IBM DB2 Information Integrator
OmniFind Edition

Programming Guide and API Reference for Enterprise Search

Version 8.2

SC18-9284-00

IBM DB2 Information Integrator
OmniFind Edition

Programming Guide and API Reference for Enterprise Search

Version 8.2

SC18-9284-00

Before using this information and the product it supports, be sure to read the general information under ″Notices.″

You can order IBM publications online or through your local IBM representative:

- To order publications online, go to the IBM Publications Center at www.ibm.com/shop/publications/order
- To find your local IBM representative, go to the IBM Directory of Worldwide Contacts at www.ibm.com/planetwide

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Chapter 1. About this guide

This guide describes how to use the Java™ application programming interfaces (APIs) provided with enterprise search. The APIs provide tools for creating a search application.

This guide includes:

- Introduction to enterprise search applications
- Documentation of the API components
- Code samples

DB2® Information Integrator OmniFind Edition provides a technology called enterprise search. The enterprise search components are installed when you install IBM® DB2 Information Integrator OmniFind Edition (DB2 II OmniFind). The term *enterprise search* is used in the documentation for DB2 II OmniFind unless references are made to CD labels or specific product components.

Who should use this guide

This guide is for application programmers who want to create customized enterprise search applications.

You should have the following skills:

- Familiarity with application programming
- Experience coding Java applications

Where to find more information

See the following resources for more information.

Product information

Information about DB2 Information Integrator OmniFind Edition is available by telephone or on the Web.

If you live in the United States, you can call one of the following numbers:

- To order products or to obtain general information: 1-800-IBM-CALL (1-800-426-2255)
- To order publications: 1-800-879-2755

On the Web, go to www.ibm.com/software/data/integration/db2ii/support.html. This site contains the latest information about:

- The technical library
- Ordering books
- Client downloads
- Newsgroups
- Fix packs
- News
- Links to Web resources

Chapter 2. Enterprise search APIs

The enterprise search APIs are a set of Java APIs for adding documents, removing documents, and searching in enterprise search collections.

API descriptions

IBM search and index API
Use the IBM search and index API (SIAPI) to build custom search applications. The enterprise search implementation of SIAPI allows for remote access to the Search Server. These APIs allow you to submit search requests, process search results, and browse taxonomy trees.

Data listener API
Use the data listener API to add or remove documents from enterprise search collections.

The data listener is a component of enterprise search that receives data from external data source crawlers. Data source crawlers connect to the data listener and push data to the targeted collection.

The data listener component can also receive documents that are added to the enterprise search queue and are then crawled internally before data is pushed to the targeted collection.

Security

The enterprise search and Browse API's communicate remotely to the ESSearchServer Enterprise Application that is installed on each Search Node's WebSphere® Application Server.

In this secure mode, the WebSphere Application Server will challenge all HTTP requests to supply a valid user name and password. The user name and password entered must be valid within the active user registry that is configured through the WebSphere Administration console. Any requests that do not contain valid user credentials will be rejected.

The enterprise search SIAPI implementation will automatically set the user name and password values for you. You can specify the user name and password values when you create an instance of the ApplicationInfo class in your application. See "Minimum required application" on page 56 for an example.

The search application names and passwords must be stored in the same repository that is used for WebSphere authentication.

Restriction:

Search API restrictions:

- Supports HTTP BASIC authentication
- No support for HTTPS (SSL v2 or v3)

Chapter 3. Search and Index API (SIAPI)

The IBM search and index API (SIAPI) is a programming interface that enables you to search and browse collections and taxonomies.

The SIAPI provides a unified programming interface that enables you to write one program that searches different IBM back-end search products.

The SIAPI supports such tasks as:

- Searching indexes
- Customizing the information that is returned in search results sets
- Searching and browsing taxonomies

Structure of a SIAPI application

An SIAPI application consists the following tasks:

- Obtains an SIAPI implementation factory object
- Obtains a SearchService object
- Obtains a Searchable object
- Issues queries
- Processes query results

Obtaining an SIAPI implementation factory object

An SIAPI-based search application begins by obtaining an implementation factory object.

```
SearchFactory factory =
SiapiSearchImpl.createSearchFactory
    ("com.ibm.es.api.search.RemoteSearchFactory");
```

The SIAPI is a factory-based Java API. All of the objects that are used in your search application either are created by calling SIAPI object-factory methods or are returned by calling methods of factory-generated objects. You can easily switch between SIAPI implementations by loading different factories.

The enterprise search SIAPI implementation is provided by the com.ibm.es.api.search.RemoteSearchFactory class.

Obtaining a SearchService object

Use the factory object to obtain a SearchService object. The SearchService object allows you to access searchable collections on the search server.

The SearchService object needs to be configured with the host name and port of the enterprise search search server, and with the required locale for receiving error messages. The locale is a either a 4 character or 5 character Java string. For example, ″enUS″ or ″en_US″ for English.

Configuration parameters are set in a java.util.Properties. The parameters are then passed to the getSearchService factory method that generates the SearchService object.

```
Properties configuration = new Properties();
configuration.setProperty("hostname", "es.mycompany.com");
configuration.setProperty("port", "80");
configuration.setProperty("locale", "en_US");
SearchService searchService =
 factory.getSearchService(configuration);
```

Obtaining a Searchable object

Use the SearchService object to obtain a Searchable object. A Searchable object is associated with a searchable collection on the search server. A Searchable object enables you to issue queries and get information about the associated collection. Each enterprise search collection has an ID.

When you request a Searchable object, you need to identify your application by using an application ID. Contact your enterprise search administrator for the appropriate application ID.

If the search server is configured with global security turned on, you will need to provide a password. The password is used to authenticate your application. For more information, see "Security" on page 3.

```
ApplicationInfo appInfo = factory.createApplicationInfo("my_application_id");
appInfo.setPassword("my_password");
Searchable searchable =
 searchService.getSearchable(appInfo, "some_collection_id");
```

Call the getAvailableSearchables method to obtain all of the Searchable object that are available for your application.

```
Searchable[] searchables =
 searchService.getAvailableSearchables(appInfo);
```

For more information about Searchable objects, see "Searchable interface" on page 23.

Issuing queries

When you have obtained a Searchable object you can issue a query against that Searchable object. To issue a query against the Searchable object:

- Create a Query object.
- Customize the Query object.
- Submit the Query object to the Searchable object.
- Get the query results, encapsulated in a ResultSet object.

```
String queryString = "big apple";
Query query = factory.createQuery(queryString);
query.setRequestedResultRange(0, 10);
ResultSet resultSet = searchable.search(query);
```

For more information, see "Controlling query behavior" on page 7 and "Query syntax" on page 8.

Processing query results

The ResultSet and Result interfaces enable you to access query results.

```
Result[] results = resultSet.getResults();
for ( int i = 0 ; i < results.length ; i++ ) {
 System.out.println
( "Result " + i + ": " + results[i].getDocumentID()
  + " - " + results[i].getTitle() );
}
```

The SIAPI has a variety of methods for interacting with the "ResultSet interface" on page 44 and individual "Result interface" on page 41 objects.

Controlling query behavior

The following methods that belong to the Query interface allow you to control all aspects of the query behavior, including how the query is processed and what metadata is returned with each result.

Table 1. Query behavior methods

Method	Description
setQueryLanguage (String language)	Specifies to use a language other than the collection default language on the search server. For example, for English, the query language parameter is en-US. For Chinese, use zh-CN for Simplified Chinese and zh-TW for Traditional Chinese.
setRequestedResultRange (int fromResult, int numberOfResults)	Controls the range of the returned results. The **fromResult** value controls which ranked document your result set starts from. For example, a value of 0 means that you are requesting the first document in the query results (**fromResult** must be either 0 or an exact multiple of numberOfResults). The **numberOfResults** value controls how many results to return in the current page of results. The maximum is 100.
setReturnedAttribute (int attributeType, boolean isReturned)	Enables or disables any of the predefined result attribute values that are returned with each Result object. By default, enterprise search returns all the predefined result attribute values, except for the metadata fields attribute (RETURN_RESULT_FIELDS).
setReturnedFields (java.lang.String[] fieldNames)	Controls which metadata fields are returned in the Result object. By default enterprise search does not return any metadata fields.
setSiteCollapsingEnabled (boolean value)	Specifies if the top results contain more than two results from the same Web site. For example, if a particular query returned 100 results from www.ibm.com and site collapsing were enabled, the ResultSet would only contain 2 of those results in the top results. The other results from that site appear only after results from other sites are listed. To retrieve more results from that same site, you can reissue the same query with the site www.ibm.com added to the query string.

Table 1. Query behavior methods (continued)

Method	Description
setSortKey (String sortKey)	Specifies the sort key. The following predefined sort key values are defined in com.ibm.siapi.search.BaseQuery: • SORT_KEY_NONE • SORT_KEY_DATE • SORT_KEY_RELEVANCE You can specify the sort key to be any other valid numeric field name for the collection that is being searched. The default sort key is SORT_KEY_RELEVANCE.
setSortOrder (int sortOrder)	Specifies the sort order as SORT_ORDER_ASCENDING or SORT_ORDER_DESCENDING. The sort order is ignored if the sort key is SORT_KEY_RELEVANCE or SORT_KEY_NONE.
setSortPoolSize (int poolSize)	Controls how many of the top relevant results will be sorted and returned in the result set. Values range from 1 to 500 (the default sort pool size is 500). Any other values will cause a SiapiException to be thrown by the search server. The sort pool size is ignored if the sortKey is SORT_KEY_RELEVANCE or SORT_KEY_NONE.
setPredefinedResultsEnabled (Boolean value)	Specifies whether query results contain predefined results in addition to the regular results. Predefined links are enabled by default.
setSpellCorrectionEnabled (Boolean value)	Specifies whether query results contain suggestions for spelling corrections for the query. Spell correction is disabled by default.
setResultCategoriesDetailLevel (int detailLevel)	Specifies the required category detail level for query results. This method is used if the categories attribute (RETURN_RESULT_CATEGORIES) is enabled. The default value is RESULT_CATEGORIES_ALL.

Query syntax

Enterprise search uses the following syntax to process queries.

Search characters

Enterprise search syntax uses special characters to refine search queries.

Character	Usage
+	Precede a term with a plus sign (+) to indicate that a document must contain the term for a match to occur. Type the plus sign immediately before the term. **Query:** `+computer +software` **Result:** This query returns documents that include the term `computer` and the term `software`.

