

**Softwarehaus Graf & Partner**

# **Documentation TopicDB.dll**

**Ver. 1.8**

**Rev. 4348**

**20.04.2010 12:49:00**

## **SOFTWAREHAUS**

**Graf & Partner**

Säulengasse 17, A-1090 Wien  
Tel. und Fax: +43 - 1 / 310 24 39  
Email: [office@grafsoft.at](mailto:office@grafsoft.at)  
Homepage: [www.grafsoft.at](http://www.grafsoft.at)  
[www.newswatch.at](http://www.newswatch.at)

## Contents:

1	Last changes .....	6
2	Getting started .....	7
2.1	What is this? .....	7
2.2	What you need .....	7
2.3	Installation .....	7
3	How To .....	8
3.1	How to call .....	8
3.2	How to connect to databases and create tables .....	8
3.3	How to insert text into the database .....	9
3.4	How to read text from files .....	10
3.5	How to create the index .....	10
3.6	How to search and show the results .....	11
3.7	How to work with frequencies .....	11
3.8	How to work with topics .....	13
3.9	How to allocate keywords automatically .....	14
3.10	How to generate automatic keyword searches .....	15
3.11	How to work with tag clouds .....	15
4	Properties and methods .....	17
4.1	Working with databases .....	17
4.1.1	Openserver method .....	17
4.1.2	Createtextdatabase method .....	17
4.1.3	Createtopicdatabase method .....	17
4.1.4	Createnamedatabase method .....	17
4.1.5	Databaseexists method .....	17
4.1.6	Createtexttables method .....	17
4.1.7	Createtopicables method .....	17
4.1.8	Createnametables method .....	18
4.1.9	Deletetextdatabase method .....	18
4.1.10	Deletetopicdatabase method .....	18
4.1.11	Textdatabaselist property .....	18
4.1.12	Topicdatabaselist property .....	18
4.1.13	Namedatabase property .....	18
4.1.14	Textdatabase property .....	18
4.1.15	Topicdatabase property .....	19
4.2	Working with text .....	19
4.2.1	Createfield method .....	19
4.2.2	Deletefield method .....	19
4.2.3	Getfieldlist property .....	19
4.2.4	Getfieldname method .....	19
4.2.5	Getfieldid method .....	19
4.2.6	Setfieldmustindex method .....	19
4.2.7	Createdocument method .....	20
4.2.8	Currentdocument property .....	20
4.2.9	Firstdocument method .....	20
4.2.10	Lastdocument method .....	20
4.2.11	Nextdocument method .....	20
4.2.12	Previousdocument method .....	20
4.2.13	Deletedocument method .....	20
4.2.14	Deletedocuments method .....	21
4.2.15	Gotodocument method .....	21
4.2.16	Documentexists function .....	21
4.2.17	Documentcount property .....	21
4.2.18	Getmaxdocnr method .....	21
4.2.19	Addtext method .....	21
4.2.20	Updatedocument method .....	21
4.2.21	Locatetext method .....	22
4.2.22	Locatetextplain method .....	22
4.2.23	Gettext method .....	22
4.2.24	Fieldtext method .....	22

4.2.25	Fieldtextstart method .....	22
4.2.26	Changetext method .....	22
4.2.27	Changefield method.....	23
4.2.28	Deletetext method.....	23
4.2.29	Emptyfield method .....	23
4.2.30	Replacefields method .....	23
4.2.31	Readfiles method .....	23
4.2.32	Killdocumentsnotindirectories method.....	24
4.3	Indexing and searching.....	24
4.3.1	Deleteindex method.....	24
4.3.2	Indexall method.....	24
4.3.3	Indextempall method .....	25
4.3.4	Copyindex function .....	25
4.3.5	Recopyindex function .....	25
4.3.6	Indexchunksize property.....	25
4.3.7	Maxbytesperround property.....	26
4.3.8	Openindexage property .....	26
4.3.9	Fieldsnotindexed property .....	26
4.3.10	Fieldstempnotindexed property .....	26
4.3.11	Hastoindex property.....	26
4.3.12	Getindexoptions method.....	26
4.3.13	Setindexoptions method .....	27
4.3.14	Addseparator method .....	27
4.3.15	Deleteseparator method .....	27
4.3.16	Setseparators method .....	27
4.3.17	Stopwords property.....	27
4.3.18	Addstopwords method .....	27
4.3.19	IndexMinLength property .....	27
4.3.20	IndexMaxLength property .....	28
4.3.21	IndexContainsNumericals property.....	28
4.3.22	IndexStartwithNumericals property.....	28
4.3.23	IndexUmlaut property .....	29
4.3.24	Maxindexwords property.....	29
4.3.25	Maxmirrorbytes property .....	29
4.3.26	Indexpos property .....	30
4.3.27	Searchfieldstring property.....	30
4.3.28	Searchsortstring property .....	30
4.3.29	Maxwordpos property .....	30
4.3.30	Setsearchselect method .....	31
4.3.31	Search method.....	31
4.3.32	Numberofhits property .....	32
4.3.33	Firsthit method .....	32
4.3.34	Lasthit method .....	32
4.3.35	Nexthit method.....	32
4.3.36	Previoushit method .....	32
4.3.37	Maxsearchtime property .....	33
4.3.38	Maxhits property .....	33
4.3.39	Minsearchwordlength property .....	33
4.3.40	Searchisinprecise property .....	33
4.3.41	Errornum property.....	33
4.3.42	Hitsasstring method .....	34
4.3.43	Hitsasstringmax method .....	34
4.3.44	Lastsearchwords property .....	34
4.3.45	Fieldtextwithtags method .....	34
4.3.46	Savehitpos method .....	34
4.3.47	Savehitposselected method.....	35
4.3.48	Loadhitpos method .....	35
4.3.49	Dohitpos property .....	35
4.3.50	Dorandom property.....	35
4.3.51	Randomseed property .....	35
4.3.52	Orderstring property.....	35

4.3.53	Hitposperdocument function .....	36
4.3.54	Defaultmask property.....	36
4.3.55	Defaultlogic property.....	36
4.3.56	Lastsoundex method .....	36
4.3.57	Soundex property .....	37
4.4	Working with frequencies.....	37
4.4.1	Analyzefrequency method .....	37
4.4.2	Analyzereativefrequency method.....	37
4.4.3	Showfrequency method .....	37
4.4.4	Showrelativefrequency method .....	38
4.4.5	Showrelativefrequencyofword method .....	38
4.4.6	Drawgraph method .....	38
4.4.7	Drawclick method .....	39
4.4.8	Drawclickx, Drawwclicky property.....	39
4.4.9	Killawordforfrequency method .....	40
4.4.10	Returnawordtofrequency method .....	40
4.4.11	Freqstopwords property.....	40
4.4.12	Savefrequency method.....	40
4.4.13	Saverrelativefrequency method .....	40
4.5	Working with topics .....	40
4.5.1	Addreplacetopic method .....	40
4.5.2	Getalltopicwords method .....	41
4.5.3	Deletewordfromtopics method.....	41
4.5.4	Deleterightwordsto method.....	41
4.5.5	Showhierarchy method .....	41
4.5.6	Associations property .....	41
4.5.7	Gettopiclayer method.....	41
4.5.8	Gettopiclayerno method.....	42
4.5.9	Getalltopiclayernos method .....	42
4.5.10	Righttopicwordstoassoc method.....	42
4.5.11	Showsynonyms method.....	42
4.5.12	Gettopicsearch method.....	42
4.5.13	Gettopicsearchpred method .....	42
4.5.14	Addkeywordsinfo method .....	43
4.5.15	Deletekeywordsinfo method .....	43
4.5.16	Deletekeywords method .....	43
4.5.17	Autokeywords method .....	43
4.5.18	Addkwsearch method .....	43
4.5.19	Getkwsearch function .....	43
4.5.20	Deletekwsearch method .....	44
4.5.21	Autokwsearch procedure .....	44
4.6	Tag Clouds.....	44
4.6.1	Cloud_Setup procedure.....	44
4.6.2	Cloud_Addword procedure .....	44
4.6.3	Cloud_Addtext procedure .....	44
4.6.4	Cloud_Addwordfreq procedure.....	44
4.6.5	Cloud_Keepfirst procedure .....	45
4.6.6	Cloud_Sort procedure.....	45
4.6.7	Cloud_Sortrandom procedure .....	45
4.6.8	Cloud_Create function .....	45
4.7	General .....	45
4.7.1	Lasterror property .....	45
4.7.2	Separator property .....	45
4.7.3	Separator2 property.....	46
4.7.4	Setinfo method.....	46
4.7.5	Getinfo method .....	46
4.7.6	Setinfofromfile method.....	46
4.7.7	Logging property.....	46
4.7.8	Logtype property.....	46
4.7.9	Loglist method.....	46
4.7.10	Killlogs method .....	47

4.7.11	Killfilesbefore method .....	47
4.7.12	Execsql method .....	47
4.7.13	Isotojis method.....	47
4.7.14	Jistois method.....	47
4.7.15	Cyrillictoiso method.....	47
4.7.16	Isotoutf8 method .....	47
4.7.17	Regexp function .....	48
4.7.18	Wordintext function .....	48
5	Appendix .....	49
5.1	Error numbers .....	49
5.2	Script examples.....	49
5.2.1	Include file to connect: .....	49
5.2.2	Searching:.....	49
5.2.3	Displaying results:.....	50
5.2.4	Searching within the last search result: .....	52
5.2.5	Showing a graph:.....	52
5.3	Database structure.....	53
5.4	Regular Expressions .....	54
5.5	Troubleshooting .....	63
5.5.1	Program seems to hang when reindexing (usually at "fieldsnotindexed").....	63
5.5.2	Program gives error message "Server has gone away" during reading files .....	63
5.5.3	Program gives error message "Server has gone away" during indexing files .....	63

## 1 Last changes

- 16 April 2010      Tag Clouds
- 15 February 2010    Autokwsearch procedure and related functions
- 30 January 2010    Drawclickx, drawclicky properties
- 22 January 2010    Searchsortstring + and -.
- 14 January 2009    Several tries when opening database server. Does not give up when first time fails.
- 1 May 2009        Cyrillcitoiso and Isotoutf8 methods.

## 2 Getting started

### 2.1 What is this?

### 2.2 What you need

- This DLL, compiled for the MySQL or MSSQL database.
- MySQL or MSSQL, according to the type of the DLL
- A programming language which can use the OLE automation: Visual Basic, Delphi, C++, ASP, PHP ...

For an Internet application

- IIS4 or higher
- User rights for IIS, to create tables, add, replace, delete ...

### 2.3 Installation

Just copy the DLL anywhere on your server and register it with the command  
"regsvr32 topicdb.dll"

## 3 How To ...

### 3.1 How to call

On every page using the object you have to create the object and connect to its database driver.

We suggest that you put these lines in an include file.

In ASP:

```
Set obj = CreateObject("Topicdb.ttopicdb")
obj.openserver ("localhost","root","")
```

In PHP:

```
$obj = new COM ("topicdb.ttopicdb") or die ("Could not create object
TopicDB");
$obj->openserver ("localhost","root","");
```

More about this in the next chapter.

### 3.2 How to connect to databases and create tables

First, you have to connect to a server and define the names of the databases. You need three databases:

- The name database knows the names and some data on all of your text and topic databases
- The text database contains your text and index information
- The topic database contains the info about topics, synonyms and the like.

First, you should tell the system where your databases are (use an include file for this).

```
<%
Set obj = CreateObject("Topicdb.ttopicdb")
' you might have to change this
' Param 1 is the server
' param 2 is the user
' param 3 is the password
b = obj.openserver ("192.168.1.2","root","")
textdatabasename = "thetext1"
topicdatabasename = "thetopic1"
namedatabasename = "thenames"
' when we know the database are here, we can comment the following lines
out
if not obj.databaseexists (namedatabasename) then
    obj.createnamedatabase (namedatabasename)
    obj.createnametables
    %>Trying to create name database<br>
    Last error: <%=obj.Lasterror%><br>
    <%
        obj.Lasterror=""
    end if
obj.createnametables
obj.lasterror = ""
' comment out until here
obj.namedatabase = namedatabasename
' when we know the database are here, we can comment the following lines
out
if not obj.databaseexists (textdatabasename) then obj.createtextdatabase
```

```
(textdatabasename)
if not obj.databaseexists (topicdatabasename) then obj.createtopicdatabase
(topicdatabasename)
' comment out until here
obj.textdatabase = textdatabasename
obj.topicdatabase = topicdatabasename
%>
```

What does this script do?

First, it creates the object and connects to the server.

Second, it verifies if the databases exist and if not, creates them. You must create the names database first, because the names of the text and topic databases are entered into these databases.

Of course, when your databases are created, you can leave these lines out. You may not have the rights to create the databases anyway.

```
<%
obj.createnametables
obj.createtexttables
obj.createtopictables
%>
```

Careful: If the tables already exist, they are replaced by empty tables!

### 3.3 How to insert text into the database

First, you have to create fields where you put the text

Here is an example:

```
<%
obj.createfield ("Filename")
obj.createfield ("Filecontent")
%>
<%=obj.Lasterror%>
Fieldlist: <br>
<%=obj.getfieldlist%><br><br>
```

If you wish to insert Text into these fields, you do it like this:

```
Obj.createdocument(false)
Obj.Addtext "filecontent","This is my text",true,false
```

If you wish to use the index on the fly, set the first Boolean parameter to true. If you add larger amounts of text, set it to false and use the `indexall` or `indextempall` method afterwards.

The last Boolean parameter indicates if you wish to remember the document when it was changed. So if you add multiple fields, we suggest that you use `updatedocument` the last time.

To navigate within the documents table, use `firstdocument`, `lastdocument`, `nextdocument` and `previousdocument` methods.

To retrieve the number of the current document, use the `currentdocument` property.

For changing and deleting documents, use the `changetext`, `deletetext`, and `deletedoc` methods.

Example:

```
Firstdocument
While nextdocument do delete (currentdocument)
```

This will delete all the documents. In this case, you could also use the deletedocuments procedure.

### 3.4 How to read text from files

The DLL can read files from office applications and PDF-files, translate them to text format and store them into the database. For this, icrosoft Office has to be installed on the server.

Of course, it can also read simple text files.

```
` first we have to specify the field with the filename and the field with
  the text:
setinfo "Watchfiles","Fields","fname<br>ftext"
` if we set subdirs to 1, subdirectories are also read
obj.setinfo "Watchfiles","Subdirs","1"
`Not we tell the system where the files are
` this could also be a list of directories, separated by the parameter
  "separator"
obj.setinfo "Watchfiles","Directories","f:\text"
` Now we tell the system whereto store the extracted information:
obj.setinfo "Watchfiles","Textfiledir","f:\tmp"
readfiles (true)
```

Sometimes, you might want to copy files from other locations of the network to your computer. Then you have to set the variable "mirror" to 1.

```
obj.setinfo "Watchfiles","Mirror","1"
```

Now you need to specify a directory, where to copy the files:

```
obj.setinfo "Watchfiles","Mirrordir","c:\inetpub\wwwroot\textfiles"
```

Even if you set "Subdirs" to 1, no subdirectories will be created in "Mirrordir"

### 3.5 How to create the index

First, you must set the index options:

```
obj.getindexoptions
obj.indexminlength = 2
obj.indexmaxlength = 35
obj.indexumlaut = true
obj.indexcontainsnumericals = false
obj.Indexstartwithnumericals = false
` use addstopwords or the stopwords property
obj.addstopwords "is<br>are<br>will<br>the<br>of<br>to<br>and<br>a<br>
...
obj.addstopwords "at<br>from<br>it<br>has<br>an<br>have<br>will<br>or"
obj.setindexoptions
```

You will only have to do this once.

Then create the index:

```
If openindexage <> 0 then indexall (true,1000000)
```

## 3.6 How to search and show the results

To search, simply use the Search method:

```
SEARCH "flatscreen* or tft"
```

Then display the results, like this:

To get to the search results, you can use different approaches. This is one of the easiest:

```
dim somestring
somestring = obj.hitsasstring
array_var = split (somestring, ",")
session ("founddocs") = array_var
```

When you are on the page that displays the results, you can use this session variable. To look at the fifth hit, you do this:

```
array_var = session ("founddocs")
j = array_var(5)
obj.gotodocument (j)
```

If you want to show the searchwords in the documents in bold etc:

```
session ("foundhits") = obj.savehitpos
```

Now you can retrieve the fields and display them:

```
obj.fieldtext ("Field1")

obj.loadhitpos (session ("foundhits"))
obj.fieldtextwithtags ("Field2", "<b>", "</b>", 200, 0, 1)
```

## 3.7 How to work with frequencies

First, you must have an indexed database.

