

FACT-Finder

Integration

FACT-Finder®
Europe's leading conversion engine

FACT-Finder Development

E-mail: support@fact-finder.com

Phone: +49 (0)7231/12597-701

Version: 6.7.0

Latest update: 03.01.2012

Table of contents

Table of contents	2
1. Introduction	5
2. Product data export.....	6
2.1. Creation	6
2.2. Updating	6
2.2.1. Pull	6
2.2.2. Push	6
3. Data integration in the FACT-Finder search	7
4. Integration of the search into the shopping system	7
4.1. Integration variants	7
4.1.1. HTML	7
4.1.2. XML	8
4.1.3. Web service / SOAP.....	9
4.1.4. JSON / JSONP	10
4.2. Comparison of variants.....	10
5. Technical documentation for integrating the Search function.....	11
5.1. General hints & notes.....	11
5.1.1. Sample code and plugins.....	11
5.1.2. Search syntax	11
5.1.3. User management.....	12
5.1.4. Search parameters.....	16
5.1.5. Troubleshooting	23
5.1.6. Caching search results	23
5.1.7. Output compression	24
5.1.8. Suppression of blank search queries	24

5.2.	HTML integration	24
5.3.	XML integration	25
5.3.1.	Result structure	25
5.4.	Web service integration	31
5.4.1.	Sending the search query	31
5.4.2.	Processing results	32
5.5.	JSON integration	32
5.5.1.	Result structure	32
5.6.	JSONP integration.....	38
6.	Technical documentation for other interfaces.....	39
6.1.	Updating databases	39
6.1.1.	HTTP interface	39
6.1.2.	Web service interface	40
6.2.	Changing, deleting and adding records	40
6.2.1.	HTTP interface	41
6.2.2.	Web service interface	41
6.3.	Reloading the database	41
6.3.1.	HTTP interface.....	41
6.3.2.	Web service interface	42
6.4.	Checking that the data is up to date	42
6.4.1.	Text interface	42
6.4.2.	XML Interface	43
6.4.3.	JSON interface.....	43
6.5.	Channel management	43
6.5.1.	HTTP interface.....	44
6.5.2.	Web service interface	44

6.6.	User management	44
6.6.1.	HTTP interface.....	44
6.6.2.	Web service interface	44
6.7.	System monitoring	44
6.7.1.	HTTP interface.....	45
7.	Optimising the search results & tips	45
7.1.	Community thesaurus	45
7.2.	Maintaining a good overview	46
8.	Best Practice	46
8.1.	General.....	46
8.1.1.	Usability	46
8.1.2.	Design.....	47
8.1.3.	Function	47
8.1.4.	Examples.....	49
8.2.	After Search Navigation.....	51
8.2.1.	Usability	51
8.2.2.	Design.....	51
8.2.3.	Function	52
8.2.4.	Examples.....	52
	Any questions?.....	53

1. Introduction

This document is intended to provide you with an overview of how you can integrate FACT-Finder with your web shop system.

Overview of the steps

The search integration can be roughly divided up into the following items:

Generating an export of product data

Integrating the product data into the FACT-Finder search engine

Integrating the search engine and other FACT-Finder modules into the shopping system

Optimising search quality

Sample URLs

In the integration documentation you can find several URLs that you will have to customise using the relevant parameters for your search environment in order to achieve the desired results. Assuming your Search instance has the application name *FACT-Finder* and is running on a server with the FQDN (fully qualified domain name) *search.fact-finder.com*, the resulting search URL would be *search.fact-finder.com/FACT-Finder/Search.ff*.

2. Product data export

Product data export forms the basis for the search. This export will later be imported into FACT-Finder and can then be searched.

2.1. Creation

To achieve the best results, you should make all search-relevant information available to FACT-finder. In addition, meta-information – such as sales rank or inventory – should also be exported. Using this data, the search result can later be controlled based on rules, so that, for example, the best-selling articles are presented first.

A precise description of the structure, the required data and tips can be found in the product data export description (additional document available from Omikron).

2.2. Updating

Since product data is constantly changing, it is very important that the search is constantly provided with updates of exported product data. Otherwise products may not be able to be found because they are not available in the (outdated) export, or products may be found that are no longer available. To avoid this, FACT-Finder offers two automatic update options.

2.2.1. Pull

In this variant your export data is stored in a location where it can be reached via a URL. The data will be automatically downloaded and imported by the FACT-Finder search server at defined intervals (e.g. daily at 5.00am). If an update is necessary outside of these intervals, it can be initiated via the Management Interface or by a URL/Web service call. You can arrange to be notified by email of any input errors that occur. This method requires little effort on your part.

If, for reasons of data protection, it is not possible for you to make the product data available via a publicly accessible URL, there is a slightly modified form of the Pull procedure. In this slightly modified form, FACT-Finder does not store the URL, but you can instruct FACT-Finder to download the product data from a specified URL by means of a web service call. To make this possible you have the option of generating a one-time URL, instructing FACT-Finder to draw the data from the URL and then removing the data afterwards. The disadvantage of this approach is that responsibility for time control of the updates lies within the shopping system.

2.2.2. Push

In the second option, you must store the data at a defined location (e.g. FTP) and initiate the FACT-Finder Import by calling a URL/Web service. This approach is particularly suitable if, for

example, the FACT-Finder search is being carried out on its own server or if you want the result from the import interface to be automatically evaluated.

3. Data integration in the FACT-Finder search

In most cases this step is carried out by Omikron. Here the previously created product export is evaluated and a search environment is created based on this evaluation. At the end of this step a fully functional search application including management interface will be available to you.

If the search is run as an ASP service (Application Service Providing), an Omikron employee will set up the application on a server and make the access data available to you. This application is normally accessible via internet..

4. Integration of the search into the shopping system

The purpose of this step is to make the results returned by the FACT-Finder search application available in your shopping system. This includes the following steps:

1. Transmitting the search query
2. Receiving the search results
3. Processing the search results
4. Displaying the search results

The practical implementation varies depending on the integration variant desired. Details of the technical implementation can be found further in this document.

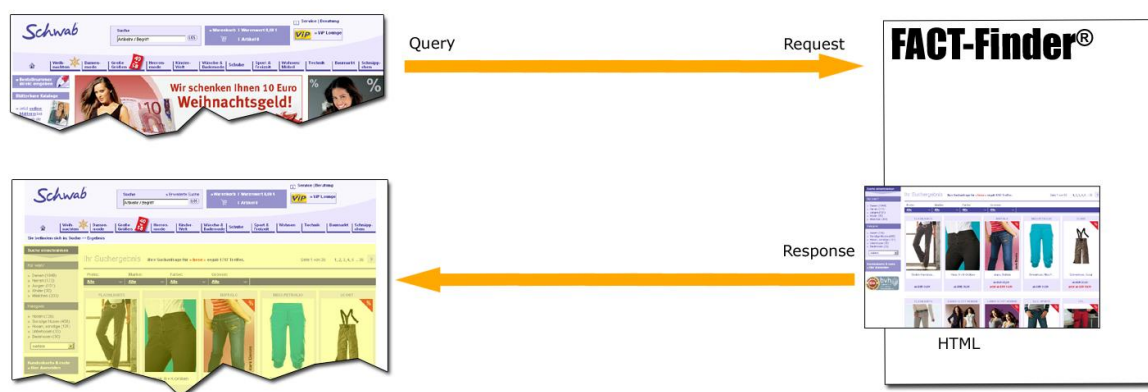
4.1. Integration variants

4.1.1. HTML

In this mode the FACT-Finder Search server returns a ready-made HTML as the search result. This type of result can be integrated into the web shop system using a FRAME or IFRAME, for example.

With this connection option, the design is determined by the company issuing the order and implemented by Omikron on the search server. The design specifications are provided as HTML templates.

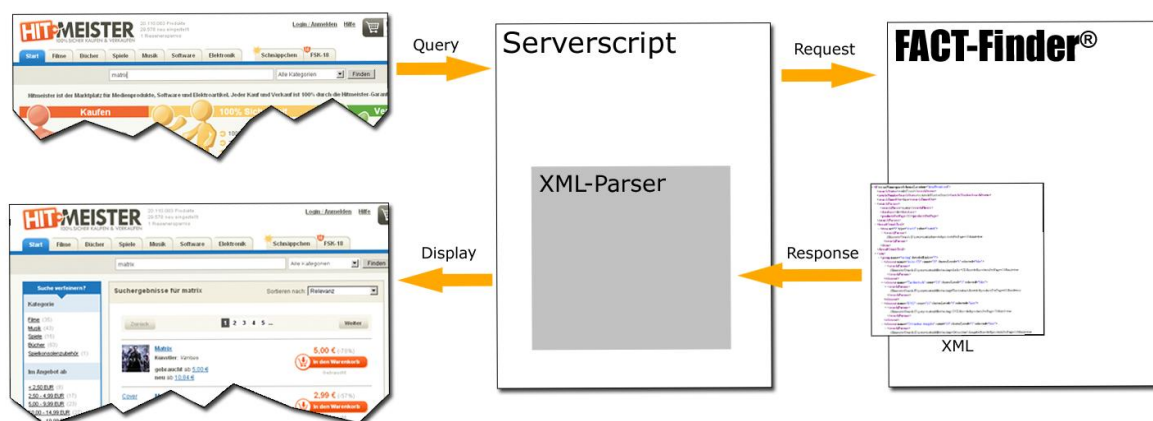
The illustration below indicates the workflow for a search query. You must ensure that when the search form is sent the FRAME/IFRAME is included.



