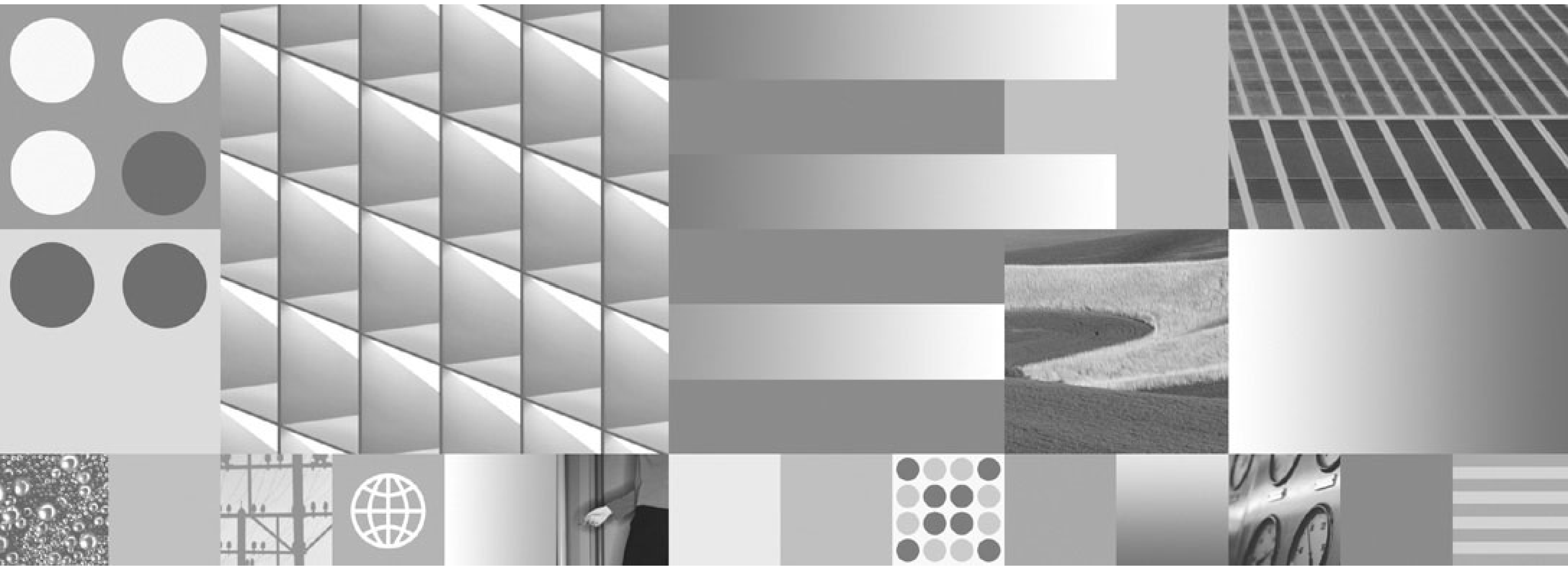
IBM OmniFind Yahoo! Edition

Version 8.4.2

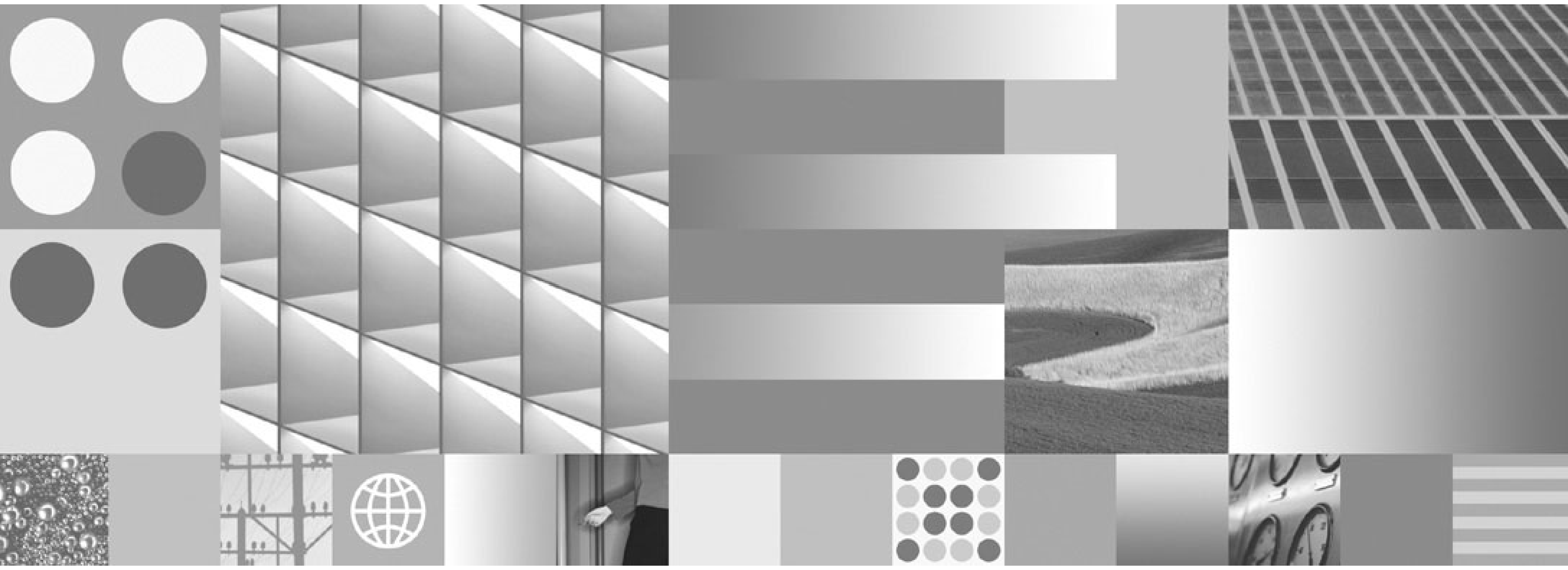


Programming Guide and API Reference

IBM OmniFind Yahoo! Edition

IBM

Version 8.4.2



Programming Guide and API Reference

Note

Before using this information and the product it supports, read the information in "Notices" on page 33.

Edition Notice

This edition applies to version 8, release 4, modification 2 of IBM OmniFind Yahoo! Edition (product number 5724-R21) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

ibm.com and related resources

Product support and documentation are available from ibm.com.

Support and assistance

Product support is available on the Web. Click Support from the product Web site at:

OmniFind Yahoo! Edition
http://www.ibm.com/software/data/enterprise-search/omnifind-yahoo/support.html

PDF publications

You can view the PDF files online using the Adobe Acrobat Reader for your operating system. If you do not have the Acrobat Reader installed, you can download it from the Adobe Web site at http://www.adobe.com.