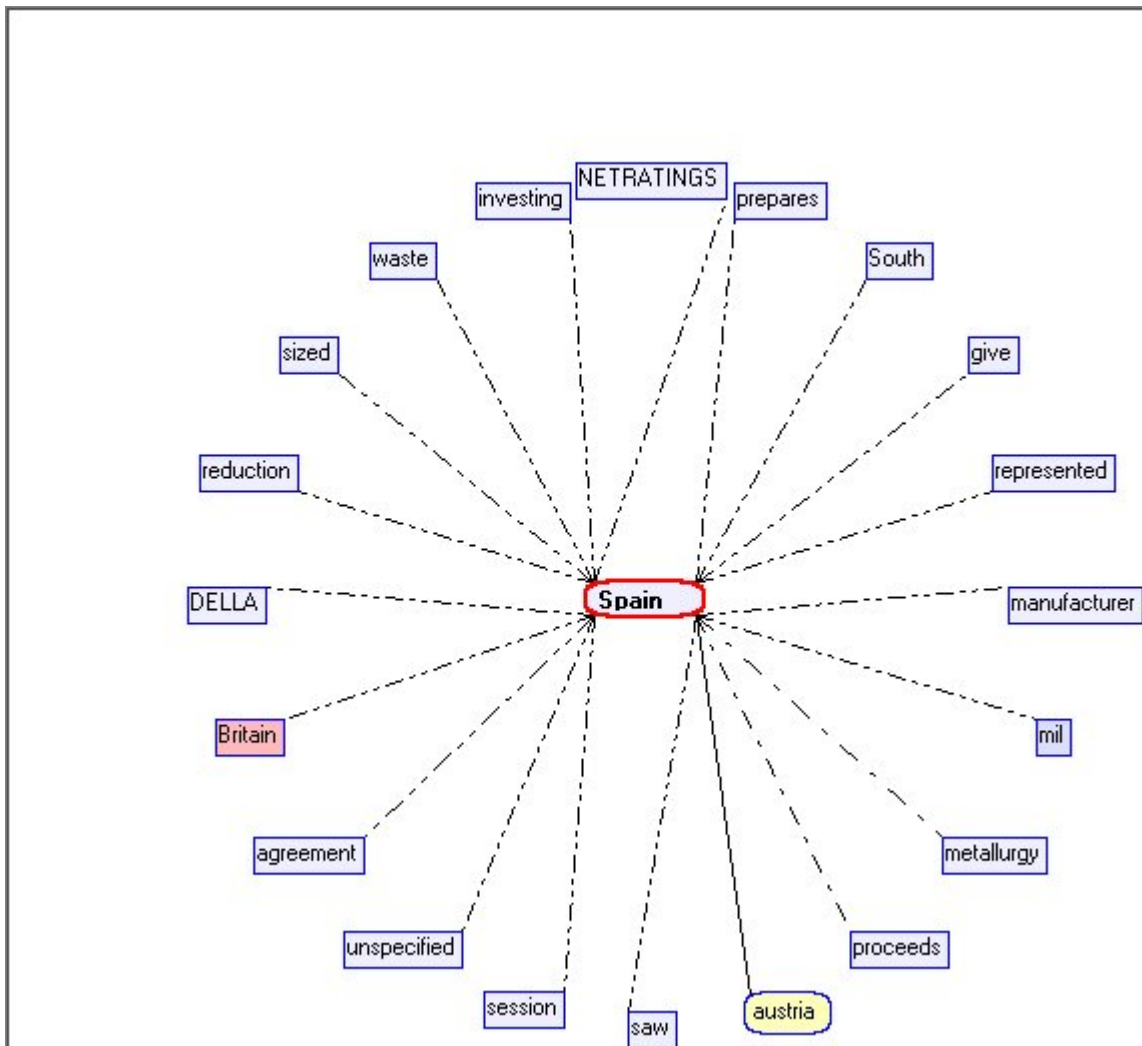
Then, use the `analyzefrequency` and the `analyzrelativefrequency` methods.

With the methods `showfrequency`, `showrelativefrequency`, `showrelativefrequencyofword`

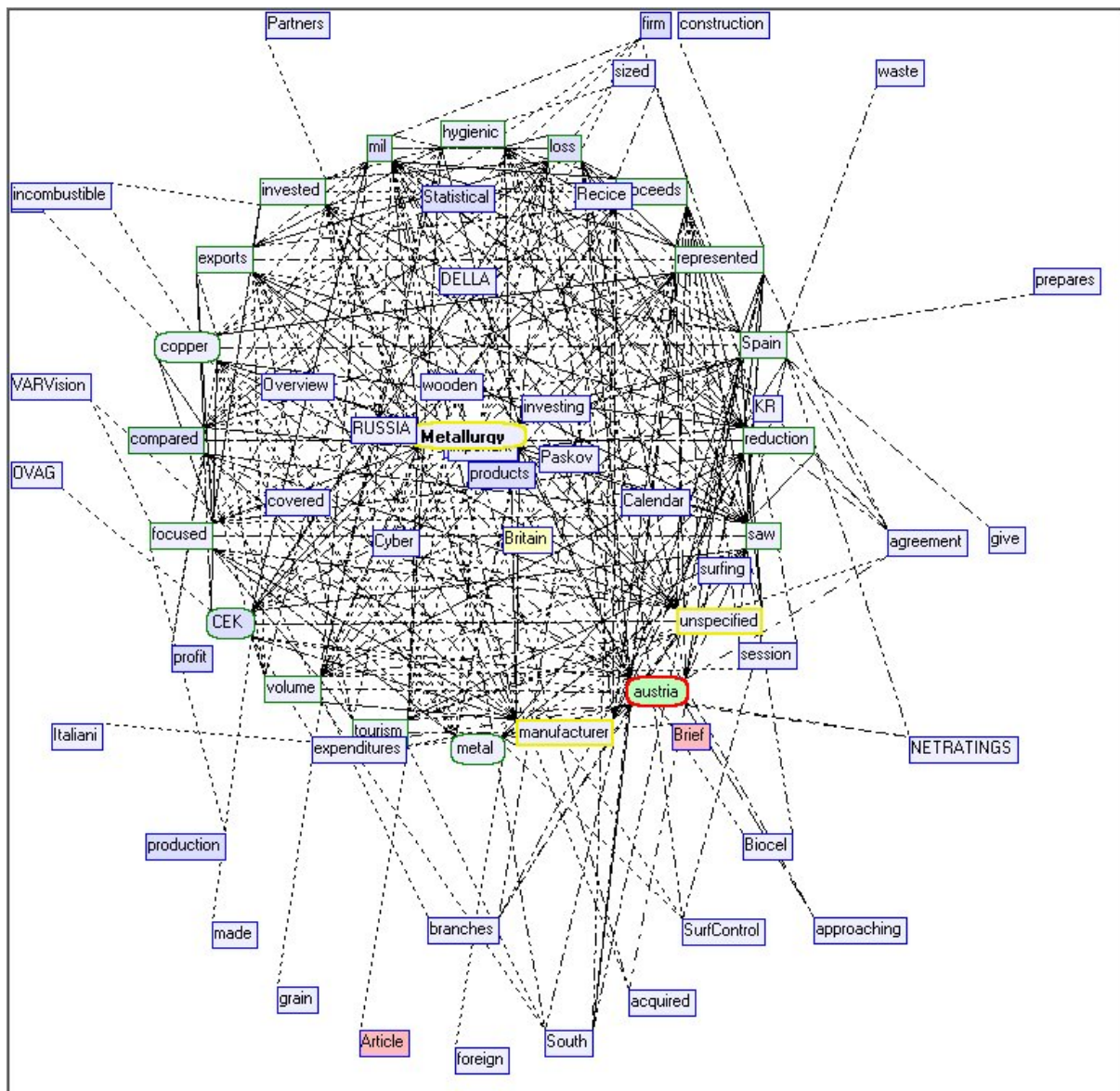
Use the `Drawgraph` method to show graphics of the relative frequencies of a word. This example shows 20 connections with a depth of 1.

Words with a low frequency appear a light blue, with higher frequency the color turns to green, yellow and red.

Strong connections (both words appear frequently in the same document) are printed in a solid line, weak connections in a dotted line.



The following example shows a depth of 3. You can see that it is a bit confusing. When using a depth of more than 1, use a lower count.



Some words have no meaning in the graph, like the word "Article". Remove them with Killawordforfrequency, return them with Returnawordtofrequency.

## 3.8 How to work with topics

What is a topic map? We use a subset of the definition of XML Topic given in <http://www.topicmaps.org/xtm/1.0/>.

But our approach is simpler:

A word is associated with another word. For example:

Copper is a metal.

A metal can be copper, iron, inc ...

In this case, the associations are "is a" and "can be".

For each association, we also have a reverse form, e.g. "Contains" and "Is contained in".

We can use this definition to define synonyms, groups and subgroups of a word. The current version is restricted to this approach, further versions will allow to use different associations.

Example:

Word 1	Association	Reverse Association	Word 2	Synonym?
Cake	contains	Is-contained_in	Flour	N
Cake	contains	Is-contained_in	Sugar	N
Cake	contains	Is-contained_in	Eggs	N
Eggs	contains	Is-contained_in	Protein	N
Cake	German_meaning	English_meaning	Kuchen	y

We can add these topics using the method `addreplacetopic`.

Using the `Sowhierarchy` method, we can see that a cake contains items from flour to protein.

Using the `showsynonyms` method, we can see that "Cake" also means "Kuchen".

The `Gettopicsearch` method would return "Cake or Kuchen or flour or sugar or eggs or protein". See also the `gettopicsearchpred` method.

You could also think of countries: When you enter the European countries as topics, and maybe the cities in the countries, you might get "Europe or France or Paris or Austria or Vienna or Graz ...".

With `Getalltopicwords` you can have a list of all the contained words, with `Deletewordfromtopic` you delete a word and all its associations.

Further versions will deal with different types of associations. But for the sake of compatibility, you can enter them already now.

### 3.9 How to allocate keywords automatically

If you have created topics for your topics database, you can create a keywords (descriptors, identifiers) – field automatically.

Look at this example:

Europe	United Kingdom	London	
		Yorkshire	York
Germany	Hessen	Frankfurt	
Austria	Tirol	Innsbruck	
			Obergurgl
		Kufstein	
Africa	Egypt	Cairo	
	Tunisia		
	Sudan		
Fruit	Apple		
Fruit	Plum		

Imagine, you have entered these words in the Topics table, using

```
addreplacetopic (Africa, Egypt, Is_Upper, Is_lower, false)
addreplacetopic (Egypt, Cairo, Is_Upper, Is_lower, false)
.. and so forth.
```

You have a text field named "Descriptors". When the text of a document contains the word "Cairo", you wish that the keyword field contains the words Africa, Egypt, and Cairo. Then you would find this article also, if you search only for the word "Africa".

How do you proceed?

- Using the command Setinfo, tell the database, which field should contain the keywords, e.g.  
`setinfo ("Keywords", "Targetfield", "Descriptors")`
- Now tell the database, which are the text fields to be analyzed, e.g.  
`setinfo ("Keywords", "Sourcefields", "Title<br>Text")`
- Now specify the association(s):  
`Deletekeywordsinfo`  
`Addkeywordsinfo ("Is_Lower", true)`  
Now you define the fields from where to search the keywords:  
`setinfo "Keywords"."Sourcefields", "ABSTRACT<BR>TITLE"`
- Now you can clear the keyword list, if you like: `Deletekeywords`
- At last, allocate the keywords: `autokeywords (1)`  
If you wish to allocate a keyword only, if it occurs more than 1 time, set the variable to 2 or higher.

### 3.10 How to generate automatic keyword searches

The above method will search the defined fields for the keywords themselves, that means that if the text contains "Apple", it will add "Fruit", but not if it contains "Apples".

Sometimes you will want to find keywords by defining complete searches for them. Look at these examples:

UK	England or Scotland or Ireland or (united adj kingdom) or (great adj Britain)
Apple	Apple or Apples
Fruit	Apple* or plum* or banana* or pear*
Berries	*berry* or *berries*
NY	(new adj york) or manhattan or ...

This will give you a much higher degree of freedom. Like in line 2 and 3, you can define the hierarchy, or you simply can define a search that will find fruit.

How do you proceed?

First, create the field where the newly generated keywords should be put: Let us call it "Autokeywords".

```
createfield ("Autokeywords")
```

Next, add the keywords and searches:

```
Addkwsearch "NY", "(new adj york) or manhattan"
```

...

Now let us not forget to tell the autokeywords procedure (if we use it at all) that it should look at a new field

```
setinfo "Keywords", "Sourcefields", "ABSTRACT<BR>"Autokeywords"
```

Now let us run the automatic search

```
Autokwsearch ("Autokeywords")
```

If we use Autokeywords, we can start the procedure now:

```
Autokeywords (1)
```

If we use autokwsearch, we do not need the autokeywords procedure. But it has one big advantage: It know hierarchies, the autokwsearch procedure does not.

### 3.11 How to work with tag clouds

You can produce tag clouds, either from the contents of your database or from the contents of your search results.

Start with cloud\_setup

You have to add the following parameters:

- Maximum words: The maximum of words to be analyzed
- Minimum font, maximum font size
- Lowestfrequency, Highestfrequency: Frequencies below the lowest and higher than the highest frequency will be ignored. If you set these values to 0, the lowest frequency is 1 and the highest frequency is infinite.
- Href: When clicked on a word in the cloud, the user will be sent to the href, the word itself is added after the href value.
- Minwordlength, maxwordlength: Words with a length below or above these values will be ignored. You can also set them to 0.

```
Cloud_Setup(1000,8,24,0,0,"http://www.newswatch.at/dosearch.aspx?s=",3,0
```

Then you can add the results of a search. If, for example, the results are stored in an array called hitarr, you would do it like this:

```
for i=lbound(hitarr) to ubound(hitarr)
    obj.gotodocument hitarr(i)
    t=obj.fieldtext ("Text")
    obj.cloud_addtext t
next
```

Then you get the result of the cloud with cloud\_create: You specify the number of words with the parameter you add.

```
response.write (obj.cloud_create (80))
```

This is an example of the output:

```
<a style="font-size:13px"
href="http://www.newswatch.at/dosearch.aspx?s=RSS">RSS</a>
<a style="font-size:23px"
href="http://www.newswatch.at/dosearch.aspx?s=Suche">Suche</a>
<a style="font-size:14px"
href="http://www.newswatch.at/dosearch.aspx?s=Sudoku">Sudoku</a>
<a style="font-size:13px"
href="http://www.newswatch.at/dosearch.aspx?s=Themen">Themen</a>
<a style="font-size:19px"
href="http://www.newswatch.at/dosearch.aspx?s=Newsletter">Newsletter</a>
<a style="font-size:18px"
href="http://www.newswatch.at/dosearch.aspx?s=Services">Services</a>
<a style="font-size:16px"
```

## 4 Properties and methods

### 4.1 Working with databases

#### 4.1.1 Openserver method

Syntax: Openserver (const ahost, auser, apass: String): OleVariant;  
Description: Opens the connection to the server.  
Variables: ahost: The host computer. Use "localhost" for your own machine, otherwise the IP address.  
auser: Your username  
apass: Your password  
Result: True if successful, otherwise false  
Example: `openserver ("localhost","root","")`

#### 4.1.2 Createtextdatabase method

Syntax: Createtextdatabase(const s: String)  
Description: Creates a text database  
Variables: s is the name of the database  
Result: none  
Example: `Createtextdatabase ("myfirstttext")`

#### 4.1.3 Createtopicdatabase method

Syntax: Createtopicdatabase(const s: String)  
Description: Creates a topic database  
Variables: s is the name of the database  
Result: none  
Example: `Createtopicdatabase ("myfirstttopic")`

#### 4.1.4 Createnamedatabase method

Syntax: Createnamedatabase(const s: String)  
Description: Creates a name database  
Variables: s is the name of the database  
Result: none  
Example: `Createnamedatabase ("myfirstttopicnames")`

#### 4.1.5 Databaseexists method

Syntax: Databaseexists (const s: String): boolean  
Description: Returns true, if the database s exists  
Variables: s is the name of the database  
Result: True or false  
Example: `if not databaseexists ("myfirstttopic") then Createtopicdatabase ("myfirstttopic")`

#### 4.1.6 Createtexttables method

Syntax: Createtexttables  
Description: Creates the tables of the text database. If they already exist, they are replaced by empty tables without any warning!  
Variables: none  
Result: none  
Example: `Createtexttables`

#### 4.1.7 Createtopictables method

Syntax: Createtopictables

Description: Creates the tables of the topic database. If they already exist, they are replaced by empty tables without any warning!