Character	Usage
-	Precede a term with a minus sign (-) to indicate that the term must be absent from a document for a match to occur. Type the minus sign immediately before the term. **Query:** `computer -hardware` **Result:** This query returns documents that include the term `computer` and not the term `hardware`.
^	Precede a term with the circumflex (^) sign to indicate that a document must contain the term (in the query), in addition to at least one more term that is not preceded by a circumflex (^) sign. **Query:** `^computer science software` **Result:** This query returns documents that include the term `computer` in the following variations: `computer science`, `computer software`, or `computer science software`.
( )	Use parentheses ( ) to indicate that a document must contain one or more of the terms within the parentheses for a match to occur. Do not use plus (+) or minus (-) signs within the parentheses. Use OR or vertical bar (|) to separate between the terms in parentheses. **Query:** `(computer OR software)` or `(computer | software)` **Result:** This query finds documents that include the term `computer`, the term `software`, or both terms `computer software`.
″ ″	Use quotation marks (″) to indicate that a document must contain the exact phrase within the quotation marks for a match to occur. **Query:** `"computer software programming"` **Result:** This query finds documents that include the exact term `computer software programming`.
~	Precede a term with a tilde sign (~) to indicate that the document can contain any term that has the same linguistic base form (also known as lemma or stem) as the term, for a match to occur. **Query:** `"~apples"` **Result:** This query finds documents that include the term `apples` or `apple` because apple is the base form of apples.

Character	Usage
=	Precede a term with the equal sign (=) to indicate the document must contain an exact match of the term, for a match to occur. **Query:** `"=apples"` **Result:** This query finds documents that include the term `apples`, but not documents that include the term `apple`.
site:*text*	If you search a collection that contains Web content, use the **site** keyword to search a specific domain. For example, you can return all pages from a particular Web site. Do not include the prefix http:// in a site query. **Query:** `+laptop site:www.ibm.com` **Result:** This query finds all documents on the `www.ibm.com` domain that contain the word `laptop`.
url:*text*	If you search a collection that contains Web content, use the **URL** keyword to find documents that contain specific words anywhere in the URL. **Query:** `URL:support` **Result:** This query finds documents with the value support anywhere in the URL, such as `http://www.ibm.com/support/fr/`.
link:*text*	If you search a collection that contains Web content, use the link keyword to find documents that contain at least one **link** to a specific Web page. **Query:** `link:http://www.ibm.com/us` **Result:** This query finds all documents that include one or more links to the page `http://www.ibm.com/us` .
field:*text*	If the documents in a collection include fields (such as the columns in a relational database), and the collection administrator made those fields searchable by field name, you can query specific fields in the collection. **Query:** `lastname:smith div:software` **Result:** This query returns all documents about employees with the last name Smith (`lastname:smith`) who works for the Software division (`div:software`).
docid:*documentid*	Use the docid keyword to find documents that have a specific URL or document ID. **Query:** `docid:http://www.ibm.com/solutions/us/` or `docid:http://www.ibm.com/products/us/` **Result:** This query finds all documents whose URL is `docid:http://www.ibm.com/solutions/us/` or `docid:http://www.ibm.com/products/us/`.

Character	Usage
taxonomy_ID::*category_ID*	To search a collection that contains categories generated by one of the enterprise search categorizers, use category terms to find documents that belong to specific category in a specific categorizer taxonomy. **Query:** `scopes::research "computer science"` **Result:** This query finds all documents that belong to the `research` scope category and contain the phrase `computer science`.
$source::*source_type*	Use the $source keyword to find documents that come from a specific source type. Source queries are useful in collections that contain documents form multiple sources. **Query:** `$source::DB2 "computer science"` **Result:** This query finds DB2 documents that contain the phrase `computer science`.
$language::*language_id*	Use the $language keyword to find documents written in a specific language. **Query:** `$language::en "computer science"` **Result:** This query finds English documents that contain the phrase `computer science`.
$doctype::*document_type*	Use the $doctype keyword to find documents of a specific document format. **Query:** `$doctype:application/pdf "computer science"` **Result:** This query finds Adobe Acrobat documents that contain the phrase `computer science`.
#field::=value	Use the numeric constraint syntax to find documents that have a numeric field with a value equal to the number specified. **Query:** `#price:=1700 laptop` **Result:** This query finds documents that contain `laptop` with a price field with a value equal to 1700.
#field::>value	Use the numeric constraint syntax to find documents that have a numeric field with a value greater than the number specified. **Query:** `#price:>1700 laptop` **Result:** This query finds documents that contain `laptop` with a price field value greater than 1700.
#field::<value	Use the numeric constraint syntax to find documents that have a numeric field with a value less than the number specified. **Query:** `#price:<1700 laptop` **Result:** This query finds documents that contain `laptop` with a price field value less than 1700.

Character	Usage
#field::>=value	Use the numeric constraint syntax to find documents that have a numeric field with a value greater than or equal to the number specified. **Query:** `#price:>=1700 laptop` **Result:** This query finds documents that contain `laptop` with a price field value greater than or equal to 1700.
#field::<=value	Use the numeric constraint syntax to find documents that have a numeric field with a value less than or equal to the number specified. **Query:** `#price:<=1700 laptop` **Result:** This query finds documents that contain `laptop` with a price field value less than or equal to 1700.
#field::>value1<value2	Use the numeric constraint syntax to find documents within a specific range. **Query:** `#price:>1700<3900 laptop` **Result:** This query finds documents that contain `laptop` with a price field value greater than 1700 and less than 3900.
#field::>=value1<=value2	Use the numeric constraint syntax to find documents within a specific range. **Query:** `#price:>=1700<=3900 laptop` **Result:** This query finds documents that contain `laptop` with a price field value greater than or equal to 1700 and less than or equal to 3900.
#field::>value1<=value2	Use the numeric constraint syntax to find documents within a specific range. **Query:** `#price:>1700<=3900 laptop` **Result:** This query finds documents that contain `laptop` with a price field value greater than 1700 and less than or equal to 3900.
#field::>=value1<value2	Use the numeric constraint syntax to find documents within a specific range. **Query:** `#price:>=1700<3900 laptop` **Result:** This query finds documents that contain `laptop` with a price field value greater than or equal to 1700 and less than 3900.

Character	Usage
ACL constraints: (ACL tokens)	Access control constraints cannot be specified in the query string as other constraints, because of security considerations. Use the setACLConstraints(String aclConstraints) method of the Query interface to specify access control constraints for the query. Parentheses, the plus character (+), minus character (-) and circumflex (^) are allowed when specifying ACL constraints, and have the same meaning as in the regular query syntax. **Query:** `thinkpad ACL constraints: (john_m | dev_group)` **Result:** This query finds documents that include the word `thinkpad` and the ACL tokens `john_m` or `dev_group`.

Building Java source code

Before you can build your Java source code, you must install and configure Apache ANT, a Java-based build tool. For more information on how to install and configure Apache ANT, refer to http://ant.apache.org/.

To build your Java source code:

1. In the command line, navigate to one of the following directories:
   - `$install_root/samples/siapi` (for example, `/opt/IBM/es/samples/siapi`)
   - `$install_root/samples//datalistener` (for example, `/opt/IBM/es/samples//datalistener`)

   Each of these directories include a `build.xml` file, that ANT uses to build the file.
2. Type `ant` and press Enter.

You will receive the `ADD TEXT HERE` message after your Java source code compiles.

SearchExample class

The SearchExample class gives a simple example of the minimum requirements needed to submit a search to the enterprise search server.

Sample

```
public class SearchExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String queryString;
    private String collectionId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
     // obtain the OmniFind specific SIAPI Search factory implementation
        SearchFactory factory =
            SiapiSearchImpl.createSearchFactory("com.ibm.es.api.search.
     RemoteSearchFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
```

```
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Search Service implementation
        SearchService searchService =
            factory.getSearchService(config);

        // obtain a Searchable object to the specified
        // collection ID
        Searchable searchable = searchService.getSearchable(appinfo, collectionId);
        if(searchable == null) {
            System.out.println("Failed to get a searchable for collection:
    " + collectionId);
            return;
        }

        // create a new Query object using the specified
        // query string
        Query q = factory.createQuery(queryString);
                q.setQueryLanguage("en_US");

// execute the search by calling the Searchable's
        // search method.  A SIAPI ResultSet object will
        // be returned
ResultSet rset = searchable.search(q);
if(rset != null) {
            // get the array of results from the ResultSet
            Result r[] = rset.getResults();
            if(r != null) {
                // walk the results list and print out the
                // document identifier
                for(int k = 0; k < r.length; k++) {
                    System.out.println("Result " + k + ": " + r[k].getDocumentID());
 }
            }
        }
   }

   public static void main(String[] args) throws Exception {
        SearchExample sc = new SearchExample();
        if (args.length < 5) {
            System.out.println("Usage: SearchExample <queryString> <collection id>
    <application name>" +
                               "<application password> <hostname> <port>");
            System.out.println("Example Usage: SearchExample lotus col1 app1
    password localhost 80");
            return;
        }
```

```
        sc.queryString = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

AdvancedSearchExample class

The AdvancedSearchExample class is an example that uses the advanced query settings and result processing options.

Sample

```
public class AdvancedSearchExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String queryString;
    private String collectionId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
     // obtain the OmniFind specific SIAPI Search factory implementation
        SearchFactory factory =
            SiapiSearchImpl.createSearchFactory("com.ibm.es.api.search.
     RemoteSearchFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Search Service implementation
        SearchService searchService =
            factory.getSearchService(config);

        // obtain a Searchable object to the specified
        // collection ID
        Searchable searchable = searchService.getSearchable(appinfo, collectionId);
        if(searchable == null) {
            System.out.println("Failed to get a searchable for collection:
     " + collectionId);
            return;
```

```
        }

        // create a new Query object using the specified
        // query string
        Query q = factory.createQuery(queryString);

        // set the result range we want to access on this query
        q.setRequestedResultRange(0, 20);
                q.setQueryLanguage("en_US");
        // designate that we do not want the Description
        // field returned for each result
        q.setReturnedAttribute(Query.RETURN_RESULT_DESCRIPTION, false);

// execute the search by calling the Searchable's
        // search method.  A SIAPI ResultSet object will
        // be returned
ResultSet rset = searchable.search(q);
if(rset != null) {
           System.out.println("");
           System.out.println("Estimated results:
    " + rset.getEstimatedNumberOfResults());
           System.out.println("Available results:
    " + rset.getAvailableNumberOfResults());
           System.out.println("Evaluation time: " + rset.getQueryEvaluationTime());
           System.out.println("");
           Result pr[] = rset.getPredefinedResults();
           if(pr != null) {
               // walk the predefinedResults list and print out the
               // document identifier and document title
               for(int k = 0; k < pr.length; k++) {
                   System.out.println("PredefinedResult " + k + ":
       " + pr[k].getDocumentID());
                   System.out.println("\tTitle: " + pr[k].getTitle());
                   System.out.println("\tDescription: " + pr[k].getDescription());
               }
           }
           System.out.println("");
           SpellCorrection sc[] = rset.getSpellCorrections();
           if(sc != null) {
               // walk the list of returned spelling corrections
               // and print out the query sub string and the
               // spelling suggestions
               for(int k = 0; k < sc.length; k++) {
                       System.out.println("SpellCorrection " + k + ": " + sc[k].
         getQuerySubstring());
                       String[] corrections = sc[k].getSuggestions();
                       if(corrections != null) {
                               for(int s = 0; s < corrections.length; s++) {
                                       System.out.println("Suggestion " + s + ":
               " + corrections[s]);
                               }
                       }
               }
           System.out.println("");
           // get the array of results from the ResultSet
           Result r[] = rset.getResults();
           if(r != null) {
               // walk the results list
               for(int k = 0; k < r.length; k++) {
                   // print out all predefined field values.
                   // NOTE:  Description will be "null"
                   // since we modified the Query above
                   // to NOT return the Description field.
                   System.out.println("Result " + k + ": " + r[k].getDocumentID());
     System.out.println("\tScore: " + r[k].getScore() + "%");
                   System.out.println("\tTitle: " + r[k].getTitle());
```

