from_path`

This parameter can be used to control whether the document language is to be determined using the path of the generated file. If this parameter is not set (or the value `false` is transferred), the Exalead server uses an internal algorithm to decide which language is assigned to a document. By setting this parameter to the value `true`, the language is determined using the language abbreviation found in the file path:

/de/content/start.html	→	Language:German
/en/content/start.html	→	Language:English
/de_1/content.pdf	→	Language:German
/media/en/docs/flyer.pdf	→	Language:English
/media/docs/flyer.pdf	→	No assigned language

However, if using automatic extraction of the document language on the basis of the file path, it must be noted that the name of each directory consisting of two letters located directly below the media directory is interpreted as the language of the documents contained in the directory.

Example:

/media/sg/styleguide.pdf	→	Language:sg

Better:

/media/guides/styleguide.pdf	→	No assigned language

In the case of documents without an assigned language, Exalead performs the analysis and assigns the document to a language.

`meta_publicurl_prefix`

This parameter is used to give the prefix for the paths of the deployed files. At the same time, the generation directory in the absolute path of the generated files is replaced by this prefix, when the files are written in the search index. Thus, in the example above, the path:

`<Generation directory>/de/unternehmen/kontakt.jsp`

becomes the path:

`http://www.mydomain.de/content/de/unternehmen/kontakt.jsp`

`meta_push_document_flag`

This optional parameter can be used to give the name of a metadata input component in FirstSpirit, via which individual documents (pages/media) can be excluded from the indexing.

The input component must be of the type `CMS_INPUT_TOGGLE`. If the checkbox is selected, the corresponding document is not indexed.

Only the value of the input component from the Site or Media Store is evaluated. Selected checkboxes of this input component in the Page Store are not taken into account in the indexing.

Example configuration in the metadata template:

```
<CMS_INPUT_TOGGLE name="md_push" singleLine="no">
    <LANGINFOS>
        <LANGINFO lang="*" label="Disable push"/>
    </LANGINFOS>
</CMS_INPUT_TOGGLE>
```

`push_connector_login`

Login for access to the push connector (an Exalead server setting)

`push_connector_name`

Name of the push connector (an Exalead server setting) `push_connector_password`

Password for access to the push connector (an Exalead server setting)

`push_delete_before_pushing`

This parameter can be used to control whether all documents written in the search index to date are removed before a renewed write process. This is recommended for a full generation; however, this should be disabled for a partial generation.

`push_max_filesize`

This optional parameter can be used to set a maximum size (in bytes) for documents, which are to be transferred to the Exalead server. If the parameter is not given, or no value is defined for it, all documents are transferred, regardless of their size.

`push_server_host`

Name of the Exalead server on the network

`push_server_port`

Port, under which the push connector can be reached (Exalead server setting)

`security_groups_prefix`

This parameter is used to inform the script which prefix it should use on setting the group permissions. The same value as the one used in the configuration of the SearchServlet should be used (see Chapter 3.7.1).

5.2.4 Removing individual pages/documents from the index

To remove pages and/or documents previously written in the index using the Exalead Content-Update module, the module provides a service, which can be accessed from any scripts (e.g. within a deletion workflow).

An instance of the service is required first before it can be used:

```
exaleadService =
context.getConnection().getService("EnterpriseSearchService");
```

In the next step the service must be initialized using the connection data of the Exalead server:

```
exaleadService.initPAPI(String exalead_hostname, String port, String
push_connector, String push_login, String push_password);
```

The parameters must be assigned the following values:

- `exalead_hostname` = Name of the Exalead server
- `port` = Push-API port of the Exalead server (Baseport + 11, e.g.: 10011)
- `push_connector` = Name of the push connector used
- `push_login` = Login for the Push-Connector (empty string, if the Push-Connector is not protected)
- `push_password` = Password for the Push-Connector (empty string, if the Push-Connector is not protected)

Example:

```
exaleadService.initPAPI(myServer, 10011, Push, myLogin, myPassword);
```

After initializing the service, it can now be used to remove pages/documents:

```
exaleadService.removeFromIndex(String document_path, String
projectId);
```

The generation path of the document must be given as `document_path` and the project ID of the project as `projectId`.