Variables: none

Result: none

Example: Createtopictables

#### 4.1.8 Createnametables method

Syntax: Createnametables

Description: Creates the tables of the named database. If they already exist, they are replaced by empty tables without any warning!

Variables: none

Result: none

Example: Createnametables

#### 4.1.9 Deletetextdatabase method

Syntax: Deletetextdatabase(const s: String)

Description: Deletes a text database with all the required tables

Variables: s is the name of the database

Result: none

Example: Deletetextdatabase ("myfirsttext")

#### 4.1.10 Deletetopicdatabase method

Syntax: Deletetopicdatabase(const s: String)

Description: Deletes a topic database with all the required tables

Variables: s is the name of the database

Result: none

Example: Deletetopicdatabase ("myfirsttopic")

#### 4.1.11 Textdatabaselist property

Syntax: Textdatabaselist: String

Description: Returns a list of text databases, separated by separator (see "General")

Interface: Read

Standard: none

Example: t = Textdatabaselist

#### 4.1.12 Topicdatabaselist property

Syntax: Topicdatabaselist: String

Description: Returns a list of topic databases, separated by separator (see "General")

Interface: Read

Standard: none

Example: t = Topicdatabaselist

#### 4.1.13 Namedatabase property

Syntax: Namedatabase: String

Description: Sets / returns the name of the named database

Interface: Read / Write

Standard: none

Example: Namedatabase = "Thenames"

#### 4.1.14 Textdatabase property

Syntax: Textdatabase: String

Description: Sets / returns the current text database

Interface: Read / Write

Standard: none

Example: t = textdatabase

### 4.1.15 Topicdatabase property

Syntax: Topicdatabase: String  
Description: Sets / returns the current topic database  
Interface: Read / Write  
Standard: none  
Example: Topicdatabase = "MyFirstttopic"

## 4.2 Working with text

### 4.2.1 Createfield method

Syntax: createfield (const s: string)  
Description: Creates a text field in the text database. In the current version, only text fields are supported.  
Variables: s is the name of the field  
Result: none  
Example: createfield ("filename")

### 4.2.2 Deletefield method

Syntax: deletefield (const s: string)  
Description: Deletes a text field in the text database and all of it's contents.  
Variables: s is the name of the field  
Result: none  
Example: deletefield ("filename")  
See also: Emptyfield method

### 4.2.3 Getfieldlist property

Syntax: Getfieldlist: String  
Description: Returns a list of fields in the current text database, separated by separator (see "General")  
Interface: Read  
Standard: none  
Example: t = getfieldlist

### 4.2.4 Getfieldname method

Syntax: function getfieldname (i: integer)  
Description: Returns the name of a field with the index i.  
Variables: I is the index of the field  
Result: The name of the field  
Example: gefieldname (1)

### 4.2.5 Getfieldid method

Syntax: function getfieldname (i: integer)  
Description: Returns the name of a field with the index i.  
Variables: I is the index of the field  
Result: The name of the field  
Example: gefieldname (1)

### 4.2.6 Setfieldmustindex method

Syntax: procedure Setfieldmustindex(const s: String; isindex: Boolean);  
Description: Determines if a field will be indexed or not. When you create a field, it is automatically set to true.  
Variables: s is the name of the field  
Isindex is true, if the field is to be indexed, else false  
Result: none

Example:        `setfieldmustindex ("filename",false)`

#### 4.2.7 Createdocument method

Syntax:        `procedure Createdocument(douupdate: WBoolean);`

Description:    Creates a new document

Variables:     If douupdate is True, the system remembers the unique number of this document. We recommend that you set it to false

Result:        none

Example:        `createdocument(false)`

#### 4.2.8 Currentdocument property

Syntax:        `Currentdocument: integer ;`

Description:    Returns the current document. This is the last created document or the document where you are with firstdocument, nextdocument ...

Interface:     Read

Standard:     none

Example:        `I = currentdocument`

#### 4.2.9 Firstdocument method

Syntax:        `procedure firstdocument`

Description:    switches to the beginning of the document table

Variables:     none

Result:        none

Example:        `firstdocument`

`If nextdocument then ...`

#### 4.2.10 Lastdocument method

Syntax:        `procedure lastdocument`

Description:    switches to the end of the document table

Variables:     none

Result:        none

Example:        `lastdocument`

`If nextdocument then ...`

#### 4.2.11 Nextdocument method

Syntax:        `procedure nextdocument`

Description:    switches to the next document

Variables:     none

Result:        True, if there is a next document

Example:        `firstdocument`

`While nextdocument do ...`

#### 4.2.12 Previousdocument method

Syntax:        `procedure previousdocument`

Description:    switches to the previous document

Variables:     none

Result:        True, if there is a previous document

Example:        `lastdocument`

`While previousdocument do ...`

#### 4.2.13 Deletedocument method

Syntax:        `procedure Deletedocument (id: Integer)`

Description:    Deletes a document

Variables:     ID is the index of the document

Result:        none

Example:        `deletedocument (4)`

#### 4.2.14 Deletedocuments method

Syntax:        `procedure deletedocuments`

Description:    Deletes all the documents

Variables:     none

Result:        none

Example:        `deletealldocuments`

#### 4.2.15 Gotodocument method

Syntax:        `procedure gotodocument(id: Integer)`

Description:    Makes the document with the index ID the current document

Variables:     ID is the index of the document

Result:        none

Example:        `gotodocument (22)`

#### 4.2.16 Documentexists function

Syntax:        `function documentexists (id: Integer)`

Description:    Returns true, if the document with the number ID exists, otherwise false

Variables:     ID is the index of the document

Result:        True or false

Example:        `if documentexists (100) then ...`

#### 4.2.17 Documentcount property

Syntax:        `Documentcount: Integer`

Description:    Returns the number of documents

Interface:     Read

Standard:     0

Example:        `For I = 1 to documentcount do ...`

#### 4.2.18 Getmaxdocnr method

Syntax:        `function getmaxdocnr: integer`

Description:    returns the index of the last document

Variables:     none

Result:        the index

Example:        `I = getmaxdocnr`

#### 4.2.19 Addtext method

Syntax:        `procedure Addtext(const fname, stext: String; Onthefly,  
Updatedoc: WBoolean)`

Description:    Adds text to the current document

Variables:     `fname`: the name of the field  
`stext`: the text itself

`Onthefly`. Should the index be updated on adding the document? If you add more than a few documents, it is much better if you set this to false and index afterwards

`Updatedoc`. If set to True, the document remembers when it has been changed.

If you add a new document and fill in the text, you may set it to false. If you add fields to an existing documents, you should set this variable to true at the last AddText.

Result:        none

Example:        `Addtext "ftext","This is my text",true,false`

#### 4.2.20 Updatedocument method

Syntax:        `procedure updatedocument`

Description:    Makes the document remember when it has been changed. Works like the last parameter in `addtext` or `changetext`

Variables: none  
Result: none  
Example: Updatedocument

#### 4.2.21 Locatetext method

Syntax: `function Locatetext(Docno: Integer; const Fieldname: String): integer`  
Description: Retrieves a text ID for a certain document and a certain field. You need this ID to retrieve or change the text.  
Variables: Docno is the unique number of the document  
Fieldname is the name of the field  
Result: The text ID  
Example: `I = locatetext (102, "Myfield")`

#### 4.2.22 Locatetextplain method

Syntax: `function Locatetextplain (Docno, Fieldid: Integer): integer`  
Description: Retrieves a text ID for a certain document and a certain field, as above. Instead of the name, you use the index of the field  
Variables: Docno is the unique number of the document  
Fieldid is the ID of the field  
Result: The text ID  
Example: `I = locatetext (102, 4)`

#### 4.2.23 Gettext method

Syntax: `function Gettext (ID: Integer): String`  
Description: Retrieves a text with a given ID  
Variables: ID is the text ID  
Result: The text  
Example: `t = gettext (1234)`

#### 4.2.24 Fieldtext method

Syntax: `function Fieldtext (fieldname: string): String`  
Description: Retrieves a text from the current field document. A shorter approach than gettext.  
Variables: Fieldname is the name of the field  
Result: The text  
Example: `t = fieldtext ("Myfield")`

#### 4.2.25 Fieldtextstart method

Syntax: `function Fieldtextstart (fieldname: string; len: integer): String`  
Description: Like fieldtext, but it returns only the first characters of the field, maximum length = len parameter. It will not break up the text in the middle of a word, but return the text with the last complete word.  
Variables: Fieldname is the name of the field, len is the max. length of the returned text  
Result: The text  
Example: `t = fieldtextstart("Myfield",100)`

#### 4.2.26 Changetext method

Syntax: `procedure Changetext(ID: integer; const: stext: String; Onthefly, Updatedoc: WBoolean)`  
Description: Changes the text of a document  
Variables: ID is the Text ID  
stext: the text itself  
Onthefly. Should the index be updated on adding the document? If you add more than a few documents, it is much better if you set this to false and index afterwards

Updatedoc. If set to True, the document remembers when it has been changed. If you add a new document and fill in the text, you may set it to false. If you add fields to an existing documents, you should set this variable to true at the last AddText.

Result: none

Example: `Changetext "Thisfield","This is the new text",true,true`

#### 4.2.27 Changefield method

Syntax: `procedure Changefield(const fname, stext: String; Onthefly, Updatedoc: WBoolean)`

Description: Changes the text of a document

Variables: fname: the name of the field

stext: the text itself

Onthefly. Should the index be updated on adding the document? If you add more than a few documents, it is much better if you set this to false and index afterwards

Updatedoc. If set to True, the document remembers when it has been changed.

If you add a new document and fill in the text, you may set it to false. If you add fields to an existing document, you should set this variable to true at the last AddText.

Result: none

Example: `Changefield "Thisfield","This is the new text",true,true`

#### 4.2.28 Deletetext method

Syntax: `procedure deletetext (ID: Integer)`

Description: Deletes a text with a given ID

Variables: ID is the text ID

Result: none

Example: `deletetext (1234)`

#### 4.2.29 Emptyfield method

Syntax: `procedure emptyfield (name: string; dokill: boolean)`

Description: Deletes all the contents of a field with a given name. Like "Deletefield", but the field name remains in the database.

Variables: name is the name of the field

if dokill is true, the contents are deleted, otherwise they are set to an empty string. Set it to false if you use the method very often.

Result: none

Example: `Emptyfield "Comment"`

See also: Deletefield method

#### 4.2.30 Replacefields method

Syntax: `procedure Replacefields (from, to: string)`

Description: Replaces the fields with the name "to" with the content of "from" and empties "from".

**There is no need to re-index.**

Why would you want to do that? For example You generate the contents of a field "keywords" from the contents of a news database automatically. You want to let your customers search the current contents of this field, while you generate its contents in another process. In this case, you could generate a field "KeywordsNew", then replace "Keywords" with "KeywordsNew".

Variables: "From" and "To" are field names.

Result: none

Example: `Replacefields "KeywordsNew","Keywords"`

#### 4.2.31 Readfiles method

Syntax: `procedure readfiles (Indexafter:Boolean)`

Description: reads the files of a specifield diretory. If the text database contains already a file with this name, the DLL checks if the contents of the file are newer than these of the text database or not.

Variables: If indexafter is TRUE, the new documents will be indexed

Result: none

Example: ` first we have to specify the field with the filename and the field with the text:  
setinfo "Watchfiles","Fields","fname<br>ftext"  
' if we set subdirs to 1, subdirectories are also read  
obj.setinfo "Watchfiles","Subdirs","1"  
' Now we tell the system where the files are  
' this could also be a list of directories, separated by the parameter "separator"  
obj.setinfo "Watchfiles","Directories","f:\text"  
' Now we tell the system where to store the extracted information:  
obj.setinfo "Watchfiles","Textfiledir","f:\tmp"  
  
' Sometimes, you might want to copy files from other locations  
' of the network to your computer. Then you have to set the  
' variable "mirror" to 1.  
obj.setinfo "Watchfiles","Mirror","1"  
' Now you need to specify a directory, where to copy the files:  
obj.setinfo  
"Watchfiles","Mirrordir","c:\inetpub\wwwroot\textfiles"  
' Even if you set "Subdirs" to 1, no subdirectories will be  
' created in "Mirrordir"  
  
obj.readfiles (true)

#### 4.2.32 Killdocumentsnotindirectories method

Syntax: procedure killdocumentsnotindirectories

Description: If you use the readfiles method to generate documents, and you always read the files from a given directory, you might want to delete the documents pointing to a given file if the file does not exist anymore. This procedure does exactly that. For this command, you must have set the information, where the filenames and the text are located (setinfo ("Watchfiles","Fields"))

Variables: none

Result: none

Example: killdocumentsnotindirectories

### 4.3 Indexing and searching

#### 4.3.1 Deleteindex method

Syntax: procedure deleteindex

Description: Deletes the word index

Variables: none

Result: none

Example: deleteindex

#### 4.3.2 Indexall method

Syntax: procedure indexall (loadwortlist: WBoolean; maximum: Integer)

Description: Indexes all the documents not yet indexed

Variables: If loadwortlist is set to TRUE, the current wordlist is loaded into memory before indexing. This will give the process more speed. Try always to set it to True. Maximum is the maximum of fields to be indexed. If you have any problems with memory or timeouts, use several indexall methods

Result: none

Example: indexall (true,1000000)

### 4.3.3 Indextempall method

<u>Syntax:</u>	procedure indextempall (loadwortlist: WBoolean; maximum: Integer)
<u>Description:</u>	Creates a new index for all the documents. Works like deleteindex and indexall, but keeps the tables of the old index, so that one process can still search while the other reindexes the database. <b>Always use it together with Copyindex, Recopyindex and Fieldstempnotindexed</b>
<u>Variables:</u>	If loadwortlist is set to TRUE, the current wordlist is loaded into memory before indexing. This will give the process more speed. Try always to set it to True. Maximum is the maximum of fields to be indexed. If you have any problems with memory or timeouts, use several indexall methods
<u>Result:</u>	none, but you can look at errornum to see if you were successful, If it is 101, there was an error with deleting and creating temporary tables. Just gibe it a new start. If it is 102, the renaming of the temporary tables did not succeed. In this case, you should delete the index and use either indexall or Indextempall.
<u>Example:</u>	<pre>` let's see how old the index is I = obj.openindexage if i = 0 then     response.write ("<h3&gt;indexing &gt;="" ("indexall")="" (obj.lasterror)="" 0="" br="" else="" end="" h3&gt;&lt;br&gt;&lt;br&gt;")="" if="" if<="" is="" obj.copyindex="" obj.fieldstempnotindexed="" obj.indextempall="" obj.recopyindex="" pre="" response.flush="" response.write="" running!&lt;="" then="" true,1000000="" wend="" while=""></h3&gt;indexing></pre>

### 4.3.4 Copyindex function

<u>Syntax:</u>	function copyindex: boolean)
<u>Description:</u>	Copies the index to temporary files and sets the "indexed flag" in all the text strings to false. For the rest see above <b>Always use it together with Indextempall</b>
<u>Variables:</u>	none
<u>Result:</u>	True if succeeds, false if fails. You can look at errornum to see if you were successful, If it is 101, there was an error with deleting and creating temporary tables. Just gibe it a new start.

### 4.3.5 Recopyindex function

<u>Syntax:</u>	function recopyindex: boolean)
<u>Description:</u>	Copies the index from temporary to productive files. For the rest see above <b>Always use it together with Indextempall</b>
<u>Variables:</u>	none
<u>Result:</u>	True if succeeds, false if fails. You can look at errornum to see if you were successful, If it is 102, the renaming of the temporary tables did not succeed. In this case, you should delete the index and use either indexall or Indextempall.