1. Search query is sent to FACT-Finder (request): the search box in the shop must be configured in such a way that the search result is shown in the FRAME or IFRAME.
2. Search results are received (response): done via the user's browser, no integration steps required.
3. Search results are processed: not required for this method.
4. Search results are generated: carried out by the FACT-Finder search server.

4.1.2. XML

In this variant, search parameters are sent from the shop to the FACT-Finder Search server in the form of a URL. FACT-Finder responds to this query in the XML format specified, which contains all the information required to build a search results page.



The illustration above shows the workflow of a search query. It should be noted that between the output pages and the FACT-Finder search server, there is a server script that must be implemented by you.

1. Search query is sent to FACT-Finder: the search parameters entered in the search box by the user (query) must be sent to the FACT-Finder search server by means of an HTTP request (request).
2. Search results are received (response): the FACT-Finder search server delivers the search results via an HTTP response.
3. Search results are processed: the returned XML must be interpreted (parsed) by the shopping system.
4. Search results page is generated (display): the structure of the HTML display of the search results page must be based on the search results information that is delivered.

The most important difference between this method and the HTML result is that the design of the search results page is created by the shop. Thus the design is centrally managed and does not have to be monitored and corrected at multiple locations.

4.1.3. Web service / SOAP

With this third connection option, the search server makes a web service interface available. Web services are standardised communication interfaces for distributed applications. The sequence here is exactly the same as with the XML variant. The difference lies in the standardized interface, so that the query is sent by means of objects rather than a URL.

1. Search query is sent to FACT-Finder: the search parameters that were entered in the search box by the user will be passed on to the web service request.

2. Search results are received (response): carried out by the program code generated via the WSDL.
3. Search results are processed: carried out by the program BE: programme code generated via the WSDL.
4. Search results are generated: the search results information is available to the programmer in the form of easily accessible objects, and the HTML page is structured around this information.

4.1.4. JSON / JSONP

The JSON or JSONP interface is another option for integrating FACT-Finder Search into your shop. In this method a URL is also invoked to pass the search parameters. As a response you receive a JSON object, with which you can display the search results pages. The sequence is the same as in the web service variant described above.

4.2. Comparison of variants

Each of the methods has advantages and disadvantages. Which of the variants is best for you therefore depends on many factors. To make this decision easier for you, we compare their advantages and disadvantages below.

	HTML	XML	Web service	JSON
Effort for shop operator	Low	Medium	Medium	Medium
Programming knowledge required	Low	High	High	Medium
Flexible changes	No	Yes	Yes	Yes
Setup costs	High	Low	Low	Low

When using the HTML integration variant, FACT-Finder returns a ready-prepared HTML page that simply has to be integrated by the shop operator. Therefore the effort required here is low.

With the other integration variants, code must be created or used to process the search results, and thus greater programming knowledge is required.

An advantage of the XML/web service and JSON integration methods is that changes can be made more flexibly as the display code is only stored in one location. With HTML integration, the page

design comes from the FACT-Finder Search server. This means that, if the search is run as an ASP service, the changes must be made by Omikron.

The setup costs charged by Omikron are lower for XML, JSON or the web service integration than for HTML integration, as the former variants do not necessitate the extra effort required to incorporate page templates.

5. Technical documentation for integrating the Search function

Depending on which variant you select, your attention will be drawn to different issues when integrating the FACT-Finder Search server into your web shop system. These issues are discussed in this section.

5.1. General hints & notes

5.1.1. Sample code and plugins

To save some of your development work, we have written various pieces of sample code that covers the various interfaces and programming languages. You can use these as the basis for your integration project. We are happy to make these available to you. Simply tell the project manager what programming language you intend to use and how you want to integrate the server.

We also have a number of ready-made plugins that make it much easier to integrate certain web shop systems. Just tell us what shop system you use so that we can check whether we already have a suitable plugin.

5.1.2. Search syntax

The Search server supports a specific syntax that allows you to get more specialised search results.

Exact search

If you want to search for the exact search term as it is entered (e.g. an article number), a plus sign can be placed before the term. If you want to search for a phrase, it must be placed in quotation marks.

```
+exact  
"exact phrase"
```

Excluding terms

If a term is preceded by a minus sign, results that contain it will be excluded from the search result.

```
SearchTerm -removedWord
```

Wildcards

Question marks (?) and asterisk characters (*) are supported. The question mark stands for a single character, the star for an arbitrary number of characters. When the search term contains a wildcard the Search server's error tolerance algorithms no longer apply.

```
Search?erm  
Search*
```

Combining

It is possible to combine two search terms with a Boolean OR operator. The symbol for this operator is two bars: ||. This is often used in combination with exact searches.

```
Trousers || Jacket
```

5.1.3. User management

The User Management functions allow you to assign precise authorisations to individual users. For example, individual users can only manage specific channels or check and maintain the associated configurations. This system of rights is based on roles that provide access to specific modules. These roles can be allocated to the user directly or indirectly via groups. Groups allow complex roles to be allocated in bulk, without having to assign them individually for each user.

If you use the web service connection, you must supply authentication data for each call. With other methods, the authorisation can be deactivated. However this is only recommended if FACT-Finder is being run on a server that is not accessible via internet. .

Roles

Users and groups can be assigned specific roles using the Management Interface. Any user who is logged in can query the Search, Suggest and Tag Cloud modules. No specific role is required for this functionality.

AfterSearchNavigationManager

Members of this role are allowed to make changes to the ASN sub-menu in the configuration editor.

BackupManager

Users with this role can restore backup copies of the configuration.

CacheManager

Users with this role can empty the search cache.

CampaignManager

This role allows the user to create and manage campaigns.

ConfiguratorManager

Members of this role can change default settings in the configuration editor. Additional roles are necessary in order for all settings to be viewed and changed.

DeploymentManager

This role allows the user to transfer files from the staging environment to the production system, assuming you have set up an additional FACT-Finder staging system.

ImportManager

Users with this role can update product data.

InstallManager

This role allows the use of the Setup Wizard.

LanguageManager

Users with this role can adjust language settings in the configuration editor.

LogfileAnalyzerManager

This role allows the user to carry out log file analyses.

MayChangePassword

This role allows users to change their own passwords.

MessagesManager

Users with this role receive information about the search application in the notification menu.

NotificationManager

This allows the user to manage email notification settings.

PreprocessorManager

Users with this role have access to the "preprocessor" module.

RecommendationEngineManager

If users have this role they can access all Recommendation Engine settings.

SearchInterfaceUser

This role allows users to work with the search engine user interface.

SearchManager

Members of this group are allowed to change search settings in the configuration editor.

ShowHelpSection

This role provides access to the "Help" section.

SuggestManager

This role allows the user to change Suggest settings and initiate a Suggest import.

ThesaurusManager

Users with this role have access to the "Thesaurus" module.

UserManager

Users with this role may manage user accounts and groups.

WhatshotManager

Users with this role can view the What's Hot analysis.

WordValuesManager

Users with this role have access to the "Stop Words" module.

Password encryption

FACT-Finder offers two options for sending passwords: a simple method and an advanced method. The advanced method is the standard setting.

The reason for the two options is that FACT-Finder interfaces do not require a secure connection. Therefore, to ensure the security of passwords as they are sent, a two-stage hash process is

applied by the advanced method. The fundamental prerequisite for this is that the same secret key is used by both the FACT-Finder side and the client.

UTF-8 must be used as the encoding when generating the hashes. MD5 is used as the standard hash algorithm, but another algorithm can also be used if desired.

The password hash is formed as follows:

Plain text password: `userpw`

From the plain text password, a hash is first created using a defined algorithm (different algorithms are available. MD5 is used by default.). In the example `userpw` produces the hash `d8118f1bb6bd9998031053176a2c4bee`. This is the string that should be sent transferred as the password when using the simple method. As this hash value, if somehow obtained, works as well as the password for authentication purposes, requests with this value should only be sent over secure connections (HTTPS).

If you prefer to use the advanced hash method, you also need the secret key and a current timestamp (expressed in milliseconds since midnight 1 January 1970 GMT). You can obtain the timestamp in Java, for example, by using the method `System.currentTimeMillis()`; This method returns a result that is already relative to the UTC time zone.

In the example we assume the following values:

Prefix: `FACT-FINDER`

Postfix: `FACT-FINDER`

Timestamp: `1270732953523`

The advanced password variant is sent as the result of the simple encryption, that is the hash of the clear text password. In addition, the prefix and the timestamp are prefixed to this password and the postfix is also added. A new hash is then generated from this string, and that is what is then sent.

Using the example above, the following string is generated:

`FACT-FINDER1270732953523d8118f1bb6bd9998031053176a2c4beeFACT-FINDER`

The result is: `167539c3e7aba8388eee252f429a4a1a`. This is sent to FACT-Finder as the password. As the timestamp changes each time the advanced hashing process is invoked, the resulting hash also changes, which leads to increased security. By default each resulting hash is valid for 60 seconds; if you would like another lifespan, please let us know. Using this approach it is not critical if the sent hash value is somehow obtained. Once the set time-out has expired, attempts to authenticate using the stolen password hash will fail. If the clocks on the web shop server and

the FACT-Finder Search server are running synchronously, the time-out value can be reduced further to increase security.

Login

The login type used depends on the interface. For web service queries an `AuthenticationToken` object must be sent, in which the values are set accordingly. With the other variants, the login data, as well as the search parameters, are passed via the URL. The parameter names for this are `username`, `password` and `timestamp`.