```
        System.out.println("\tLanguage: " + r[k].getLanguage());
        System.out.println("\tType: " + r[k].getDocumentType());
                    System.out.println("\tSource: " + r[k].getDocumentSource());
                    System.out.println("\tDate: " + r[k].getDate());
        System.out.println("\tDescription: " + r[k].getDescription());
                    // walk any categories that this result belongs to
                    ResultCategory[] cats = r[k].getCategories();
                    if(cats != null) {
                     for(int s = 0; s < cats.length; s++) {
                            CategoryInfo info = cats[s].getInfo();
                            // print out the taxonomy id, the category id, and the
             category label
                            // for each ResultCategory returned
                            System.out.println("\tTaxonomy : " + cats[s].
              getTaxonomyID());
                            if(info != null) {
                                System.out.println("\tCategory " + s + ":
               " + info.getID());
                                System.out.println("\t\tLabel " + s + ":
               " + info.getLabel());
                            }
                     }
                    }
                    // print out any additional field names and their
                    // values (document metadata)
                    NameValuePair[] fields = r[k].getFields();
                    if(fields != null) {
                        for(int s = 0; s < fields.length; s++) {
                            NameValuePair nvp = fields[s];
                            if(nvp != null) {
                                System.out.print("\tField " + nvp.getName() + ":
               " + nvp.getValue());
                            }
                        }
                    }
        System.out.println("");
                  }
                }
            }
        }

        public static void main(String[] args) throws Exception {
            AdvancedSearchExample sc = new AdvancedSearchExample();
            if (args.length < 5) {
                System.out.println("Usage: AdvancedSearchExample <queryString>
         <collection id> <application name>" +
                                   "<application password><hostname><port>");
                System.out.println("Example Usage: AdvancedSearchExample lotus col1 app1
         password localhost 80");
                return;
            }
            sc.queryString = args[0];
            sc.collectionId = args[1];
            sc.applicationName = args[2];
            sc.applicationPassword = args[3];
            sc.hostname = args[4];
            sc.portNumber = args[5];
            sc.execute();
        }
    }
```

BrowseExample class

The BrowseExample class gives an example of accessing a collection's taxonomy tree and printing out some of the basic navigation properties

Sample

```
public class BrowseExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String collectionId;
    private String taxonomyId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
        // obtain the OmniFind specific SIAPI Browse factory implementation
 BrowseFactory factory =
  SiapiBrowseImpl.createBrowseFactory("com.ibm.es.api.browse.RemoteBrowseFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Browse service implementation
 BrowseService browseService = factory.getBrowseService(config);

        // get a TaxonomyBrowser forthe specified taxonomy id and collection id
        TaxonomyBrowser browser = browseService.getTaxonomyBrowser(appinfo,
   collectionId,
   taxonomyId);
        if(browser == null) {
            System.out.println("Failed to get a taxonomy for taxonomy id:
     " + taxonomyId +
                               " from collection: " + collectionId);
            return;
        }

        //Display the Taxonomy Info
        TaxonomyInfo taxonomyInfo = browser.getTaxonomyInfo();
        System.out.println("Taxonomy label: " + taxonomyInfo.getLabel());

        // get the root category
        Category rootCategory = browser.getRootCategory();

        // display the root category
        System.out.println("Root category label: " + rootCategory.getInfo().
    getLabel());
        System.out.println("child categories:");
        CategoryInfo[] childrenInfo  = rootCategory.getChildren();
        for (int i = 0; i < childrenInfo.length; i++) {
            System.out.println("\t" + childrenInfo[i].getLabel());
        }
```

```
        // Now get the root's first child category
        Category childCategory = browser.getCategory(rootCategory.getChildren()
   [0].getID());

        // Display the child category and it's path from root
        System.out.println("Root's first child's label: " + childCategory.getInfo()
   .getLabel());
        System.out.println("It's path from root is : ");
        CategoryInfo[] pathFromRoot = childCategory.getPathFromRoot();
        for (int i = 0; i < pathFromRoot.length; i++) {
            System.out.println("-->" + pathFromRoot[i].getLabel());
        }
    }

    public static void main(String[] args) throws Exception {
        BrowseExample sc = new BrowseExample();
        if (args.length < 5) {
            System.out.println("Usage: BrowseExample <taxonomy id> <collection id>
     <applicationname>");
System.out.println("Example Usage: BrowseExample tax1 col1 Default password
localhost 80");
            return;
        }
        sc.taxonomyId = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

SIAPI calls and methods

The SIAPI uses the following classes and interfaces.

SiapiVersion class

The SiapiVersion class manages the SIAPI version.

getSiapiVersion method

Returns the version of SIAPI.

Syntax

```
public String getSiapiVersion()
```

SiapiSearchImpl class

The SiapiSearchImpl class obtains a search factory for the enterprise search SIAPI implementation.

createSearchFactory method

Creates a search factory for an SIAPI implementation by the implementation class name.

Call this method with `com.ibm.es.api.search.RemoteSearchFactory`.

Syntax

```
public static SearchFactory createSearchFactory(String factoryImplClassName)
  throws SiapiException
```

Parameters

factoryImplClassName
Full name of the implementation factory class.

Returns

A search factory for an SIAPI implementation.

SearchFactory interface

The SearchFactory interface is the entry point to an SIAPI search implementation.

getSearchService method

Obtains a search service object for a certain application and a certain search server.

Syntax

```
public SearchService getSearchService(Properties config) throws SiapiException
```

Parameters

config
Enterprise search supports three configuration properties:

- hostname (required) - the hostname of the Enterprise search search server.
- port (optional) - the port number of the enterprise search search server (the default port number is 80).
- locale (optional) - the locale settings used for returning translated error messages (the default value is the default java locale).

Returns

A search service.

Throws

SiapiException if you cannot obtain the search service.

createQuery method

Creates a Query object with the specified query text.

The query text must comply with the query syntax that is supported in Enterprise Search. See "Query syntax" on page 8 for more information on creating a query.

Syntax

```
public Query createQuery(String queryText) throws SiapiException
```

Parameters

queryText
Text of the query.

Returns

The new query object.

createApplicationInfo method

Creates an application information object for the specified application ID. After you obtain an ApplicationInfo object, you can use the ApplicationInfo.setPassword(String) or the ApplicationInfo.setToken(String) to set the application's password or security-token.

Syntax

```
public ApplicationInfo createApplicationInfo(java.lang.String applicationID)
                                      throws SiapiException
```

Throws

SiapiException if the applicationID is invalid.

getVersion method

Returns the version of the specific SIAPI implementation. Version data can be managed differently in each SIAPI implementation. The value returned here can be different than the version of the SIAPI.

Syntax

```
public java.lang.String getVersion()
```

CategoryInfo interface

The CategoryInfo interface provides descriptive information on a category.

getID method

Returns the category ID.

Syntax

```
public java.lang.String getID()
```

getLabel method

Returns the category label.

Syntax

```
public java.lang.String getLabel()
```

CollectionInfo interface

The CollectionInfo interface provides descriptive information about a collection.

getCollectionInfo method

Returns the searchable's ID and label.

Syntax

```
public CollectionInfo getCollectionInfo()
```

Returns

A CollectionInfo interface that contains the collection ID and label.

getID method

Returns the collection ID.

Syntax

```
public java.lang.String getID()
```

getLabel method

Returns the collection label.

Syntax

```
public java.lang.String getLabel()
```

FieldInfo interface

The FieldInfo interface represents a field definition.

A field is a named text portion of a document's content or metadata.

getID method

Returns the field name.

Syntax

```
public java.lang.String getID()
```

getAvailableFields method

Returns information about the metadata fields that are defined for documents in this collection.

Syntax

```
public FieldInfo[] getAvailableFields()
```

Returns

Array of field information objects.

isContentSearchable method

Returns true if fields of this type are content-searchable.

A content-searchable field is a field with searchable text that uses non-fielded-search queries. For example, a query for the word `apple` will return any document with a content-searchable field that contains the word `apple`.

Syntax

```
public boolean isContentSearchable()
```

Returns

True if fields of this type are content-searchable; false otherwise.

isFieldSearchable method

Returns true if fields of this type are field-searchable.

Syntax

A field-searchable field is a field whose text is searchable by fielded-search queries. For example, a query for `keywords:apple` will return any document that has a field-searchable field keywords that contains the word `apple`.

```
public boolean isFieldSearchable()
```

Returns

True if fields of this type are field-searchable; false otherwise.

isReturnable method

Returns true if the content of fields of this type can be returned with search results.

For example, if a document has an `author` metadata field that is returnable, then if that document is returned as a query result, the content of the author field would be returned with the result.

Use the setReturnedFields method of the Query interface to specify the names of fields to be returned with query results. Use the getFields method to obtain the content of returned fields from Result objects.

Syntax

```
public boolean isReturnable()
```

Returns

True if content of fields of this type can be returned with search results; false otherwise.

isParametric method

Returns true if fields of this type are parametric.

A parametric field is a field whose value can be translated to a numeric value. See "Query syntax" on page 8 to create query constraints that restrict query results in the specified parametric field value ranges.

Syntax

```
public boolean isParametric()
```

Returns

True if fields of this type are parametric; false otherwise.

Searchable interface

The Searchable interface allows you to search a collection's index and provides information about the collection.

ATTRIBUTE_LANGUAGE constant

Pass this constant to the getAvailableAttributeValues method to retrieve the list of document languages which restricts queries to this Searchable

Syntax

```
public static final int ATTRIBUTE_LANGUAGE
```

ATTRIBUTE_SOURCE constant

Pass this constant to the getAvailableAttributeValues method to retrieve the list of document sources which restricts queries to this Searchable.

Syntax

```
public static final int ATTRIBUTE_SOURCE
```

ATTRIBUTE_DOCTYPE constant

Pass this constant to the getAvailableAttributeValues method to retrieve the list of document sources which restricts queries to this Searchable

Syntax

```
public static final int ATTRIBUTE_DOCTYPE
```

search method

Runs a query and returns the set of results.

Syntax

```
public ResultSet search(Query query) throws SiapiException
```

Parameters

query
Query that you want to search.

Returns

A ResultSet object that contains the search results.

Throws

SiapiException if the search operation could not be performed.

count method

Returns an exact count of all of the documents that match the specified query. The count includes documents that include all of the required query terms and none of the forbidden terms.

Syntax

```
public int count(Query query) throws SiapiException
```

Parameters

query
Query to be searched.