See the following PDF publications Web sites:

Product	**Web site address**
IBM OmniFind Discovery Edition	http://www-1.ibm.com/support/docview.wss?rs=3035&uid=swg27008552
IBM OmniFind Enterprise Edition	http://www-1.ibm.com/support/docview.wss?rs=63&uid=swg27007911
IBM OmniFind Yahoo! Edition	http://www.ibm.com/support/docview.wss?rs=3193&uid=swg27010191

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

Send your comments by using the online reader comment form at https://www14.software.ibm.com/webapp/iwm/web/signup.do?lang=en_US&source=swg-rcf.

Contacting IBM

To contact IBM customer service in the United States or Canada, call 1-800-IBM-SERV (1-800-426-7378).

To learn about available service options, call one of the following numbers:

- In the United States: 1-888-426-4343
- In Canada: 1-800-465-9600

For more information about how to contact IBM, see the Contact IBM Web site at http://www.ibm.com/contact/us/.

Application programming

You can use application programming interfaces (APIs) to integrate IBM® OmniFind™ Yahoo! Edition with custom applications.

The APIs offer the following functions:

- Send queries and receive search results. For example, you can embed the results directly into a Web page or you can format the results according to the look and feel of your Web site.
- Add documents to a collection. For example, you can add content from a data source that cannot be crawled by one of the crawlers that are included with the product, such as an enterprise content management system.
- Delete documents from a collection. For example, you can delete documents that you no longer want users to see in the search results.
- Start or stop a crawler. The crawler management API can be used by scheduling programs to start or stop crawling at specific times.
- Assign values to the metadata fields when you add documents to a collection. If metadata fields are configured for a collection, you can specify the field values.
- List the names of all the metadata fields in a collection.
- List the names of all the collections in your search system.

API requests are based on the HTTP standard, which makes the APIs programming-language independent.

Search API

The `search` API supports search requests that are formatted as **HTTP GET** commands and returns search results as Atom feeds or HTML snippets.

Search requests
An **HTTP GET** request returns documents that match the search criteria.

Search results
You can customize search results that are returned in the Atom 1.0 Syndication format by specifying a stylesheet in the search request. When search results are returned as an HTML snippet, you can embed the HTML search results into an existing Web page.

Search requests

Your search application can provide a search box that issues an **HTTP GET** command to the server.

Related reference

"Error responses" on page 25

HTTP GET search requests

The search request is a standard **HTTP GET** command.

You can form the search request URL by combining the following properties:

- Host name
- Port
- Path
- Search request parameters, a collection of name-value pairs () that are separated by ampersand (&) characters

The host name is the host name of the search engine server. The port is the port number for the search application, a value that is specified initially when the search engine server is installed. The path to send your search requests to is always `/api/search`.

Examples of HTTP GET search requests

The following example shows a URL format that searches the Default collection and returns the first five results that match the query *Siamese*. Results are returned in the default Atom output format.

```
http://hostname:port/api/search?query=Siamese&collection=Default&results=5
```

The following example shows a URL format that searches the Default collection and returns the first 20 results that match the query *Siamese*. All results are returned in either Spanish (es) or German (de):

```
http://hostname:port/api/search?query=Siamese&collection=Default
&results=20&resultLang=es|de
```

The following example shows a URL format that searches the Default collection and returns the first 10 results that match the query *fiesta*. The query term is in Spanish (es).

```
http://hostname:port/api/search?query=fiesta&collection=Default
&queryLang=es
```

The following example shows a URL format that searches the Employees collection for the query term *manager* and returns results that number 11-20. Also, the Atom results that are returned are formatted by using the specified XSLT stylesheet that is located at `http://myserver.com/stylesheet/atom.xsl`.

```
http://hostname:port/api/search?query=manager&collection=Employees
&start=10&results=10&stylesheet=http://myserver.com/stylesheet/atom.xsl
```

The following example shows a URL format that searches the Employees collection for the query term *manager* and returns results that number 11-20. Results are returned in the HTML snippet output format:

```
http://hostname:port/api/search?query=manager&collection=Employees
&start=10&results=10&output=htmlsnippet
```

Search request parameters

You can use various options in search requests.

The order of the parameters in the requests does not matter. The parameter names are case-sensitive, and they must be entered in the documented format. Any unknown or unsupported parameters that are submitted as part of a request are ignored.

The following table shows the supported parameters for search requests:

Table 1. Search request parameters

Parameter name	Description	Default value	Comments
collection	The name of collection to search.		Required. The value must be UTF-8 encoded and URL-escaped. You can use the administration console or the `collections` API to see the names of all of the collections that are enabled for search.
fields	Mandatory metadata field values to be returned for each result, regardless of the query terms.		Use the colon (:) character to separate the mandatory fields to be returned. For example: `fields=author:keywords`. The value should be URL-escaped.
filter	Filters the search results to detect duplicate documents.	`true`	The supported values are `true` and `false`. When set to `true`, documents that are exactly alike are collapsed so that a single result is displayed in the search results. A single result is also displayed for documents that have a matching title and summary.

Table 1. Search request parameters (continued)

Parameter name	Description	Default value	Comments
locale	The client locale.	Server locale	Returns messages in the language of the client locale. Supported values: `de_DE` - German `en_US` - English `es_ES` - Spanish `fr_FR` - French `hu_HU` - Hungarian `it_IT` - Italian `ja_JP` - Japanese `ko_KR` - Korean `nl_NL` - Dutch `pl_PL` - Polish `pt_PT` - Portuguese `pt_BR` - Brazilian Portuguese `sv_SE` - Swedish `zh_CN` - Chinese (Simplified) `zh_TW` - Chinese (Traditional)
output	The format of the message body in the server response.	`atomxml`	The supported values are `atomxml` and `htmlsnippet`.
oyeFieldFormat	Use for deprecated fielded search response formats.	`false`	Prior to IBM OmniFind Yahoo! Edition Version 8.4.2, the metadata fields in Atom search responses are represented in the deprecated **/feed/entry/omnifind:field** format. Set the value to `true` for search responses to continue using this format. See "Atom feeds" on page 7 for information about the new format.
query	The query string.		Required. This value must be URL-escaped.