If we take up the previous example again and the password belongs to the user `user`, the following URL results:

```
../Search.ff?timestamp=1270732953523&username=user&password=
167539c3e7aba8388eee252f429a4a1a&...
```

You can also assign this value on the Management Interface pages, so that a "Single Sign On" system can be created.

5.1.4. Search parameters

Search parameters are used to tell FACT-Finder what product data to return. In general there are two different ways of sending these parameters. The HTML, XML and JSON interfaces each respond to a query URL with the appropriate parameters. In the web service interface, program objects and associated attributes are passed through the interface.

Whenever data is passed to FACT-Finder via a URL, it is important that the encoding is set correctly. If this is not the case, special symbols and accented characters will be processed incorrectly. The default URL encoding scheme for FACT-Finder is UTF-8. The available search parameters are listed below.

5.1.4.1. Basic parameters

This list corresponds to the `Params` class when using the web service.

	Description	URL parameter	Web service
Search term	The search term indicates the term or phrase that is to be used to search in the FACT-Finder database. This and the channel parameter are the only mandatory parameters.	The parameter name is <code>query</code> .	Set using the <code>Params</code> object and the <code>setQuery</code> method.

Channel	If your search environment consists of multiple channels, you can define which channel to search here. If no channel is specified, the first channel in the channel list is used.	<code>channel=NAME</code>	<code>Params.setChannel</code>
Page	If a search result contains many results these will be divided into pages. This limits the amount of data that has to be sent in one go. You can indicate which page should be returned. Page numbering starts at 1.	<code>page=NUMBER</code>	<code>Params.setPage</code>
Results per page	In the FACT-Finder Management Interface you can define how many results will be returned on a page by default. If you prefer another number, you can set it with this parameter.	<code>productsPerPage=NUMBER</code>	<code>Params.setResultPerPage</code> You must always set this value; otherwise you will get back one article per page by default.
Sorting	By default the result that is returned has been sorted for relevance. However, you can specify a different sort order here. It is also possible to sort by more than one criterion. Each sort criterion is applied sequentially.	The parameter is made up of the prefix <code>sort</code> and the name of the field by which the data should be sorted (<code>sortFIELDNAME</code>). The sort direction is indicated by the value <code>asc</code> for ascending order and <code>desc</code> for descending order.	Each sort criterion is described using a <code>SortItem</code> object. A list of these objects can be set in <code>Params.</code>
Filter	When a filter parameter is sent to FACT-Finder, the Search engine only returns results that correspond to the filter indicated. The filter value will not register a hit if only a	As with the sort criteria, the name of the parameter is composed of a prefix and a fieldname. Here, however, the prefix is	Each <code>Filter</code> object represents one filter for each field. This object can contain several <code>FilterValue</code> objects. As well as a

	<p>partial match is detected.</p> <p>It is also possible to combine field filters using AND or OR.</p> <p>Moreover you can send a filter that excludes the given criteria. For example, return all products that are NOT red.</p>	<p><code>filter.</code></p> <p>The filter value is provided as the value of the parameter. If you want to apply multiple filters, you need to insert a separator string between each of the filter values. To combine the filters using a logical AND operator the separator is normally 3 underscores (<code>___</code>). If you want to use a logical OR operator, the separator is 3 tildes (<code>~~~</code>). Example: <code>Red~~~Blue</code> These separator strings can be configured in <code>config.xml</code>.</p> <p>If you want the filter to exclude specific values (a logical NOT operator), you need to add a prefix before the filter value. By default this is an exclamation mark (<code>!</code>). Example: <code>!Red</code> This string is also configured in <code>config.xml</code>.</p>	<p>filter value, it contains information about how multiple <code>FilterValues</code> are combined (<code>type</code>). The object attribute <code>exclude</code> indicates whether or not the filter should be applied to exclude the value.</p> <p>In turn, multiple Filter objects can be added to the <code>Params</code> object.</p>
Search field	<p>Normally FACT-Finder searches all fields defined as searchable. However it is possible to search only one specific field as well.</p>	<p><code>searchField=FIELDNAME</code></p> <p><code>E</code></p>	<p><code>Params.setSearchField</code></p> <p><code>Id</code></p>

Article number search	Normally an article number search is carried out if the search term matches one or more set formats. You can also use parameters to prevent an article number search from being carried out.	If you pass the parameter <code>noArticleNumberSearch</code> with the value <code>true</code> , the article number search will not be carried out.	The <code>Params</code> object has an attribute <code>noArticleNumberSearch</code> through which searching by article number can be prevented. Using the attribute <code>isArticleNumber</code> you can also stipulate explicitly that an article number search should be carried out.
-----------------------	--	--	---

5.1.4.2. Parameters used to control the search process

	Description	URL parameter	Web service
Create ASN	Controls whether or not the ASN (after-search navigation block) is created. <code>true</code> = create ASN, <code>false</code> = do not create ASN. The default value is <code>true</code> .	<code>useAsn=true</code>	<code>SearchControlParams.useAsn</code>
Return found words	FACT-Finder is able to return the words that were used to find the data record for the located records. Determining these words takes up CPU time. For this reason, the function is disabled by default. Set the parameter to <code>true</code> to generate the search words or <code>false</code> if you do not want to generate search words.	<code>useFoundWords=true</code>	<code>SearchControlParams.useFoundWords</code>

	The default value is <code>false</code> .		
Suppress campaigns	<p>If you want to prevent the Campaign Manager from checking whether the search query matches a campaign, use this parameter. <code>true</code> = campaigns are evaluated and returned as appropriate, <code>false</code> = campaigns are ignored.</p> <p>More detailed information about the Campaign Manager can be found in the specific documentation.</p>	<code>useCampaigns=true</code>	<code>SearchControlParameters.useCampaigns</code>
Shop navigation	<p>FACT-Finder can also replicate your entire shop navigation. No search term is necessary if you want to use this function. You simply pass the corresponding request.</p>	<p>The <code>query</code> parameter is superfluous here. You just need the parameter <code>catalog=true</code>.</p>	<p>This parameter is not available via the web service. Please use <code>*</code> as the search term.</p>
Hide field contents	<p>The result normally contains all field information about the products that have been found. If you only need the IDs, you can request for the field contents not to be returned, thus saving bandwidth. <code>True</code> = do not pass field content. The default value is <code>false</code>.</p>	<code>idsOnly=true</code> .	<code>SearchControlParameters.idsOnly</code>
Generate SEO keywords	<p>FACT-Finder is able to generate product keywords that can be used for SEO¹.</p>	<code>useKeywords=false</code>	<code>SearchControlParameters.useKeywords</code>

¹ Search Engine Optimisation

	<code>true</code> = SEO keywords are evaluated and added to the product records, <code>false</code> = SEO keywords are not returned. The default value is <code>false</code> .		
Return complete question/answer tree for advisor campaigns	This parameter is used in conjunction with advisor campaigns. For more information, please consult the Campaign Manager documentation. <code>true</code> = returns the complete question/answer tree along with the advisor campaign. <code>false</code> = only the currently active questions and their answers are returned. The default value is <code>false</code> .	<code>generateAdvisorTree=false</code>	<code>SearchControlParameters.generateAdvisorTree</code>
Disable cache	This parameter controls whether or not the search result cache is used. <code>true</code> = cache is ignored, <code>false</code> = cache is used. The default value is <code>false</code> .	<code>disableCache=false</code>	<code>SearchControlParameters.disableCache</code>

5.1.4.3. Authentication parameters

	Description	URL parameters	Web service
Username	As FACT-Finder calls must be authorised, you need to pass a user.	The the username is passed using the parameter <code>username</code> .	<code>AuthenticationToken.setUsername</code>
Password	A password for the specified user must be sent. It must be encrypted using the formula described above.	<code>password=HASH</code>	<code>AuthenticationToken.setPassword</code>

Timestamp	If you want to use the advanced encryption method, the timestamp used for the hash must also be sent.	<code>timestamp=TIMESTAM MP</code>	<code>AuthenticationToken .setTimestamp</code>
-----------	---	--	--

5.1.4.4. Shop visitor information

The parameters below are used to pass information about the shop visitor. This parameter must be passed if you wish to use the Recommendation Engine. However, we recommend that you also pass this parameter even if the Recommendation Engine is not being used. We plan to use this information for behavioural-targeting functions in future releases of FACT-Finder.

	Description	URL parameters	Web service
Session ID	Use this to pass the user's session identifier. This can be anonymised. The only requirement is that the ID does not change during the user's visit.	<code>sid</code>	<code>UserInformation. sessionId</code>
User ID	Use this to pass a user identifier. As with the session ID, this can be anonymised. The user identifier differs from the sid in that it remains constant for a single user over multiple visits.	<code>userId</code>	<code>UserInformation. userID</code>
Cookie ID	You use this parameter to pass a token that identifies the user over a longer period of time, even when not logged in to the shop. Typically this is stored in a browser cookie.	<code>cookieId</code>	<code>UserInformation. cookieID</code>

5.1.4.5. XML/JSON parameters

The following parameters are only used for calls passed using XML/JSON.

	Description	URL parameters
Return format	Since the HTML, XML and JSON interfaces are invoked via URL, a parameter is required to indicate the format in which the result should be delivered.	The parameter has the name <code>format</code> . If it is not assigned a value, the result will be returned in HTML. Possible values are <code>xml</code> , <code>json</code> and <code>jsonp</code> .
Hide application name	The returned search parameter URLs normally start with the name of the FACT-Finder server application. However, this can be disabled. This function may be useful in load-balanced installations in which the different FACT-Finder applications have different names, for example. It can also be used in "normal" deployment scenarios in order to reduce the volume of data being sent.	If you pass <code>omitContextName = true</code> the URLs start with the action name rather than the application name.