Example:

```
exaleadService.removeFromIndex("de/aboutus/company/company.jsp",
2574);
```

This call must be repeated for each project language and for each output channel:

```
exaleadService.removeFromIndex("de/aboutus/company/company.jsp",
2574);
exaleadService.removeFromIndex("en/aboutus/company/company.jsp",
2574);
exaleadService.removeFromIndex("de_1/aboutus/company/company.jsp",
2574);
exaleadService.removeFromIndex("en_1/aboutus/company/company.jsp",
2574);
```

After removing the required documents, a manual commit is required to transfer these changes to the live index of the Exalead server:

```
exaleadService.commitChanges();
```

Without a manual commit, the changes to the index are transferred with the next automatic commit (the interval for the automatic commit can be configured on the Exalead server side).

5.3 The Java Push-API helper class

The Push-API can also be called manually to cover special project requirements. A suitable Java helper class for performing the necessary operations is located in the `exalead_push.jar` file, which is part of the FirstSpirit "Exalead Content-Update" module.

5.3.1 Initialization

The helper class must now be initialized first in order to use the Push-API via Java:

```
de.espirit.ps.exalead.pushapi.PushAPIHelper pHelper = new
de.espirit.ps.exalead.pushapi.PushAPIHelper("myserver", "10011",
"Push", "test", "test");
```

The first parameter of the constructor stands for the server name and the second parameter for the port (`baseport + 11` is the default port of the push service). The third parameter gives the name of the Push-Connector created. The fourth and fifth parameters give the login data assigned in Chapter 5.1.

5.3.2 Public methods

5.3.2.1 sendDocument(PushAPIDocumentValues docValues)

This method passes any individual document (local file system or HTTP resource) into the index. To do this, a PushAPIDocumentValues object is generated first, which must be filled with appropriate values.

Example: Push a file from the local file system

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();

// Required indication: URI = location of the file
pav.setUri("E:/PushTest/MyDocument.pdf");
// Required indication: PublicURL = publicly available path
pav.setPublicURL("http://www.myserver.de/media/pdf/MyDocument.pdf"
);
```

```
// Optional indications:
pav.setAuthor("Me");
pav.setLanguage("de");
pav.setTitle("My document");

// Pushing the document
pHelper.sendDocument(pav);
```

Example: Push an HTTP resource

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();

// Required indication: URI = location of the page
pav.setUri("http://www.myserver.de/content/de/index.html");
// Required indication: PublicURL = publicly available path
pav.setPublicURL("http://www.myserver.de/content/de/index.html");

// Optional indications:
pav.setAuthor("Me");
pav.setLanguage("de");
pav.setTitle("My document");

// Pushing the document
pHelper.sendDocument(pav);
```

In addition, it is also possible to directly specify the content of a document. To do this, an additional byte array can be passed to the PushAPIDocumentValues object. On pushing, the contents of the document are no longer read from the location specified as the URI and the contents of the byte array are passed instead.

Example: Push a byte array

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();

// Required indication: the content as byte array
pav.setContent(myByteArray);
// Required indication: URI (is required internally despite of
indication of the byte array
pav.setUri("http://www.myserver.de/myByteArray.html ");
// Required indication: PublicURL
pav.setPublicURL("http://www.myserver.de/myByteArray.html");
// Required indication: Date
pav.setDate(new Date());
// Required indication: Filename = name of the document
pav.setFilename("myByteArray.html");
// Required indication: Stamp = unique indication for identifying
changes at the document, for example string composed of creation
date and length of the document
pav.setStamp((new Date()).toString + myByteArray.length);

// Optional indications:
pav.setAuthor("Me");
pav.setLanguage("de");
pav.setTitle("My Byte Array Document");

// Pushing the document
pHelper.sendDocument(pav);
```

If a language abbreviation is not set using the setLanguage() method, Exalead performs an internal content check to try to establish the language of the document. If you want to prevent this for documents to which a language has not been assigned, the language can be set to unknown using setLanguage("xx"). Exalead will then no longer check the contents of the document for an ascertainable language.

5.3.2.2 sendDirectory(PushAPIDocumentValues docValues, boolean recursive, String excludeExtensions)

This method passes the complete contents of a directory into the index. The parameter `excludeExtensions` can be used to define a comma-separated list of file extensions, which are excluded from the transfer to the index. It is also possible to define whether or not all subdirectories are to be transferred too.

Example: Push a directory