### 4.3.6 Indexchunksize property

<u>Syntax:</u>	Indexchunksize: Integer
<u>Description:</u>	Sets the standard for maximum field count per run when reading files and indexing. It corresponds to the value set with indexall.
<u>Interface:</u>	Read / Write

Example:     indexchunksize = 2000 ^ this is the standard

### 4.3.7 Maxbytesperround property

Syntax:       Maxbytesperround: Integer

Description:   Sets the standard for maximum bytes per run when reading files. When indexing, this value is divided by 5.

If you do not define this value, it is calculated on the basis of the total free memory.

Interface:     Read / Write

Example:       maxbytesperround = 300000000 ^ this is the standard

### 4.3.8 Openindexage property

Syntax:       Openindexage: Integer

Description:   In a multiuser environment, you should not index the database while another indexing process is running. This property is  
-1, if no indexing is under way  
0, if an indexing process has been started today, and is still running  
1..n, if a process has started n days before, but was not closed. This is unlikely, perhaps the machine running the indexing has crashed.

Interface:     Read

Example:       if openindexage <> 0 then Indexall (true,100)

### 4.3.9 Fieldsnotindexed property

Syntax:       Fieldsnotindexed: Integer

Description:   Retrieves the number of fields yet to be indexed

Interface:     Read

Example:       While fieldsnotindexed > 0  
                  Indexall (true,100)  
                  Wend

### 4.3.10 Fieldstempnotindexed property

Syntax:       Fieldstempnotindexed: Integer

Description:   Retrieves the number of fields yet to be indexed, like Fieldsnotindexed.  
**Works only with Indextempall, Copyindex and Recopyindex!** (See there)

Interface:     Read

Example:       While fieldsnotindexed > 0  
                  Indexall (true,100)  
                  Wend

### 4.3.11 Hastoindex property

Syntax:       Hastoindex: Boolean

Description:   Determines if the database has fields yet to index  
Can be used like fieldsnotindexed, but is faster

Interface:     Read

Example:       While hastoindex  
                  Indexall (true,100)  
                  Wend

### 4.3.12 Getindexoptions method

Syntax:       procedure getindexoptions

Description:   Loads the index options from the database into memory

Variables:    none

Result:       none

Example:       getindexoptions

### 4.3.13 Setindexoptions method

Syntax: procedure setindexoptions  
Description: saves the index options  
Variables: none  
Result: none  
Example: setindexoptions

### 4.3.14 Addseparator method

Syntax: procedure addseparator (c: char)  
Description: Separators are characters which separate words, like blank, dot, hyphen ...  
The standard-Separators are Charecters 0 to 46, 58 to 64, 91 to 96, 123 to 127, 143 to 153 and 160 to 191.  
Use this procedure to add another separator.\_  
Variables: c is a character.  
Result: none  
Example: addseparator ("\_")

### 4.3.15 Deleteseparator method

Syntax: procedure deleteseparator (c: char)  
Description: Separators are characters which separate words, like blank, dot, hyphen ...  
Use this procedure to delete a separator.  
Variables: c is a character.  
Result: none  
Example: deleteseparator("-")

### 4.3.16 Setseparators method

Syntax: procedure setseparators  
Description: Use this procedure to save the separators after changing them with addseparator an deleteseparataor  
Variables: none  
Result: none  
Example: setseparators

### 4.3.17 Stopwords property

Syntax: Stopwords: String  
Description: Retrieves or sets the stopword list, separated by separator (see "General")  
Interface: Read / Write  
Standard: empty  
Example: Stopwords = "the<br>is<br>are<br>will<br>can<br>do"

### 4.3.18 Addstopwords method

Syntax: procedure addstopwords (s: string)  
Description: Adds stopwors, separated by the property separator  
Variables: none  
Result: none  
Example: addstopwords  
"out<br>their<br>also<br>they<br>had<br>than<br>up<br>who<br>In<br>one"

### 4.3.19 IndexMinLength property

Syntax: Indexminlength: Integer  
Description: Retrieves or sets the minimum length of a word to be indexed  
Interface: Read / Write  
Standard: 1  
Example: Indexminlength = 3

**Caution:** Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect  
indexminlength,  
indexmaxlength,  
Indexstartwithnumericals,  
indexcontainsnumericals,  
indexumlaut,  
indexpos,  
maxmirrorbytes,  
soundex and  
maxindexwords.

#### 4.3.20 IndexMaxLength property

**Syntax:** Indexmaxlength: Integer  
**Description:** Retrieves or sets the maximum length of a word to be indexed  
**Interface:** Read / Write  
**Standard:** unlimited  
**Example:** Indexmaxlength = 35  
**Caution:** Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect  
indexminlength,  
indexmaxlength,  
Indexstartwithnumericals,  
indexcontainsnumericals,  
indexumlaut,  
indexpos,  
maxmirrorbytes,  
soundex and  
maxindexwords.

#### 4.3.21 IndexContainsNumericals property

**Syntax:** Indexcontainsnumericals: Boolean  
**Description:** Retrieves or sets a Boolean variable which indicates if words to be indexed may contain numbers  
**Interface:** Read / Write  
**Standard:** true  
**Example:** Indexcontainsnumericals = false  
**Caution:** Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect  
indexminlength,  
indexmaxlength,  
Indexstartwithnumericals,  
indexcontainsnumericals,  
indexumlaut,  
indexpos,  
maxmirrorbytes,  
soundex and  
maxindexwords.

#### 4.3.22 IndexStartwithNumericals property

**Syntax:** Indexstartwithnumericals: Boolean  
**Description:** Retrieves or sets a Boolean variable which indicates if words to be indexed may start with a number. Note: If Indexcontainsnumericals is set to false, this variable will automatically be registered as false.  
**Interface:** Read / Write  
**Standard:** true  
**Example:** Indexstartwithnumericals = false  
**Caution:** Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect

indexminlength,  
indexmaxlength,  
Indexstartwithnumericals,  
indexcontainsnumericals,  
indexumlaut,  
indexpos,  
maxmirrorbytes,  
soundex and  
maxindexwords.

### 4.3.23 IndexUmlaut property

Syntax: Indexumlaut: Boolean

Description: Retrieves or sets a Boolean variable, determining if German umlauts “ä,ö,ü ...” are transferred into “ae, oe, ue ...” when indexing, so that you can search for “ä” or “ae” and retrieve the same result.

Interface: Read / Write

Standard: false

Example: Indexumlaut = true

Caution: Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect  
indexminlength,  
indexmaxlength,  
Indexstartwithnumericals,  
indexcontainsnumericals,  
indexumlaut,  
indexpos,  
maxmirrorbytes,  
soundex and  
maxindexwords.

### 4.3.24 Maxindexwords property

Syntax: Maxindexwords: Integer

Description: Retrieves or sets the maximum word count to be indexed within a field.

Interface: Read / Write

Standard: 2.147.483.647

Example: Maxindexwors = 1000

' this will index only the first 1000 words of a field.

Caution: Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect  
indexminlength,  
indexmaxlength,  
Indexstartwithnumericals,  
indexcontainsnumericals,  
indexumlaut,  
indexpos,  
maxmirrorbytes,  
soundex and  
maxindexwords.

### 4.3.25 Maxmirrorbytes property

Syntax: Maxmirrorbytes: integer

Description: Maxmirrorbytes is the value for the maximum of bytes to be mirrored in one round. You will need it if you have a lot of files to mirror, so that the machine can mirror a part of the files, and another the next day.

Interface: Read / Write

Standard: 0, this means that no maximum is defined.

Example: obj.Maxmirrorbytes = 100000000

Caution: Always set this value and then use setindexoptions! In order to retrieve the current

values, use `getindexoptions`. This will affect `indexminlength`, `indexmaxlength`, `Indexstartwithnumericals`, `indexcontainsnumericals`, `indexumlaut`, `indexpos`, `maxmirrorbytes`, `soundex` and `maxindexwords`.

#### 4.3.26 Indexpos property

Syntax: `Indexpos: integer`

Description: Retrieves or sets 1, if the positions of all the words in a field will be registered during indexing, and 0 if only the position of the first word in the field will be registered. When set to 1, you can do frequency analysis, or show the searched terms in a different color. When set to 0, the indexing will be faster.

Interface: Read / Write

Standard: 1

Example: `Indexpos = 0`

Caution: Always set this value and then use `setindexoptions`! In order to retrieve the current values, use `getindexoptions`. This will affect `indexminlength`, `indexmaxlength`, `Indexstartwithnumericals`, `indexcontainsnumericals`, `indexumlaut`, `indexpos`, `maxmirrorbytes`, `soundex` and `maxindexwords`.

#### 4.3.27 Searchfieldstring property

Syntax: `Searchfieldstring: String`

Description: Defines the field in which to search. If not set, the search will occur in all fields.

Interface: Read / Write

Standard: empty

Example: `searchfieldstring = "author"`

#### 4.3.28 Searchsortstring property

Syntax: `Searchsortstring: String`

Description: Defines the sort field for a search output. If set to "+", the oldest documents will be first, if it is "-", then the newest.  
It can also be used to randomize a result: Use the value "\_random".  
If you wish to "seed" then randomization process, use "\_random" plus any string of characters, e.g. "\_random\_this\_is\_me".

Interface: Read / Write

Standard: empty

Example: `searchsortstring = "country"`  
`searchsortstring = _random`

#### 4.3.29 Maxwordpos property

Syntax: `maxwordpos: integer;`

Description: Restricts a word search to a position within a field.

Interface: Read / Write

Standard: 0

Example: `maxwordpos = 5`  
' will only find documents where the seachword is within the

first 5 words of a field.

### 4.3.30 Setsearchselect method

<u>Syntax:</u>	Procedure setsearchselect (select: string; ainclude: Boolean)
<u>Description:</u>	Restricts a word search to a given array of document numbers or excludes this array from the search.
<u>Variables:</u>	Select is the array of document numbers, separated by commas. If ainclude is true, the search result may contain only numbers included in "select". If it is false, the numbers included in "select" are excluded from the search result. If you do not wish not to restrict the search to a given array, set "Select" to empty.
<u>Standard:</u>	empty
<u>Example:</u>	setsearchselect "2,4,6",false setsearchselect "",true

### 4.3.31 Search method

<u>Syntax:</u>	Procedure search (s: string): integer
<u>Description:</u>	Searches for the string s
<u>Variables:</u>	s is the searchstring
<u>Result:</u>	Number of search results
<u>Example:</u>	Search ("topic*")

The Search string is a string with a search term. You can also use complex queries with the following features:

AND	finds documents containing both words
OR	finds documents containing one or more of the words
NEAR	the search terms must be near each others, maximum 8 words in between.
NOT	finds documents with the first word, where the last word does not occur.

In addition, you can use

(, )	(brackets) to combine more than one operator and to define the order of the processing,
*	to truncate a search
.	to perform a field search: Searchword.fieldname
:	to specify the max. position of a word in the field. Example: red:5.description finds documents with the word red in the field description, but only if it occurs in the position 1 to 5. If you do not wish to specify a field, enter "red:5". See also the Maxwordpos property.
empty	if you wish to search for empty fields: empty.fieldname. NOTE: To find these, you must have entered an empty string into the field before
notempty	if you wish to search for not empty fields: empty.fieldname.

AND, OR, NOT can be replaced by "+", ",", "-".

SPEED UP your search with ",:":

If you search for "red or blue or green" you can enter "red;blue;green", which will be processed much faster. Leave nor blanks between the words.

You can also combine with truncation and fields, e.g. "red\*;green\*.color"

### Japanese characters:

There are 2 ways to search Japanese text:

#### Substring search:

This takes longer, because it analyzes the whole text of a field, or all the text.

The syntax is the following:

```
Search ("substr:word")
```

Or

```
Search ("substr:word.field")
```

**Note:** "substr" must be lowercase.

#### Word search:

In Japanese, no blanks are used. Therefore it is difficult to split the text into single words. Word endings are always written in the character set hiragana.

Therefore, Hiragana is ignored, the index contains only Kanji, Katakana and Roman characters.

**Note:** The Japanese word search is right truncated automatically.

### 4.3.32 Numberofhits property

Syntax:           Numberofhits: integer  
Description:   Returns the number of hits of the last search, just like the result of the search method.  
Interface:       Read  
Standard:        0  
Example:         for I:=1 to numberofhits do ...

### 4.3.33 Firsthit method

Syntax:           procedure firsthit  
Description:   switches to the beginning of the search results  
Variables:       none  
Result:          Number of search hits  
Example:         firsthit  
                    If nexthit then ...

### 4.3.34 Lasthit method

Syntax:           procedure lasthit  
Description:   switches to the end of the search results  
Variables:       none  
Result:          Number of search hits  
Example:         lasthit  
                    If nexthit then ...

### 4.3.35 Nexthit method

Syntax:           procedure nexthit  
Description:   switches to the next hit  
Variables:       none  
Result:          True, if there is a next hit  
Example:         firsthit  
                    While nexthit do ...

### 4.3.36 Previoushit method

Syntax:           procedure previoushit  
Description:   switches to the previous hit  
Variables:       none  
Result:          True, if there is a previous hit  
Example:         lasthit  
                    While previoushit do ...

### 4.3.37 Maxsearchtime property

Syntax: maxsearchtime: integer;  
Description: Restricts the search time to a given value in milliseconds  
Interface: Read / Write  
Standard: 0  
Example: maxsearchtime = 2000  
` restricts the search time to 2 seconds. If it takes longer, Errornum will return 3.

### 4.3.38 Maxhits property

Syntax: maxhits: integer  
Description: When you allow a large database to be searched on the internet, you might not want users to retrieve a large number of documents, because the server might generate a timeout. In this case, set maxhits to approx. 100-500.  
Interface: Read / Write  
Standard: 2.147.483.647  
Example: maxhits = 200...

### 4.3.39 Minsearchwordlength property

Syntax: minsearchwordlength: integer  
Description: If you search for a very short masked string, e.g. "e\*", the search can take up to several seconds. Use this property to avoid this. If you set it to 4, "com\*" will be processed, "co\*" will not. "co" will be processed because it is not masked. If a search contains such a too short string, the searchisinprecise property will be set to true and the number of hits will be 0.  
Interface: Read / Write  
Standard: 1  
Example:  
obj.minsearchwordlength = 4  
numberofhits = obj.search (s)  
if obj.searchisinprecise > 0 then  
    response write ("search string too short!")  
end if

### 4.3.40 Searchisinprecise property

Syntax: Searchisinprecise: integer  
Description: When the search terminates due to an excession of the values set in maxsearchtime, or minsearchwordlength, this property returns 1, otherwise 0. Furthermore, the variable errornum is set.  
Interface: Read  
Standard: 0  
Example: if searchisinprecise = 1 then  
Response.write ("The results exceeds the max. number of hits!")

### 4.3.41 Errornum property

Syntax: Errornum: integer  
Description: When the search or any other process terminates due to an error, searchisinprecise is set and the errornum indicates then type of error, otherwise errornum is 0.  
Values: 0 if no error occurs  
1 if a search term is shorter than minsearchwordlength  
2 if the search result contains more than maxhits  
3 if maxsearchtime is exceeded  
  
See other error numbers in chapter "error numbers"  
Interface: Read  
Standard: 0  
Example: if searchisinprecise = 1 then

```
If errornum = 1 then
    Response.write ("The results exceeds the max. number of
hits!")
```

#### 4.3.42 Hitsasstring method

**Syntax:** function hitsasstring: string  
**Description:** returns all the hits of your last search as string.  
**Variables:** none  
**Result:** The hits, separated by comma  
**Example:**  
dim somestring  
somestring = obj.hitsasstring  
array\_var = split (somestring, ",")  
` now we have an array which we can use anywhere in our project

#### 4.3.43 Hitsasstringmax method

**Syntax:** function hitsasstringmax (max: integer): string  
**Description:** like hitsasstring, but returns the first *max* hits of your last search as string.  
**Variables:** Max is an integer, maximum of hits to return  
**Result:** The hits, separated by comma  
If you wish randomized results, look at the searchsortstring property  
**Example:**  
dim somestring  
somestring = obj.hitsasstringmax (500)  
array\_var = split (somestring, ",")  
` now we have an array which we can use anywhere in our project

#### 4.3.44 Lastsearchwords property

**Syntax:** Lastsearchwords: string  
**Description:** Returns the search terms of the last search, separated by the standard separator (default: "<br>")  
**Interface:** Read  
**Standard:** ""  
**Example:** Search for "Red\* or white". Lastsearchwords will return "red\*<br>white".

