



# **Insuma*Focus* Specification**

Insuma GmbH

December 8, 2005

Insuma reserves the right to make changes in the specifications and other information contained in this publication without prior notice. In case of doubt, the reader should consult Insuma to determine whether any such changes have been made. The software described in this document is furnished under a license and may be used or copied in accordance with the terms of such license.

## **Trademarks**

Insuma and its logo are registered trademarks of Insuma GmbH. All other product and company names are either trademarks or registered trademarks of their respective companies.

Insuma GmbH  
Sand 13  
72076 Tübingen  
Germany  
Phone: +49 (7071) 3 65 96 10  
Fax: +49 (7071) 3 65 96 14  
E-mail: [info@insuma.de](mailto:info@insuma.de)  
<http://www.insuma.de>

# Contents

<b>1. Introduction</b>	<b>6</b>
1.1. Tasks addressed by InsumaFocus	6
1.2. How InsumaFocus is built	7
1.3. InsumaFocus in your software environment	8
<b>2. Search interface</b>	<b>9</b>
2.1. Interface address	9
2.2. XML queries	9
2.3. XML responses	10
<b>3. Indexing interface</b>	<b>11</b>
3.1. Full reindexing of XML	11
3.2. Full reindexing of HTML	11
3.3. Regular reindexing	12
3.4. Incremental indexing	12
3.5. Incremental indexing of websites	12
3.6. Changing index attributes	13
<b>4. Optional modules</b>	<b>14</b>
4.1. Morphology for major European languages	14
4.2. Correction of mistypes	14
4.3. Phonetic search	14
4.4. Thesaurus support	14
4.5. Query by example text	14
4.6. Wildcard search	15
4.7. Phrase search	15
4.8. Clustering	15
4.9. Summarization	15
4.10. Classification	15
4.11. Taxonomy, third party taxonomies	15
4.12. VIP search	16
<b>5. Control Centre</b>	<b>17</b>
5.1. Starting URLs for import	17
5.2. Category mappings	17
5.2.1. Mapping URLs to categories	17

5.2.2. Meta tag mapping . . . . .	17
5.3. Crawling domain definition . . . . .	18
5.3.1. URL inclusion . . . . .	18
5.3.2. URL exclusion . . . . .	18
5.3.3. URL modification . . . . .	18
<b>A. Code examples</b>	<b>20</b>
A.1. Calling InsumaFocus from Perl . . . . .	20
A.2. Calling InsumaFocus from Java . . . . .	20

# Executive summary

This document describes the *InsumaFocus* product, a search engine for Web portals and intranets. The overall functionality is described as well as the separate plugins and their function. Detailed description of interfaces allows technical specialists to integrate *InsumaFocus* in their company's IT environment.

# 1. Introduction

InsumaFocus is a search engine. Its central functionality is to make the textual content searchable. The source where the content is situated can be, but not limited to: websites or a corporate portal, databases, file system, proprietary software or office document repositories.

## 1.1. Tasks addressed by InsumaFocus

The amount of information being stored grows in a non-linear manner. In the organisations, the new information is being steadily added in form of:

- database records
- web and intranet pages
- emails
- elements of document management software
- or simply stored at hard disks in form of office documents.

Enabling information by making it searchable increases its usability and, therefore, its value for both customers and employees. Centrally searchable information is actively used by the employees and customers

### **The value for employees:**

- saves working time spent for searching
- saves creative resources by recycling of texts
- supports logical document structure and updating
- increases interoperability of divisions
- empowers fast-to-make decisions
- shortens learning curve for new employees
- increases qualification by resource awareness

**The value for portal owners:**

- increases customer satisfaction
- make the visitors come again to the website
- decreases number of support tickets
- increases quality of support tickets
- makes new documents come faster to the users
- gives control on what customers find first
- enables geographically distributed information
- enables multi-format information search.

To address the problem of infoglut, many companies of all sizes create business portals as a central access point to company-wide information. Insuma*Focus* offers a component which cares of three principal tasks: search, classification and categorization.

**Search**

Insuma*Focus* offers the widest range of advanced search features, from simple keyword search to semantic search by example.

**Classification**

Insuma*Focus* provides highly tunable automatic classification technique based on state-of-the-art research in information retrieval

**Categorization**

InsumaFocus provides open XML interfaces allowing unlimited assigning of (hierarchical) categories: by topic, access rights, document format, geographic location, departments, or individually based.