Table 1. Search request parameters (continued)

Parameter name	Description	Default value	Comments
queryLang	The language of the query string.	Server locale	Supported values: `ar` - Arabic `cs` - Czech `da` - Danish `de` - German `el` - Greek `en` - English `es` - Spanish `fi` - Finnish `fr` - French `he` - Hebrew `it` - Italian `ja` - Japanese `ko` - Korean `nl` - Dutch `no` - Norwegian `pl` - Polish `pt` - Portuguese `ru` - Russian `sv` - Swedish `zh_CN` - Chinese (Simplified) `zh_TW` - Chinese (Traditional)
queryTimeout	The maximum evaluation time in seconds for the query request.	`0` (unlimited)	Specify a value for this parameter to limit the amount of time that query requests are evaluated.
resultLang	The languages in which search results are to be filtered and returned.		Use the vertical bar (|) to separate language strings. For example, `en|de|fr`. This value must be URL-escaped. See the list of allowed values in the **queryLang** parameter comments section. In addition, the following value is supported: `tr` - Turkish
results	The number of search results to be returned for any single search request.	`10`	The minimum value is 0. The maximum number of results that are returned for any request is 1250.
start	The offset to the first result to return in the search results.	`0`	If the specified value is negative, the value by default is 0. If the specified value is greater than or equal to the number of results, no results are returned.

Table 1. Search request parameters (continued)

Parameter name	Description	Default value	Comments
stylesheet	The fully qualified URL to the XSL stylesheet that formats the search results.		If the output parameter value is `htmlsnippet`, the **stylesheet** value is ignored. This value must be URL-escaped. The XSL style sheet that you specify is not processed on the search engine server. The client application must apply the transformation rules that are found in the XSL stylesheet to the Atom feed search results. The client application can simply be an XSLT-compliant Web browser, feed reader, or your own custom XSLT application.

Search results

The `search` API supports search results as Atom 1.0 feeds and HTML snippets.

Atom feeds
: You can customize the appearance of the feed in a browser by specifying an XSL stylesheet in the search request.

HTML snippets
: You can embed the HTML search results into an existing Web page. To do that, you can provide a search box that issues an **HTTP GET** request to the server:

The `search` API also provides a service interface that returns an OpenSearch description document and enables client applications to discover the IBM OmniFind Yahoo! Edition search interface.

If an error occurs during the search request, a message is returned containing the error message ID and a description of the error.

Related reference

"Error responses" on page 25

Atom feeds

You can request search results to be returned as an Atom feed.

For information about Atom 1.0, see The Atom Syndication Format at http://atompub.org/rfc4287.html. IBM OmniFind Yahoo! Edition uses OpenSearch 1.0 data formats to extend the Atom feed format with extra metadata that is needed to return search results. For more information about OpenSearch 1.0, see OpenSearch response elements at http://www.opensearch.org.

The following table describes the elements that are returned in the `search` API results:

Table 2. Atom and OpenSearch elements and returned API results

Elements and attributes	Description
/feed	The container element for metadata and data that is associated with the search results feed.

Table 2. Atom and OpenSearch elements and returned API results (continued)

Elements and attributes	Description
/feed/title	Value: `Search results for query 'query' on collection collection_name`
/feed/link@href	If the `rel` attribute value in the `href` element is: • `self`: The reference is to the URL that generated this feed. • `first`: The reference is to the first set of search results. • `previous`: The reference is to the previous set of search results relative to this set. • `next`: The reference is to the next set of search results relative to this set. • `last`: The reference is to the last set of search results. • `alternate`: The reference is to an alternative format for this set of search results. • `search`: Points to an OpenSearch description document. • `unconstrained` Reference is to a set of unfiltered search results. Search results might be filtered due to duplicate results or to exceeding a specified query evaluation time limit.
/feed/author/name	Value: `IBM OmniFind API Web Service`
/feed/id	The URL that the client application issued to generate this feed.
/feed/category	Conveys information about the collection that is associated with the search results.
/feed/category@term	The name of the collection that this search request was issued for (the collection parameter in the search request).
/feed/category@label	See the description for /feed/category@term. This attribute is used for display in feed readers.
/feed/updated	The date and time that the query was issued. The value is in UTC in the format: `YYYY-MM-DDThh:mm:ssZ`.
/feed/opensearch:totalResults	The total number of results for the submitted query.
/feed/opensearch:Query	Contains information about the query that was submitted by the user.
/feed/opensearch:Query@role	If the `role` attribute value is: • `request`: The `searchTerms` attribute value is the submitted query (only 1 per feed). • `correction`: The `searchTerms` attribute value represents a spelling suggestion. There can be 0 or more spelling suggestions in an Atom feed.
/feed/opensearch:Query@searchTerms	Represents the query that was submitted or represents a spelling suggestion for the submitted query that was returned by the search engine server.
/feed/opensearch:startIndex	The initial result number for the search results that are returned in this feed.
/feed/opensearch:itemsPerPage	The number of search results that are returned in this feed.
/feed/entry	Encompasses the information for a single search result.
/feed/entry/category@term	Exists for entries that represent a featured link rather than a text result. The attribute value is `featured link`.

Table 2. Atom and OpenSearch elements and returned API results (continued)

Elements and attributes	Description
/feed/entry/title	The result title.
/feed/entry/link	Defines a reference to the search result resource.
/feed/entry/link@rel	If the rel attribute value is: • alternate: The href value is the result document URI. • via: The href value is a cached version of the result document. There can be two link elements with a rel attribute value of via if the original document is not of type text/html. One link element represents the cached version of the original document. The second link element represents the HTML version of the document (the type attribute has a value of text/html). The link elements with the rel attribute of via exist only if caching is enabled.
/feed/entry/link@href	The URI link to document.
/feed/entry/link@type	The content type of the URI document link.
/feed/entry/link@hreflang	The language of the URI document link.
/feed/entry/opensearch:relevance	The document score.
/feed/entry/updated	The last modified date for the document. The value is in UTC in the format: YYYY-MM-DDThh:mm:ssZ.
/feed/entry/id	The document URI.
/feed/entry/summary	The summary that is generated by the search engine for this document.
/feed/entry/omnifind:*field*	The metadata value for searches on fields. Possible values for *field*: abstract, author, creator, description, doctype, fileext, keywords, language, owner, subject, title, url. This element is applied only when the search API parameter **oyeFieldFormat** is set to true.
/feed/entry/omnifind:field	The metadata value for fielded searches.
/feed/entry/omnifind:field@name	The name of the metadata field.