```
PushAPIDocumentValues pav = new PushAPIDocumentValues();

// Required indication: URI = the directory which is to be pushed
pav.setUri("E:/PushTest/dir");
// Required indication: PublicURL = publicly available path
pav.setPublicURL("http://www.myserver.de/content/" +
PushAPIHelper.PUBLIC_URL_PLACEHOLDER);
// Recursive pushing of the directory
pHelper.sendDirectory(pav, true, "css,js");
```

As, on pushing a complete directory, it is not possible to specify the PublicURL for each individual document, in this case it is also possible to use a wildcard, which is then replaced by the file name of the respective document when the individual documents are pushed. The file name is always the complete path, which begins after the given URI.

Example:

The following file structure exists:

E:\PushTest\dir\a.html
E:\PushTest\dir\b.html
E:\PushTest\dir\subdir1\c.html
E:\PushTest\dir\subdir1\d.html
E:\PushTest\dir\subdir1\subdir2\e.html
E:\PushTest\dir\subdir1\subdir2\f.html

The PublicURLs of these documents are then as follows:

http://www.myserver.de/content/a.html
http://www.myserver.de/content/b.html
http://www.myserver.de/content/subdir1/c.html
http://www.myserver.de/content/subdir1/d.html
http://www.myserver.de/content/subdir1/subdir2/e.html
http://www.myserver.de/content/subdir1/subdir2/f.html

5.3.2.3 resetAllDocuments()

This method is used to remove all pushed documents from the index.

5.3.2.4 long getTransferRate()

This method can be used to query the transfer rate in kb/s of all documents pushed since initialization of the helper class.

5.3.2.5 Date getGlobalStartDate()

Date on which the helper class is initialized

5.3.2.6 long getGlobalByteCounter()

Bytes transferred since initialization of the helper class

5.3.2.7 long getErrorCounter()

Number of errors which have occurred since initialization of the helper class

5.3.2.8 long getDocCounter()

Number of documents pushed since initialization of the helper class

5.3.2.9 resetCounters()

Resets all counters and resets GlobalStartDate to the current date.

5.3.2.10 setMaxFileSize(int maxFileSize)

Maximum size of the documents to be pushed in bytes. Documents which exceed this value are not written in the index.

5.3.2.11 int getMaxFileSize()

Query the set maximum size

5.3.2.12 setCheckpointAndTriggerIndexing()

Calling this method causes Exalead to immediately index the transferred documents.

Otherwise indexing is carried out according to the Build Groups settings.

5.3.2.13 setUserPrefix(String userPrefix)

This method is used to inform the Push-API helper class which prefix it should use on setting the user permissions. The same value as the one used in the configuration of the SearchServlet should be used (see Chapter 3.7.1).

5.3.2.14 setGroupsPrefix(String groupsPrefix)

This method is used to inform the Push-API helper class which prefix it should use on setting the group permissions. The same value as the one used in the configuration of the SearchServlet should be used (see Chapter 3.7.1).

5.3.2.15 setNoIndexStart(String noIndexStart)

This method overwrites the default value for the label of the start of an area within an HTML document which is not to be indexed. The default value is:

```
<!-- NOINDEX -->
```

5.3.2.16 setIndexStart(String indexStart)

This method overwrites the default value for the label of the end of an area within an HTML document which is not to be indexed (= start of the area to be indexed). The default value is:

```
<!-- INDEX -->
```

5.3.2.17 setTextTypeExtensions(String extensions)

This method is used to define a comma-separated list of file extensions of documents saved in text format. Areas excluded from the indexing can only be defined for such documents. The default value is:

```
html,jsp
```

5.4 Assigning permissions via the Push-API

If you would like to assign documents permissions on pushing them via the Push-API, this is done by setting the permissions in the corresponding PushAPIDocumentValues object:

```
List<String> users = new ArrayList<String>();
users.add("user1");
users.add("user2");
pav.setUsers(users);
List<String> groups = new ArrayList<String>();
groups.add("group3");
groups.add("group4");
pav.setGroups(groups);
```

In this case, unlike setting permissions via MetaTags (see Chapter 4.2), a prefix is not specified, as this is automatically added by the Push-API helper class (see Chapter 5.3.2.13 and 5.3.2.14).

5.5 Excluding areas from the indexing

Within documents written in the index via the Push-API, it is possible to use special NoIndex tags to exclude certain areas from the indexing. This is useful, e.g. if you do not want to index the navigation within an HTML page.

The start of an area that is not to be indexed is labeled with the following tag:

```
<!-- NOINDEX -->
```

The end of such an area is labeled with the following tag:

```
<!-- INDEX -->
```

Comment: It is **not** possible to nest this tag, which must be suitably taken into account when creating FirstSpirit templates.

Comment: These tags correspond to the default setting and can be overwritten using the relevant methods of the Push-API helper class (see Chapter 5.3.2.15 and Chapter 5.3.2.16).

Comment: Such areas are only recognized in documents saved in text format, not in binary format. Therefore, it is necessary to make the file types that exist in text format known to the Push-API helper class using the appropriate method (see

Chapter 5.3.2.17).

6 Autocompletion (SuggestService)

With SuggestService, Exalead Cloudview provides an interface, which enables suggestions for autocompletion of the search term to be made to the user in realtime while they are entering a search term. Examples for the interface are shown in Figure 6-1 and Figure 6-2.

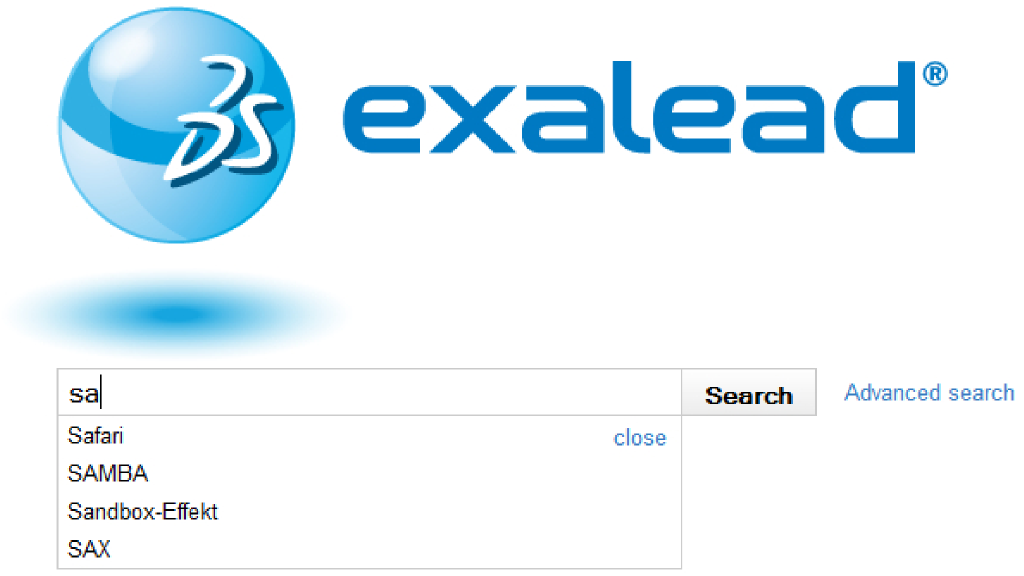


Figure 6-1: Search suggestions in the Exalead search interface

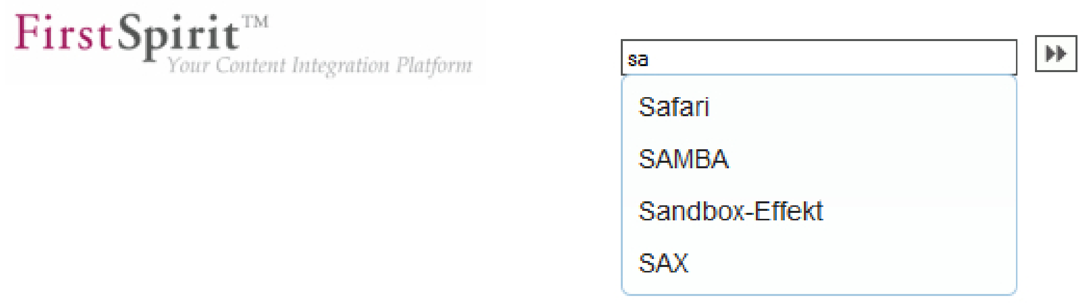


Figure 6-2: Search suggestions in the Exalead Live module

The setting up of a SuggestService is divided into the following four steps:

1. Creating a dictionary file
2. Compiling the dictionary file
3. Setting up the SuggestService on the Exalead server
4. Adjusting the search template

The last step only has to be performed once perSuggestService (it is possible to set up several SuggestServices, however, of which only one can be selected at any time). The first two steps on the other hand must be performed with each update of the SuggestService.

6.1 Creating a dictionary file

The SuggestService of Exalead is based on a suggestion dictionary, which must exist in the form of a special XML file:

```
<SuggestDictionary xmlns="exa:com.exalead.mot.suggest.v10"
maxEntries="10">
    <SuggestDictEntry entry="Safari" score="1" />
    <SuggestDictEntry entry="SAMBA" score="1" />
    <SuggestDictEntry entry="Sandbox-Effekt" score="1" />
    <SuggestDictEntry entry="SAX" score="1" />
</SuggestDictionary>
```

This XML file can be created manually, by FirstSpirit using a template provided for this purpose or by Exalead itself. In the case of an XML file created by Exalead itself, after it has been created it can also be manually edited, before it is used by Exalead as a dictionary for the SuggestService (for further information on the structure of the XML file, please refer to the Exalead documentation for the SuggestService).

The creation of the XML file by Exalead is initiated by the following command-line command:

```
cvconsole create-suggest-file-from-index fieldName suggest.xml
```

where `fieldName` stands for an index field, which is to be used as the source for the search term to be suggested. This index field should optimally be a field, which is filled with only a single word per document during indexing.

The second parameter gives the name of the XML file to be generated and can be freely selected.

Example:

```
D:\Exalead Cloudview\data\bin>cvconsole create-suggest-file-from-
index mymetafield mymetasuggest.xml
```

6.2 Compiling the dictionary file

After a dictionary file has been created in XML format in the first step, in the second step it must be transferred to a binary file and installed as a source for the SuggestService. This is done with the following command-line command:

```
cvconsole compile-suggest /path/to/source.xml qshare:///destDir
```

The command expects the path to the XML file as the first parameter and the target directory in which the binary file is to be written as the second parameter. The target directory is always defined in the form: `qshare:/// + <any directory name>`

Example:

```
D:\Exalead Cloudview\data\bin>cvconsole compile-suggest
mymetasuggest.xml qshare:///mysuggest
```

This call has created a new `mysuggest` directory under `D:\Programme\Exalead Cloudview\data\build\qshare\dic0`, which contains the compiled dictionary.

If the directory has already been created by a previous call of this command and the corresponding SuggestService in Exalead is already set up (step 3), a third command can be transferred to the command, which specifies the name of the SuggestService, which is to be restarted after compiling the XML file, so that the changes to the dictionary are immediately visible to the user.

Example:

```
D:\Exalead Cloudview\data\bin>cvconsole compile-suggest
mymetasuggest.xml qshare:///mysuggest mysuggestservice
```

6.3 Setting up the SuggestService on the Exalead server

After creating, compiling and installing the dictionary, a corresponding SuggestService must now be set up on the Exalead server. The operations required for this are performed online in the Manage area of Exalead's API Console (accessible under `http://<exalead-server-host>:<baseport>+1`):

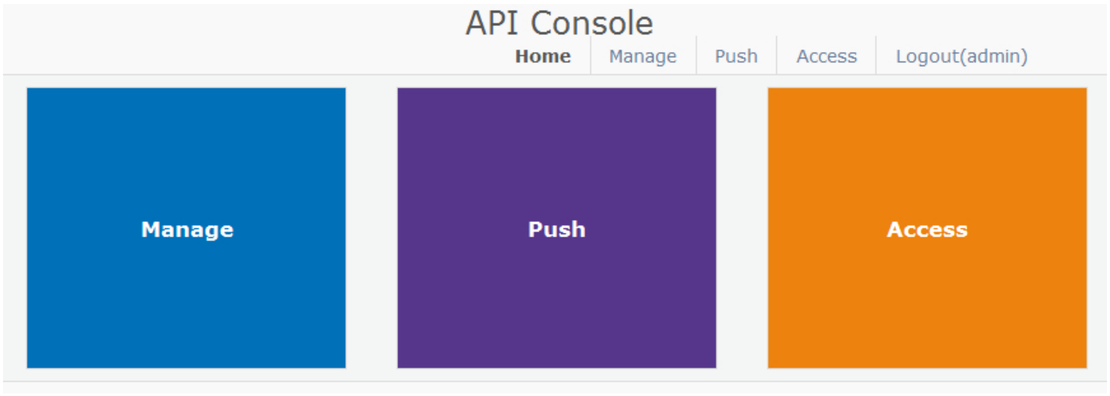
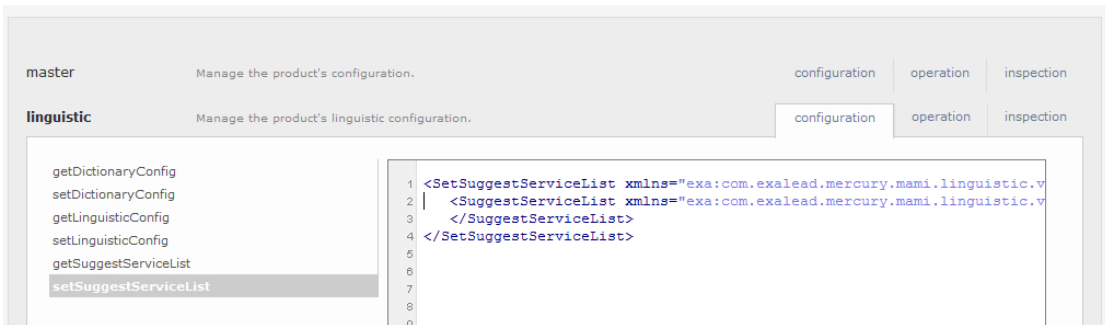


Figure 6-3: API Console

First, a new SuggestService must be created. To do this, call the setSuggestServiceList command in the linguistic > configuration area:



```
<SetSuggestServiceList xmlns="exa:com.exalead.mercury.mami.linguistic.v
   <SuggestServiceList xmlns="exa:com.exalead.mercury.mami.linguistic.v
   </SuggestServiceList>
</SetSuggestServiceList>
```

Figure 6-4: Adding a SuggestService

An entry for the new SuggestService is now created within the SuggestServiceList tag by adding the following tag:

```
<SuggestService serviceName="mysuggestservice"
resourceDir="qshare:///mysuggest">
</SuggestService>
```

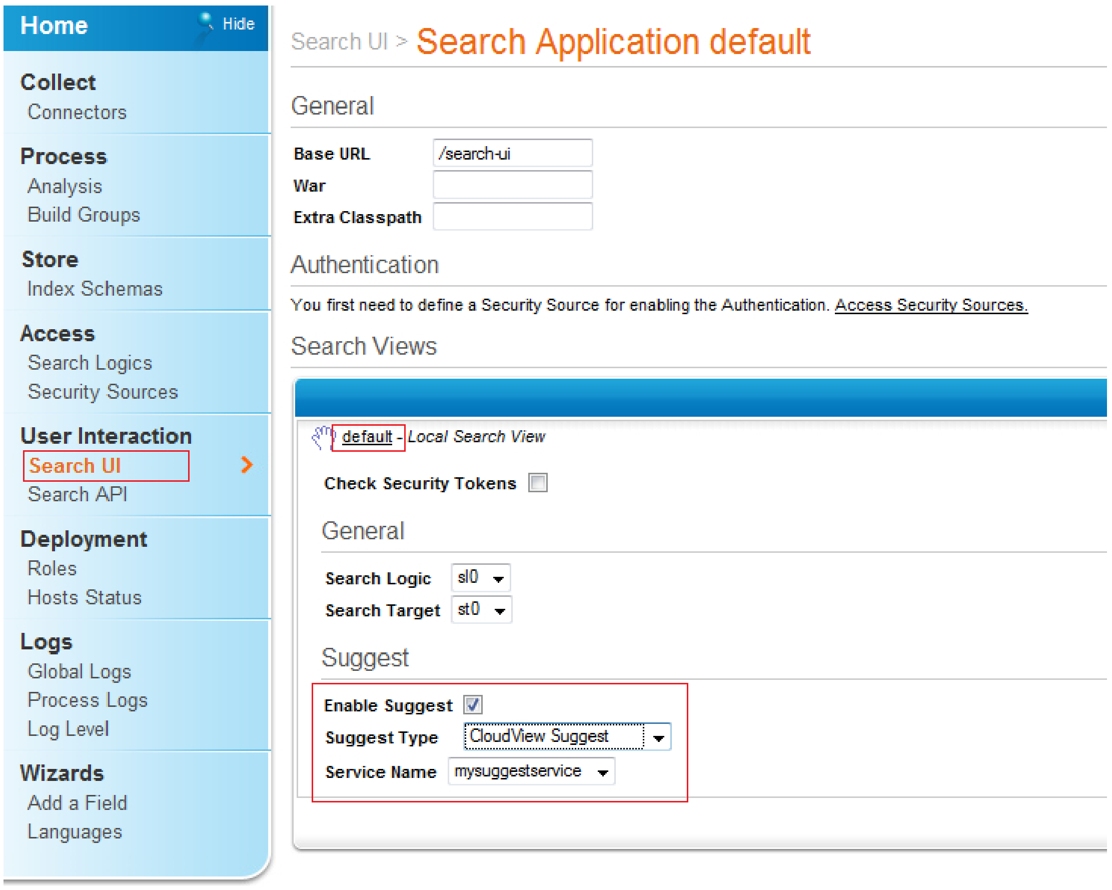
Any name can be chosen for the service. The ResourceDir must point to the directory into which the compiled dictionary was copied.

Click the "send data" button to save the changes. The "applyConfiguration" command must now be run in the master > operation area (also by clicking the corresponding "send data" button), so that the changes are applied.

To use the SuggestService with the Exalead Live module, the name of the SuggestService created must be given in the configuration of the AutocompleteServlet in the web.xml (see Chapter 3.7.4 AutocompleteServlet).

If the SuggestService is also to be made available in the standard Exalead GUI, it must be activated first in the Exalead administration (see Figure 6-5: Activating the SuggestService). This is not necessary if the Exalead Live module only is used for searching.

In some cases the previously created SuggestService is not shown in the Exalead administration. This can be corrected by rebooting the server.



Base URL /search-ui
War
Extra Classpath

You first need to define a Security Source for enabling the Authentication. Access Security Sources.

default - Local Search View

Check Security Tokens ☐

Search Logic sl0
Search Target st0

Enable Suggest ✓
Suggest Type CloudView Suggest
Service Name mysuggestservice

Figure 6-5: Activating the SuggestService

6.4 Adjusting the search template

To use the autocomplete function together with the Exalead Live module, a final step is necessary: the template of the search page must be adjusted.

The page's HTML header must be extended to include the following three lines:

```
<link type="text/css" href="css/redmond/jquery-ui-1.8.6.custom.css"
rel="stylesheet" />
<script type="text/javascript" src="js/jquery-
1.4.2.min.js"></script>
<script type="text/javascript" src="js/jquery-ui-
1.8.6.custom.min.js"></script>
```

In addition, the autocomplete function must be registered for the input field of the search form:

```
<script>
$(function() {
            $("#searchquery").autocomplete({
                  source: "do.autocomplete"
            });
      });
</script>
```

In this example the input field is addressed via the `searchquery` ID. The information specified for `source` must correspond to the mapping of the AutocompleteServlet, as given in the web.xml.

If these adjustments have been made to the template, the form now supports autocompletion of the search entries, as seen in Figure 6-6.

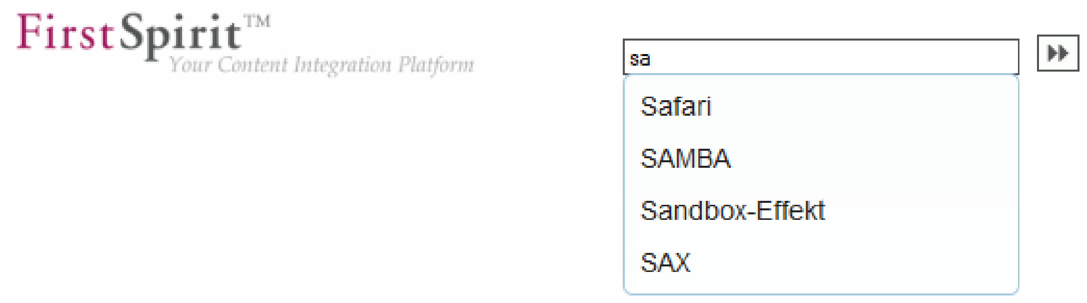


Figure 6-6: Autocompletion

7 Migration guidelines from Version 1.x to 2.0

If an earlier version of the FirstSpirit Exalead Live and FirstSpirit Exalead Content-Update modules have already been used, the following items must be noted when updating to Version 2.0:

7.1 web.xml

- The web.xml must be extended to include the global parameter for defining the Exalead server (see Chapter 3.5 page 16).
- The parameters of the SearchServlet must be adjusted according to the new servlet description (see Chapter 3.7.1 page 19).
- The parameters of the SearchDownload servlet must be adjusted according to the new servlet description (see Chapter 3.7.2 page 20).

7.2 Tag Library

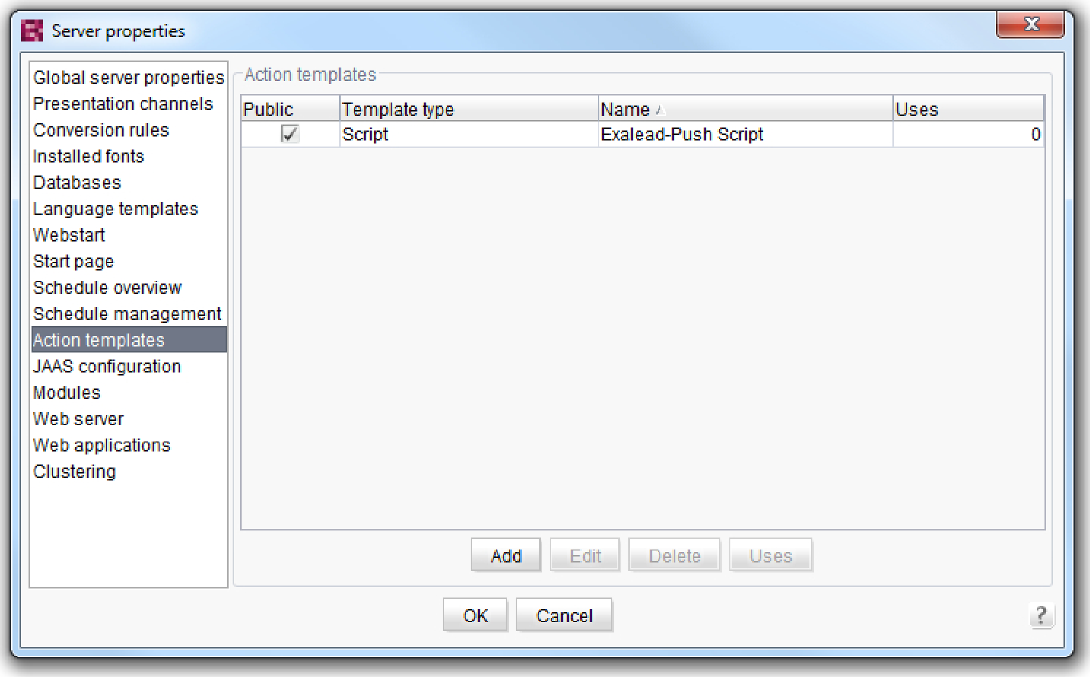
- The display of thumbnails for the search results has been changed (see Chapter 3.12.11 page 47).
- The `has_meta_data` tag is no longer supported. The `hits_field` tag should be used instead (see Chapter 3.12.8 page 46).
- The `hits_field_attributes` tag has also been replaced by the `hits_field` tag.
- The "path" PageContext variable of the `refinements` tag has been renamed "title".

7.3 Personalized Search Results

Extensive changes have been made to the use of personalized search results, which mainly affect the configuration of the Exalead server. All the important information on this is given in Chapter 4, page 55.

7.4 Installing the FirstSpirit "Exalead Content-Update" module

When updating the Exalead Content-Update module from a previous version (< 2.0) to the current version, please note that before the module is updated the Exalead-Push action template previously installed with the old module must be removed manually (see Figure7-1).



Public	Template type	Name	Uses
✓	Script	Exalead-Push Script	0

Figure7-1 Action templates

All other important information concerning the installation of the module is given in Chapter 5.1 on page 59 and Chapter 5.2 on page 61.