#### 4.3.45 Fieldtextwithtags method

**Syntax:** function fieldtextwithtags const fname, tagbefore, tagafter: String; kwic, Firstlines, Options: Integer)  
**Description:** Like fieldtext, but better suited to present search results  
**Variables:** Fname is the name of the field  
Tagbefore and Tagafter are placed before and behind the words you have searched. KWIC are the characters to be returned around your search terms. If you set it to 20, every search terms will be surrounded by about 20 characters. If set to 0, the whole text will be returned.  
Firstlines: Set this to 10, if you only want the first 10 lines of text. If you want the whole text, use 0  
Options: if set to 1, CR/LF sequences will be replaced by <br>. Further options may follow with later versions.  
You can also use the replacestring function to get a similar result.  
**Result:** The text  
**Example:** t = fieldtextwithtags ("Myfield", "<b>", "</b>", 100, 20, 1)

#### 4.3.46 Savehitpos method

**Syntax:** function savehitpos: string  
**Description:** You need this if you want to use the above function "fieldtextwithtags". Savehitpos stores information about the positions of the search terms in your search results.  
**Variables:** none  
**Result:** A string, which you can use with the procedure loadhitpos.

Example:     `obj.search ("this word")  
session ("foundhits") = obj.savehitpos`

#### 4.3.47 Savehitposselected method

Syntax:       `function savehitposselected (s: string): string`  
Description:   Like savehitpos, but you use it if you only want the positions for a subset of your search.  
Variables:     s is a string containing the document numbers from which you wish to get the positions, separated by commas. You can begin or end the string with commas, but you need not.  
Result:        A string, which you can use with the procedure loadhitpos.  
Example:       `obj.search ("this word")  
session ("foundhits") = obj.savehitposselected ("1,2,5,11,13,")`

#### 4.3.48 Loadhitpos method

Syntax:        `procedure loadhitpos (s: string)`  
Description:   You need this if you want to use the above function "fieldtextwithtags". The procedure stores the information retrieved with savehitpos back to the DLL.  
Variables:     S is the string with the position information  
Result:        none  
Example:       `obj.loadhitpos (session ("foundhits"))  
t=obj.fieldtextwithtags (ftextfield,"<b>","</b>",>75,0,1)`

#### 4.3.49 Dohitpos property

Syntax:        `Dohitpos: integer`  
Description:   If you do not need the positions of your hits (see the above methods), set the value to 0. In this case, Loadhitpos will retrieve an empty string. Otherwise, use 1.  
Interface:     Read / Write  
Standard:     1  
Example:       `obj.dohitpos=0`

#### 4.3.50 Dorandom property

Syntax:        `Dorandom: integer`  
Description:   If you wish to restrict your searchresults (see: maxhits) and want them randomly selected, set this value to 1. Otherwise, use 0. You can also set the seed of the random process (see: randomseed).  
Interface:     Read / Write  
Standard:     0  
Example:       `obj.dorandom=1  
obj.randomseed=1067  
obj.maxhits=3000  
hits=obj.search ("red and wine")`

#### 4.3.51 Randomseed property

Syntax:        `Randomseed: integer`  
Description:   Works with dorandom (see there) and sets the seed of the randomizing process.  
Interface:     Read / Write  
Standard:     0  
Example:       `obj.dorandom=1  
obj.randomseed=1067  
obj.maxhits=3000  
hits=obj.search ("red and wine")`

#### 4.3.52 Orderstring property

Syntax:        `Orderstring: string`  
Description:   Orders your search result

Use WORDPOS to order your search results by the position of your search term in a field.  
Use WORDPOS to order your search results by the position of your search term in the complete document.  
More examples will follow.

Interface: Read  
Standard: "  
Example: `obj.orderstring='abspos'`

### 4.3.53 Hitposperdocument function

Syntax: `Hitposperdocument (ID: integer; lastsearchstring: string)`  
Description: Like savehitpos. You can use it if you have set dohitpos to 0.  
Variables: ID is the index of the document, lastsearchstring is the string received by the lastsearchstring method (see there).  
Result: A string containing the hits and their positions.  
Example:  
`obj.dohitpos=0  
obj.search ("this or that")  
T = obj.hitsasstring  
Arr=split (t,",")  
T = obj.hitposperdocument (arr(0))  
obj.loadhitpos (t)`

### 4.3.54 Defaultmask property

Syntax: `Defaultmask: integer`  
Description: Determines if a search term is masked automatically or not. Possible values:  
`nomask = 0  
rightmask = 1  
leftmask = 2  
bothmask = 3`  
Interface: Read / Write  
Standard: `nomask`  
Example: `Defaultmask = 3` (a search term "word" will now be searched as `"*word"` and find words beginning or ending with "word").

### 4.3.55 Defaultlogic property

Syntax: `Defaultlogic: integer`  
Description: Determineshot to proceed if no operator is entered between two search terms. Possible values:  
`AND = 1  
OR = 2`  
Interface: Read / Write  
Standard: `AND`  
Example: `Defaultlogic = 2` (a search like "one two" will now be searched as "one or two").

### 4.3.56 Lastsoundex method

Syntax: `function lastsoundex (logic: Boolean): string`  
Description: finds words in the database similar to your last search term(s)  
Variables: 0 if it may contain numbers, 1 if not  
Result: Possible search terms, separated by the standard separator  
Example:  
`obj.search ("watr")  
t = lastsoundex (0)`  
This will retrieve words like "water" or "waiter", if they are parts of your database.

### 4.3.57 Soundex property

<u>Syntax:</u>	soundex
<u>Description:</u>	Determines how to work with the soundex algorithm 0: during indexing, no soundex will be stored in the words table. The method lastsoundex retrieves words according to the classical (English) soundex algorithm. 1: during indexing, the soundex value will be stored in the words table for faster performance. The method lastsoundex retrieves words according to the classical (English) soundex algorithm. 2: Like 1, but the soundex algorithm is replaced with an algorithm optimized for the German language.
<u>Interface:</u>	Read / Write
<u>Standard:</u>	0
<u>Example:</u>	obj.Getindexoptions obj.soundex = 2 obj.Setindexoptions
<u>Caution:</u>	Always set this value and then use setindexoptions! In order to retrieve the current values, use getindexoptions. This will affect indexminlength, indexmaxlength, Indexstartwithnumericals, indexcontainsnumericals, indexumlaut, indexpos, maxmirrorbytes, soundex and maxindexwords.

## 4.4 Working with frequencies

### 4.4.1 Analyzefrequency method

<u>Syntax:</u>	Analyzefrequency
<u>Description:</u>	Determines how often the words (except stopwords) occur in the database. For every word, the documents in which it occurs are counted.
<u>Variables:</u>	none
<u>Result:</u>	none
<u>Example:</u>	Analyzefrequency

### 4.4.2 Analyzerrelativefrequency method

<u>Syntax:</u>	Analyzerrelativefrequency (minimum: integer)
<u>Description:</u>	Determines how often two words occur in the same document.
<u>Variables:</u>	minimum determines a minimal occurrence count. You should not set it as low as 1, because this would take quite long in a large database. Choose a value of 3-10 for a database with < 10 MB text, 5-20 for a larger one By default, this procedure it will only analyze the first 100 words of a field in a document. If you want to set another value, use Setinfo 'Analyzerrelativefrequency', 'Maxwords', 'nnnn' where nnnn is the value. Please note that this function can eat um a lot of processing time and memory. A value of 1000 might be too high. If you need more "power" in this function, please ask us, perhaps we can help ypu.
<u>Result:</u>	none
<u>Example:</u>	Analyzerrelativefrequency (5)

### 4.4.3 Showfrequency method

<u>Syntax:</u>	function showfrequency (maximum: integer) string
<u>Description:</u>	Returns a string with the words and their frequencies. The lines are separated by

`separator`, the word and the number are separated by `separator2`. The output is sorted by frequency

Variables: Maximum determines the maximum number of lines.

Result: The string with the words and their frequencies.

Example:  
`obj.separator = "<br>"`  
`obj.separator2 = " / "`  
`obj.showfrequency (10)`

The output might be:

`word1 / 100`  
`word2 / 97`

...

#### 4.4.4 Showrelativefrequency method

Syntax: `function showrelativefrequency (maximum: integer) string`

Description: Returns a string with both words and their frequencies. The lines are separated by `separator`, the words and the number are separated by `separator2`. The output is sorted by frequency

Variables: Maximum determines the maximum number of lines.

Result: The string with the words and their frequencies.

Example:  
`obj.separator = "<br>"`  
`obj.separator2 = " / "`  
`obj.showrelativefrequency (10)`

The output might be:

`word1 / word2 / 33`  
`word2 / word4 / 25`

...

#### 4.4.5 Showrelativefrequencyofword method

Syntax: `function showrelativefrequencyofword (const s: string; maximum: integer) string`

Description: Returns a string with the words and their frequencies. The lines are separated by `separator`, the word and the number are separated by `separator2`. The output is sorted by frequency

Variables: Maximum determines the maximum number of lines.

Result: The string with the words and their frequencies.

Example:  
`obj.separator = "<br>"`  
`obj.separator2 = " / "`  
`obj.showrelativefrequencyofword ("Austria", 10)`

The output might be:

`word1 / 33`  
`word2 / 25`

...

#### 4.4.6 Drawgraph method

Syntax: `function drawgraph (const s: string; count,depth: integer; orderbyfrequency, orderbyconnections. Boolean; width, height, quality: integer; const nameSeed: String): string;`

Description: Instead of displaying the relative frequencies as a table, you can create a graph with words and connections between them.

Variables: `s` is the word, with which the analysis starts.

`count`: for each iteration (see `depth`), `count` words are analyzed.

if `depth = 1`, then the `count` words with the highest co-frequency to the word `s` are displayed. If `depth = 2`, this process is repeated for all of the words found in the last iteration. If it is 3, one more repetition:

For good performance, you should keep the value of depth low or the value of count very low.

If `orderbyfrequency` is set to true, the words are ordered by the number of documents, in which they occur.

If `orderbyconnections` is set to true, the words are ordered by the number of connections they have to the other words in the picture.

`Nameseed` is the path and the first part of the name of the graphic file.

**Result:** The name of the graphic file (jpeg). The system takes the variable `nameseed` and adds the date and time and a number.

**Example:** `t = drawgraph ("Austria",10,2,false,false,"e:\tmp\graph")`  
would produce a filename like  
`e:\tmp\graph2003-03-24-14-45-30-123456-1.jpg`

#### 4.4.7 Drawclick method

**Syntax:** `function drawclick (const s: string; count,depth: integer; orderbyfrequency, orderbyconnections: Boolean; width, height, quality: integer; const outdir, nameseed, stringparam, countparam, depthparam, orderparam: String): string;`

**Description:** Works like `drawgraph`, but it produces a HTML file with a HTML-map. When you click on the words, a new graph with the relative frequencies of this word will appear.

**Variables:** `s` is the word, with which the analysis starts.

`count`: for each iteration (see `depth`), `count` words are analyzed.

if `depth = 1`, then the `count` words with the highest co-frequency to the word `s` are displayed. If `depth = 2`, this process is repeated for all of the words found in the last iteration. If it is 3, one more repetition:

For good performance, you should keep the value of depth low or the value of count very low.

If `orderbyfrequency` is set to true, the words are ordered by the number of documents, in which they occur.

If `orderbyconnections` is set to true, the words are ordered by the number of connections they have to the other words in the picture.

`outdir` is the directory, where the graphs are located

`Nameseed` is the path and the first part of the name of the graphic file.

`Stringparam` contains the first part of the URL to be generated, which contains the word to be analyzed.

`Countparam` contains the next part of the URL, followed by `Depthparam` and `Orderparam`. Look on the example below:

**Result:** The name of the HTMLfile. The system takes the variable `nameseed` and adds the date and time and a number.

**Example:** `t = obj.drawclick (s,wi,de,orfre,orco,800,600,90,"d:\tmp\graph","temp", "showgraph.asp?thestring=", "&count=", "&depth=", "&order=")`  
would produce a filename like

`d:\tmp\graph2003-03-24-14-45-30-123456-1.html`

When you click on one of the words, a response like this could be generated:

`showgraph.asp?thestring=xxx&count=10&depth=1&order=0`

#### 4.4.8 Drawclickx, Drawclicky property

**Syntax:** `drawclickx=nn-nn`

**Description:** When working with `drawclick`, you have sometimes to adjust the position of the clickable areas. The values below are a good example.

**Interface:** Read / Write

**Standard:** 1

**Example:** `obj.drawclickx=1.03`  
`obj.drawclicky=1.1`

#### 4.4.9 Killawordforfrequency method

**Syntax:** `procedure killawordforfrequency (const s: string)`  
**Description:** When you do frequency analysis, especially when producing graphics, you will find that some words are not relevant in this respect. Especially verbs might have no meaning concerning the connections between words. With this method you can eliminate them:  
**Variables:** s is the word you wish to eliminate  
**Result:** none  
**Example:** `killawordforfrequency ("requires")`

#### 4.4.10 Returnawordtofrequency method

**Syntax:** `procedure returnawordtofrequency (const s: string)`  
**Description:** Does the opposite from killawordforfrequency: The word will show up in the graphic again.  
**Variables:** s is the word you wish to add  
**Result:** none  
**Example:** `returnawordtofrequency ("requires")`

#### 4.4.11 Freqstopwords property

**Syntax:** `freqstopwords = s`  
**Description:** Instead of using the above 2 methods, you can work with a list of "stopwords" for frequencies.  
**Interface:** Read / Write  
**Standard:** 1  
**Example:** `returnawordtofrequency ("requires")`

#### 4.4.12 Savefrequency method

**Syntax:** `procedure savefrequency`  
**Description:** Enables you to save the frequencies of today to a table "afreqdate". You want to do that if you wish to create time series.  
**Variables:** none  
**Result:** none  
**Example:** `savefrequency`

#### 4.4.13 Saverelativefrequency method

**Syntax:** `procedure saverelativefrequency`  
**Description:** Enables you to save the frequencies of today to a table "arelfreqdate". You want to do that if you wish to create time series.  
**Variables:** none  
**Result:** none  
**Example:** `saverelativefrequency`

### 4.5 Working with topics

#### 4.5.1 Addreplacetopic method

**Syntax:** `procedure Addreplacetopic(const aleft, aright, Assocleft, Assocright: String; Isasynonym: Boolean)`  
**Description:** Adds or replaces a topic. For a detailed description, please see "how to work with Topics"  
**Variables:** Aleft, aright are the words ww enter  
Assocleft, Assocright are the associations. You might also think of them as verbs.  
Isasynonym determines, whether both words are synonyms.  
**Result:** none  
**Example:** `addreplacetopic "Vegetable", "potato", "can_be", "is_a", false.`

## 4.5.2 Getalltopicwords method

Syntax: function getalltopicwords: string  
Description: Returns all the words entered as topics, separated by separator  
Variables: none  
Result: The string  
Example: t = getalltopicwords

## 4.5.3 Deletewordfromtopics method

Syntax: procedure deletewordfromtopics (const s: string)  
Description: Deletes a word and all the associations from the topics database."  
Variables: s is the word to be deleted.  
Result: none  
Example: deletewordfromtopic ("beer")

## 4.5.4 Deleterightwordsto method

Syntax: procedure deleterightwordsto (const s,p: string)  
Description: Deletes all the associations with s as a left word and p as an association.  
**NOTE: The words themselves are not deleted from the database.**  
Variables: s is the left word,  
P is the association.  
Result: none  
Example: deleterightwordsto ("Shell company", "has\_employees")

## 4.5.5 Showhierarchy method

Syntax: function showhierarchy (const s: string): string  
Description: Shows all the subgroups of a given word."  
Variables: s is the word to be analyzed.  
Result: A list of words  
Example: t = showhierarchy ("drink")  
might produce  
beer, wine, milk ...

## 4.5.6 Associations property

Syntax: Property associations: string  
Description: Returns all the associations of the topics database, separated by the current separator  
Interface: Read  
Standard: empty  
Example: t = associations

## 4.5.7 Gettopiclayer method

Syntax: function gettopiclayer (const s,assoc: string; lookforright: integer): string  
Description: Shows a subgroup of a given word with a given association."  
If s is empty, it shows the highest hierarchy part or the lowest, depending if lookforright is set to 1 or not.  
Variables: s is the word to be analyzed.  
if you leave the words free, the words of the highest hierarchy are displayed  
assoc is the verb, or the association  
lookforright is 1, if we look for the right association (see method addreplacetopic), otherwise 0  
Result: A list of words  
Example: t = gettopiclayer ("Austria","HasCountries",0)  
might produce "Wien, Salzburg, Kärnten ... "  
t = gettopiclayer ("","HasCountries",0)  
might produce "Asia,Europe, America "

#### 4.5.8 Gettopiclayerno method

**Syntax:** `function gettopiclayerno (const n: integer; const assoc: string; lookforright: integer): string`

**Description:** Like gettopiclayer, but instead of s it needs the number of the word in the words database. N must not be zero.