5.1.5. Troubleshooting

If the FACT-Finder user interface displays a general error page (a code 500 page), the parameter `verbose=true` can be used to access additional information. Attach the parameter to the URL and refresh the page. The HTML source code then contains a stack trace for the error. If the error message is of no further assistance, please contact the FACT-Finder support team.

5.1.6. Caching search results

FACT-Finder has a search results cache that allows it to process queries that have already been asked before more quickly. The passed search parameters are verified to determine whether or not this is the same query.

For this reason, please do not pass any parameters that are not needed by FACT-Finder to the search engine, since this can impair the caching function and the search queries will take longer than they need to.

Naturally, you can also cache FACT-Finder results in your shop system. When doing this you should remember: There are various situations in which FACT-Finder considers its own cache to be obsolete and deletes it, such as when the configuration is changed or product data is altered.

The online shop will not learn about these changes. This means that in such cases, the online shop will hold cached results that are out of date.

5.1.7. Output compression

When a large number of results are produced, along with many filter options, the amount of data returned by FACT-Finder rises quickly. This can cause the network data transmission time to form a significant component of the overall search response time.

If supported by the web server, the server response can be sent after having been compressed using the GZIP algorithm to reduce the data volume.

Depending on the programming language used, however, it may be necessary to use client-side code to support this compression method as well. The client must explicitly request compressed output.

5.1.8. Suppression of blank search queries

Many visitors to web shops simply click the Search button without entering anything in the search field. Instead of launching a search query, the visitor should receive a message explaining that a search term should be entered in the text box. For example, you could display an information message with the corresponding explanation while simultaneously switching the focus to the search box. This will avoid blank queries (or searches using the default search term) and reduce annoyance for your users.

5.2. HTML integration

If you integrate FACT-Finder using this interface, in most cases you integrate the returned HTML into your page as an IFRAME. You must therefore program your search box in such a way that it invokes a FACT-Finder search. Normally only the search term is sent; the search parameters are listed in the description of the XML integration.

In addition, you must define the following method in the `parent` frame.

```
function setFrameHeight(h) {
  if (document.all) {
    document.all.search_result.style.height = h;
  } else if (!document.all) {
    document.getElementById("search_result").height = h;
  }
}
```

This method ensures that the FRAME or IFRAME that has the ID `search_result` is only as large as it needs to be. The function is invoked by the returned data.

Due to browser security restrictions (to prevent cross-site scripting) the page returned by FACT-Finder can only invoke methods on the parent page if it is integrated in the FRAME/IFRAME as a subdomain, and the `document.domain` page attribute of the parent page and return page is set to the principle domain. Therefore, please insert these lines as the first JavaScript block in your HTML <head> section:

```
document.domain = "yourDomain.com";
```

The domain for the search server must also be a subdomain of this principle domain. Example: `"search.yourDomain.com"`.

If you want to use the Campaign Manager module, it is also necessary to define when a forwarding campaign will be shown in the IFRAME area and when it will be shown on the complete page. You can do this using the syntax used to create the campaign or by means of a parameter.

5.3. XML integration

When integrating via XML you must convert the search query parameters into a URL and receive the search result back in a defined XML format. The result is obtained by submitting a URL such as:

```
../Search.ff?query=term&format=xml
```

5.3.1. Result structure

```
<ff>
  <searchStatus>resultsFound</searchStatus>

<articleNumberSearchStatus>noArticleNumberSearch</articleNumberSearchSta
tus>
  <searchTimedOut>>false</searchTimedOut>
  <searchParams>
    <searchPhrase>animation dvd</searchPhrase>
    <filters>
      <filter name="Category" exclude="false">DVD</filter>
    </filters>
    <channel>de</channel>
    <productsPerPage>20</productsPerPage>
  </searchParams>
  <productsPerPageOptions default="12" selected="24">
    <option value="12">
      <searchParams>...searchParams...</searchParams>
    </option>
```

```

        <option value="24">
            ...
        </option>
    </productsPerPageOptions>
    <breadcrumbTrail>
        <item no="0" type="search" value="dvd">
            <searchParams>...</searchParams>
        </item>
        ...
    </breadcrumbTrail>
    <singleWordSearch>
        <item nr="0" word="dvd" count="23">
            <record ...>...</record>
            ...
        </item>
        ...
    </singleWordSearch>
    <campaigns>
        ...
    </campaigns>
    <asn>
        <group name="Category">
            <element name="DVD" count="0" clusterLevel="0" selected="true">
                <searchParams>...</searchParams>
            </element>
            ...
        </group>
        ...
    </asn>
    <paging pageCount="3" currentPage="1" productsPerPage="20">
        <searchParams>...</searchParams>
    </paging>
    <sorting>
        <sort name="Relevancy" description="Relevancy" method="desc"
selected="true">
            <searchParams>...</searchParams>
        </sort>
        ...
    </sorting>
    <results count="42">
        <record nr="0" id="71004851">
            <field name="EAN">4011846012801</field>
            <field name="Category">DVD</field>
            <field name="Description">...here is the description of this
product...</field>

```

```
<field name="Title">products title</field>
...
</record>
...
</results>
</ff>
```

An XSD schema description file exists to allow you to generate processing code. Many elements have a `searchParams` tag – this contains the search parameters associated with this step in URL format.

Status information

The value carried in the `searchStatus` tag indicates whether or not a result was found. Possible values are `resultsFound`, `nothingFound` and `errorOccured`.

The `articleNumberSearchStatus` tag shows whether or not an article number search was carried out. Possible values here are `resultsFound`, `nothingFound` or `noArticleNumberSearch`.

If a timeout occurred during the search, the value `true` will be returned in the `searchTimedOut` tag, otherwise the value `false` is returned. A search that has timed out may not contain all matching products.

searchParams Block

This block contains the search parameters that were passed. The query parameters for visualisation of the response data can be deduced from this section, and therefore you can also check that the parameters were correctly recognised.

productsPerPageOptions Block

This block lists the options defined in the configuration regarding the number of products that can be displayed per page. The default option (`default`) and the currently selected option (`selected`) are passed as attributes. Each child `option` block also contains the search parameters used to select the corresponding option.

breadCrumbTrail Block

This block lists the user's previous steps (e.g. search query, filters set). It is normally used to create a breadcrumb trail that allows the user to jump back to a previous point.

```
<breadCrumbTrail>
  <item no="0" type="search" value="12">
    <searchParams>
      ... here is the url for the initial search ...
    </searchParams>
  </item>
</breadCrumbTrail>
```

```

</item>
<item no="1" type="filter" value="DVD" fieldName="Category">
  <searchParams>
    ... here is the url for the filter ...
  </searchParams>
</item>
</breadcrumbTrail>

```

singleWordSearch Block

This section is only displayed if the option has been activated to search for the individual words of the search request in the event of no hits or poor results. In such cases, the individual words of the search request are separated and searched for individually, so that the user can view the results of these searches. This feature is disabled by default for performance reasons.

Each component that would return a result is returned as an `item` element, which has an index (`nr`), the word (`word`) and the expected number of hits (`count`). You can also display a specific number of preview products for the individual terms, which are then returned as the respective `record` tags within the `item` tag. The structure is identical to the tag in `records`.

```

<singleWordSearch>
  <item nr="0" word="dvd" count="23">
    <record ...>...</record>
    ...
  </item>
  ...
</singleWordSearch>

```

campaigns Block

This block is only contained in the result if you have activated the Campaign Manager module and a campaign is available for the search query that was submitted. You can find more detailed information about this block in the documentation for the associated module.

asn Block

This abbreviation stands for "After Search Navigation". The block contains the filter options that are available to allow the user to further refine the search results. The ASN contains groups (e.g. price, colour, category) and their filter elements (e.g. blue, red, yellow).

```

<group name="Category" detailedLinks="7" style="DEFAULT">
  <element name="DVD" count="0" clusterLevel="0" selected="true">
    <searchParams>
      ... here is the url for deselecting the filter ...
    </searchParams>
  </element>
</group>

```

```

    </element>
    <element name="Drama" count="25" clusterLevel="1" selected="false"
previewImage="ImageURL">
      <searchParams>
        ... here is the url for selecting the filter ...
      </searchParams>
    </element>
    ...
  </group>
  <group name="Price" detailedLinks="5" unit="EUR" style="SLIDER">
    <element name="Price" count="0" clusterLevel="0" selected="false"
selectedMin="2"
selectedMax="1599" absoluteMin="2" absoluteMax="1599">
      <searchParams>
        ... here is the url for selecting the filter ...
      </searchParams>
    </element>
  </group>

```

A `group` tag represents a filter group, for which the `name` attribute is the name of the group. The attribute `detailedLinks` indicates how many links should be shown in detail. All others are normally shown in a Select box. The value can be defined using the Management Interface so that the settings can easily be changed. If a unit is defined for the group, the attribute `unit` is also present. The value of this feature should appear after each element name.

Each possible filter element in the group is returned in an `element` tag. The attributes contain, apart from the name (name), also the number of expected hits if they were limited/restricted to this (count).. The `clusterLevel` indicates the level on which the element is to be found. This can be used to indent categories, for example. The Boolean value `selected` indicates whether or not the element has been filtered. If the element has been selected, the URL to remove the filter is returned in `searchParams`. FACT-Finder offers the option of displaying thumbnails for filter elements. If this feature is active, the preview URL for the filter is the value `previewImage`.