Atom feed sample

The following sample of Atom 1.0 search results shows what is returned by the search application for a query that searches for documents that contain the phrase "united nations" in the keywords, author, or creator metadata fields. In the search application, this query is:

```
keywords:"united nations" OR author:"united nations" OR creator:"united nations"
```

The URL-encoded format of this query is:

```
http://hostname:port/api/search?query=keywords%3A%22united+
 nations%22+OR+author%3A%22united+nations%22+OR+creator%3A%22united+nations%22
 &collection=Default
```

The output returned from this query is:

```
<?xml version="1.0" encoding="utf-8" ?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
      xmlns:omnifind="http://omnifind.ibm.yahoo.net/api/spec/1.0/">
  <title>Search results for query 'creator:"united nations" OR author:"united nations"'
  on collection Default</title>
  <link href="http://hostname:port/api/search?collection=
  Default&query=creator:%22united%20nations%22%20OR%20
  author:%22united%20nations%22" rel="self" type="application/atom+xml"/>
  <author>
 <name>IBM OmniFind API Web Service</name>
  </author>
  <id>http://hostname:port/api/search?query=keywords%3A%22
united+nations%22+40OR+author%3A%22united+nations%22+OR+creator%3A%22united+nations%22&collection=Default
  <category term="Default" label="Default" />
  <updated>2007-02-06T02:42:22Z</updated>
  <opensearch:totalResults>2</opensearch:totalResults>
  <opensearch:Query role="request" searchTerms="creator:"united nations"
  OR author:"united nations""/>
  <opensearch:startIndex>1</opensearch:startIndex>
  <opensearch:itemsPerPage>2</opensearch:itemsPerPage>
  <entry>
 <link href="http://unbisnet.un.org/" rel="alternate" type="text/html" hreflang="en" />
 <link href="http://hostname:port/search/?query=cache::http%3A%2F%2Funbisnet.un.org%2F&output=binary"
  rel="via" type="text/html" hreflang="en" />
 <opensearch:relevance>2.38</opensearch:relevance>
 <title type="html">UNBISnet - UN Bibliographic Information System</title>
 <updated>2006-02-06T19:21:05Z</updated>
 <id>http://unbisnet.un.org/</id>
 <summary type="html"><SPAN class="ellipsis">... </SPAN> Catalogue of <SPAN class="highlight">
  <SPAN class="hlTerm0">United Nations</SPAN></SPAN>(UN) documents and publications indexed by the
  UN Dag Hammarskjöld Library and the Library of the UN Office at Geneva. Also included are commercial
  publications and <SPAN class="ellipsis">... </SPAN></summary>
  <omnifind:creator type="html"><SPAN class="highlight"><SPAN class="hlTerm0">
  United Nations</SPAN></SPAN></omnifind:creator>
   <omnifind:author type="html">Authored by <SPAN class="highlight"><SPAN class="hlTerm0">
   United Nations</SPAN>
  </SPAN></omnifind:author>
  </entry>
  <entry>
 <link href="http://testresult.un.org/" rel="alternate" type="text/html" hreflang="en" />
 <link href="http://hostname:port/search/?query=cache::http%3A%2F%2Ftestresult.un.org%2F&output=
  binary" rel="via" type="text/html" hreflang="en" />
 <opensearch:relevance>2.08</opensearch:relevance>
 <title type="html">UN test result with only author matching</title>
 <updated>2006-02-06T19:21:05Z</updated>
 <id>http://testresult.un.org/</id>
 <summary type="html"><SPAN class="ellipsis">... </SPAN> Summary for a <SPAN class="highlight">
  <SPAN class="hlTerm0">United Nations</SPAN></SPAN>(UN) result <SPAN class="ellipsis">... </SPAN>
  </summary>
   <omnifind:author type="html"><SPAN class="highlight"><SPAN class="hlTerm0">United Nations</SPAN>
  </SPAN></omnifind:author>
  </entry>
</feed>
```

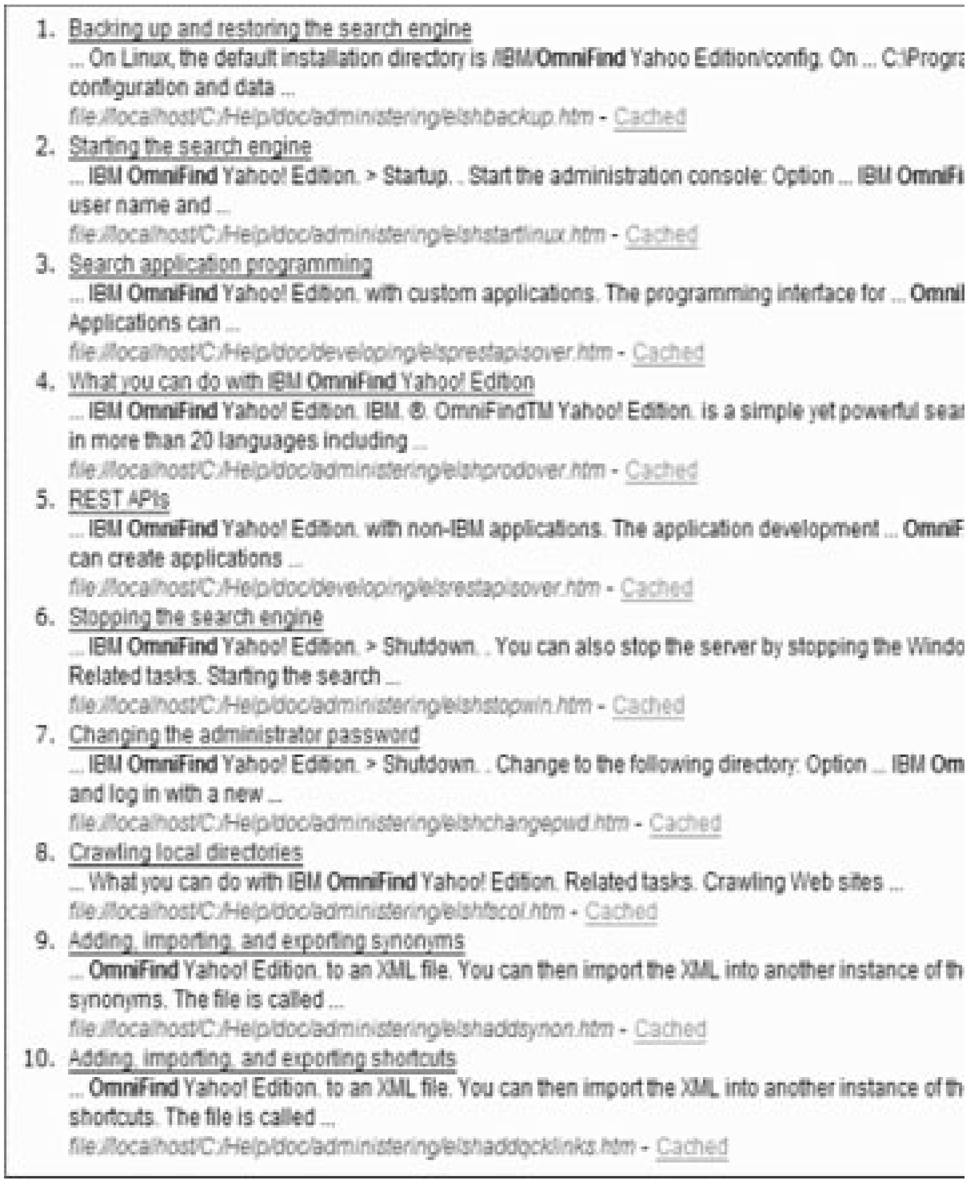
HTML snippets

You can request that search results be returned as HTML snippets.