Returns

The number of documents that match the specified query.

Throws

SiapiException if the count operation could not be performed.

getSpellCorrections method

Returns a list of spell correction suggestions for the specified query string, or null if no suggestions are available.

If spell correction is enabled see the method "setSpellCorrectionEnabled method" on page 37 in the Query interface.

Syntax

```
public SpellCorrection[] getSpellCorrections()
```

Parameters

queryText
The query string to check.

Returns

Returns a list of spell correction suggestions for the specified query string.

getDefaultLanguage method

Retrieves the default language used by this searchable to process queries.

Syntax

```
public String getDefaultLanguage()
```

getAvailableAttributeValues method

Returns a list of values that you can use to create attribute query constraints of the specified type.

Syntax

```
public java.lang.String[] getAvailableAttributeValues(int attributeType)
```

Parameters

attributeType
The attribute type.

Returns

Returns the list of available values for the specified attribute type.

Note: For some attribute types, the returned list might not be exhaustive or might not exist. In these cases, a null value will be returned.

Valid attribute types are:

- ATTRIBUTE_DOCTYPE
- ATTRIBUTE_LANGUAGE
- ATTRIBUTE_SOURCE

For example, a call to getAvailableValues(Searchable.ATTRIBUTE_SOURCE) returns the list of document sources for this collection. For more information see, "Query syntax" on page 8.

getAvailableFields method

Returns information about the metadata fields that are defined for documents in this collection.

Syntax

```
public FieldInfo[] getAvailableFields()
```

Returns

Array of field information objects.

setProperty method

Sets the value of a searchable property.

Using these properties, you can modify the behavior of a searchable. To learn which properties are supported, call the "getProperties method."

Restriction: Enterprise search does not support any properties and will throw an exception when this method is called.

Syntax

```
public void setProperty(String name,
                        String value) throws SiapiException
```

Parameters

name
Name of the searchable property.

value
Value of the searchable property.

Throws

SiapiException if the property cannot be modified at this time, or if the value is illegal.

getProperty method

Returns the value of a searchable property.

Syntax

```
public String getProperty(String name)
```

Parameters

name
Name of the searchable property to retrieve.

getProperties method

Returns all of the searchable properties, or null if none exist.

Syntax

```
public Properties getProperties()
```

getCollectionInfo method

Returns the searchable's ID and label.

Syntax

```
public CollectionInfo getCollectionInfo()
```

Returns

A CollectionInfo interface that contains the collection ID and label.

getAvailableSearchables method

Obtains all of the available references to collections.

Syntax

```
public Searchable[] getAvailableSearchables(ApplicationInfo appInfo)
 throws SiapiException
```

Parameters

appInfo
The requesting application's authentication and authorization info.

Returns

Searchable interfaces for collections.

Throws

SiapiException if the action cannot be performed.

getSearchable method

Obtains a reference to a collection by the collection ID.

Syntax

```
public Searchable getSearchable(ApplicationInfo appInfo,
                                String collectionID) throws SiapiException
```

Parameters

appInfo
The requesting application's authentication and authorization info.

collectionID
ID of the collection.

Returns

A Searchable.

Throws

SiapiException if cannot obtain the searchable reference.

SearchService interface

Use this interface to obtain searchable objects.

getAvailableSearchables method

Obtains all of the available references to collections.

Syntax

```
public Searchable[] getAvailableSearchables(ApplicationInfo appInfo)
 throws SiapiException
```

Parameters

appInfo
The requesting application's authentication and authorization info.

Returns

Searchable interfaces for collections.

Throws

SiapiException if the action cannot be performed.

getSearchable method

Obtains a reference to a collection by the collection ID.

Syntax

```
public Searchable getSearchable(ApplicationInfo appInfo,
                                String collectionID) throws SiapiException
```

Parameters

appInfo
The requesting application's authentication and authorization info.

collectionID
ID of the collection.

Returns

A Searchable.

Throws

SiapiException if cannot obtain the searchable reference.

ApplicationInfo interface

The ApplicationInfo interface controls the information that is used to authorize access to the collections.

getID method

Returns the application ID.

Syntax

```
public java.lang.String getId()
```

getPassword method

Returns the password that is associated with the application.

Syntax

```
public String getPassword()
```

getToken method

Returns the token value that is associated with the application.

Syntax

```
public String getToken()
```

setPassword method

Sets the password that is associated with the application. Use this method in security environments that require authentication by password.

Syntax

```
public void setPassword(String password) throws SiapiException
```

Parameters

password
Password that you want to set.

Throws

SiapiException if the password is invalid.

setToken method

Sets the token that is associated with the application.

Use this method in security environments that require tokens to be passed, such as single sign-on environments.

Syntax

```
public void setToken(String token) throws SiapiException
```

Parameters

token
Token to be set

Throws

SiapiException if the token is invalid.

createApplicationInfo method

Creates an application information object for the specified application ID. After you obtain an ApplicationInfo object, you can use the ApplicationInfo.setPassword(String) or the ApplicationInfo.setToken(String) to set the application's password or security-token.

Syntax

```
public ApplicationInfo createApplicationInfo(java.lang.String applicationID)
                                      throws SiapiException
```

Throws

SiapiException if the applicationID is invalid.

createApplicationInfo method

Creates an application info object for the specified application ID.

After you obtain an ApplicationInfo object, you can use setPassword or setToken to set the application's password or security-token.

Syntax

```
public ApplicationInfo createApplicationInfo(String applicationID)
  throws SiapiException
```

Parameters

ID ID of the application.

Query interface

The Query interface controls query behavior.

createQuery method

Creates a Query object with the specified query text.

The query text must comply with the query syntax that is supported in Enterprise Search. See "Query syntax" on page 8 for more information on creating a query.

Syntax

```
public Query createQuery(String queryText) throws SiapiException
```

Parameters

queryText
Text of the query.

Returns

The new query object.

SORT_KEY_RELEVANCE constant

Pass this constant to the setSortKey method to sort results by relevance.

Syntax

```
public static final java.lang.String SORT_KEY_RELEVANCE
```

SORT_KEY_DATE constant

Pass this constant to the setSortKey method to sort results by date.

Syntax

```
public static final java.lang.String SORT_KEY_DATE
```

SORT_KEY_NONE constant

Pass this constant to the setSortKey method to get unsorted results.

Syntax

```
public static final java.lang.String SORT_KEY_NONE
```

SORT_ORDER_ASCENDING constant

Pass this constant to the setSortOrder method to sort results in ascending order.

Syntax

```
public static final int SORT_ORDER_ASCENDING
```

SORT_ORDER_DESCENDING constant

Pass this constant to the setSortOrder method to sort results in descending order.

Syntax

```
public static final int SORT_ORDER_DESCENDING
```

RESULT_CATEGORIES_ALL constant

Pass this constant to the setResultCategoriesDetailLevel method to set full result categories detail level.

Syntax

```
public static final int RESULT_CATEGORIES_ALL
```

RESULT_CATEGORIES_NO_PATH_TO_ROOT constant

Pass this constant to the setResultCategoriesDetailLevel method to set partial result categories detail level.

Syntax

```
public static final int RESULT_CATEGORIES_NO_PATH_TO_ROOT
```

RETURN_RESULT_DESCRIPTION constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result descriptions.

Syntax

```
public static final int RETURN_RESULT_DESCRIPTION
```

RETURN_RESULT_TITLE constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result titles.

Syntax

```
public static final int RETURN_RESULT_TITLE
```

RETURN_RESULT_FIELDS constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result stored fields.

Syntax

```
public static final int RETURN_RESULT_FIELDS
```

RETURN_RESULT_CATEGORIES constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result categories.

Syntax

```
public static final int RETURN_RESULT_CATEGORIES
```

RETURN_RESULT_TYPE constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result document types.

Syntax

```
public static final int RETURN_RESULT_TYPE
```

RETURN_RESULT_SOURCE constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result document source.

Syntax

```
public static final int RETURN_RESULT_SOURCE
```

RETURN_RESULT_LANGUAGE constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result language.

Syntax

```
public static final int RETURN_RESULT_LANGUAGE
```

RETURN_RESULT_DATE constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result dates.

Syntax

```
public static final int RETURN_RESULT_DATE
```

RETURN_RESULT_SCORE constant

Pass this constant to the setReturnedAttribute method to enable or disable returning result scores.

Syntax

```
public static final int RETURN_RESULT_SCORE
```

getText method

Returns the text of this query.

See "Query syntax" on page 8 for more information.

Syntax

```
public String getText()
```

setText method

Sets the text for this query.

See "Query syntax" on page 8 for more information.

Syntax

```
public void setText(String text) throws SiapiException
```

Parameters

text
New text of this query.

getQueryLanguage method

Returns the language of this query as previously specified by calling the setQueryLanguage method, or null if the language was not specified.

Syntax

```
public String getQueryLanguage()
```

getReturnedFields method

Returns the names of the fields to be returned for the results of this query (as previously specified by a calling the setReturnedFields method). Returns null if no fields were specified.

For each document, some field elements can be marked as returnable, which means that they are stored within the index for that document. These field elements can be restored by the user when such a document comes up as a search result.

If no returned field was specified, this method returns null, and no fields will be returned for the query results.

You can specify exactly which fields to be retrieved by calling the "resetReturnedFields method" on page 33.

Syntax

```
public String[] getReturnedFields()
```

setReturnedFields method

Sets the names of returned metadata fields. By default, all fields are returned.

Use the setReturnedAttribute method to enable retrieval of metadata fields (using the RETURN_RESULT_FIELDS constant) before calling this method.

Syntax

```
public void setReturnedFields(String[] fieldNames) throws SiapiException
```

Parameters

fieldNames
Names of required fields

resetReturnedFields method

Resets the returned metadata fields data for this query.

Syntax

```
public void resetReturnedFields()
```

getNumRequestedResults method

Returns the number of results that are requested for this query.

Syntax

```
public int getNumRequestedResults()
```

getFirstRequestedResult method

Returns the rank, ordinal position, of the first result that is required for this query.

For example, having first-result 20 and num-requested-results 10 will retrieve 10 results, those that were ranked 20 to 30. Ranks start at 0.

Syntax

```
public int getFirstRequestedResult()
```

setRequestedResultRange method

Sets the rank (ordinal position) of the first result that is requested for this query, and the number of requested results. The first result has to be either 0 or an exact multiple of the number of requested results.

For example, having first-result 17 and num-requested-results four will retrieve four results, those that were ranked 17 to 20. Ranks start at 0.

Syntax

```
public void setRequestedResultRange(int fromResult,
                                    int numberOfResult) throws SiapiException
```

Parameters

fromResult
Rank of the first requested result.

numberOfResults
Number of requested results (the maximum value allowed for this parameter is 100).

getProperty method

Returns the value of a searchable property.

Syntax

```
public String getProperty(String name)
```

Parameters

name
Name of the searchable property to retrieve.

setProperty method

Sets the value of a searchable property.

Using these properties, you can modify the behavior of a searchable. To learn which properties are supported, call the "getProperties method" on page 26.