If the group has been configured as a slider group in the Management Interface, the attribute value of `style` in the `group` tag is `SLIDER`. The element in the group then has the additional attributes `selectedMin`, `selectedMax`, `absoluteMin` and `absoluteMax`. The values of the attributes refer to the minimum and maximum values that are displayed in the slider. If no group selection has been made, both the minimum and maximum values are identical. If a range has been selected, it is returned with `selectedMin` and `selectedMax`. The range in `absoluteMin` and `absoluteMax` specifies the maximum range of the slider, so that a wider range can be selected under certain circumstances.

If you wish to use the slider to restrict a particular range, you will need to attach that range to the URL using `searchParams`. The value is derived from the minimum and maximum values which are separated by a hyphen (e.g.: 5 - 23).

paging block(Groß- und Kleinschreibung variiert bei den folgenden Überschriften)

This block contains information regarding the page of the search results on which you are currently located (attribute `currentPage`) and how many pages there are (`pageCount`). The page count begins with 1. The maximum number of products that can be shown on a page is also indicated (`productsPerPage`).

If you want to call up another page, you only need to attach the desired page number to the URL specified in `searchParams`.

```
<paging pageCount="3" currentPage="1" productsPerPage="20">
  <searchParams>
    ... here's the url which can be used for the paging. You only have to
    add the page number at the end ...
  </searchParams>
</paging>
```

sorting block

If there are multiple sort criteria, they are listed as `sort` elements in this block. The attribute `name` contains either the value `Relevancy` or the corresponding field name. `description` contains the name of the criterion, which can be shown on the interface. The remaining attributes provide information on the sort order (`method`) and whether the criterion has been selected (`selected`).

```
<sorting>
  <sort name="Relevancy" description="Relevancy" method="desc"
  selected="true">
    <searchParams>
      ... Here's the url for choosing the sorting ...
    </searchParams>
  </sort>
  ...
</sorting>
```

results Block

This section contains the details of the articles that have been found on this page. The `count` attribute indicates how many articles were found in total.

Each `record` tag represents an article. These have the attributes for the index (`nr`), the record ID (`id`) and similarity to the search term (`relevancy`). If you have not activated the `idsOnly`

parameter, each *record* also contains the field information for the corresponding product. This information is returned in the child *field* element. The field name is given as an attribute (*name*) and the field content the value of the tag.

```
<results count="42">
  <record no="0" id="71004851" relevancy="100.0">
    <field name="EAN">4011846012801</field>
    <field name="Category">DVD</field>
    <field name="Description">...here is the description of this
product...</field>
    <field name="Title">products title</field>
    ...
  </record>
  ...
</results>
```

5.4. Web service integration

From FACT-Finder version 6.5 we stabilised the web service interface. The interface made available with version 6.5 will remain in newer versions. This approach makes dealing with FACT-Finder updates easier for you. You do not need to change your own code in any way. If you want to integrate the options provided by new versions, however, you must update your integration scripts.

The path to the web service varies depending on which web service you want to invoke. For example, if you want to use the web service interface of version 6.5, the following pattern applies. Based on this, you can either arrange for separate processing classes to be generated or use the ones we have already made available.

```
../webservice/ws67/Search?wsdl
```

This documentation should make it clear how to use the interface. No sample code or implementation detail is shown here, but we will be happy to send you sample implementations as well as a JavaDoc.

5.4.1. Sending the search query

The methods *getResult* and *getResultWithCampaigns* are provided to query a search result. If you have licensed the Campaign Manager module, we recommend that you use the latter method, otherwise please use the first one.

These methods have parameters that comprise the search parameters as `Params` object and an `AuthenticationToken` object with which you must pass login data. The `AuthenticationToken` object must set the username, timestamp and encrypted password as described above.

The settings for the search query are set in the `Params` object using simple Setter methods. For the filters, it should be noted that if you want to allow selection of multiple elements in a field, you must first set multiple `FilterValue` objects in the appropriate `Filter` objects and then add this to the parameters.

If you want to send special parameters to FACT-Finder so that a custom solution can be implemented, please set these using the `customParameters`. However, these cases should be discussed in detail with an Omikron consultant.

5.4.2. Processing results

In response to the methods described above being called, the server will return a `Result` object. This contains all the search results information you need to display the data to your users.

The status information is among the most important data in the search result. This indicates whether a search result was found and whether an article number search was carried out. Based on this you can decide what results page to show the user. The search results of type `SearchRecord` are saved in a list and can be displayed accordingly. These objects contain all fields saved in the FACT-Finder database; they can be queried using `getRecord()`.

Please consult the XML integration description or the JavaDoc for a detailed description of the information.

5.5. JSON integration

The return structure of the JSON interface follows the XML structure closely, but here there are JavaScript objects and lists that you must query. To receive a JSON result, you must pass the `format` parameter with the value `json`:

```
../Search.ff?format=json
```

5.5.1. Result structure

```
{
  "searchResult": {
    "resultStatus": "resultsFound",
    "resultArticleNumberStatus": "noArticleNumberSearch",
    "timedOut": false,
    "resultCount": 42,
```



```

    "searchTime": 68,
    "simiFirstRecord": 9874,
    "simiLastRecord": 9683,
    "errorMessage": null,
    "campaigns": [ ...This object holds information about the matching
campaigns...],
    "pushedProducts": [...This object holds information about the pushed
products...],
    "singleWordResults": [
        {
            "word": "red",
            "recordCount": 513,
            "previewRecords": [{...list with records, see description
below...}]
        },
        ...
    ],
    "breadCrumbTrailItems": [
        {
            "text": "dvd",
            "type": "search",
            "value": "dvd",
            "searchParams": "... URL with search parameters to return to
this step ..."
        },
        ...
    ],
    "sortsList": [
        {
            "description": "sort.relevanceDescription",
            "name": null,
            "order": "desc",
            "selected": true,
            "searchParams": "... URL with parameters for the sorting..."
        },
        ...
    ],
    "resultsPerPageList": [
        {
            "value": 12,
            "selected": true,
            "default": true,
            "searchParams": "... URL with parameters for that amount of
records per page..."
        },
        ...
    ],

```

```

    ...
  ],
  "paging": {
    "pageCount": 2,
    "resultsPerPage": 12
    "currentPage": 2,
    "firstLink": {
      "caption": "1",
      "currentPage": false,
      "number": 1,
      "searchParams": "... URL with search parameters for first
page..."
    },
    "lastLink": ...Object for last link of the search result...,
    "previousLink": ...Object for previous link of the search
result...,
    "nextLink": ...Object for next link of the search result...,
    "pageLinks": [ ...Array of objects with paging information... ]
  },
  "groups": [
    {
      "name": "Category",
      "detailedLinks": 5,
      "groupOrder": 1,
      "unit": "",
      "filterStyle": "DEFAULT",
      "showPreviewImages": false,
      "type": "text",
      "groupTarget": null,
      "deselectText": "field.defaultDeselectLinkDescription",
      "moreElementsDescription": "",
      "elements": [
        {
          "name": "Drama",
          "recordCount": 3,
          "clusterLevel": 0,
          "selected": false,
          "associatedFieldName": "category1",
          "previewImageURL": "...If enabled, here's the preview image
URL...",
          "searchParams": "... URL to select the filter parameters
..."
        }
      ],
    }
  ],
  ...
],

```

```

    "selectedElements": [
      ... Array with all selected elements for this group.
      The searchParams attribute holds the URL to deselect the
      filter...
    ]
  },
  ...
],
"records": [
  {
    "id": "11915",
    "position": 1,
    "searchSimilarity": 98.74,
    "simiMalusAdd": 126,
    "foundWords": [],
    "record": {
      "products_name": "Easton XC Two Disc MTB Set",
      ...field information of the found record...
    }
  },
  ...
]
}
}

```

As the interface is URL-based, the secondary `searchParams` elements contain URLs. In the case of filter objects, for example, this is the URL you must submit in order to filter the corresponding value or to remove the filter.

Status information

The keys `resultStatus` and `resultArticleNumberStatus` indicate whether or not a result has been found and whether or not an article number search was carried out (as well as its status). Possible values are `resultsFound`, `nothingFound`, `errorOccured` and `noArticleNumberSearch` (only `resultArticleNumberStatus`).

If a timeout occurred during the search, you receive back `true` as the value of `timedOut`. Searches that have timed out have been aborted and maybe incomplete.

The value of `resultsCount` shows the total number of results found. `searchTime` returns the search time required in ms. The numbers in `simiFirstRecord` and `simiLastRecord` provide information as to the similarity of the first and last records in the result. The value range extends from 0 to 10000.

Campaign Manager

The objects returned in the keys `campaigns` and `pushedProducts` relate to the Campaign Manager Module. Details on this can be found in the associated documentation.

singleWordResults

```
"singleWordResults": [
  {
    "word": "red",
    "recordCount": 513,
    "previewRecords": [{...list with records, see description
below...}]
  },
  ...
],
```

If this feature is enabled and the search does not return a result, or the similarity of the best item falls below a set threshold, a search is conducted for the individual words of the search expression. The returned single-word objects have as their attributes the respective word (`word`), the number of hits achieved when searching for that word (`recordCount`), and optionally a certain number of products from the corresponding search result (`previewRecords`). Their structure is identical to the standard search results. However, the number of words returned is limited in a given configuration.

breadcrumbTrailItems

Objects that can be used to generate a breadcrumb trail are provided here. Each object in this list corresponds to a step in the trail. The value in `type` allows you to determine what action this step led to. The name of the step, which corresponds to the search term or the filter value, can be found as the value of `text`. The value of the key `searchParams` is the URL that must be invoked to return to this point.

sortsList

Each of these objects corresponds to one of the result's sorting options. In `description` you will normally find a key that can be used for an internationalised language version. When sorting by relevance, the key `name` contains the value `null`. In other cases, this is the name of the field according to which the products are sorted. The sort order is indicated in `order`; possible values here are `asc` and `desc`. The criterion that has been selected is indicated by the value `selected`. The URL in `searchParams` provides the URL that generates the specified sort criteria.

resultsPerPageList

The Management Interface allows you to configure the maximum number of articles that should be shown on a results page. Each configured number will be returned as the `value` of one of these objects. The Boolean operator in `selected` indicates whether this value is selected, while the operator in `default` indicates whether this number is the default value.

paging

The values that are returned here allow you to create the page navigation. The number of pages is given in `pageCount`, while the maximum number of results per page is in `resultsPerPage`. The current page number is contained in `currentPage`.