An HTML snippet is different from a full HTML page in that it does not contain all the elements of a complete HTML page. There are no <HTML> or <BODY> tags. The snippet of HTML that is returned in the API search results is meant to be embedded within a full HTML page. If you want to add your own styles to the snippet, you need to parse the HTML yourself.

The following sample of HTML snippet shows the formatted search results that are returned by the search application for the request:

```
http://hostname:port/api/search?query=OmniFind&collection=Default
&start=0&results=10&output=htmlsnippet
```



1. Backing up and restoring the search engine
   ... On Linux, the default installation directory is /IBM/**OmniFind** Yahoo Edition/config. On ... C:\Progra
   configuration and data ...
   file://localhost/C:/Help/doc/administering/eishbackup.htm - Cached
2. Starting the search engine
   ... IBM **OmniFind** Yahoo! Edition. > Startup. . Start the administration console: Option ... IBM **OmniFi**
   user name and ...
   file://localhost/C:/Help/doc/administering/eishstartlinux.htm - Cached
3. Search application programming
   ... IBM **OmniFind** Yahoo! Edition. with custom applications. The programming interface for ... **Omnil**
   Applications can ...
   file://localhost/C:/Help/doc/developing/eisprestapisover.htm - Cached
4. What you can do with IBM **OmniFind** Yahoo! Edition
   ... IBM **OmniFind** Yahoo! Edition. IBM. ®. OmniFindTM Yahoo! Edition. is a simple yet powerful sea
   in more than 20 languages including ...
   file://localhost/C:/Help/doc/administering/eishprodover.htm - Cached
5. REST APIs
   ... IBM **OmniFind** Yahoo! Edition. with non-IBM applications. The application development ... **OmniF**
   can create applications ...
   file://localhost/C:/Help/doc/developing/eisrestapisover.htm - Cached
6. Stopping the search engine
   ... IBM **OmniFind** Yahoo! Edition. > Shutdown. . You can also stop the server by stopping the Windo
   Related tasks. Starting the search ...
   file://localhost/C:/Help/doc/administering/eishstopwin.htm - Cached
7. Changing the administrator password
   ... IBM **OmniFind** Yahoo! Edition. > Shutdown. . Change to the following directory: Option ... IBM **Om**
   and log in with a new ...
   file://localhost/C:/Help/doc/administering/eishchangepwd.htm - Cached
8. Crawling local directories
   ... What you can do with IBM **OmniFind** Yahoo! Edition. Related tasks. Crawling Web sites ...
   file://localhost/C:/Help/doc/administering/eishfscol.htm - Cached
9. Adding, importing, and exporting synonyms
   ... **OmniFind** Yahoo! Edition. to an XML file. You can then import the XML into another instance of th
   synonyms. The file is called ...
   file://localhost/C:/Help/doc/administering/eishaddsynon.htm - Cached
10. Adding, importing, and exporting shortcuts
   ... **OmniFind** Yahoo! Edition. to an XML file. You can then import the XML into another instance of th
   shortcuts. The file is called ...
   file://localhost/C:/Help/doc/administering/eishaddqcklinks.htm - Cached

OpenSearch description document

The OmniFind API web service provides a service interface for OpenSearch-compatible clients. The interface returns an introspection document that allows OpenSearch-compatible clients to discover the search interface.

An advantage of this interface is that client applications are not compelled to be hardcoded specifically to the IBM OmniFind Yahoo! Edition search interface.

Request format

Use the following request to retrieve the format for the OpenSearch description document:

```
http://hostname:8888/api/search/opensearchdescription
```

The OpenSearch description document is returned. For example:

```
<?xml version="1.0" encoding="utf-8"?>
<OpenSearchDescription xmlns="http://a9.com/-/spec/opensearch/1.1/"
          xmlns:omnifind="http://omnifind.ibm.yahoo.net/api/spec/1.0/">
  <ShortName>OmniFind</ShortName>
  <Description>API Web Service for the IBM OmniFind Enterprise Search Engine</Description>
  <Url type="application/atom+xml"
   indexOffset="0"
   template="http://<hostname:port>/api/search?query={searchTerms}&amp;results={count?}
    &amp;start={startIndex?}&amp;
   resultLang={language?}&amp;collection={omnifind:collection}&amp;queryLang={omnifind:queryLang?}&amp;
   locale={omnifind:locale?}"/>
  <Url type="text/html"
   indexOffset="0"
   template="http://<hostname:port>/api/search?query={searchTerms}&amp;results={count?}
    &amp;start={startIndex?}&amp;
   resultLang={language?}&mp;collection={omnifind:collection}&amp;queryLang={omnifind:queryLang?}&amp;
   locale={omnifind:locale?}&amp;output=htmlsnippet"/>
 <Query role="example"
   searchTerms="cat"
   omnifind:collection="Default"/>
 <Query role="example"
   searchTerms="cat OR mouse"
   omnifind:collection="Default"
   omnifind:stylesheet="http://my.server.com/stylesheets/atom.xsl"
   count="20"/>
 <!--result language-->
 <Language>ar</Language>
 <Language>cs</Language>
 ...
</OpenSearchDescription>
```

Guidelines

The OpenSearch description document is extended with the XML namespace http://omnifind.ibm.yahoo.net/api/spec/1.0/. The namespace prefix is `omnifind`. The extension is needed to define certain search request template parameters that are not defined in the core set of OpenSearch search parameter names.

The elements of interest are the `Url` and `Query` elements. Each `Url` element specifies a template attribute. The attribute value contains a search URL template for client applications. The OpenSearch description document contains the following templates:

- A template that returns an `application/atom+xml` type response for Atom feed responses.
- A template that returns a `text/html` type response for HTML snippet responses.

For each `Url` element, the `indexOffset` attribute value is set to `0`. This is done because the first search result is numbered 1 by the OpenSearch default setting. The OpenSearch document description overrides the default value because OmniFind Yahoo! Edition uses a starting value of `0` for search results.

Each `Query` element contains example queries that can be performed by search clients. The example queries use the defined custom namespace prefix, `omnifind`. One query example issues a search request for the keyword `cat` in the Default collection. The second query example issues a search request for the keywords `cat OR mouse` in the Default collection. The stylesheet available at http://my.server.com/stylesheets/atom.xsl is used to format the results, and 20 results are returned in each response.