Restriction: Enterprise search does not support any properties and will throw an exception when this method is called.

Syntax

```
public void setProperty(String name,
                        String value) throws SiapiException
```

Parameters

name
Name of the searchable property.

value
Value of the searchable property.

Throws

SiapiException if the property cannot be modified at this time, or if the value is illegal.

getProperties method

Returns all of the searchable properties, or null if none exist.

Syntax

```
public Properties getProperties()
```

getSortKey method

Returns the key that the results of this query are sorted by.

The special predefined keys are:

SORT_KEY_RELEVANCE
Sorts results by relevance (default).

SORT_KEY_DATE
Sorts results by date.

SORT_KEY_NONE
No sorting.

Other valid key values include any names of fields that the search engine supports sorting by.

Syntax

```
public String getSortKey()
```

getSortPoolSize method

Returns the number of top relevant results that the search engine sorts by the sort key. The default setting for sort pool size is 500.

Syntax

```
public int getSortPoolSize()
```

Returns

The sort pool size.

setSortPoolSize method

Sets the number of top relevant results that the search engine should sort by the sort key.

The default setting for this is implementation dependent and may differ from one SIAPI implementation to another.

Syntax

```
public void setSortPoolSize(int sortPoolSize) throws SiapiException
```

Parameters

sortPoolSize
Sort pool size

Throws

SiapiException if the sort pool size is negative or too large.

getSortOrder method

Returns the order in which query results are sorted.

Valid values are:

SORT_ORDER_ASCENDING
The sort order is ascending.

SORT_ORDER_DESCENDING
The sort order is descending.

Syntax

```
public int getSortOrder()
```

Returns

The sort order.

setSortKey method

Sets the key by which to sort the results of this query.

The special predefined keys are:

SORT_KEY_RELEVANCE
Sets the sort results by relevance (default).

SORT_KEY_DOCUMENT_ID
Sets the sort results by document ID.

SORT_KEY_DATE
Sets the sort results by date.

SORT_KEY_NONE
Specifies no sorting.

Other valid values include any names of fields that the search engine supports through sorting.

Syntax

```
public void setSortKey(String sortKey) throws SiapiException
```

Parameters

sortKey
Field name or sort method to sort by

isSiteCollapsingEnabled method

Returns true if site collapsing should be applied when computing the results for this query.

Results are collapsed if the top results should contain more than two results from the same Web site

Syntax

```
public boolean isSiteCollapsingEnabled()
```

setSiteCollapsingEnabled method

Sets the site collapsing behavior for this query.

Syntax

```
public void setSiteCollapsingEnabled(boolean value) throws SiapiException
```

Parameters

value
True if site collapsing should be applied when computing the results for this query; false otherwise.

isPredefinedResultsEnabled method

Returns true if predefined links are returned for this query.

Syntax

```
public boolean isPredefinedResultsEnabled()
```

setPredefinedResultsEnabled method

Sets the predefined links behavior for this query.

Syntax

```
public void setPredefinedResultsEnabled(boolean value) throws SiapiException
```

Parameters

value
True if predefined links should be returned for this query, false otherwise

setSpellCorrectionEnabled method

Sets the spell correction to be enabled or not in this query.

Syntax

```
public void setSpellCorrectionEnabled(boolean enable) throws SiapiException
```

Parameters

enable
True if spell correction should be enabled, false otherwise

Throws

If spell correction enabling could not be set.

isSpellCorrectionEnabled method

Returns true if spell correction is enabled for this query. Returns false otherwise.

Syntax

```
public boolean isSpellCorrectionEnabled()
```

setResultCategoriesDetailLevel method

Sets the required category detail level for query results.

Syntax

Use the setReturnedAttribute method with the RETURN_RESULT_CATEGORIES constant to enable or disable retrieval of result categories.

Valid detailLevel values are:

RESULT_CATEGORIES_ALL
Each result category is returned along with its complete path-from-root information (RESULT_CATEGORIES_ALL is the default value).

RESULT_CATEGORIES_NO_PATH_TO_ROOT
Each result category is returned without path-from-root information (that is, getPathFromRoot returns null).

```
public void setResultCategoriesDetailLevel(int detailLevel)
```

setReturnedAttribute method

By default, all result attributes are returned, except for Fields.

Valid attributeType values are:

RETURN_RESULT_TITLE
getTitle returns null if isReturned is set to false.

RETURN_RESULT_DESCRIPTION
getDescription returns null if isReturned is set to false.

RETURN_RESULT_FIELDS
getFields returns null if isReturned is set to false.

RETURN_RESULT_CATEGORIES
getCategories returns null if isReturned is set to false.

RETURN_RESULT_TYPE
getDocumentType returns null if isReturned is set to false

RETURN_RESULT_SOURCE
getDocumentSource returns null if isReturned is set to false.

RETURN_RESULT_LANGUAGE
getLanguage returns null if isReturned is set to false.

RETURN_RESULT_DATE
getDate returns null if isReturned is set to false.

RETURN_RESULT_SCORE
getScore returns 0.0 if isReturned is set to false.

Syntax

```
public void setReturnedAttribute(int attributeType,
                                 boolean isReturned)
```

Parameters

attributeType
Result attribute type.

isReturned
True if attributes of the specified type should be returned, false otherwise.

isAttributeReturned method

Returns true if query result objects will contain the specified attribute type. By default, all result attributes are returned.

Valid attributeType values are:

RETURN_RESULT_TITLE
getTitle will return null if isReturned is set to false (by default, all result attributes are returned, except for fields).

RETURN_RESULT_DESCRIPTION
getDescription will return null if isReturned is set to false.

RETURN_RESULT_FIELDS
getFields will return null if isReturned is set to false.

RETURN_RESULT_CATEGORIES
getCategories will return null if isReturned is set to false.

RETURN_RESULT_TYPE
getDocumentType will return null if isReturned is set to false.

RETURN_RESULT_SOURCE
getDocumentSource will return null if isReturned is set to false.

RETURN_RESULT_LANGUAGE
getLanguage will return null if isReturned is set to false.

RETURN_RESULT_DATE
getDate will return null if isReturned is set to false.

RETURN_RESULT_SCORE
getScore will return 0.0 if isReturned is set to false.

Syntax

```
public boolean isAttributeReturned(int attributeType)
```

Parameters

attributeType
Result attribute type

Returns

True if the specified attribute is returned; false otherwise.

setQueryID method

Assigns an application-defined identifier to this query. This identifier can be used by the search engine to log information that pertains to the execution of this query.

Syntax

```
public void setQueryID(java.lang.String queryID)
```

Parameters

queryID
Application-defined query identifier.

getQueryID method

Returns the application defined query identifier set for this query, or null if no identifier was set.

Syntax

```
public java.lang.String getQueryID()
```

Returns

Returns the application-defined query identifier.

setACLConstraints method

Sets the access control constraints for this query.

See "Query syntax" on page 8 for more information on ACL constraints syntax.

Syntax

```
public void setACLConstraints(java.lang.String aclConstraints)
```

Parameters

aclConstraints
A string that specifies the access control constraints.

getACLConstraints method

Returns the access control constraints string that is set for this query. Returns null if no access control constraints are set.

See "Query syntax" on page 8 for more information on ACL constraints syntax.

Returns

Returns the access control constraints string set for this query.

setQueryLanguage method

Specifies a language other than the collection default language on the search server.

Syntax

```
public void setQueryLanguage(String lang) throws SiapiException
```

Parameters

lang
The language for this query. You can use either two letter codes (for example, en, ja) and four letter codes (for example, en-US, fr-FR) to specify the query language.

getResultCategoriesDetailLevel method

Returns the category detail level for query results.

The category detail level for query results. Possible return values are:

RESULT_CATEGORIES_ALL
Each result category is returned with its complete path-from-root information.

RESULT_CATEGORIES_NO_PATH_TO_ROOT
Each result category is returned without path-from-root information.

Syntax

```
public int getResultCategoriesDetailLevel()
```

Returns

setText method

Sets the text for this query.

See "Query syntax" on page 8 for more information.

Syntax

```
public void setText(String text) throws SiapiException
```

Parameters

text
New text of this query.

NameValuePair interface

The NameValuePair interface contains a name and an associated value for a document.

Each NameValuePair represents a document field, or metatag. A Result implementation stores an array of NameValuePair objects to represent the complete list of fields for a particular document.

getName method

Returns the name that is associated with this value.

Syntax

```
public String getName()
```

Returns

The name associated with this value.

getValue method

Returns the value.

Syntax

```
public String getValue()
```

Returns

Returns the value.

getFields method

Returns the name and value of all fields that were returned for this result. Returns null if no fields exist, or if the user requests that no fields return.

Syntax

```
public NameValuePair[] getFields()
```

Result interface

The Result interface contains a single search result.

Each result holds information about the unique ID of the document that corresponds to this result. The result can also contain the document title, description (if available), score, document type, source, and language, depending on the attributes the you requested for retrieval.

Each result can also contain a set of dynamic field properties that are represented as NameValuePairs, and a list of categories to which the document is currently assigned.

getCategories method

Returns the categories to which this result's document belongs. The method returns null if no categories exist for this result, or if the user requests no return for the categories.

Syntax

```
public ResultCategory[] getCategories()
```

getDate method

Returns null if no date exists, or if the user specified that date should not be returned, using the setReturnedAttribute method of the Query object.

Syntax

```
public Date getDate()
```

getDescription method

Returns the description (also known as a summary) of this result's document. Returns null if no field exists, or if the user specified that fields should not be returned, using the setReturnedAttribute method of the Query object

Syntax

```
public String getDescription()
```

getDocumentID method

Returns the unique name of this result's document.

Syntax

```
public String getDocumentID()
```

getDocumentSource method

Returns the source of this result's document. Returns null or unknown if no field exists, or if the user requests that no fields return.

Syntax

```
public String getDocumentSource()
```

getDocumentType method

Returns the type of this result's document. Returns null or unknown if no field exists, or if the user requests that no fields return.

Syntax

```
public String getDocumentType()
```

getFields method

Returns the name and value of all fields that were returned for this result. Returns null if no fields exist, or if the user requests that no fields return.

Syntax

```
public NameValuePair[] getFields()
```

getFields method

Returns the values of fields that were returned for this result, and whose name is fieldName. Returns null if none are available.

Syntax

```
public String[] getFields(String fieldName)
```

getLanguage method

Returns the language of this result's document. Returns null if unknown, or if the user requests not to return the language.

Syntax

```
public String getLanguage()
```

getProperties method

Returns all of the searchable properties, or null if none exist.

Syntax

```
public Properties getProperties()
```

getProperty method

Returns the value of a searchable property.

Syntax

```
public String getProperty(String name)
```

Parameters

name
Name of the searchable property to retrieve.

getScore method
Returns the score assigned by the index to this result. Returns null if the user requests not to return the score.

Syntax

```
public double getScore()
```

getTitle method
Returns the title of this result's document. Returns null if no title is available or if the user requests not to return the title.