The remaining items return either an object or a list of objects, each of which describes a page link. These contain the text that should be shown for the page as the value of the `caption` key. As a result, instead of showing the page number (1, 2, 3, etc.) it is also possible to generate the range of items (1-10, 11-20, etc.). Therefore this may differ from the value of `number`. The value of `currentPage` indicates whether the page is the one currently being displayed or not.

group

The `group` tag contains an element for each ASN group. The name of the group is sent as the value of the `name` key. The number of filter elements that should be shown in detail is described by the value `detailedLinks`. Normally all elements that exceed this value will be shown in a drop-down box. The value in `unit` is also important for displaying groups. This should be shown after each element value.

The two lists `elements` and `selectedElements` provide all of the filter elements belonging to the group by which the results can be refined. Each object in this list contains information about its name (`name`), the expected number after the filter refinement (`recordCount`) is applied, the associated field (`associatedFieldName`) and whether it has been selected or not. The value in `clusterLevel` indicates the level of the element. This should be used to move back up through the category paths. If preview images are to be shown for the elements of the groups (`showPreviewImages` in the group object), the URL of the preview images for each element can be found as the value of `previewImageURL`.

ASN groups that are to be displayed as sliders have a special characteristic, with the value `SLIDER` for the attribute `filterStyle`. In addition, the returned element has the attributes `selectedMinValue`, `selectedMaxValue`, `absoluteMinValue` and `absoluteMaxValue`. See the description of the XML interface for more details. If you wish to restrict a particular range, you will need to attach a parameter for the selected range and a parameter for the absolute range to the URL, using `searchParams`. The parameter values are derived from the range values separated by a hyphen (e.g.: 5 - 23). The parameter names are derived from the prefix `filter` and the related

field name. The suffix `Absolute` must be attached to the parameter name for the absolute range. For example, a field with the name Price would result in the following parameters:

```
...&filterPrice=5+-+23&filterPriceAbsolute=1+-+50.
```

records

In `records` you will find the search results that are to be shown on the selected page. Each search results object contains the ID of the product in the FACT-Finder database (`id`), its position in the results list (`position`), the calculated similarity to the search term (`searchSimilarity`), as well as the product importance devaluation (`simiMalusAdd`). There are many reasons why a product can be devalued; the most common are business rules. Each product's field information can be found in `record`. This list contains the field name and its value as a pair.

5.6. JSONP integration

In addition to the JSON interface, the JSONP interface allows you to pass the name of a callback method that can be called on the client system.

The JSONP interface is called using the parameter `format=jsonp`, with the method name passed in the `callback` parameter.

```
../Search.ff?format=jsonp&callback=METHOD_NAME
```

You can use JSONP for all interfaces that support JSON.

6. Technical documentation for other interfaces

This chapter contains documentation for the other interfaces that can be used instead. It is not always necessary to use one of these, but in many cases it makes sense to do so.

Documentation for the individual FACT-Finder modules (e.g. Campaign Manager, Recommendation Engine) can be found in separate documents.

Please ensure that the user you use for authentication purposes when sending requests to this interface holds the relevant rights. You will receive an error message if this is not the case.

6.1. Updating databases

In most cases FACT-Finder handles data updates itself. However, depending on the definition of the update process, the import will not be initiated by FACT-Finder itself but via the Import interface. This makes sense especially if the data is provided using the Push method.

If a URL from which your export data can be drawn is stored in the FACT-Finder Search configuration, it is possible to download this before the import, so that it can be loaded.

6.1.1. HTTP interface

You can start a FACT-Finder data import by addressing the URL below. The parameters influence how the import is carried out and the results achieved:

```
../Import.ff
```

If no specific channel is defined, the import is carried out for all channels. If you only want to update the database for a specific channel, you must send the name of the channel as the value of the parameter `channel`.

The parameter `download=true` causes the import file to be updated first. If no URL is stored in the FACT-Finder configuration, this parameter has no effect.

In addition to the data import for normal search data, you can also start an import for the Suggest database; the parameter `type=suggest` must be sent for this purpose.

Normally this interface returns an HTML page. If the parameter `format=xml` is sent, the result is returned in XML format, which is better for machine processing. If you only want to receive a response if an error occurs during the import, attach the parameter `quiet=true` (this parameter is independent of the `format` parameter). The response has the following format:

```
<importProgress>
  <currentChannel/>
  <progress>0</progress>
  <percentage/>
  <totalPercentage/>
  <finished>false</finished>
  <running>true</running>
  <status>
    <message>... A Status message ...</message>
  </status>
  <errors>
    <message>...An error message...</message>
  </errors>
</importProgress>
```

6.1.2. Web service interface

As with the search, the path to the WSDL depends on the version of FACT-Finder used.

```
../webservice/ws67/Import?wsdl
```

You start the import processes using the methods `startImport`, `startImports`, `startSuggestImport` and `startSuggestImports`. As a result you will receive a map that can contain `ERRORS` or `STATUS` as a key. A list of reports will be returned as the value of the key. If you call `startImports`, the result is stored in another map, which contains the particular channel name as the key.

If an update URL is configured for the product data, some methods of the product export file can be updated in advance via the `download` flag. If this is not the case the method `downloadProductExport` can be used for updating the product data. This method accepts as parameters the channel name, the file name (`fileType`), the URL and the user data. The file name is configured permanently in the FACT-Finder search. By default it is `productData`, but ask to be sure or look in the configuration.

6.2. Changing, deleting and adding records

This interface allows individual or multiple records in the search database to be updated, deleted or added. It should be noted, however, that changes will be lost when a data import is performed via this interface. The product data export provided by your shop system is always used as the basis of the import. Therefore only use this interface temporarily to make a quick correction to a record.

All changes are immediately available in the search. By saving the database the changes made will be written to the hard drive, so that the changes will also be available after a restart. The saving process requires processing power, so it should not be carried out for every change call.

6.2.1. HTTP interface

Each of the update methods is invoked using a different URL:

```
../Update.ff
../Delete.ff
../Insert.ff
```

In the `id` parameter you must indicate the record ID of the record that should either be deleted or updated. If it is a new article, this parameter will cause a new ID to be generated in the database.

If you want to update or create an article, you must send its fields as parameters. For this purpose, the parameter name is the field name and the parameter value of the associated field value.

If `save=true` is sent as a parameter, the database will also be saved after the change.

6.2.2. Web service interface

The WSDL of this interface is identical with that of the import interface, but here the methods `insertRecord`, `insertRecords`, `deleteRecord`, `deleteRecords` and `updateRecord` are used.

All methods send the channel name, the login data and the save flag. The delete methods will also send either an ID or a list of IDs that are to be deleted. Furthermore, the update and addition methods have one or more `ImportRecord` objects.

6.3. Reloading the database

Normally a search database is reloaded if it was changed. This behaviour can be deactivated to improve performance. In this case the database loading procedure must be initiated manually after a change in the database.

It is possible to arrange for the search and Suggest databases to be reloaded together or separately. In addition, you can define whether a database will be reloaded immediately or only after the next query.

6.3.1. HTTP interface

The interface acts differently depending on which parameters and values are passed on to it. For example, the basic URL is:

```
../RefreshDatabases.ff
```

The parameter `do` defines which database should be reloaded. Possible values are `refreshDatabases` to reload only the search database, `refreshSuggestDatabases` for the suggest database and `refreshAllDatabases` for both. In addition, with `refreshDatabasesOnNextRequest` it is possible to reload both databases when the next query is received.

In all versions, except for `refreshDatabasesOnNextRequest`, it is also possible via the `channel` parameter to arrange for the database to only be reloaded for a specific channel. If it should apply to several channels, the channel names must be entered, separated by a comma.

If the request was successful, you will receive an empty page as the result. If an error occurred, you will receive an error page.

6.3.2. Web service interface

The WSDL of this interface can be reached at:

```
../webservice/ws67/RefreshDatabases?wsdl
```

The available methods accept the user data as well as the desired channel names or a list of these.

6.4. Checking that the data is up to date

In a search service, it is enormously important that the underlying data is up to date. To check this, there is an interface that supplies this information. In the FACT-Finder configuration, a period of time is indicated for each database. If the database is not updated within this period, FACT-Finder considers it out of date and delivers corresponding information via this interface.

6.4.1. Text interface

This function is only available via an HTTP interface and can be invoked at the following URL:

```
../DatabaseExpiration.ff
```

Please enter an appropriate user as a parameter. In addition, using the parameter `channel`, you can specially query this interface for one or more channels. If you want to query several, you must enter the channel names separated by a comma.