## 1.2. How InsumaFocus is built

Insuma*Focus* consists of a number of scripts and runtime programs: core modules responsible for indexing and searching, and a number of plugins. Plugins are responsible for special tasks e.g. for filtering HTML or office documents, accessing databases, parsing language specific morphology, fuzzy search etc. Plugins can be combined voluntarily according to the ordered configuration. Customized plugins (from Insuma or from third party software manufacturers customized specifically for the client) can be added to the installation.

Complexity of plugins varies. The simplest consists of few wrapper lines around open source tool. Most complex plugins incorporate dozens man-years of state-of-the-art research. Missing plugins can be developed on demand.

Insuma*Focus* accesses information on regular basis (hourly, daily etc.) using crawling scripts (also called fetch recipes). Fetch recipes can be configured for accessing heterogenous information sources: (remote) databases, file systems, web sites, document management systems etc.

Search front end (website, intranet application, Java program etc.) may call InsumaFocus via XML or, optionally, through widely supported SOAP interface and represent search results in appropriate layout.

Insuma*Focus* can be installed at several servers in a distributed manner. Every collection updates a local index, the collections co-operate to allow central search. Collections can communicate via XML over HTTP or via CORBA.

### 1.3. Insuma*Focus* in your software environment

InsumaFocus is a standalone program which communicates with the outer world via XML messages. The XML messages can be transferred by the HTTP protocol, or using the UNIX inter-process communication.

- Insuma*Focus* indexes the content which it receives as XML files.
- The search queries must be given to InsumaFocus in XML format, typically over HTTP
- The returned search results are again in XML, using the same transport.

The XML schemas used for communication with Insuma*Focus* are described in a separate document called “XML Handbook”. Several plugins allow for sophisticated search: crawling and parsing of HTML-formatted web pages, morphological parsing, phonetic tolerance, etc. Category, access group, and other attributes can be assigned to every document which allows for searches limited to the particular attribute values.

## 2. Search interface

Search is performed in a SOAP-like manner<sup>1</sup>: an XML-encoded query is sent to InsumaFocus over HTTP (or any other reliable transport), the XML-encoded response is returned via the same channel.

### 2.1. Interface address

Search queries should be posted to the interface address. If HTTP (plain or secure) is used, the search interface entry point is some particular URL, e.g.:

```
http://www.insuma.de/<customer>/web2xml.py
```

The URL above should be used if you have ordered an “Application Service Provider” (ASP) installation. In this case you communicate with InsumaFocus which remains at the Insuma ASP server. The XML query must be submitted using the POST method, the Content-Type to be used is "multipart/form-data". The same Content-Type is used for Form-based File Upload as specified in RFC 1867.

The name field used by the posting must have the value of ‘xml\_query’.

The whole HTTP request submitted must look approximately like the one shown in Figure 2.1.

You can look up a short example script written in Perl in the section A. “Code examples”.

If your frontend application runs at the same machine as InsumaFocus you can post search queries directly to the script:

```
python /usr/local/lib/insuma/focus/if_search.py
```

The directory path may vary upon your installation. Start this program from command line to see all necessary parameters.

### 2.2. XML queries

The search query which you post to InsumaFocus should be formatted in XML. Please refer to the detailed description of XML schema in the separate document “XML Handbook”.

---

<sup>1</sup>This is not exactly SOAP because of minor formalities like namespaces not used. A compliant SOAP interface can be produced if ordered.

```
POST http://www.insuma.de/cgi-bin/<customer>/web2xml.py

Content-Length: 388
Content-Type: multipart/form-data; boundary="6G+f"

--6G+f
Content-Disposition: form-data; name="xml_query"
Content-Type: text/xml

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE query SYSTEM "insuma_query.dtd">
<query show_attrs="title description">
  <condition predicate="match" attr="body"
    value="Bundeskanzler"/>
</query>

--6G+f--
```

Figure 2.1.: POST example

### 2.3. XML responses

The result of the search is returned by InsumaFocus in XML format. Please refer to the detailed description of XML schema in the separate document “XML Handbook”.

## 3. Indexing interface

The indexing interface serves to communicate between the Insuma search engine and the text source (backend database, website etc.). In general, you should format your data in XML format and provide a valid address from which those data can be fetched. Alternatively, InsumaFocus can harvest the website(s) directly. Updates can be done periodically (e.g. every night at 6 o'clock). The schema of the XML file describing the changes in the document source, as well as an example of such file, can be found in the InsumaFocus XML Handbook.