Syntax

```
public String getTitle()
```

isFirstOfASite method
Returns true if this result is the first of a sequence of results of the same site.

Syntax

```
public boolean isFirstOfASite()
```

ResultCategory interface
Category of a result document.

Syntax

```
public interface ResultSet extends Serializable
```

getConfidence method
Returns the confidence level that a document belongs to this category. The confidence level is between 0 to 100.

Syntax

```
public double getConfidence()
```

getInfo method
Returns the ID and label of this category.

Syntax

```
public CategoryInfo getInfo()
```

getPathFromRoot method
Returns the IDs and labels of all ancestor categories, beginning with the root category, and ending with the direct parent of this category.

Syntax

```
public CategoryInfo[] getPathFromRoot()
```

Returns

Ancestor category information.

getTaxonomyID method

Returns the taxonomy ID for this category.

Syntax

```
public String getTaxonomyID()
```

Returns

The taxonomy ID.

MAX_CONFIDENCE constant

Maximum confidence level (the value of MAX_CONFIDENCE is 100.0).

Syntax

```
public static final double MAX_CONFIDENCE
```

MIN_CONFIDENCE constant

Minimum confidence level (the value of MIN_CONFIDENCE is 0.0).

Syntax

```
public static final double MIN_CONFIDENCE
```

ResultSet interface

The ResultSet interface contains the search results for the executed query.

In addition to the results of the query, the ResultSet contains administrator assigned predefined links, the estimated total number of results for this request, the number of results that are available, the total execution time, and the spell correction suggestions. The search method that is exposed by using the Searchable interface returns a ResultSet instance to the caller upon completion of the query.

getAvailableNumberOfResults method

Returns the number of results that can be retrieved for this query.

This number may differ from the estimated number of results. For example, when the estimated number of results exceeds the maximum of 500 sorted, getAvailableNumberOfResults will return 500. When the query results are not sorted (sort key is set to SORT_KEY_NONE), this method returns 0 if this is the last result page, and 1 if there are more results.

Syntax

```
public int getAvailableNumberOfResults()
```

getEstimatedNumberOfResults method

Returns an estimate of the total number of matching results for this query, which can be different than the value thatgetAvailableNumberOfResultsreturns. Returns -1 if this number is not known.

This estimation is the number of documents which include all the required query terms and none of the forbidden terms. Enterprise search does not provide a number-of-results estimate for unsorted results. Therefore, 0 is returned when the sort key is SORT_KEY_NONE.

Syntax

```
public int getEstimatedNumberOfResults()
```

getPredefinedResults method

Returns an array of predefined query results. Returns null if no predefined results are available or if you request that the predefined links not be returned (see Query.setPredefinedResultsEnabled).

This allows a search engine to return a highlighted set of special results for a query.

Syntax

```
public Result[] getPredefinedResults()
```

getProperties method

Returns all of the searchable properties, or null if none exist.

Syntax

```
public Properties getProperties()
```

getProperty method

Returns the value of a searchable property.

Syntax

```
public String getProperty(String name)
```

Parameters

name
Name of the searchable property to retrieve.

getQueryEvaluationTime method

Returns the amount of time, in milliseconds, required to process the query and obtain this result set.

Syntax

```
public long getQueryEvaluationTime()
```

getResults method

Returns an array of the available results.

Syntax

```
public Result[] getResults()
```

getSpellCorrections method

Returns a list of spell correction suggestions for the specified query string, or null if no suggestions are available.

If spell correction is enabled see the method “setSpellCorrectionEnabled method” on page 37 in the Query interface.

Syntax

```
public SpellCorrection[] getSpellCorrections()
```

Parameters

queryText
The query string to check.

Returns

Returns a list of spell correction suggestions for the specified query string.

hasUnconstrainedResults method

Returns an indication of whether the constraints applied during the search caused this query to have no results in the result set. Unconstrained results are results that would be returned if constraints are removed from the query.

Constraints for this matter can be:

- ACL
- Categories
- Range constraints
- Result languages
- Document type constraints
- Document source constraints

Syntax

```
public int hasUnconstrainedResults()
```

Returns

A positive value to indicate that unconstrained results exist, 0 (zero) to indicate that no unconstrained results exist, and a negative value to indicate that it is unknown whether unconstrained results exist or not.

isEvaluationTruncated method

Returns true if the process of evaluating the search for obtaining this search result set had to be truncated at least one time during the search process.

Syntax

```
public boolean isEvaluationTruncated()
```

SpellCorrection interface

The SpellCorrection interface contains the original query substring that is corrected, and suggestions for corrections of that substring, ordered by likelihood.

getQuerySubstring method

Returns the substring of the query for which the spelling corrections are suggested.

Syntax

```
public String getQuerySubstring()
```

getSuggestions method

Returns the spell correction suggestions, ordered by likelihood, or null if none exist.

Syntax

```
public String[] getSuggestions()
```

BrowseFactory interface

The BrowseFactory interface is the entry point to enterprise search implementation of the SIAPI Taxonomy Browsing functionality.

getBrowseService method

Obtains a browse service object.

Syntax

```
public BrowseService getBrowseService(java.util.Properties config)
                                throws SiapiException
```

Parameters

config
configuration information of the search server

Enterprise search supports 3 configuration properties (see "Structure of a SIAPI application" on page 5 for more information on obtaining a SearchService):

- **hostname** (required) - the hostname of the enterprise search search server.
- **port** (optional) - the port number of the enterprise search search server (the default port number is 80).
- **locale** (optional) - the locale settings used for returning translated error messages (the default value is the default java locale).

createApplicationInfo method

Creates an application info object for the specified application ID.

After you obtain an ApplicationInfo object, you can use setPassword or setToken to set the application's password or security-token.

Syntax

```
public ApplicationInfo createApplicationInfo(String applicationID)
  throws SiapiException
```

Parameters

ID ID of the application.

getVersion method

Returns the version of the specific SIAPI implementation.

Versioning data can be managed differently in each SIAPI implementation. The value returned here can be different than the version of the SIAPI API.

Syntax

```
public String getVersion()
```

BrowseService interface

The BrowseService interface provides taxonomy browsing services for collections.

getTaxonomyBrowser method

Returns an interface for browsing a taxonomy of the specified collection.

Syntax

```
public TaxonomyBrowser getTaxonomyBrowser(ApplicationInfo appInfo,
                                          java.lang.String collectionID,
                                          java.lang.String taxonomyID)
                                   throws SiapiException
```

Parameters

appInfo
Authentication and authorization information for the requesting application.

collectionID
ID of the collection.

taxonomyID
ID of the required taxonomy.

Returns

The specified collection's taxonomy browser SiapiException.

getAvailableTaxonomyBrowsers method

Returns an array of all available TaxonomyBrowsers for the specified collection.

Syntax

```
public TaxonomyBrowser[] getAvailableTaxonomyBrowsers(ApplicationInfo appInfo,
                                                      java.lang.String collectionID)
                                               throws SiapiException
```

Parameters

appInfo
Authentication and authorization information for the requesting application.

collectionID
ID of the collection.

Returns

Array of the available TaxonomyBrowsers SiapiException.

Category interface

The Category interface encapsulates a category's taxonomy data.

getInfo method

Returns the ID and label of this category.

Syntax

```
public CategoryInfo getInfo()
```

getChildren method

Returns the IDs and labels of all child categories.

Syntax

```
public CategoryInfo[] getChildren()
```

Returns

Child category IDs.

getPathFromRoot method

Returns the IDs and labels of all ancestor categories, beginning with the root category, and ending with the direct parent of this category.

Syntax

```
public CategoryInfo[] getPathFromRoot()
```

Returns

Ancestor category information.

getRootCategory method

Returns the tree's root category.

Syntax

```
public Category getRootCategory()
```

Returns

The root category.

getCategory method

Returns the category with the specified category ID.

Syntax

```
public Category getCategory(java.lang.String categoryID)
```

Parameters

categoryID
ID of the category to return.

Returns

The specified category.

TaxonomyBrowser interface

The TaxonomyBrowser interface provides taxonomy browsing and navigation interfaces. This interface is for read-only, non-administrative taxonomy-browsing tasks.

getRootCategory method

Returns the tree's root category.

Syntax

```
public Category getRootCategory()
```

Returns

The root category.

getCategory method

Returns the category with the specified category ID.

Syntax

```
public Category getCategory(java.lang.String categoryID)
```

Parameters

categoryID
ID of the category to return.

Returns

The specified category.

getTaxonomyInfo method

Returns the taxonomy information (ID and label).

Syntax

```
public TaxonomyInfo getTaxonomyInfo()
```

SiapiException class

General SIAPI exception.

This exception can occur as a result of the search and indexing API methods. This is the only SIAPI exception, and it covers many errors by setting the appropriate severity and appropriate type.

Each SiapiException is associated with a severity, an error code, and an optional embedded exception. Each error code is associated with a parametrized message, and an SIAPI implementation can also add arguments that resolve it.

This exception is implemented so that the original cause for the exception (that is, the original lowest exception) is kept as an embedded exception within the SiapiException. Therefore, when the stack trace is printed, the call stack of the original exception is printed. This implementation is useful for identifying the original cause for a problem.

SEVERITY_ERROR constant

Non fatal error severity.

Syntax

```
public static final int SEVERITY_ERROR
```

SEVERITY_FATAL_ERROR constant

Fatal error severity.

Syntax

```
public static final int SEVERITY_FATAL_ERROR
```

TYPE_UNKNOWN_ERROR constant

An unexpected internal error.

Message: internal error: %1
Message parameters:
%1: generic message

Syntax

```
public static final int TYPE_UNKNOWN_ERROR
```

TYPE_DOC_EXIST_ERROR constant

An attempt was made to create a document that already exists.

Message: ″document %1 already exists″
Message parameters:
%1: document ID

Syntax

```
public static final int TYPE_DOC_EXIST_ERROR
```

TYPE_DOC_NOT_FOUND_ERROR constant

An attempt was made to access or remove a document that does not exist.

Message: ″document %1 does not exist″
Message parameters:
%1: document ID

Syntax

```
public static final int TYPE_DOC_NOT_FOUND_ERROR
```

TYPE_SEARCH_ENGINE_STATE_ERROR constant

An illegal attempt to interact with the search engine. The state of the search engine did not match the requested operation.

Message: ″search engine state error: %1″
Message parameters:
%1: generic message

Syntax

```
public static final int TYPE_SEARCH_ENGINE_STATE_ERROR
```

TYPE_ILLEGAL_VALUE_ERROR constant

An attempt to set an illegal value (for example, setting an illegal value to a property).

Message: ″illegal value error: %1″
Message parameters:
%1: the illegal name and value

Syntax

```
public static final int TYPE_ILLEGAL_VALUE_ERROR
```

TYPE_IO_ERROR constant

An interaction with the search engine failed due to an I/O error. For example, the file could not be created because the disk is full.

Message: ″IO error″
Message parameters:
none

Syntax

```
public static final int TYPE_IO_ERROR
```

TYPE_IMPL_FACTORY_ERROR constant

An attempt to create the factory object of a SIAPI implementation failed.