If all data is up to date you will receive nothing back. If one or more of the queried data files is out of date, you will receive a response that looks similar to the following:

```
expired file for channel de: productData - Tue Apr 27 10:06:07 CEST 2010
expired file for channel de: database - Tue Apr 27 10:06:10 CEST 2010
```

6.4.2. XML Interface

You can also receive the obsolescence report in an XML format. If this is what you request, you must add the parameter `format=XML` to your call. The data will be returned in the format shown in this example:

```
<ff>
  <messages channel="de">
    <message>images - Wed Mar 23 16:52:52 CET 2011</message>
    <message>productData - Fri Apr 08 10:29:07 CEST 2011</message>
    ...
  </messages>
  ...
</ff>
```

The messages (`message`) are structured into `messages` tags, one for each channel. If no messages exist you will only receive the `ff` tag in response.

6.4.3. JSON interface

You can pass the parameter `format=json` to request the response in JSON format.

```
{
  "de": [
    "productData - Fri Apr 08 10:29:07 CEST 2011",
    "images - Wed Mar 23 16:52:52 CET 2011",
    ...
  ],
  ...
}
```

The output consists of a map that contains the channel name as the key and a value comprising a list of messages. If no messages exist the map is returned empty.

6.5. Channel management

This interface makes it possible to display all available channels in the search environment, to add new ones and to delete specific ones.

6.5.1. HTTP interface

To carry out one of the points mentioned, please use the Setup Wizards on the Management Interface.

6.5.2. Web service interface

To access the information via the Web service interface you must invoke the following WSDL:

```
../webservice/ws67/ChannelManagement?wsdl
```

Here the methods `getAllChannels`, `createChannel` and `deleteChannel` are available to you. As parameters, these methods expect the user data and the channel name to be created or deleted.

6.6. User management

This interface was introduced to create or delete users or to adjust their rights. Any number of roles can be brought together in a group through which users can be provided with rights indirectly. This simplifies the maintenance of large numbers of users with identical rights.

6.6.1. HTTP interface

To manage users and rights, please use the Management Interface.

6.6.2. Web service interface

The WSDL for this interface can be found at:

```
../webservice/ws67/UserManagement?wsdl
```

Here it should be noted that the password of a new user can only be transferred with encryption in accordance with the simple variant. Transferring it over an insecure connection is therefore not advised.

6.7. System monitoring

This interface provides information about some system characteristics so that the Fact-Finder search operating environment can be monitored. For example, it indicates how heavy the CPU load is or how high the memory usage is.

6.7.1. HTTP interface

You have the option of querying this information to be returned, prepared in a human-readable form, via the Management Interface or in a machine-readable form in XML format. You will receive the XML data by querying the following URL:

```
../restfulservices/system/getSystemInformation
```

7. Optimising the search results & tips

FACT-Finder is designed to function well straightaway and with little configuration. To achieve the best results with FACT-Finder, however, it is necessary to help the search out a little in a few places. FACT-Finder provides you with the tools you need to do exactly that.

We recommend optimising the search results as soon as technical integration is complete. Before starting the optimisation process, please check whether the display order of the search results is identical in FACT-Finder and your interface. The product sequence should be the same. If this is not the case, please find the reason for the difference before starting the optimisation. A common cause of error is a filter parameter that is sent by default from the shop to FACT-Finder (e.g. name). This should never happen. FACT-Finder sorts its hits according to relevance. A standard sort based on other criteria will deliver confused results.

Once it is ensured that the integration and the FACT-Finder Demo interface does not deliver different results, the optimisation can begin. For this step, it is worth testing terms that have recently been searched most often and checking their results. If for some of these FACT-Finder delivers no results or does not deliver the desired results, make a note of this and then begin with the optimisation.

As many optimisations are very specific, this document will not describe them in greater detail. You can obtain a document from Omikron that deals exclusively with search result optimisations. If you have questions about specific problems, we will be happy to assist you.

7.1. Community thesaurus

Most problems can be solved using either the thesaurus or the pre-processor. The Community Thesaurus Project provides a database with checked thesaurus entries. These are categorised according to topic and language to make it easy for you to find and apply appropriate entries for your shop.

7.2. Maintaining a good overview

We recommend that from time to time you examine the search results of your search more closely. The log file analysis and the What's Hot module are best suited to do this. You can set the Management Interface to automatically send you a search report by email at specific time intervals. In addition to the top search terms, this contains the search terms that have produced the worst or no results. You should concentrate on these and draw conclusions from them. The trend analysis section in this report is also interesting. It highlights search terms that have recently been searched unexpectedly frequently or infrequently.

You can also arrange to be notified if problems occur during the import.

8. Best Practice

The purpose of this section is to provide suggestions and ideas on how to integrate FACT-Finder. The items below should be viewed as hints and are intended to explain the possibilities for your implementation. Depending on your shop and sector, it may be that some of the items below are not suitable or can be ignored.

8.1. General

8.1.1. Usability

- Usability studies have demonstrated that a very large proportion of shop visitors enter the shop directly through the search function. Therefore, we recommend that you make the search box relatively large and prominent on your page, so that it immediately attracts the eye. In addition, the focus should be placed directly onto the search box so that users can enter their search term straight away.
- The way results are displayed may also differ according to the type of products on offer. For example, if your shop sells items of fashion a gallery view will be most suitable to allow you to emphasise the design. On the other hand, retailers of products to customers who are more interested in specifications will benefit to a greater extent from a list view.
- If your visitors are able to access specific additional information (such as 360° views, product videos, etc.) from the details page of a product, you should guide them to this on the results page, possibly even giving them the option to view these extras from there.
- In the event that users are unable to find the desired product, they should be presented with a "not-found" page, which also provides further hints or product recommendations. Another option at this point is to display the entire product range, but to indicate that this list does not contain products relating to the search query.

- You can also improve the users' search experience by integrating other FACT-Finder modules. In particular, the Suggest, TagCloud and Campaign Manager modules offer significant value to customers.

8.1.2. Design

- The search box should attract the visitor's eye immediately, so it is recommended it should be located in the left-hand region or in the middle of the header region.
- If you want to refer to product characteristics on a target basis (features such as "new", "bargain", "video", etc.) then you can use relevant icons in the search results. Remember, though, to keep the amount of information to a reasonable level, since it is very easy to overload the results page and annoy customers with a crowded, difficult to read list.
- In some cases it may be advantageous to offer the customer multiple view options, allowing them to switch between a list view and a gallery view.

8.1.3. Function

Adapting the article number search

FACT-Finder can be set up in such a way that if the search term corresponds to one or more formulas, a specific field will be searched exactly. Hence an article number search will return only the exact matching product.

If no product matches the article number entered, FACT-Finder normally searches in an error-tolerant manner. If you do not want to display articles found like this but would prefer to display the Nothing Found page instead, you must evaluate the search results status appropriately.

Showing information on the "not-found" page

Thanks to its error tolerance and search algorithms, FACT-Finder is frequently able to avoid returning an empty search result to your customers. However, this situation cannot be avoided completely. As a result, you need to consider what you want to show to customers who are unable to find what they seek.

A range of different options is available. In all cases you should display an informative text to the effect that no matching products have been found and offer the opportunity to repeat the search using different terms.

It is also a good idea to use this normally rather empty page to provide the customer with additional data. For example, you could display the most frequently used search terms or a list of top-selling products in order to awaken the user's interest. The FACT-Finder Campaign Manager and FACT-Finder Tag Cloud modules are very useful in this respect.

Only evaluate customer search queries

In order to achieve greater accuracy in your evaluation of customer search patterns, we recommend that you exclude the searches made by you and by your employees from the evaluation. It may also be useful to exclude searches that come in through specific advertising paths.

To do this, pass the `log` parameter in addition to the search query. The value in this parameter is used to describe the log file entry and can be viewed when analysing the log file. You can use values such as "internal" or "adwords".

The parameter and its value can be passed just like any URL parameter or CustomParameter.

Allowing users to change view

If your product range is very large, it may be helpful to your visitors to allow them to switch between gallery views and list views of the search results. This function should be easy to access by clicking an appropriate icon.

You can also select the view dynamically, using attributes associated with the hit list. For example, if the search results mainly comprise items of fashion, you would want to show a gallery view as the default view. On the other hand, if most products are technical items it is better to display a list view.

Failover options

Despite all of the mechanisms and systems you put in place to prevent this from happening, sometimes your search facility may not be accessible. In such cases, the user should be presented with an error page that clearly explains the situation in layman's terms. This will generate greater acceptance than displaying purely technical information.











8.1.4. Examples

Start ← Click here

notebook

Relevance

Selected items ...