**Variables:** n is the number of the word to be analyzed.  
assoc is the verb, or the association  
lookforright is 1, if we look for the right association (see method addreplacetopic), otherwise 0

**Result:** A list of numbers of words. You can look up the words in the twords table.

#### 4.5.9 Getalltopiclayerenos method

**Syntax:** `function getalltopiclayerenos (const n: integer; const assoc: string; lookforright: integer): string`

**Description:** Like gettopiclayerno, but returns all the numbers throughout the hierarchies.

**Variables:** n is the number of the word to be analyzed.  
assoc is the verb, or the association  
lookforright is 1, if we look for the right association (see method addreplacetopic), otherwise 0

**Result:** A list of numbers of words. You can look up the words in the twords table.

#### 4.5.10 Righttopicwordstoassoc method

**Syntax:** `function righttopicwordstoassoc (const s: string): string`

**Description:** Shows all the right words of a given association with the left word s.

**Variables:** s is the word to be analyzed.

**Result:** A list of words

**Example:**  
`t = righttopicwordstoassoc ("traffic")`  
might produce  
railway, road traffic, air traffic

#### 4.5.11 Showsynonyms method

**Syntax:** `function showsynonyms (const s: string): string`

**Description:** Shows all the synonyms of a given word."

**Variables:** s is the word to be analyzed.

**Result:** A list of words

**Example:**  
`t = showsynonyms ("railway")`  
might produce  
Eisenbahn, Ferrovia ...

#### 4.5.12 Gettopicsearch method

**Syntax:** `function gettopicsearch (const s: string): string`

**Description:** Combines showhierarchy and showsynonyms methods and produces a search string containing all the supergroups, subgroups and synonyms. Note: If the search term is truncated, the other terms will also be truncated the same way.

**Variables:** s is the search string.

**Result:** A new search string

**Example:**  
`t = gettopicsearch ("Europe and Traffic")`  
might produce  
(Europe or Europa or France or Austria or Germany or Deutschland) and (traffic or railway or aviation or airline)

#### 4.5.13 Gettopicsearchpred method

**Syntax:** `function gettopicsearchpred (const s,p: string; depth: integer): string`

**Description:** Like gettopicsearch, but supergroups will not be included. It is also limited to a given association, or verb.

Variables: s is the search string.  
p is the association.  
Depth is the depth of the search. Use 1 or more.

Result: A new search string

Example: t = gettopicsearchpred ("dog", "has\_subgroup",1)  
might produce  
dog or dachshund or terrier or sheperd

#### 4.5.14 Addkeywordsinfo method

Syntax: procedure Addkeywordsinfo(const s: String; Goleft: Integer)

Description: This belongs to the automatic allocation of keywords to a text. You use it to add an association (see chapter "Automatic allocation of keywords")

Variables: s is the assooiation.  
Goleft is 0, if you use the roght association, otherwise 1..

Result: None

Example: Addkeywordsinfo ("IsUpper",true)

#### 4.5.15 Deletekeywordsinfo method

Syntax: procedure Deletekeywordsinfo

Description: Deletes the above mentioned information.

Variables: None

Result: None

Example: Deletekeywordsinfo

#### 4.5.16 Deletekeywords method

Syntax: procedure Deletekeywords

Description: Deletes the field containing the automatically allocated keywords.

Variables: None

Result: None

Example: Deletekeywords

#### 4.5.17 Autokeywords method

Syntax: procedure Autokeywords (Min: Integer)

Description: Automatically allocates keywords to text

Variables: Min is the minimum occurrence count of the word in one text.

Result: None

Example: Autokeywords (1)

#### 4.5.18 Addkwsearch method

Syntax: procedure Addkwsearch (keyword, searchtext: string)

Description: Adds a search string to a keyword.  
**For this and the next 3 functions, pleas read "How to generate automatic keyword searches"**

Variables: Keyword is the keyword, where you wish to add a search text. It has to exist already.  
Searchtext is the search to be performed.

Result: None

Example: Addkwsearch ("Flower", "Rose\* or lily or flower\*")

#### 4.5.19 Getkwsearch function

Syntax: function Getkwsearch (keyword: string): string

Description: Gets a search string for a keyword.

Variables: "Keyword" is the keyword.

Result: The allocated search text.

Example: s = getkwsearch ("Flower")

## 4.5.20 Deletekwsearch method

Syntax: procedure deletewsearch (keyword: string)  
Description: Deletes a search string for a keyword.  
Variables: "Keyword" is the keyword.  
Result: None  
Example: Deletewsearch ("Flower")

## 4.5.21 Autokwsearch procedure

Syntax: procedure autokwsearch (Fieldname: string)  
Description: Performs the automatic keyword search.  
Variables: "Fieldname" is name of the field where the newly generated keywords are stored.  
Result: None  
Example: autokwsearch ("KW")

# 4.6 Tag Clouds

## 4.6.1 Cloud\_Setup procedure

Syntax: procedure Cloud\_Setup(Maxwords, Minfontsize, Maxfontsize, Lowestfreq, Highestfreq: Integer; const ahref: WideString; Minwordlength, Maxwordlength: Integer);  
Description: Creates a cloud object. Please read the "Howto"-chapter to this issue.  
Variables: Maximum words: The maximum of words to be analyzed  
Minfontsize, Maxfontsize: Minimum font, maximum font size  
Lowestfreq, Highestfreq: Frequencies below the lowest and higher than the highest frequency will be ignored. If you set these values to 0, the lowest frequency is 1 and the highest frequency is infinite.  
ahref: When clicked on a word in the cloud, the user will be sent to the href, the word itself is added after the href value.  
Minwordlength, maxwordlength: Words with a length below or above these values will be ignored. You can also set them to 0.  
Result: None  
Example Cloud\_Setup (1000,8,24,0,0, "[http://www.newswatch.at/dosearch.aspx?s="](http://www.newswatch.at/dosearch.aspx?s=), 3,0

## 4.6.2 Cloud\_Addword procedure

Syntax: Cloud\_Addword(const s: WideString)  
Description: Adds a word to the cloud.  
Variables: s is the word you want to add.  
Result: None  
Example Cloud\_Addword ("Wire")

## 4.6.3 Cloud\_Addtext procedure

Syntax: Cloud\_Addtext(const s: WideString)  
Description: Adds a text to the cloud. The text will be split up into words and then added.  
Variables: s is the text you want to add.  
Result: None  
Example Cloud\_Addtext (fieldtext ("Abstract"))

## 4.6.4 Cloud\_Addwordfreq procedure

Syntax: Cloud\_Addwordfreq(const s: WideString; freq: Integer)  
Description: Adds a word and a frequency (the count of occurrences in the cloud)..  
Variables: s is the word you want to add, freq is the frequency.  
Result: None  
Example Cloud\_Addword ("Wire",19)

### 4.6.5 Cloud\_Keepfirst procedure

Syntax: Cloud\_Keepfirst(count, Highest: Integer)  
Description: Deletes all but the words with the highest or lowest frequency.  
Variables: Count is the word count, Highest defines if you mean the highest or lowest frequency.  
Result: None  
Example Cloud\_Keepfirst (100,True)

### 4.6.6 Cloud\_Sort procedure

Syntax: Cloud\_Sort  
Description: Sorts the values by frequency  
Variables: None  
Result: None  
Example Cloud\_Sort

### 4.6.7 Cloud\_Sortrandom procedure

Syntax: Cloud\_Sortrandom  
Description: Sorts the values by random  
Variables: None  
Result: None  
Example Cloud\_Sortrandom

### 4.6.8 Cloud\_Create function

Syntax: function Cloud\_Create(Count: Integer): string  
Description: Produces the cloud values in HTML.  
Variables: Count is the number of words to produce.  
Result: The HTML-output, see the example below.  
Example: s = Cloud\_Create (50)  
This might produce an output like this:

```
<a style="font-size:23px"
href="http://www.newswatch.at/dosearch.aspx?s=Suche">Suche</a>
<a style="font-size:14px"
href="http://www.newswatch.at/dosearch.aspx?s=Sudoku">Sudoku</a>
<a style="font-size:19px"
href="http://www.newswatch.at/dosearch.aspx?s=Newsletter">Newsletter</a>
```

## 4.7 General

### 4.7.1 Lasterror property

Syntax: Lasterror: String  
Description: Returns the last error, SQL or otherwise. You can also set it, e.g. if you want to make it empty  
Interface: Read / Write  
Standard: empty  
Example: t = lasterror

### 4.7.2 Separator property

Syntax: Separator: String  
Description: When you retrieve a list, the items of the list will be separated by this separator  
Interface: Read / Write  
Standard: <br>  
Example: separator = " / "

### 4.7.3 Separator2 property

Syntax: Separator2: String  
Description: Some of the list received will use this variable  
Interface: Read / Write  
Standard: <br>  
Example: separator2 = " / "

### 4.7.4 Setinfo method

Syntax: procedure setinfo (section,topic,value: string)  
Description: saves a string into the information table.  
You will need this for procedures like readfiles, but you can store other information as well (e.g. logging info)  
Variables: Section, Topic and Value are strings. You can use them like in INI-Files  
Result: none  
Example: setinfo "Watchfiles","Fields","First<br>Second"

### 4.7.5 Getinfo method

Syntax: function getinfo (section,topic: string)  
Description: reads a string from the information table  
Variables: Section and Topic are strings. You can use them like in INI-Files  
Result: The info string  
Example: t = getinfo ("Watchfiles","Fields")

### 4.7.6 Setinfofromfile method

Syntax: procedure setinfofromfile (section,topic,filename: string)  
Description: saves a list from a file into the information table.  
Variables: Section, Topic and Filename are strings.  
Result: none  
Example: setinfofromfile "Watchfiles","Fields","Mytable.txt"

### 4.7.7 Logging property

Syntax: Logging: boolean  
Description: some functions create logging information for debugging. Set logging to true if you need this.  
Interface: Read / Write  
Standard: false  
Example: logging = true  
readfiles  
logging = false

### 4.7.8 Logtype property

Syntax: Logtype: integer  
Description: If set to 0, the logging information will be stored in a string. If set to 1, it will be written in the table ADBInf in the text database.  
If set to 2, the logging events will be restricted.  
Interface: Read / Write  
Standard: f0  
Example: logging = true  
logtype = 1  
readfiles  
logging = false

### 4.7.9 Loglist method

Syntax: function loglist: string  
Description: Returns the complete log list.

Variables: none  
Result: The list, separated by separator  
Example: t = loglist

#### 4.7.10 Killogs method

Syntax: procedure killogs (s: string)  
Description: kills the logs older than the string s, which has the format "yyyy:mm:dd hh:mm:ss.zzz"  
Variables: If s = empty, the complete loglist is killed.  
Result: none  
Example: killogs ("2004-10-13")

#### 4.7.11 Killfilesbefore method

Syntax: procedure killfilesbefore (const mask. String; ageindays: Integer)  
Description: When using the drawgraph method, you will produce graphic files which are usually only downloaded once. Kill the older files with this method.  
Variables: Ageindays ist the age of the file in days. 0 would kill al the files, 1 the files 1 or more days old and so forth.  
Result: none  
Example: killfilesbefore (2)

#### 4.7.12 Execsql method

Syntax: function execsql (s,res: string): string  
Description: Executes a SQL statement. If res is defined, then an SQL search is performed, and the fields res are returned as a string, separated by ";".  
Variables: s is the SQL statement, res is the name of the field to be returned from a search  
Result: None or the results of the search  
Example: result = execsql ("select ID from atable where name = 'Miller'")

#### 4.7.13 Isotojis method

Syntax: function isotojis(s: string): string  
Description: Works only with Japanese characters. There are two methods of defining Japanese characters: ISO and JIS. This function converts strings from one method to the other.  
Variables: s is the string in ISO  
Result: the string in JIS  
Example: response.write (isotojis ("&#12371;&#12428;&#12431;&#26085;"))

#### 4.7.14 Jistoiso method

Syntax: function jistoiso (s: string): string  
Description: Works only with Japanese characters. There are two methods of defining Japanese characters: ISO and JIS. This function converts strings from one method to the orther.  
Variables: s is the string in JIS  
Result: the string in ISO  
Example: response.write (jistoiso (request.form ("theline")))

#### 4.7.15 Cyrillictoiso method

Syntax: function cyrillictoiso (s: string): string  
Description: Works only with Cyrilliccharacters..  
Variables: s is the string in 8859-5 character code  
Result: the string in ISO  
Example: response.write (cyrillictoiso (request.form ("theline")))

#### 4.7.16 Isotoutf8 method

Syntax: function isotoutf8 (s: string): string

Description: Works only with Cyrilliccharacters..  
Variables: s is the string in ISO character code  
Result: the string in UTF-8  
Example: `response.write (Isotoutf8 (request.form ("theline")))`

#### 4.7.17 Regexp function

Syntax: `function regexp (needle, haystack: string; ignorecase, addblank: boolean): boolean`  
Description: Performs a subset of regular expressions.  
Variables: needle is the string to be found  
haystack is the string where to search  
if addblank is true, then blanks are added to the beginning and the end of haystack, so that "\b" will not only mean word boundaries, but also beginning and end of text.  
Result: true, if needle is found in haystack  
Example: `response.write (regexpr ("*word", "this is a word", true, false))`

#### 4.7.18 Wordintext function

Syntax: `function wordintext (needle, haystack: string; ignorecase: boolean): boolean`  
Description: Determines if a word is contained in a given text, using a subset of regular expressions.  
Variables: needle is the string to be found  
haystack is the string where to search  
Result: true, if needle is found in haystack  
Example: `response.write (regexpr ("*red", "green, red, blue, yellow", false))`

## 5 Appendix

### 5.1 Error numbers

0	no error
1	Search: a search term is shorter than minsearchwordlength
2	Search: the search result contains more than maxhits
3	Search: maxsearchtime is exceeded
10	Cannot access text database
11	Cannot access topic database
101	Copyindex: Cannot create or rename files
102	Recopyindex: Cannot copy files back
103	Error with function hastonidex ("select aid from atext where isindexed=0 limit 1")

There are more error messages than error numbers. Use the Lasterror property

### 5.2 Script examples

#### 5.2.1 Include file to connect:

```
<%  
Set obj = CreateObject("Topicdb.ttopicdb")  
' you might have to change this  
' Param 1 is the server  
' param 2 is the user  
' param 3 is the password  
b = obj.openserver ("192.168.1.2","root","")  
textdatabasename = "thetext"  
topicdatabasename = "thetopic"  
namedatabasename = "QT_Databases"  
watchfiledirs = "e:\inetpub\wwwroot\docs"  
watchfiletextdirs = "e:\inetpub\wwwroot\temp"  
outdir = "temp"  
outtextdir = "docs"  
%>
```

#### 5.2.2 Searching:

```
searchstring=request("SearchString")  
  
' anzahl anzuzeigender treffer  
MaxDocs=request("MaxDocs")  
if MaxDocs="" then  
    MaxDocs=10  
end if  
' ab welchem treffer anzeigen  
Page=request("Page")  
if Page="" then  
    Page=1  
end if  
  
' Fehlerbehandlung  
if searchstring = "" then  
    Response.Redirect("Error.asp?ErrorString=" & Server.URLEncode("Kein  
Suchstring angegeben!"))
```

```
end if

if len(searchstring) < 2 then
    Response.Redirect("Error.asp?ErrorString=" &
Server.URLEncode("Suchstring muß mind. 2 Zeichen lang sein!"))
end if

s = obj.gettopicsearch (searchstring)
obj.search (s)
NumberOfHits = obj.numberofhits
dim somestring
somestring = obj.hitsasstring
array_var = split (somestring,",")
session ("founddocs") = array_var

Response.Redirect("showresults.asp?searchstring=" & Server.URLEncode(s) &
"&Page=" & Page & "&MaxDocs=" & MaxDocs & "&NumberOfHits=" & NumberOfHits)
%>
</html>
```