### 3.1. Full reindexing of XML

In the case of full reindexing InsumaFocus downloads the XML files with updates from your server using the HTTP protocol (plain or secure). You should provide a URL pointing to the data repository. It can be a PHP script which accesses your database and wraps XML tags around the fields on the fly. It can be a static page, prepared by your software.

### 3.2. Full reindexing of HTML

InsumaFocus can harvest the websites directly and index the HTML formatted documents. Additionally, documents in other formats can be indexed (PDF, Word, etc.), depending on your configuration. Attributes of the documents can be handed on to InsumaFocus in form of meta tags. Parts of HTML can be excluded from the indexing by the tags:

```
<insuma_ignore>Ignored text</insuma_ignore>
```

Reindexing can be started from the command line as follows:

```
python /usr/local/insuma/focus/if_crawler.pyc
```

The path can vary upon your installation.

### 3.3. Regular reindexing

InsumaFocus can perform reindexing automatically at the regular basis. Following modules should be started from cron system of the machine:

```
/usr/local/insuma/focus/daemon/cc_daemon.pl - should run every
5 minutes
/usr/local/insuma/focus/daemon/cc_reindex.pl - should run ev-
ery night
```

The module `cc_daemon` monitors the status of the system, checks if reindexing has been initiated via control center and runs the necessary modules which are responsible for the harvesting and file manipulation. The module `cc_reindex` initiates the reindexing.

### 3.4. Incremental indexing

The XML update file should be posted to the indexing interface. The usual entry point for the operation while the search engine runs as ASP at the Insuma server is

```
http://www.insuma.de/<customer>/web2xml.py
```

The XML file with updates must be submitted in the way similar to submitting a query. The POST method must be used, with the Content-Type of "multipart/form-data". The same Content-Type is used for Form-based File Upload as specified in RFC 1867.

The name field used by the posting must, however, have the value of 'xml\_update'.

This will update entries for changed documents and add entries for new documents incrementally. This approach is suitable for changing a relatively small number of documents (e.g, 5%) at a time. The advantage is that the newly changed documents become searchable instantly.

### 3.5. Incremental indexing of websites

In case of incremental indexing the crawler updates index entries of the modified web pages only. Also newly appeared pages will be added during incremental reindexing. To allow incremental indexing the web server should report modification date (`Last-Modified`) in the HTTP header. This is not the case if the pages are dynamically produced by a script. If you own the website, ensure that the script posts the modification date to HTTP header based on modification date of the source file or the corresponding database record.

If incremental indexing is not possible, than InsumaFocus will harvest and reindex the entire website all over.

### 3.6. Changing index attributes

The file, which describes mapping of document attributes to the index database must be specified in the configuration file (normally `focus.conf`) as the value of the parameter "mapping", for example:

```
mapping = /var/local/insuma/focus/html_mapping.xml
```

The mapping file lists the attributes in the index and specifies their properties in XML 1.0 format. Default mapping includes HTML derived attributes like "body", "title", etc. Search engine administrator can add further attributes. The changes in the attribute mapping take effect after resetting the index:

```
python if_collection.pyc --reset
```

## 4. Optional modules

Optional modules, or plugins, may be used to achieve improved search quality. These modules allow for a fuzzy search: correction of mistypes, phonetic tolerance with proper nouns, etc.

### 4.1. Morphology for major European languages

Morphology processing is language dependent. Morphology plugin for German cares about compound words, like *Donaudampfschiffgesellschaft*, ignore plurals, etc. Morphology plugin(s) - also for further European languages - can be added to your configuration optionally.

### 4.2. Correction of mistypes

Plugin Typo is responsible for correction of typing mistakes (*Spiegle* -> *Spiegel* and the like), including missing or redundant, as well as swapped or mistyped letters. This plugin can be added as an option to your configuration.

### 4.3. Phonetic search

Phonetic search allows finding the misspelled but phonetically close proper nouns (example: *Alladin* -> *Aladdin*, *Hulet-Pekart* -> *Hewlett-Packard* etc.). This plugin can be added as an option to your configuration.

### 4.4. Thesaurus support

Thesauri compliant with ISO 2788 or ISO 5964 can be integrated in the search. This allows the user to find the words which are defined as associated to a word in the search query. This plugin can be added as an option to your configuration.