<input type="checkbox"/>		Toshiba Satellite Pro C660-16H Notebook ▶ Reference No. PSCOME-01S00SEN Bechtie No. 691241-01 Version: English		£ 287.00 £ 344.40	⇅
<input type="checkbox"/>		Toshiba Satellite Pro C660-1V0 Notebook ▶ Reference No. PSCOME-02300SEN Bechtie No. 684730-01 Version: English		£ 300.00 £ 360.00	⇅
<input type="checkbox"/>		Toshiba Satellite Pro C660-219 Notebook ▶ Reference No. PSCORE-01L01MEN Bechtie No. 703266-01 Version: English		£ 320.00 £ 384.00	⇅
<input type="checkbox"/>		Lenovo TP X121e 3051-62G Notebook Top ▶ Reference No. NWS62UK Bechtie No. 706083-01 Version: English		£ 326.00 £ 391.20	⇅
<input type="checkbox"/>		HP 630 i3-370M Notebook ▶ Reference No. A1F77ES#ABU Bechtie No. 717851-01 Version: English		£ 333.00 £ 399.60	⇅

Replacement (2)
Cables / Connectivity (2)
Audio Visual (1)
Printers / Printing Solutions (1)
Furniture / Safes (1)
PCs / Thin Clients (1)

Manufacturer

Go to

Product group

Manufacturer

Service Chat online | TEL: 0845 089 64 65 + Monday-Friday 8.00 am - 5.00 pm | service@fashionforhome.co.uk

[Login](#) | [My Account](#) | [My Watchlist](#)



Modern design's new home



Shopping Basket

Furniture Categories Living Dining Bedroom Office Accessories NEW Bestsellers Brands

Search entire store here... About FASHION FOR HOME New Designs

Accano Search

25 results



Arcano Extendable Dining Table
£319.00 £609.00



Arcano Bookcase with 2 Doors
£339.00 £659.00



Arcano Desk
£419.00 £809.00



Arcano Set of Chairs
£129.00 £239.00



Arcano TV Cabinet
£219.00 £429.00




Arcano Cabinet
£339.00 £659.00






Arcano Small Dining Table
£119.00 £219.00



Arcano Large Mirror
£69.00 £139.00




[Newsletter](#) | [gift-certificates](#) | [FAQ](#) | [Customer Service](#) | [My Account](#)


 Free Shipping & Return Shipping ²



SHOPPING BASKET

29.12.11

0 article 0,00 € ¹





PRODUCTS

BRANDS

% SALE %

KIDS BIKES

FULL SUSPENSION MTB

BIKES

SUPER LOW PRICES

BIKE EQUIPMENT

TOP SELLER AT KILLER PRICES

BIKE PARTS

ASSORTMENT OF MORE THAN 40000 GOODS

CLOTHING

MANY GOODS GREATLY REDUCED

Oops Sorry!

The page or product you've searched for couldn't be found.

Please try our search feature to help you find products, categories or brands. You can also go to the [Homepage](#) to browse through our large selection of Bikes and bits.

If you have any questions, please take a look in our [FAQ](#)'s or feel free to contact our customer [services](#).

backpacks bicycle lock bikes+girl bottle bottom bracket boys+bikes carbon+bike carbon+road+bike cassette child+bikes city+bikes city
bike colnabi **crank** cube cycling dirt dmr dutch+bike fixed+wheel+bikes folding+bicycle folding bicycle folding mtb tyres freeride+downhill full
suspension bike girls+bikes helmets hybrid+bicycle hybrid bike kids+bikes kids+mountain+bike kind shock damper climbmax 1.0 38 mm
suspension fork ladies+bikes ladies+cycle ladies+road+bike ladies mountain bike marathon maxi maxi micro maxi micro lila mens+bike mens bike micro micro scooter
mtb ortler lindau men pedals poc puma **race+bike** road+bike road+race+bikes roadbike scooter sportbike standard grip/commu gel sram tacy tacy 2200 tacy
and elite roller trainers ~~the+bikes~~ tools ~~tools~~ touring+bike touring bikes triathlon ~~tools~~ ~~tools~~ wheels women cross bike womens+bikes

8.2. After Search Navigation

8.2.1. Usability

- Customers should be able to limit their search results using a range of attributes. However, the number of attributes available should not detract from the actual search result. Therefore, you should limit the available options. This can either be done by defining dependencies between attribute groups or by only displaying certain groups if a certain proportion of the search result supports the corresponding attribute filter.
- When it comes to numeric groups there are a couple of other peculiarities that are worth considering. For example, the easiest way (for the customer) of limiting a specific range of values is to use a slider control. If you do not want to use this control, then you should present logical ranges that visitors can use to filter the results.
- With some attributes you should also permit multiple selection so that the user can find it easier to browse to the desired product.
- Frequently used filter options such as category and manufacturer should be easy for visitors to find and to use. On the other hand, attribute filters such as size and storage capacity can be displayed as drop-down boxes.
- The filter elements themselves should be sorted by the number of hits. There are, of course, exceptions to this, so it makes sense to sort manufacturers alphabetically. When it comes to clothes sizes, we also recommend that a more logical sort order is used.

8.2.2. Design

- The ASN should normally be located to the left of the search results. If you display it above the search results you may find that users then do not immediately see any products and have to scroll down.
- There is an option to return and display preview images for the individual filter options. These images can provide shop visitors with a quick overview of what the filtered results contain.
- Some attributes are well suited to this form of visual presentation, which rapidly tells users what the filter does. For example, you could display a colour filter using different coloured swatches. In addition, users often appreciate the meaning of small icons used in filter elements (e.g. tailoring cuts).

8.2.3. Function

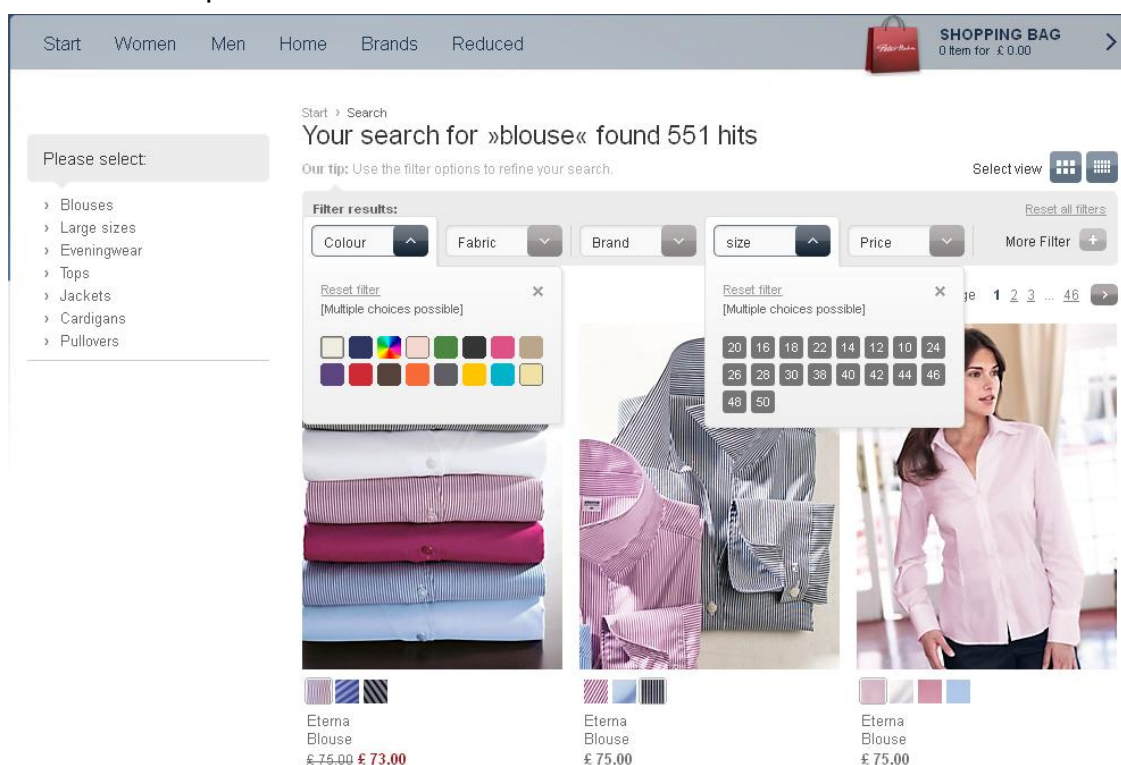
Extracting attributes from descriptive texts

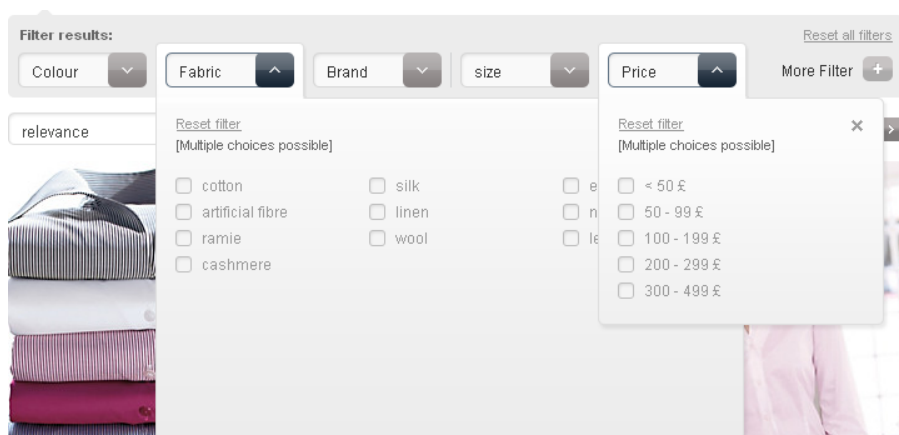
If your shop system does not have attributes in a clearly structured form and any product characteristics are rather "hidden" in the descriptions, you may find the Attribute Generator module helpful. This module extracts the relevant attributes from brief and long descriptions and prepares the product data for browsing using filters.

Simple limitation of numerical groups

There are many approaches when it comes to limiting ranges of numeric filters. For example, you can offer the customer predefined or dynamic price ranges, a slider control and/or input boxes. As a FACT-Finder customer you have access to the One-Touch Slider, which we developed ourselves. This slider tool allows users to select a desired range of values simply by clicking and moving the mouse.

8.2.4. Examples





Any questions?

If you have questions or suggestions about the documentation, please call us at: +49 (0)7231/12597-701 or send us an email at support@fact-finder.com. A qualified employee will be happy to assist you.