### 5.2.3 Displaying results:

```
<%
    searchstring=request("SearchString")

    ' anzahl anzuzeigender treffer
    MaxDocs=CInt(request("MaxDocs"))
    ' ab welchem treffer anzeigen
    Page=CInt(request("Page"))
    ' wieviele Treffer
    NumberOfHits=CInt(request("NumberOfHits"))
    ' Fehlerprüfungen
%>

<html>
<head>
<link rel="stylesheet" href="style.css" type="text/css">
</head>
<body >
<ul>
<h3>Topic Web Suchergebnis</h3>
<p>

<b>Topic Web hat <%=NumberOfHits%> Dokumente gefunden!</b>

<br><p>

<%
    ' is it the first or last page to show for a result?
    IsFirstPage = False
    if Page<=0 then
        Page=1
    end if
    if Page=1 then
        IsFirstPage = True
    end if

    if MaxDocs = 0 then
        MaxDocs = NumberOfHits
    end if
```

```
' how much and what documents should be shown?
FirstDoc = ((Page-1) * MaxDocs) + 1
LastDoc = FirstDoc + MaxDocs - 1
IsLastPage = False

if (NumberOfHits - LastDoc) <= 0 then
    LastDoc = NumberOfHits
    IsLastPage = True
end if

%>

<br>
<br>

<!--#INCLUDE FILE="./Include_HitNavigation.asp"-->
<!--#INCLUDE FILE="./Include_DBServer.asp"-->
<p>

<%' jetzt werden die MaxDocs Treffer angezeigt
' dazu brauchen wir das objekt aber wieder, sonst kriegen wir den inhalt
nicht raus!!
Dim Obj
' objekt erzeugen
' set Obj = CreateObject(???) (tuwas!)
for i = FirstDoc to LastDoc
' hier wird aus der sessionvariable ausgelesen
' obj.gotodocument(wert aus der sessionvariable mit dem index i) (tuwas!)
array_var = session ("founddocs")
j = array_var(i-1)
obj.gotodocument (j)
%>

<div CLASS="HitInd"> Treffer Nummer: <%=i%></div>
<table CLASS="BTABLE" width=90%>
  <tr>
    <td align=right width=90>
      Dateiname
    </td>
    <td align=left>
      <% t = obj.fieldtext (fnamefield)
      ' t = escape (t)
      t1 = outtextdir + right (t, len (t) - len (watchfiledirs))
      t2 = right (t, len (t) - len (watchfiledirs))
      %>
      <A HREF=<%=t1%>><%=t2%></A>
    </td>
  </tr>
  <tr>
    <td align=right>
      Dateinhalt
    </td>
    <td align=left>
      <%=obj.fieldtextwithtags (ftextfield,"<b>","</b>",<%=200,0,1%>)>
      <%' =obj.fieldtext (ftextfield)%>
    </td>
  </tr>
</table>
```

```
<br>
```

```
<%next  
%>
```

```
<!--#INCLUDE FILE="./Include_HitNavigation.asp"-->
```

```
<p>
```

```
<p><strong><font size=-1>Powered by TopicMap</a><br>
```

## 5.2.4 Searching within the last search result:

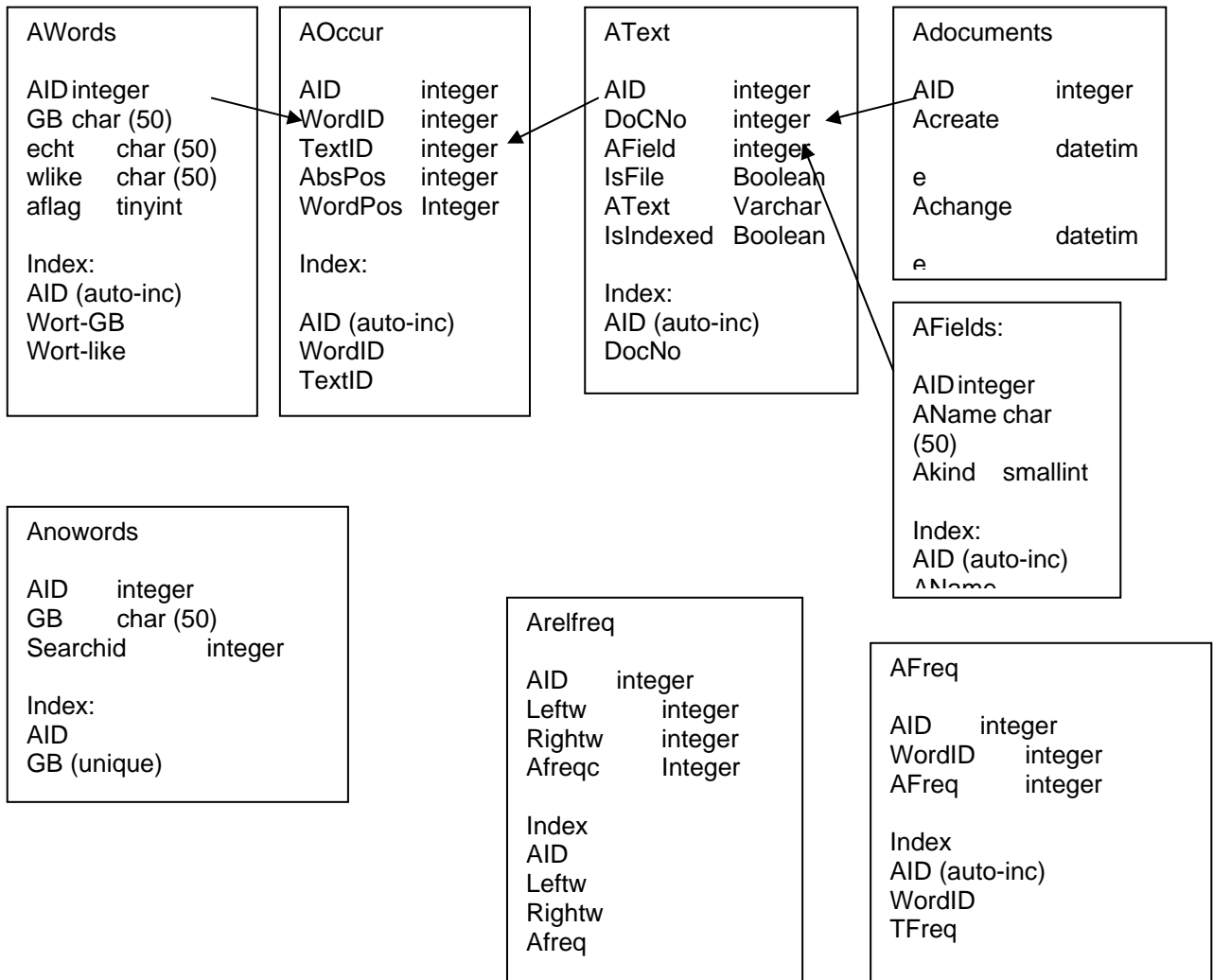
```
` this is the first search  
n = obj.search ("this")  
t = obj.hitsasstring  
obj.setsearchselect (t,true)  
` this is the second search  
n = obj.search ("that")
```

## 5.2.5 Showing a graph:

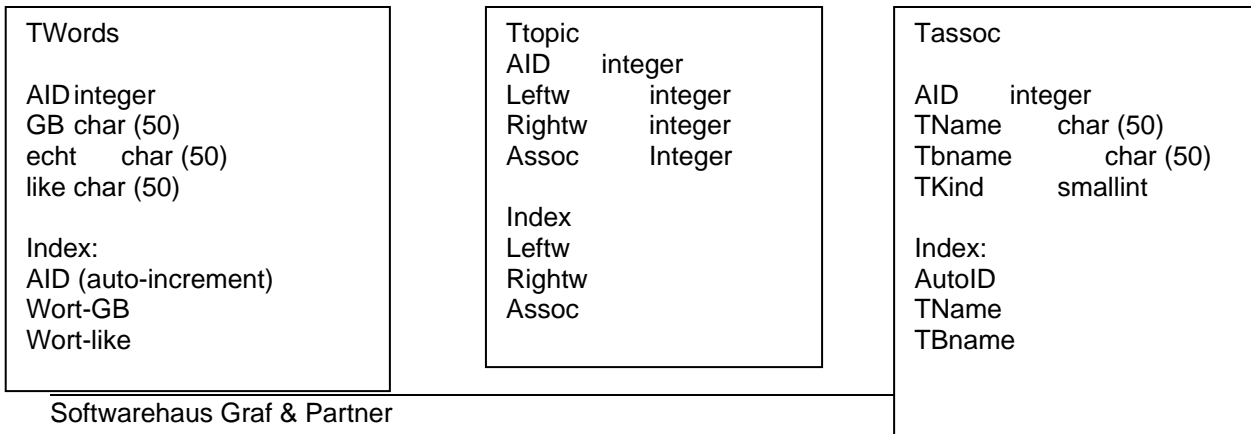
```
<html>  
<head><title>Responding to a form</title></head>  
<body>  
<!--#INCLUDE FILE="./Include_DBServer.asp"-->  
<% s=Request.Form("thestring")  
wi=Request.Form("count")  
if wi="" then wi="20"  
de=Request.Form("Depth")  
if de="" then de="1"  
op=Request.Form("order")  
orfre = false  
orco = false  
if op = "1" then orfre = true  
if op = "2" then orco = true  
' tdir=obj.getinfo ("Watchfiles","Textfiledir") + "\Graph"  
tdir=watchfiletextdirs + "\Graph"  
obj.logging = true  
obj.killfilesbefore tdir + "*.jp*",1  
t = obj.drawgraph (s,wi,de,orfre,orco,800,600,90,tdir)  
i = instr (t,"\Graph")  
t = outdir + right (t, len (t) - i + 1)  
  
obj.logging = false  
response.redirect (t)  
  
%>  
<A HREF=<%=t%>>View Pic</A>  
  
</body>  
</html>
```

### 5.3 Database structure

#### Text-DB



#### Topic-DB



## 5.4 Regular Expressions

Unterstützt nur einen teil, nimmt automatisch wortgrenzen an.

„\*baum“ in „das ist ein apfelbaum“

Einführung

Reguläre Ausdrücke werden weitum verwendet, um Textmuster zu beschreiben, nach welchen dann gesucht wird. Spezielle Metazeichen erlauben das Definieren von Bedingungen, beispielsweise soll ein bestimmter gesuchter String am Anfang oder am Ende einer Zeile vorkommen, oder ein bestimmtes Zeichen soll n mal Vorkommen.

Reguläre Ausdrücke sehen üblicherweise für Anfänger ziemlich kryptisch aus, sind aber im Grunde genommen sehr einfache (nun, üblicherweise einfache ;) ), handliche und enorm mächtige Werkzeuge.

Ich empfehle Dir wärmstens, dass Du mit dem Demo-Projekt in TestRExp.dpr etwas mit den regulären Ausdrücken herumspielst - es wird Dir enorm dabei helfen, die hauptsächlichen Konzepte zu erfassen. Darüberhinaus findest Du viele vorgegebene und kommentierte Beispiele in TestRExp.

Also, starten wir in die Lernkurve!

Einfache Treffer

Jedes einzelne Zeichen findet sich selbst, ausser es sei ein Metazeichen mit einer speziellen Bedeutung (siehe weiter unten).

Eine Sequenz von Zeichen findet genau dieses Sequenz im zu durchsuchenden String (Zielstring). Also findet das Muster (= reguläre Ausdruck) "bluh" genau die Sequenz "bluh" irgendwo im Zielstring. Ganz einfach, nicht wahr?

Damit Du Zeichen, die üblicherweise als Metazeichen oder Escape-Sequenzen dienen, als ganz normale Zeichen ohne jede Bedeutung finden kannst, stelle so einem Zeichen einen "\" voran. Diese Technik nennt man Escaping. Ein Beispiel: das Metazeichen "^" findet den Anfang des Zielstrings, aber "\"^" findet das Zeichen "^" (Circumflex), "\"\" findet also "\" etc.

Beispiele:

foobar            findet den String 'foobar'

\"^FooBarPtr    findet den String '^FooBarPtr'

Escape-Sequenzen

Zeichen können auch angegeben werden mittels einer Escape-Sequenz, in der Syntax ähnlich derer, die in C oder Perl benutzt wird: "\\n" findet eine neue Zeile, "\\t" einen Tabulator etc. Etwas allgemeiner: "\\xnn", wobei nn ein String aus hexadezimalen Ziffern ist, findet das Zeichen, dessen ASCII Code gleich nn ist. Falls Du Unicode-Zeichen (16 Bit breit kodierte Zeichen) angeben möchtest, dann benutze "\\x{nnnn}", wobei 'nnnn' - eine oder mehrere hexadezimale Ziffern sind.

\\xnn            Zeichen mit dem Hex-Code nn (ASCII-Text)

`\x{nnnn}` Zeichen mit dem Hex-Code nnnn (ein Byte für ASCII-Text und zwei Bytes für Unicode-Zeichen)

`\t` ein Tabulator (HT/TAB), gleichbedeutend wie `\x09`

`\n` Zeilenvorschub (NL), gleichbedeutend wie `\x0a`

`\r` Wagenrücklauf (CR), gleichbedeutend wie `\x0d`

`\f` Seitenvorschub (FF), gleichbedeutend wie `\x0c`

`\a` Alarm (bell) (BEL), gleichbedeutend wie `\x07`

`\e` Escape (ESC), gleichbedeutend wie `\x1b`

#### Beispiele:

`foo\x20bar` findet 'foo bar' (beachte den Leerschlag in der Mitte)

`\tfootbar` findet 'foobar', dem unmittelbar ein Tabulator vorangeht

#### Zeichenklassen

Du kannst sogenannte Zeichenklassen definieren, indem Du eine Liste von Zeichen, eingeschlossen in eckige Klammern [], angibst. So eine Zeichenklasse findet genau eines der aufgelisteten Zeichen Zeichen im Zielstring.

Falls das erste aufgelistete Zeichen, das direkt nach dem "[", ein "^" ist, findet die Zeichenklasse jedes Zeichen ausser denjenigen in der Liste.

#### Beispiele:

`foob[aeiou]r` findet die Strings 'foobar', 'foober' etc. aber nicht 'foobbr', 'foobcr' etc.

`foob[^aeiou]r` findet die Strings 'foobbr', 'foobcr' etc. aber nicht 'foobar', 'foober' etc.

Innerhalb der Liste kann das Zeichen "-" benutzt werden, um einen Bereich oder eine Menge von Zeichen zu definieren. So definiert `a-z` alle Zeichen zwischen "a" and "z" inklusive.

Falls das Zeichen "-" selbst ein Mitglied der Zeichenklasse sein soll, dann setze es als erstes oder letztes Zeichen in die Liste oder schütze es mit einem vorangestellten "\" (escaping). Wenn das Zeichen "]" ebenfalls Mitglied der Zeichenklasse sein soll, dann setze es als erstes Zeichen in die Liste oder escape es.

#### Beispiele:

`[-az]` findet 'a', 'z' und '-'

`[az-]` findet 'a', 'z' und '-'

`[a\z]` findet 'a', 'z' und '-'

`[a-z]` findet alle 26 Kleinbuchstaben von 'a' bis 'z'

`[\n-\x0D]` findet eines der Zeichen #10, #11, #12 oder #13.

`[\d-t]` findet irgendeine Ziffer, '-' oder 't'.

`[]-a]` findet irgendein Zeichen von ']'..'a'.

#### Metazeichen

Metazeichen sind Zeichen mit speziellen Bedeutungen. Sie sind die Essenz der regulären Ausdrücke. Es gibt verschiedene Arten von Metazeichen wie unten beschrieben.

#### Metazeichen - Zeilenseparatoren

`^` Beginn einer Zeile  
`$` Ende einer Zeile  
`\A` start of text  
`\Z` end of text  
`.` irgendein beliebiges Zeichen

#### Beispiele:

`^foobar` findet den String 'foobar' nur, wenn es am Zeilenanfang vorkommt  
`foobar$` findet den String 'foobar' nur, wenn es am Zeilenende vorkommt  
`^foobar$` findet den String 'foobar' nur, wenn er der einzige String in der Zeile ist  
`foob.r` findet Strings wie 'foobar', 'foobbr', 'fooblr' etc.

Standardmässig garantiert das Metazeichen "^" nur, dass das Suchmuster sich am Anfang des Zielstrings befinden muss, oder am Ende des Zielstrings mit dem Metazeichen "\$". Kommen im Zielstring Zeilenseparatoren vor, so werden diese von "^" oder "\$" nicht gefunden.