### 4.5. Query by example text

The search can be initiated by giving a piece of text which user considers relevant. This gives InsumaFocus much more information for searching than just keywords.

## 4.6. Wildcard search

Wildcard search allows using masks in the search query (example: *Benzol\** etc.). This plugin can be added as an option to your configuration

## 4.7. Phrase search

Phrase search (string search) matches phrases in documents like “Brandenburger Tor” or “Bill Gates”. This plugin can be added as an option to your configuration.

## 4.8. Clustering

Clustering performs sorting of documents into thematic groups. The documents in such groups contain similar semantic content. Clustering is used for identifying thematically very close documents in the result set. Such documents can be e.g. older versions of a document, or versions with different formatting. This plugin can be added as an option to your configuration.

## 4.9. Summarization

This plugin produces summaries of documents based on search terms. The search terms or their stems become highlighted which helps the user to understand why particular document were found. The summaries can be used for presenting the result set to the user. The summary information can also be used for highlighting the search terms in the preview of documents provided the document format allows such highlighting.

## 4.10. Classification

Classification module automatically ascribes content to pre-defined thematic categories. The categories can be defined by keywords and by sample documents. The sample documents must be identified for each category by human experts. The classification process happens while indexing. New documents being added to the index will be automatically routed to the best matching categories. The automatic classification can be corrected by a human expert manually. The classification filter will take this feedback into account for later automatic classification.

## 4.11. Taxonomy, third party taxonomies

Automatic classification can be done based on a defined taxonomy. Such taxonomy represents a (hierarchical) catalogue of subject areas. with sample documents in each of them. A taxonomy

can be provided by a third party vendor. Taxonomy should match the subject area of the indexed contents. A possible third party taxonomy is e.g. the so-called “Lebenslagenmodell” which is widely used in municipal or governmental portals.

## 4.12. VIP search

VIP is an integer property defined in the XML mapping. It can be specified in an HTML page as a meta tag, e.g. `<meta name="vip" value="34">`. It is supported by standard XML queries. It allows for graduate or so called soft boosting. This means that actual scores of VIP URLs are added with VIP values given in the meta tag. A xml query using soft boosting may be formulated like this:

```
<or>
  <and>
    <condition predicate=...>
    <condition predicate="gt" attr="vip" value="5">
  </and>
  <condition predicate=...>
</or>
```

The exemplary xml query ranks higher each URL whose VIP attribute contains a value greater than 5.

To show only VIP entries (hard boost), following xml query should be used instead:

```
<and>
  <condition predicate=...>
  <condition predicate="gt" attr="vip" value="2">
</and>
```

## 5. Control Centre

Control Centre is a web-based configuration and administration tool. It allows the admin to change starting URLs, looking at the search statistics, re-launch the indexing, and perform other functions defined in your installation.

### 5.1. Starting URLs for import

You should specify starting URL(s) of the data source. InsumaFocus will start harvesting from this URL. Starting URL can be (re)configured through the control centre.

### 5.2. Category mappings

The search may be limited to a category. Categories can be given to InsumaFocus along with the documents over the XML interface or they can be encoded in your HTML pages. The search can be later limited to certain values in the categories.

#### 5.2.1. Mapping URLs to categories

In case of web harvesting you may assign categories by URL in the control center. For example you can assign categories according to the website - source of information:

```
http://www.insuma.de/          search engines
http://news.bbc.co.uk/world/   latest news
```

The category name is to be separated from the URL by white space. If missing, the last word in the URL is considered to be the category name. For example, it would be 'world' if 'latest news' was missing from the last snippet above.

#### 5.2.2. Meta tag mapping

In case of web harvesting the categories can be defined over the meta tags in every HTML page:

```
<meta name="category" content="marketing">
<meta name="subcategory" content="fairs">
```

The category name follows the meta tag separated by white space. Both names and values in the meta tags can be defined freely by the user. If the category is missing, the value attribute of the meta tag in the source text is taken as category name.

### 5.3. Crawling domain definition

In order to include, exclude specific URLs from crawling or modify URLs before being crawled, regular expressions are used to set the appropriate values. Regular expressions (regexps) comply with POSIX regexps, the same as used by Perl and Python. For a detailed explanation on how to use POSIX regexps, see e.g. <http://www.python.org/doc/current/lib/re-syntax.html>

Make sure to escape dots (\.), question marks (\?) and asterisks (\\*) in any regexp.