Possible causes:

- The factory implementation class could not be found and loaded.

- The factory implementation object could not be instantiated.

Message: ″factory implementation error″
Message parameters:
none

Syntax

```
public static final int TYPE_IMPL_FACTORY_ERROR
```

TYPE_UNSUPPORTED_OPERATION constant

Indicates that the SIAPI implementation is missing some functionality.

Message: ″unsupported operation: %1″
Message parameters:
%1 name of the unsupported operation

Syntax

```
public static final int TYPE_UNSUPPORTED_OPERATION
```

TYPE_INDEX_DOES_NOT_EXIST constant

An index with the specified name does not exist.

Message: ″collection %1 does not exist″
Message parameters:
%1: collection ID

Syntax

```
public static final int TYPE_INDEX_DOES_NOT_EXIST
```

TYPE_TOO_MANY_VALUES constant

A multi-valued variable was specified with more than the allowed number of values.

Message: ″too many values for an argument: %1″
Message parameters:
%1: argument name

Syntax

```
public static final int TYPE_TOO_MANY_VALUES
```

TYPE_TOO_FEW_VALUES constant

A multi-valued variable was specified with less than the required number of values.

Message: ″too few values for an argument: %1″
Message parameters:
%1: argument name

Syntax

```
public static final int TYPE_TOO_FEW_VALUES
```

TYPE_ILLEGAL_RESULTS_RANGE constant

The requested results range exceeds the value allowed by the search engine.

Message: ″the specified result range is illegal: %1″
Message parameters:
%1: the specified result range

Syntax

```
public static final int TYPE_ILLEGAL_RESULTS_RANGE
```

TYPE_INDEX_CORRUPTED constant

The index is corrupted.

Message: ″collection %1 has a corrupted index″
Message parameters:
%1: collection ID

Syntax

```
public static final int TYPE_INDEX_CORRUPTED
```

TYPE_UNKNOWN_ENCODING constant

Certain data is unavailable because character encoding is unknown.

Message: ″data unavailable because encoding %1 is unknown″
Message parameters:
%1: encoding name

Syntax

```
public static final int TYPE_UNKNOWN_ENCODING
```

TYPE_QUERY_SYNTAX_ERROR constant

A query syntax error occurred.

Message: ″query syntax error: %1″
Message parameters:
%1: the illegal query string

Syntax

```
public static final int TYPE_QUERY_SYNTAX_ERROR
```

getSeverity method

Returns the severity for this exception (can be either SEVERITY_ERROR or SEVERITY_FATAL_ERROR).

Syntax

```
public int getSeverity()
```

Returns

Returns the severity of this exception.

getSeverityDescription method

Returns a human readable name of the severity for this exception.

Syntax

```
public java.lang.String getSeverityDescription()
```

Returns

Returns a human readable name of the severity of this exception.

getType method

Returns the type of this exception.

can be either SEVERITY_ERROR or SEVERITY_FATAL_ERROR

can be either SEVERITY_ERROR or SEVERITY_FATAL_ERROR

Syntax

```
public int getType()
```

Returns

Returns the type of this exception.

getTypeDescription method

Returns a human readable name of the type of this exception.

Syntax

```
public java.lang.String getTypeDescription()
```

Returns

Returns a human readable name of the type of this exception.

printStackTrace method

Prints the stack trace of the embedded exception. Otherwise, if this does not exist, it prints the stack of this exception.

Syntax

```
public void printStackTrace()
```

Overrides

Overrides printStackTrace in class java.lang.throwable.

addArgument method

Adds an additional name-value pair as an argument of this exception.

Syntax

```
public void addArgument(NameValuePair arg)
```

Parameters

arg
An additional name-value pair.

getArguments method

Returns the arguments for this exception.

Syntax

```
public java.util.ArrayList getArguments()
```

Returns

Returns the arguments for this exception and an array list of arguments.

getMessage method

Returns the error message string of this exception.

Syntax

```
public java.lang.String getMessage()
```

Returns

Returns the error message string of this exception.

Overrides

Overrides getMessage in class java.lang.throwable.

getLocalizedMessage method

Returns a localized message for this exception.

Syntax

```
public java.lang.String getLocalizedMessage()
```

Returns

Returns a localized message for this exception.

Syntax

public java.lang.String getLocalizedMessage()

Overrides

getLocalizedMessage in class java.lang.throwable

TaxonomyInfo interface

The TaxonomyInfo interface provides the ID and label of a taxonomy.

getID method

Returns the taxonomy ID.

Syntax

```
public java.lang.String getID()
```

getLabel method

Returns the taxonomy label.

Syntax

```
public java.lang.String getLabel()
```

getTaxonomyInfo method

Returns the taxonomy information (ID and label).

Syntax

```
public TaxonomyInfo getTaxonomyInfo()
```

Sample SIAPI applications

The enterprise search SIAPI includes several sample applications.

The sample applications demonstrate:

- The minimum required application to submit a search query to the search server
- The basic search tasks
- The search tasks that are typical of programming needs
- The basic taxonomy browsing and navigation tasks

Minimum required application

The SearchExample class gives a simple example of the minimum required application to submit a search query to the search server.

The following sample demonstrates how to:

- Access the service
- Specify a collection
- Specify an application
- Submit a query
- Process the returned results

SearchExample class

The SearchExample class is already compiled in the esapi.jar file. You can run the class by executing the following command from a command line.

UNIX

```
java -classpath <installroot>/lib/esapi.jar:<installroot>>/lib/siapi.jar
SearchExample
```

Windows

```
javac -classpath <installroot>\lib\esapi.jar;<installroot>>\lib\siapi.jar
SearchExample
```

The default host is assumed to be the local host if the host is not specified. The default port is assumed to be 80 if the port is not specified.

Obtain a factory

The first line of any program written against the SIAPI should create an instance of the Factory class. The SearchFactory class implementation in enterprise search is: com.ibm.es.api.search.RemoteSearchFactory.

Obtain a factory of the enterprise search specific SIAPI implementation.

```
SearchFactory factory =
          SiapiSearchImpl.createSearchFactory
  ("com.ibm.es.api.search.RemoteSearchFactory");
```

Create a valid application ID

The search server uses the application ID to authorize this access to the collection.

```
ApplicationInfo appinfo = factory.createApplicationInfo("app1");
appinfo.setPassword("password");
```

Create a new Properties object

```
Properties config = new Properties();
```

Set the host name

Set the host name of the SIAPI search server. In this case it is an enterprise search search server. The host name should be the host name that is assigned to the enterprise search WebSphere installation.

```
config.setProperty("hostname", hostname);
```

Set the port number

Set the port number of the SIAPI search server. The default value is port 80 (the Web server port).

```
config.setProperty("port", portNumber);
```

Set the locale

Set the locale of the client. The underlying SIAPI implementation uses this value to return translated error messages to the client for the appropriate language.

```
config.setProperty("locale", "en_US");
```

Obtain the SearchService implementation

```
SearchService searchService =
    factory.getSearchService(config);
```

Obtain a Searchable object to the specified collection ID

```
Searchable searchable =
    searchService.getSearchable(appinfo, collectionId);
if (searchable == null) {
    System.out.println("Failed to get a searchable
   for collection: " + collectionId);
    return;
}
```

Create a new Query object using the specified query string

```
Query q = factory.createQuery(queryString);
```

Set the result range

Set the result range that you want to access on this query.

```
q.setRequestedResultRange(0, 10);
```

Execute the search

Execute the search by calling the Searchable's search method. An SIAPI ResultSet object is returned.

```
ResultSet rset = searchable.search(q);
System.out.println("returned from search call");
if (rset != null) {
```

Get the results from the ResultSet

```
Result r[] = rset.getResults();
            if (r != null) {
```

Print the results

Walk the results list and print out the document identifier and document title.

```
for (int k = 0; k < r.length; k++) {
                  System.out.println
      ("Result " + k + ": " + r[k].getDocumentID() + " - " + r[k].getTitle());
                }
            }
        } else {
            System.out.println("result set was null");
        }
    }
```

For testing purposes only

```
public static void main(String[] args) throws Exception {
   SearchExample sc = new SearchExample();
   if (args.length < 2) {
      System.out.println
("Usage: SearchExample<queryString><collection ID>optionally:<hostname><port>");
System.out.println("Example Usage: SearchExample lotus col1");
System.out.println("\tdefault host and port is localhost:80");
System.out.println("Example Usage: SearchExample lotus col1 localhost 80");
      return;
        }
        sc.queryString = args[0];
        sc.collectionId = args[1];
        if (args.length > 2) {
            sc.hostname = args[2];
        } else {
            sc.hostname = "localhost";
        }
        if (args.length > 3) {
            sc.portNumber = args[3];
        } else {
            sc.portNumber = "80";
        }
        sc.execute();
    }
}
```

Browse and navigation application

This sample code is an SIAPI example of basic taxonomy browsing and navigation tasks.

This example demonstrates how to:

- Obtain the browse factory
- Obtain a browse service
- Obtain a browser reference
- Get and display the root category
- Get the root's first child category
- Display the child category and it's path from root