For more information about the syntax and semantics of the OpenSearch search description document, see http://www.opensearch.org/Specifications/OpenSearch/1.1.

Add and delete document APIs

The client application can use the APIs to add documents to or delete documents from a collection.

The API requests to add and delete documents are standard HTTP requests. The requests are secured with HTTP basic authentication. The user ID value is ignored. You can get the API password from the administration console. Contact the search administrator or, from the administration console Manage Collections page, click **Change Password**.

The responses for document APIs are standard HTTP response messages. If an error occurs in the request, the response message body contains details on the error. If the request is successful, the message body is empty.

Related tasks

"Generating API passwords" on page 27

Related reference

"Error responses" on page 25

"Java command line examples" on page 29

Add and delete document API request format

You can use **HTTP POST** requests to add documents to a collection and **HTTP DELETE** requests to delete documents from a collection.

Add document request

The `addDocument` API is an **HTTP POST** request. It adds or replaces a document in the specified collection. This request is synchronous. When the request returns, the document is successfully added to the collection, or an error message is returned.

The following example shows an `addDocument` request:

```
POST /api/document HTTP/1.1
     Host: hostname:port
     action: addDocument
     collection: Default
     docId: document1
     docType: application/x-mspowerpoint
     docLang: en
     lastModified: 2006-01-26T16:37:44-04:00
     Authorization: Basic OnY2eEdyQWM9
     Content-Length: 2048

[body here]
```

Documents that are added into a collection by the `addDocument` API cannot be tracked in the Document Status window in the administration console. Any error that occurs when the document is added is reflected in the HTTP response.

Also, if the `docId` value is not a valid URI, the document will not be a clickable result in the search results page.

Add metadata values to documents

If an administrator configured metadata fields for a collection, you can assign values to the metadata fields when you add documents to the collection. The `addDocument` request cannot define the metadata field type or attributes, which must be configured with the administration console, but the request can assign field values.

In the following example, an administrator configured two metadata fields named product and price:

```
name = "product"
type = "text"
name = "price"
type = "decimal"
```

The `addDocument` request can specify the metadata field names as additional parameters and assign values to the fields. The metadata field names are prefaced with the **X-** parameter to indicate that they are user-defined fields. For example:

```
POST /api/document HTTP/1.1
     Host: hostname:port
     action: addDocument
     collection: Default
     docId: document1
     docType: application/x-mspowerpoint
     docLang: en
     X-product: movie
     X-price: 19.99
     lastModified: 2006-01-26T16:37:44-04:00
     Authorization: Basic OnY2eEdyQWM9
     Content-Length: 2048

[body here]
```

To retrieve a list of all of the metadata fields that are available in a collection, use the `metadataFields` API. To retrieve a list of all the collections that are available to add documents to, use the `collectionsList` API

Delete document request

The `deleteDocument` API is an **HTTP DELETE** request. The request deletes a document from the specified collection. This request is synchronous. However, on return, it does not guarantee that the document is no longer searchable.

The following example shows a `deleteDocument` request:

```
DELETE /api/document HTTP/1.1
     Host: hostname:port
     action: deleteDocument
     collection: Default
     docId: document1
     Authorization: Basic OnY2eEdyQWM9
```

The time that is required for the document to be no longer searchable depends on the search server load when the delete request is issued.

Related reference

"Metadata fields API" on page 21

"Collection list API" on page 23

Add and delete document API request parameters

You can use various parameter options in your requests to add or delete documents.

The following table describes the supported parameters for document requests:

Table 3. Supported parameters for requests to add or delete documents

Parameter name	Description	Default value	Supported action	Comments
action	The action to be performed.		All actions	Required. Supported values: `addDocument` and `deleteDocument`.
collection	The name of the collection to be updated.		All actions	Required. This value must be UTF-8 encoded and URL-escaped.
Content-length	The size of the document body to be added, in bytes.		`addDocument`	Required. The value must be greater than or equal to zero bytes.
docId	The document identifier.		All actions	Required. If you want users to be able to click the search result to retrieve the document, the value must be a valid URI. This value must be URL-escaped.
docKnownLang	The known language of the message content (document content)	Determined by the server.	`addDocument`	The **docKnownLang** value is used to force the server to use the specified language as the document language. See the description of **locale** for supported values.
docLang	The fallback language of the message content (document content).	Determined by the server.	`addDocument`	The **docLang** value is used if the server cannot determine the document language, and there is no `docKnownLang` value specified. See the description of **locale** for supported values.
docType	The fallback type and subtype of the message content (document content).		`addDocument`	Required. If the server cannot determine the document type, the **docType** value is used for the document type. The format is of *type/sub-type*, for example `text/html`. See RFC1341 for valid values.
lastModified	The date and time that the document was last modified.	The date and time that the document is received.	`addDocument`	This value must be in the ISO-8601 format: *YYYY-MM-DD*T*hh:mm:ssTZD*. For example: `2006-01-26T16:37:44-04:00` or `2006-01-26T20:37:44Z`

Table 3. Supported parameters for requests to add or delete documents (continued)

Parameter name	**Description**	**Default value**	**Supported action**	**Comments**
locale	The client locale.	The server locale.	All actions	Returns messages in the language of the client locale. Supported values: `de_DE` - German; `en_US` - English; `es_ES` - Spanish; `fr_FR` - French; `hu_HU` - Hungarian; `it_IT` - Italian; `ja_JP` - Japanese; `ko_KR` - Korean; `nl_NL` - Dutch; `pl_PL` - Polish; `pt_PT` - Portuguese; `pt_BR` - Brazilian Portuguese; `sv_SE` - Swedish; `zh_CN` - Chinese (Simplified); `zh_TW` - Chinese (Traditional)
X-*field name*	The value for metadata field.		`addDocument`	Both the parameter name and parameter value must be UTF-8 encoded and URL-escaped.

Administration APIs

The client application can use the administration APIs to start and stop crawlers, obtain a list of all metadata fields that are configured for a collection, and obtain the names of all collections in the search system.

The administration API requests are standard HTTP requests. The requests are secured with HTTP basic authentication. The user ID value is ignored. You can get the API password from the administration console. Contact the search administrator or from the administration console, click **Manage System** → **Manage Authentication**.

The responses for document APIs are standard HTTP response messages. If an error occurs in the request, the response message body contains details on the error. If the request is successful, the message body is blank.

Related tasks

"Generating API passwords" on page 27

Crawler management API

Use the crawler management API to start or stop a crawler.