#### 5.3.1. URL inclusion

A URL inclusion is a list of regexps that are allowed for crawling. If left empty, everything is allowed for crawling. If not empty, only those URLs that match given regexps are allowed. The input must be line separated. Every line is a regexp of its own. A URL is crawled, if it matches at least one of the existing regexps.

#### 5.3.2. URL exclusion

URL exclusion works in the same manner as URL inclusion. If left empty, nothing is excluded. If the URL matches at least one of the existing regexps, the URL will be excluded from being crawled.

#### 5.3.3. URL modification

This is a re-writing rule for URLs. It is applied from first to the last rule. Rules consist of two elements separated by white spaces. The second element may be missing. The first one is a regexp, the second one is a string which will be substituted instead of the part of the URL that matches the regexp. In order to translate e.g. shadow pages to real pages within the URL, the following rule should be applied:

```
http://www\mysite\.de/shadow/ http://www.mysite.de/real/
```

Rules are applied one by one. That means that rules may be applied to the result of the previous substitution.

Some examples:

- sessionid=[^&]\*&?

will erase parts of URLs from the first occurrence of 'sessionid=' until first occurrence of ampersand or until end of the URL

- &\$

will erase trailing ampersand

- \?\$

will erase trailing question mark

## A. Code examples

### A.1. Calling InsumaFocus from Perl

```
#!/usr/bin/perl
use strict;
use HTTP::Request::Common;
use LWP;

die "Usage:  xml_post.pl XML_query_file [service_url]\n"

    if (scalar(@ARGV) == 0);

my $url = 'http://www.insuma.de/cgi-bin/focus/web2xml.py';
$url = $ARGV[1]
    if (scalar(@ARGV) >= 1);

my $ua = LWP::UserAgent->new;
# submit the request over HTTP
my $response = $ua->request(
    # use the POST method
    POST $url,
    # the Content-Type is 'multipart/form-data'
    Content_Type => 'form-data',
    # the field name is 'xml_query',
    # the field value is the XML query.
    Content => [ xml_query => [$ARGV[0]], ]
) or die "Could not send the request\n";

# print the search result to standard output (in XML)
print $response->content, "\n";
```

### A.2. Calling InsumaFocus from Java

The example below uses only the functionality available in the Standard Edition of Java. The Enterprise edition offers much better support for multipart MIME messages used for submitting

the query. The way of choosing the boundary between the parts is **unsafe** and presented for demonstration purposes only.

```
import java.net.*;
import java.io.*;

public class connect {
    public static void main( String args[] )
        throws MalformedURLException, IOException, ProtocolException
    {
        // open connection
        URL requestURL =
            new URL("http://www.insuma.de/cgi-bin/focus/web2xml.py");
        HttpURLConnection connection =
            (HttpURLConnection) requestURL.openConnection();

        // Use the POST method
        connection.setDoOutput(true);

        // Set up the HTTP headers.
        // The boundary must not be present in the query.

        connection.setRequestProperty("Content-Type",
            "multipart/form-data; boundary=\"cRaZy\"");

        // The XML query to post.
        // Note the field name, xml_query, specified explicitly.
        String postString = "--cRaZy\n"
            + "Content-Disposition: form-data; name=\"xml_query\"\n"
            + "Content-Type: text/xml\n\n"
            + "<?xml version=\"1.0\" encoding=\"iso-8859-1\"?>\n"
            + "<!DOCTYPE query SYSTEM \"insuma_query.dtd\"\n"
            + "<query show_attrs=\"title description\"\n"
            + "<condition predicate=\"match\" attr=\"body\"
                value=\"cheatcode handy\"/>\n"
            + "</query>\n"
            + "--cRaZy--\n";

        // POST the XML query
        OutputStream outputStream = connection.getOutputStream();
        OutputStreamWriter out = new OutputStreamWriter(outputStream);

        out.write(postString);
        out.flush();
    }
}
```

```
        System.out.print( "Posted.  The return code is "
);
        System.out.println( connection.getResponseCode()
);

        System.out.println( "The output from the server is:"
);
        int c;
        InputStream inStream = connection.getInputStream();

        while ((c = inStream.read()) > 0) {
            System.out.print( (char)c );
        }
        System.out.println( "" );
    }
}
```