Du kannst allerdings den Zielstring als mehrzeiligen Puffer behandeln, so dass "^" die Stelle unmittelbar nach, und "\$" die Stelle unmittelbar vor irgendeinem Zeilenseparator findet. Du kannst diese Art der Suche einstellen mit dem Modifikator /m.

The \A and \Z are just like '^' and '\$', except that they won't match multiple times when the modifier /m is used, while '^' and '\$' will match at every internal line separator.

Das "." Metazeichen findet standardmässig irgendein beliebiges Zeichen, also auch Zeilenseparatoren. Wenn Du den Modifikator /s

ausschaltest, dann findet '.' keine Zeilenseparatoren mehr.

TRegExpr geht mit Zeilenseparatoren so um, wie es auf [www.unicode.org](http://www.unicode.org) (<http://www.unicode.org/unicode/reports/tr18/>) empfohlen ist:

"^" ist am Anfang des Eingabestrings, und, falls der Modifikator /m gesetzt ist, auch unmittelbar folgend einem Vorkommen von \x0D\x0A oder \x0A or \x0D (falls Du die Unicode-Version von

TRegExpr benutzt, dann auch nach \x2028 oder \x2029 oder \x0B oder \x0C oder \x85). Beachte, dass es keine leere Zeile gibt in den Sequence \x0D\x0A. Diese beiden Zeichen werden atomar behandelt.

"\$" ist am Anfang des Eingabestrings, und, falls der Modifikator /m gesetzt ist, auch unmittelbar vor einem Vorkommen von \x0D\x0A oder \x0A or \x0D (falls Du die Unicode-Version von TRegExpr benutzt, dann auch vor \x2028 oder \x2029 oder \x0B oder \x0C oder \x85). Beachte, dass es keine leere Zeile gibt in den Sequence \x0D\x0A. Diese beiden Zeichen werden atomar behandelt.

"." findet ein beliebiges Zeichen. Wenn Du aber den Modifikator /s ausstellst, dann findet "." keine Zeilenseparatoren \x0D\x0A und \x0A und \x0D mehr (falls Du die Unicode-Version von TRegExpr benutzt, dann auch \x2028 und \x2029 und \x0B und \x0C and \x85).

Beachte, dass "^.\*\$" (was auch eine leere Zeile findet können sollte) dennoch nicht den leeren String innerhalb der Sequence \x0D\x0A findet, aber es findet den Leerstring innerhalb der Sequenz \x0A\x0D.

Die Behandlung des Zielstrings als mehrzeiliger String kann leicht Deinen Bedürfnissen angepasst werden dank der TRegExpr-Eigenschaften LineSeparator und LinePairedSeparator. Du kannst nur den UNIX-Stil Zeilenseparator \n benutzen oder nur DOS-Stil Separatoren \r\n oder beide gleichzeitig (wie schon oben beschrieben und wie es als Standard gesetzt ist). Du kannst auch Deine eigenen Zeilenseparatoren definieren!

Metazeichen - vordefinierte Klassen

\w ein alphanumerisches Zeichen inklusive "\_"

\W kein alphanumerisches Zeichen, auch kein "\_"

\d ein numerisches Zeichen

\D kein numerisches Zeichen

\s irgendein wörtertrennendes Zeichen (entspricht [ \t\n\r\f])

\S kein wörtertrennendes Zeichen

Du kannst \w, \d und \s innerhalb Deiner selbstdefinierten Zeichenklassen benutzen.

Beispiele:

foob\dr findet Strings wie 'foobl', 'foob6r' etc., aber not 'foobar', 'foobbr' etc.

foob[\w\s]r findet Strings wie 'foobar', 'foob r', 'foobbr' etc., aber nicht 'foobl', 'foob=r' etc.

TRegExpr benutzt die Eigenschaften SpaceChars und WordChars, um die Zeichenklassen \w, \W, \s, \S zu definieren. Somit kannst Du sie auch leicht umdefinieren.

### Metazeichen - Wortgrenzen

- \b findet eine Wortgrenze
- \B findet alles ausser einer Wortgrenze

Eine Wortgrenze (\b) ist der Ort zwischen zwei Zeichen, welcher ein \w auf der einen und ein \W auf der anderen Seite hat bzw. umgekehrt. \b bezeichnet alle Zeichen des \w bis vor das erste Zeichen des \W bzw. umgekehrt.

### Metazeichen - Iteratoren

Jeder Teil eines regulären Ausdrucks kann gefolgt werden von einer anderen Art von Metazeichen - den Iteratoren. Dank dieser Metazeichen kannst Du die Häufigkeit des Auftretens des Suchmusters im Zielstring definieren. Dies gilt jeweils für das vor diesem Metazeichen stehende Zeichen, das Metazeichen oder den Teilausdruck.

- \* kein- oder mehrmaliges Vorkommen ("gierig"), gleichbedeutend wie {0,}
- + ein- oder mehrmaliges Vorkommen ("gierig"), gleichbedeutend wie {1,}
- ? kein- oder einmaliges Vorkommen ("gierig"), gleichbedeutend wie {0,1}
- {n} genau n-maliges Vorkommen ("gierig")
- {n,} mindestens n-maliges Vorkommen ("gierig")
- {n,m} mindestens n-, aber höchstens m-maliges Vorkommen ("gierig")
- \*? kein- oder mehrmaliges Vorkommen ("genügsam"), gleichbedeutend wie {0,}?
- +? ein oder mehrmaliges Vorkommen ("genügsam"), gleichbedeutend wie {1,}?
- ?? kein- oder einmaliges Vorkommen ("genügsam"), gleichbedeutend wie {0,1}?
- {n}? genau n-maliges Vorkommen ("genügsam")
- {n,}? Mindestens n-maliges Vorkommen ("genügsam")
- {n,m}? mindestens n-, aber höchstens m-maliges Vorkommen ("genügsam")

Also, die Ziffern in den geschweiften Klammern in der Form {n,m} geben an, wieviele Male das Suchmuster im Zielstring gefunden muss, um einen Treffer zu ergeben. Die Angabe {n} ist gleichbedeutend wie {n,n} und findet genau n Vorkommen. Die Form {n,} findet n oder mehr Vorkommen. Es gibt keine Limiten für die Zahlen n und m. Aber je grösser sie sind, desto mehr Speicher und Zeit wird benötigt, um den regulären Ausdruck auszuwerten.

Falls eine geschweifte Klammer in einem anderen als dem eben vorgestellten Kontext vorkommt, wird es wie ein normales Zeichen behandelt.

Beispiele:

foob.\*r findet Strings wie 'foobar', 'foobalkjdfllkj9r' und 'foobr'  
foob.+r findet Strings wie 'foobar', 'foobalkjdfllkj9r', aber nicht 'foobr'  
foob.?r findet Strings wie 'foobar', 'foobbr' und 'foobr', aber nicht 'foobalkj9r'  
fooba{2}r findet den String 'foobaar'  
fooba{2,}r findet Strings wie 'foobaar', 'foobaaar', 'foobaaaar' etc.  
fooba{2,3}r findet Strings wie 'foobaar', or 'foobaaar', aber nicht 'foobaaaar'

Eine kleine Erklärung zum Thema "gierig" oder "genügsam". "Gierig" nimmt soviel wie möglich, wohingegen "genügsam" bereits mit dem ersten Erfüllen des Suchmusters zufrieden ist. Beispiel: 'b+' und 'b\*' angewandt auf den Zielstring 'abbbbc' findet 'bbbb', 'b+?' findet 'b', 'b\*?' findet den leeren String, 'b{2,3}?' findet 'bb', 'b{2,3}' findet 'bbb'.

Du kannst alle Iteratoren auf den genügsamen Modus umschalten mit dem Modifizier /g.

#### Metazeichen - Alternativen

Du kannst eine Serie von Alternativen für eine Suchmuster angeben, indem Du diese mit einem '|' trennst. Auf diese Art findet das Suchmuster fee|fie|foe eines von "fee", "fie", oder "foe" im Zielstring - dies würde auch mit f(e|i|o)e erreicht.

Die erste Alternative beinhaltet alles vom letzten Muster-Limiter "(", "[" oder natürlich der Anfang des Suchmusters) bis zum ersten "|". Die letzte Alternative beinhaltet alles vom letzten "|" bis zum nächsten Muster-Limiter. Aus diesem Grunde ist es allgemein eine gute Gewohnheit, die Alternativen in Klammern anzugeben, um möglichen Missverständnissen darüber vorzubeugen, wo die Alternativen beginnen und enden.

Alternativen werden von links nach rechts gepüft, so dass der Treffer im Zielstring zusammengesetzt ist aus den jeweils zuerst passenden Alternativen. Das bedeutet, dass Alternativen nicht notwendigerweise "gierig" sind. Ein Beispiel: Wenn man mit "(foo|foot)" im Zielstring "barefoot" sucht, so passt bereits die erste Variante. Diese Tatsache mag nicht besonders wichtig erscheinen, aber es ist natürlich wichtig, wenn der gefundene Text weiterverwendet wird. Im Beispiel zuvor würde der Benutzer nicht "foot" erhalten, wie er eventuell beabsichtigt hatte, sondern nur "foo".

Erinnere Dich auch daran, dass das "|" innerhalb von eckigen Klammern wie ein normales Zeichen behandelt wird, so dass [fee|fie|foe] dasselbe bedeutet wie [feio|].

#### Beispiele:

foo(bar|foo) findet die Strings 'foobar' oder 'foofoo'.

#### Metazeichen - Teilausdrücke

Das Klammerkonstrukt (...) wird auch dazu benutzt, reguläre Teilausdrücke zu definieren (nach dem Parsen findest Du Positionen, Längen und effektive Inhalte der regulären Teilausdrücke in den TRegExpr-Eigenschaften MatchPos, MatchLen und Match und kannst sie ersetzen mit den Template-Strings in TRegExpr.Substitute).

Teilausdrücke werden nummeriert von links nach rechts, jeweils in der Reihenfolge ihrer öffnenden Klammer. Der erste Teilausdruck hat die Nummer 1, der gesamte reguläre Ausdruck hat die Nummer 0 (der gesamte Ausdruck kann ersetzt werden in TRegExpr.Substitute als '\$0' oder '\$&').

Beispiele:

(foobar){8,10} findet Strings, die 8, 9 oder 10 Vorkommen von 'foobar' beinhalten

foob([0-9]|a+)r findet 'foob0r', 'foobl'r', 'foobar', 'foobaar', 'foobaar' etc.

Metazeichen - Rückwärtsreferenzen

Die Metacharacters \1 bis \9 werden in Suchmustern interpretiert als Rückwärtsreferenzen. \<n> findet einen zuvor bereits gefundenen Teilausdruck #<n>.

Beispiele:

(.)\1+ findet 'aaaa' und 'cc'.

(.+)\1+ findet auch 'abab' und '123123'

(["']?)(\d+)\1 findet "13" (innerhalb "), oder '4' (innerhalb ') oder auch 77, etc.

Modifikatoren

Modifikatoren sind dazu da, das Verhalten von TRegExpr zu verändern.

Es gibt viele Wege, die weiter unten beschriebenen Modifikatoren zu nutzen. Jeder der Modifikatoren kann eingebettet werden im Suchmuster des regulären Ausdrucks mittels des Konstruktes (?...).

Du kannst allerdings auch die meisten Modifikatoren beeinflussen, indem Du den entsprechenden TRegExpr-Eigenschaften die passenden Werte zuweist (Beispiel: Zuweisung an ModifierX oder ModifierStr für alle Modifikatoren zugleich).

Die Standardwerte für neue Instanzen von TRegExpr-Objekte sind definiert in globalen Variablen. Beispielsweise definiert die globale Variable RegExprModifierX das Verhalten des Modifikators X und damit die Einstellung der TRegExpr-Eigenschaft ModifierX bei neu instantiierten TRegExpr-Objekten.

i

Führe die Suche Schreibweisen-unabhängig durch (allerdings abhängig von den Einstellungen in Deinem System, Lokale Einstellungen), (beachte auch die InvertCase)

m

Behandle den Zielstring als mehrzeiligen String. Das bedeutet, ändere die Bedeutungen von "^" und "\$": Statt nur den Anfang oder das Ende des Zielstrings zu finden, wird jeder Zeilenseparator innerhalb eines Strings erkannt (beachte auch die Zeilenseparatoren)

s

Behandle den Zielstring als einzelne Zeile. Das bedeutet, dass "." jedes beliebige Zeichen findet, sogar Zeilenseparatoren (beachte auch Zeilenseparatoren), die es normalerweise nicht findet.

g

Modifikator für den "Genügsam"-Modus. Durch das Ausstellen werden alle folgenden Operatoren in den "Genügsam"-Modus. Standardmässig sind alle Operatoren "gierig". Wenn also der Modifikator /g aus ist, dann arbeitet '+' wie '+?', '\*' als '\*?' etc.

x

Erweitert die Lesbarkeit des Suchmusters durch Whitespace und Kommentare (beachte die Erklärung unten).

r

Modifikator. Falls er gesetzt ist, beinhaltet die Zeichenklasse à-ÿ zusätzliche russische Buchstaben ' ', À-ß beinhaltet zusätzlich ' ', und à-ß beinhaltet alle russischen Symbole.

Sorry für fremdsprachliche Benutzer, er ist gesetzt standardmässig. Falls Du ihn ausgeschaltet haben willst standardässig, dann setze die globale Variable RegExprModifierR auf false.

Der Modifikator /x selbst braucht etwas mehr Erklärung. Er sagt TRegExpr, dass er allen Whitespace ignorieren soll, der nicht escaped oder innerhalb einer Zeichenklasse ist. Du kannst ihn benutzen, um den regulären Ausdruck in kleinere, besser lesbare Teile zu zerlegen. Das Zeichen # wird nun ebenfalls als Metazeichen behandelt und leitet einen Kommentar bis zum Zeilenende ein. Beispiel:

```
(  
  (abc) # Kommentar 1  
    |   # Du kannst Leerschläge zur Formatierung benutzen - TRegExpr  
        ignoriert sie  
  (efg) # Kommentar 2  
)
```

Dies bedeutet auch, wenn Du echten Whitespace oder das # im Suchmuster haben möchtest (ausserhalb einer Zeichenklasse, wo sie unbehelligt von /x sind), dann muss der entweder escaped oder mit der hexadezimalen Schreibweise angegeben werden. Beides zusammen sorgt dafür, dass reguläre Ausdrücke besser lesbar werden.

Perl Erweiterungen

(?imsxr-imsxr)

Dies kann benutzt werden in Regulären Ausdrücken, um Modifikatoren innerhalb eines Ausdruckes im Flug zu ändern. Wenn dieses Konstrukt innerhalb eines Teilausdruckes erscheint, betrifft er auch nur diesen.

Beispiele:

(?i)Saint-Petersburg Petersburg'	findet 'Saint-petersburg' und 'Saint-
(?i)Saint-(?-i)Petersburg petersburg'	findet 'Saint-Petersburg', aber nicht 'Saint-
(?i)(Saint-)?Petersburg petersburg'	findet 'Saint-petersburg' und 'saint-
((?i)Saint-)?Petersburg petersburg'	findet 'saint-Petersburg', aber nicht 'saint-

(?#text)

Ein Kommentar, der Text wird ignoriert. Beachte, dass TRegExpr den Kommentar abschliesst, sobald er eine ")" sieht. Es gibt also keine Möglichkeit, das Zeichen ")" im Kommentar zu haben.

## 5.5 Troubleshooting

### 5.5.1 Program seems to hang when reindexing (usually at “fieldsnotindexed”)

If your database was created before version 1.4.2 of Oct. 2006, try to execute the following statement in your text database:

```
alter table atext add index atext(atext(1))
```

### 5.5.2 Program gives error message “Server has gone away” during reading files

This happens when the server keeps too much information in memory before writing to the database.

Try setting Indexchunksize and maxbytesperround to a lower level, e.g. 500 and 10.000.000.

```
Obj.indexchunksize = 500  
Obj.maxbytesperround = 10000000
```

### 5.5.3 Program gives error message “Server has gone away” during indexing files

This happens when the server keeps too much information in memory before writing to the database.

If you are using readfiles, the remedies mentioned above should be enough.

If not, you can set maxbytesperround (see above) or set the value maximum for Indexall or Indextempall to a lower value.

```
Obj.maxbytesperround = 10000000  
Obj.indexall true,100000
```