The request to start or stop a crawler is a standard **HTTP POST** request. The API uses HTTP basic authentication to secure the requests. The password value is the API token that is retrieved from the administration console. The user name value is ignored.

The HTTP request format is:

```
POST /api/admin HTTP/1.1
Host: hostname:port
action: action
locale: locale
collection: collection_name
crawlType: crawler_type
Authorization: Basic password
```

Table 4. Crawler management request parameters

Parameters	Comments
action	Required. The action to be performed. Supported values are `startCrawl` or `stopCrawl`.

Table 4. Crawler management request parameters (continued)

Parameters	Comments
locale	Optional. The client locale. The default value is the server locale. Supported values: `de_DE` - German `en_US` - English `es_ES` - Spanish `fr_FR` - French `hu_HU` - Hungarian `it_IT` - Italian `ja_JP` - Japanese `ko_KR` - Korean `nl_NL` - Dutch `pl_PL` - Polish `pt_PT` - Portuguese `pt_BR` - Brazilian Portuguese `sv_SE` - Swedish `zh_CN` - Chinese (Simplified) `zh_TW` - Chinese (Traditional)
collection	Required. The name of the collection that the crawler belongs to. The value should be UTF-8 encoded and URL-escaped.
crawlType	Required. The crawler type. Supported values are `file`, `jdbc`, or `web`.

manageCrawler tool

You can also use the **manageCrawler** tool to start and stop crawlers. For information about administering crawlers from the command line, enter `manageCrawler -?` on the search server command line or see the IBM OmniFind Yahoo! Edition administration documentation.

Start crawler example

This example uses the crawler management API to request that the File system crawler be started for the Default collection:

```
POST /api/admin HTTP/1.1
Host: http://JKEnterprises.com:8888
action: startCrawl
collection: Default
crawlType: file
Authorization: Basic 6eKvCms=
```

To create this same request from the command line, you might enter the following command:

```
manageCrawler -h http://JKEnterprises.com:8888 -a start -c Default -t file
-p ″6eKvCms=″ -o output.txt
```

Stop crawler example

This example uses the crawler management API to request that the Web crawler be started for the Employees collection, using the French locale:

```
POST /api/admin HTTP/1.1
Host: http://JKEnterprises.com:8888
action: stopCrawl
locale: fr_FR
collection: Employees
crawlType: web
Authorization: Basic 6eKvCms=
```

To create this same request from the command line, you might enter the following command:

```
manageCrawler -h http://JKEnterprises.com:8888 -a stop -l fr_FR -c
Employees -t web -p ″6eKvCms=″ -o output.txt
```

Metadata fields API

Use the `metadatafields` API to retrieve a list of all of the metadata fields that are configured for a collection.

Request format

You can use the `metadatafields` API with search requests to determine what fields available to search. You can also use the `metadatafields` API with `addDocument` requests to determine what fields available for setting metadata field values.

The request to retrieve the names of all metadata fields in a collection is a standard **HTTP GET** request. The `metadatafields` request is formed by combining the following properties:

- Host name
- Port
- Path
- Request parameters, a collection of name-value pairs () that are separated by ampersand (&) characters

The host name is the host name of the search engine server. The port is the port number for the search application, if you are using the API to determine the metadata fields to search, or the port number for administration application, if you are using the API to determine metadata fields when you add documents to a collection.

The path to send your request to is one of the following:

- `/api/search/metadatafields`. This request returns the User-Defined, Pre-Defined, and Built-In fields (all of the fields that are available for search).
- `/api/document/metadatafields`. This request returns only the User-Defined fields because these are the only fields that client applications can set values for when adding documents.

Response format

The response to a `metadatafields` request is in XML format. For each metadata field in the collection, the response includes the field name and the field type. If any attributes are configured for the field, such as whether the field can be searched by field name or whether the field value can be shown in the search results, the response also includes the attribute data.

Request parameters

Table 5. Get metadata fields request parameters

Parameters	Comments
collection	Required. The name of the collection that you want to retrieve metadata field names from. The value should be UTF-8 encoded and URL-escaped.
locale	Optional. The client locale. The default value is the server locale. Supported values: `de_DE` - German `en_US` - English `es_ES` - Spanish `fr_FR` - French `hu_HU` - Hungarian `it_IT` - Italian `ja_JP` - Japanese `ko_KR` - Korean `nl_NL` - Dutch `pl_PL` - Polish `pt_PT` - Portuguese `pt_BR` - Brazilian Portuguese `sv_SE` - Swedish `zh_CN` - Chinese (Simplified) `zh_TW` - Chinese (Traditional)

Example request and response

The following request retrieves the names of all metadata fields available for search that are configured for the Sample collection:

```
http://JKEnterprises.server.com:8888/api/search/
metadatafields?collection=Sample
```

The example response shows that two metadata fields (price and product) are configured for the Sample collection:

```
<?version = 1.0 encoding="UTF-8"?>
<fields version="1.0">
<collection>Sample</collection>
<field>
   <name>author</name>
   <type>text</type>
</field>
<field>
   <name>doctype</name>
   <type>text</type>
</field>
<field>
   <name>docdate</name>
   <type>date</type>
</field>
...
</fields>
```

Related reference

"Add and delete document API request format" on page 15

Collection list API

Use the `collections` API to retrieve the names of all of the collections that exist in your search system.

The request to retrieve the names of all collections is a standard **HTTP GET** request. The `collections` request is formed by combining the following properties:

- Host name
- Port
- Path

The host name is the host name of the search engine server. The port is the port number for the search application. The path to send your request to is always `/api/search/collections`. You can use the `collections` API with both search and document requests when you determine which collection you want to take an action on.

For example, the following request obtains the names of all collections:

```
http://JKEnterprises.server.com:8889/api/search/collections
```

The response is in XML format. This example shows that the search system has two collections named Marketing and Sales. The response indicates that the Marketing collection is enabled for search and is the default collection on the search server. The Sales collection is not enabled for search (an administrator can specify whether a collection is available for search).

```
<?xml version="1.0" encoding="UTF-8"?>
<collections version="1.0">
  <collection enabled="true" default="true">
     <name>Marketing</name>
  </collection>
  <collection enabled="false">
     <name>Sales</name>
  </collection>
</collections>
```

Related reference

"Add and delete document API request format" on page 15

Error responses

An error response is returned for an unsuccessful API request.

The error responses for API requests are standard HTTP response codes. The HTTP response body contains error messages, each of which contains the ID and detailed description of the error.

All API requests return errors in XML format. The `search` API can also return errors in HTML snippet format, depending on the value of the **output** parameter in the search request.

The following sample shows an XML formatted error response:

```
<APIResponse version="1.0">
  <Error>
   <Message>
      <Id>IQQR0016E</Id>
         <Text>The search API request cannot be processed.</Text>
   </Message>
   <Message>
      <Id>IQQS0032E</Id>
         <Text>The query cannot be processed because it has incorrect
          syntax.
         </Text>
   </Message>
  </Error>
</APIResponse>
```

Only the message text is displayed in the API error response. You can view the full message (with explanation and user response sections) in the product documentation.

Related reference

"Search requests" on page 3

"Search results" on page 7

"Add and delete document APIs" on page 15

HTTP response codes

Standard HTTP error response codes indicate the general type of error that occurred. The HTTP body contains additional details about the error.

The following table below maps the HTTP error response codes to the associated error condition.

Table 6. HTTP response codes and situations when the error might occur

Error code and name	Error situation	Examples of error situations
400 - Bad Request	The input provided in the request body does not comply with the expected format or expected valid values.	The client does not include the required **collection** parameter in the search request or the client specifies an invalid collection name.

Table 6. HTTP response codes and situations when the error might occur (continued)

Error code and name	**Error situation**	**Examples of error situations**
401 - Unauthorized	When the request is processed, an access control check performed by the REST API service implementation fails.	An invalid API password is provided in the HTTP request to add a document to the collection.
404 - Not Found	1. The URI provided in the request, including parameters, does not match any of the URIs specified in the REST API interface. 2. A syntactically correct URI addresses a resource that cannot be found by the REST API service implementation.	1. A path element or parameter name might contain a typographical error. 2. A URL that was saved as a bookmark in the browser points to a resource that was deleted.
405 - Method Not Allowed	The REST API service does not support the operation implied by the HTTP method for the resource addressed by the URI that is provided in the request.	A PUT request on a URI that defines only GET and POST commands in the REST API interface.
500 - Server Error	An exception occurs internally during request processing that is based on an incorrect setup.	This situation might occur during test periods, but should not occur in a production environment.

Generating API passwords

You need an API password to use the administration APIs.

To get the API password, contact the search administrator. An API password is displayed in the Manage Authentication page in the administration console.

To generate a new API password, contact the search administrator. If you can access the administration console, follow these steps to generate a new API password:

1. From the administration console, click **Manage System** → **Manage Authentication**.
2. In the Manage Authentication window, click **Generate New API Password**.
3. Copy and paste the API password to your application code.

If you generate a new password, the old API password is invalid for existing applications that use the administration APIs. If your application cannot access the search system, ensure that the API password in the application matches the API password that is displayed in the administration console.

Related reference

"Add and delete document APIs" on page 15

"Administration APIs" on page 19

Java, XSL, and PHP examples

You can use the provided Java™, XSL, and PHP examples to create custom search applications.

Java, XSL, and PHP examples are in the *INSTALL_ROOT*/examples directory.

A Java software development kit (SDK) is not provided with the search engine. Do not develop applications by using the included Java Virtual Machine. The included Java Virtual Machine contains only the Java Runtime Environment.

Java command line examples

You can use the provided Java API examples to help build a custom Java search application.

Java API examples and the associated Java class files are provided in the *INSTALL_ROOT*/examples/java/commandline directory, where INSTALL_ROOT is the IBM OmniFind Yahoo! Edition installation directory. To run a Java example, use the command line to navigate to the *INSTALL_ROOT*/examples/java directory.

Before you run any of the command line examples, add whitney_core.jar to your CLASSPATH statement. The whitney_core.jar file is in the *INSTALL_ROOT*/lib directory.

Search

The **Search** command line example runs a search and returns the search results as an Atom feed, which is displayed in the command line window. If a local XSL file is specified as an argument, the XSL stylesheet is applied to the returned Atom feed, and the formatted result is also displayed in the command line window.

The usage statement is:

```
Search hostname port collection_name query
local_XSL_file_path
```

For example:

```
commandline.Search localhost 8080 Default NFL
"C:\\Program Files\\IBM\\OmniFindYahooEdition\\examples\\xsl\\atom2text.xsl"
```

AddDocument

The **AddDocument** command line example adds a document to the collection.

The usage statement is:

```
AddDocument hostname port collection_name document_ID
local_file mime_type username password
```

For example:

```
commandline.AddDocument localhost 8080 Default "My MS Word Document"
"C:\\temp\\My Document.doc" application/msword admin "fhWJhgo="
```

DeleteDocument

The **DeleteDocument** command line example deletes a document from the collection.

The usage statement is:

```
DeleteDocument hostname port collection_name document_ID
username password
```

For example:

```
commandline.DeleteDocument localhost 8080 Default "My MS Word Document"
admin "fhWJhgo="
```

Related reference

"Add and delete document APIs" on page 15

XSL style sheet example

XSL style sheets define standard formatting for the display of XML output, such as an Atom feed.

The XSL style sheet example file is in the *INSTALL_ROOT*/examples/xsl directory. The XSL style sheet example transforms an Atom feed into a text format.

PHP search application example

You can use the provided PHP example to create a custom PHP search application.

The PHP search application example is in the *INSTALL_ROOT*/examples/php directory.

To run the example application, you must have PHP and a PHP-compatible Web server installed on your system. After these components are installed, create a context root directory for the PHP search application in the Web server root directory. For example, create an `OYE` directory in the Web server root directory. Then copy and paste the contents of the *INSTALL_ROOT*/examples/php directory into the new `OYE` directory. Edit `search.php` to change the variable *$oyeUrl* to the URL for your IBM OmniFind Yahoo! Edition system.

The PHP search application example includes a style sheet, two images, and two PHP files, `search.php` and `oye.php`. The style sheet contains CSS classes that control the appearance of the PHP search application example. The PHP search application uses two image files, `fp_bg.png` and `front-page-header.png`, in the application page banner. The file `search.php` contains the HTML code to display the search form and the search results. The file `oye.php` contains functions to perform a search by using the search REST API and to process the results.

For example, if you want to display featured links in your PHP search application, run a search by using `search($queryString)`, then pass the *$feed* variable to the `getFeaturedLinks($feed)` function.

The following functions are available in the `oye.php` include file:

search($queryString)
Returns an object that points to the beginning of the XML data.

getTotalResults($feed)
Returns the total number of results that is expressed as an integer.

getSearchTerms($feed)
Returns the string of search terms.

getSpellCorrections($feed)
Returns an array of strings that represent the spelling corrections.

getStartIndex($feed)
Returns the first result that is expressed as an integer.

getItemsPerPage($feed)
Returns the number of search results to display per page expressed as an integer value.

getSearchResults($feed)
Returns an array of result objects that represent the search results.

getFeaturedLinks($feed)
Returns an array of result objects that represent the featured links.

Notices

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web

Trademarks

Index

Printed in USA