Browse and navigation example

```
 */
import java.util.Locale;
import java.util.Properties;

import com.ibm.siapi.browse.BrowseFactory;
import com.ibm.siapi.browse.BrowseService;
import com.ibm.siapi.browse.Category;
import com.ibm.siapi.browse.SiapiBrowseImpl;
import com.ibm.siapi.browse.TaxonomyBrowser;
import com.ibm.siapi.common.ApplicationInfo;
import com.ibm.siapi.common.CategoryInfo;
import com.ibm.siapi.common.TaxonomyInfo;

/*
 * The BrowseExample class gives an example of accessing a collection's
 * taxonomy tree and printing out some of the basic navigation properties.
 *
 */
public class BrowseExample {
    private String hostname     = "localhost";
    private String portNumber   = "80";
    private String collectionId;
    private String taxonomyId;
    private String applicationName;
    private String applicationPassword;

    public void execute() throws Exception {
        // obtain the OmniFind specific SIAPI Browse factory implementation
 BrowseFactory factory =
  SiapiBrowseImpl.createBrowseFactory("com.ibm.es.api.browse.RemoteBrowseFactory");

        // create a valid Application ID that will be used
        // by the Search Node to authorize this
        // access to the collection
        ApplicationInfo appinfo = factory.createApplicationInfo(applicationName);
        appinfo.setPassword(applicationPassword);

        // create a new Properties object.
        Properties config = new Properties();
        // set the hostname of the OmniFind Search Node.  The hostname
        // should be the hostname that is assigned to the
        // Search Node's WebSphere installation
        config.setProperty("hostname", hostname);
        // set the port number - the
        // default value is port 80 (the web server port).
        config.setProperty("port", portNumber);
        // set the locale of the "client".  This value is
        // used by the underlying SIAPI implementation to
        // return translated error messages to the client
        // for the appropriate language.
        // NOTE:  this should be the 2 letter ISO-639
        // language code followed by an underscore
        // followed by the ISO-639 2 letter country code
        config.setProperty("locale", "en_US");

        // obtain the Browse service implementation
 BrowseService browseService = factory.getBrowseService(config);

        // get a TaxonomyBrowser forthe specified taxonomy id and collection id
        TaxonomyBrowser browser = browseService.getTaxonomyBrowser(appinfo,
   collectionId, taxonomyId);
        if(browser == null) {
            System.out.println("Failed to get a taxonomy for taxonomy id: "
     + taxonomyId +
                               " from collection: " + collectionId);
            return;
```

```
        }

        //Display the Taxonomy Info
        TaxonomyInfo taxonomyInfo = browser.getTaxonomyInfo();
        System.out.println("Taxonomy label: " + taxonomyInfo.getLabel());

        // get the root category
        Category rootCategory = browser.getRootCategory();

        // display the root category
        System.out.println("Root category label: " + rootCategory.getInfo()
   .getLabel());
        System.out.println("child categories:");
        CategoryInfo[] childrenInfo  = rootCategory.getChildren();
        for (int i = 0; i < childrenInfo.length; i++) {
            System.out.println("\t" + childrenInfo[i].getLabel());
        }

        // Now get the root's first child category
        Category childCategory = browser.getCategory(rootCategory.getChildren()[0]
   .getID());

        // Display the child category and it's path from root
        System.out.println("Root's first child's label: " + childCategory.getInfo()
   .getLabel());
        System.out.println("It's path from root is : ");
        CategoryInfo[] pathFromRoot = childCategory.getPathFromRoot();
        for (int i = 0; i < pathFromRoot.length; i++) {
            System.out.println("-->" + pathFromRoot[i].getLabel());
        }
    }

    public static void main(String[] args) throws Exception {
        BrowseExample sc = new BrowseExample();
        if (args.length < 5) {
            System.out.println("Usage: BrowseExample <taxonomy id> <collection id>
     <application name>");
System.out.println("Example Usage: BrowseExample tax1 col1 Default password
localhost 80")
return;
}
sc.taxonomyId = args[0];
        sc.collectionId = args[1];
        sc.applicationName = args[2];
        sc.applicationPassword = args[3];
        sc.hostname = args[4];
        sc.portNumber = args[5];
        sc.execute();
    }
}
```

Retrieve all search results

The following sample demonstrates how to set a query to return unsorted results and loop over the query results. Enterprise search allows you to obtain a maximum of 500 sorted results for your queries, however, you can obtain all unsorted results.

Obtain a SearchFactory and a Searchable

Obtain a SearchFactory and a Searchable object as explained in "Minimum required application" on page 56 sample.

```
SearchFactory factory;
Searchable searchable;

... // obtain a SearchFactory and Searchable object
```

Create a new Query object

```
Query q = factory.createQuery("big apple");
```

Set the query to return unsorted results

```
q.setSortKey(Query.SORT_KEY_NONE);
```

Execute the search

Execute the query in a loop to obtain a page of results at a time. The maximum result page size allowed in enterprise search is 100.

Note: When you receive the results pages you need to interpret the getAvailableNumberOfResults and getEstimatedNumberOfResults differently from the way you would interpret them for sorted query results:

- getEstimatedNumberOfResults will always return 0, since enterprise search does not provide a number-of-results estimate for unsorted results.
- getAvailableNumberOfResults will return one of two values: 0 if this is the last result page, and 1 if there are more results.
- You can use the length of the array returned by getResults to find out how many results are actually within this result page.

```
int fromResult = 0;
int pageSize = 100;
boolean moreResults = true;

// loop over query results, pageSize results at a time
while (moreResults) {

 // set the result range for the next page of results
 q.setRequestedResultRange(fromResult, pageSize);

 // execute the search
 ResultSet resultPage = s.search(q);

 // loop over the results from the ResultSet
 Result[] results = resultPage.getResults();
 for (int i=0;i<results.length;i++) {
... // process result
 }

 // check if there are more available results
 moreResults = (resultPage.getAvailableNumberOfResults() == 1);

 // modify the range for getting the next page of results
 fromResult += pageSize;
}
```

Chapter 4. Data listener API

Data Listener is a enterprise search component that accepts data from a known port, and sends this data to a collection. By calling data listener APIs, the user can add pages to a collection, remove URIs from a collection, or instruct the Web crawlers of a collection to visit or revisit URLs.

Enterprise search overview

The enterprise search system can collect data from different data sources, create indexes for the collected data, and provide search capabilities. Data sources include the Web, news servers, database tables, Lotus Notes® databases, content management systems, and file systems.

Figure 1 shows an overview of the search behavior of the enterprise search system. The users submit queries to the search server, and the index server periodically refreshes the data on the search server.

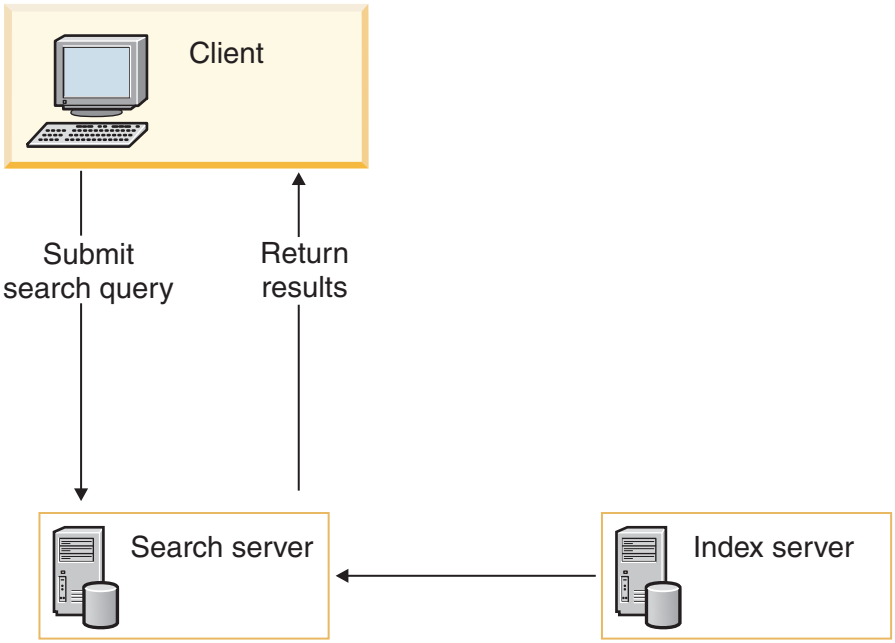

Figure 1. Enterprise search behavior overview

Data listener API overview

Data listener touches the following components in the enterprise search system:

Client machine
Submits search queries.

Search server
Accepts queries and provides results to the client.

Index server
Stores data that has been parsed from the crawler.

Crawler
Retrieves documents from a data source.

The Data Listener API enables you to submit data requests to the enterprise search system.

The client connects to the data listener component to push data to the appropriate server. When the client connects to the data listener component, the data listener component verifies the client ID and password and that the client is authorized to push data to the specified collection.

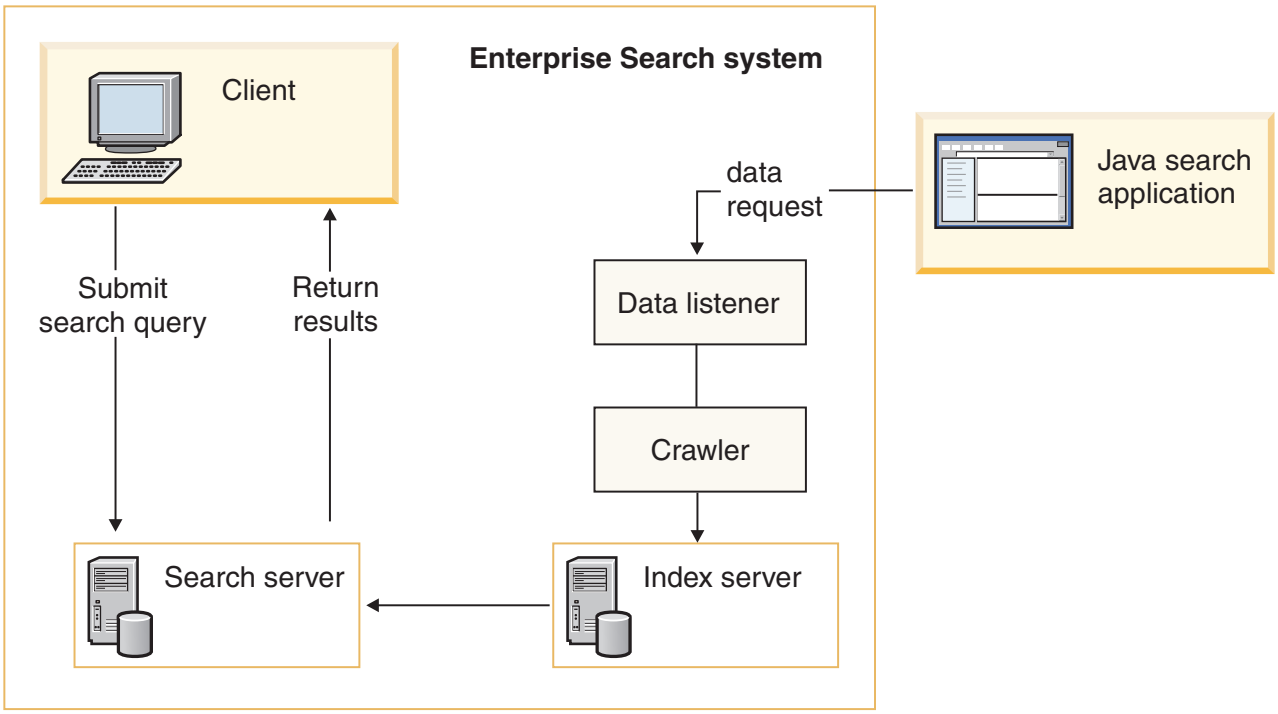

Figure 2. How the data listener API works with the enterprise search system

Data listener API properties

To push data to the data listener component, you must provide the following information:

Table 2. Data listener API properties

Property	Definition
Hostname	Host name of the data listener server.
Port	Port number that the server listens to.
ID	Client ID.
PW	Client password.
Col	ID of the target collection.
URI	URI of the data.
Metadata	Metadata object of the data, of type DataSourceMetada.
Content	Main body of the data.

Data control

The data listener API adds and removes content by submitting URI and URL requests to the collection servers and crawler.

universal resource identifier (URI)
An encoded address that represents any resource, such as an HTML document, image, video clip, or program, on the Web. The URI superclass includes URLs.

universal resource locator (URL)
A Web address, which offers a way of naming and locating specific items on the Web.

Removing data with the data listener API

There are two ways to remove data from the search collections by using the data listener API:

- Removing a URI from the search server and index
- Removing URI patterns from the index

Removing a URI from the search server and index

You can use the data listener API to remove a specific URI. The request is sent to the data listener and the URI is removed from the index server and search server as shown in the figure below.

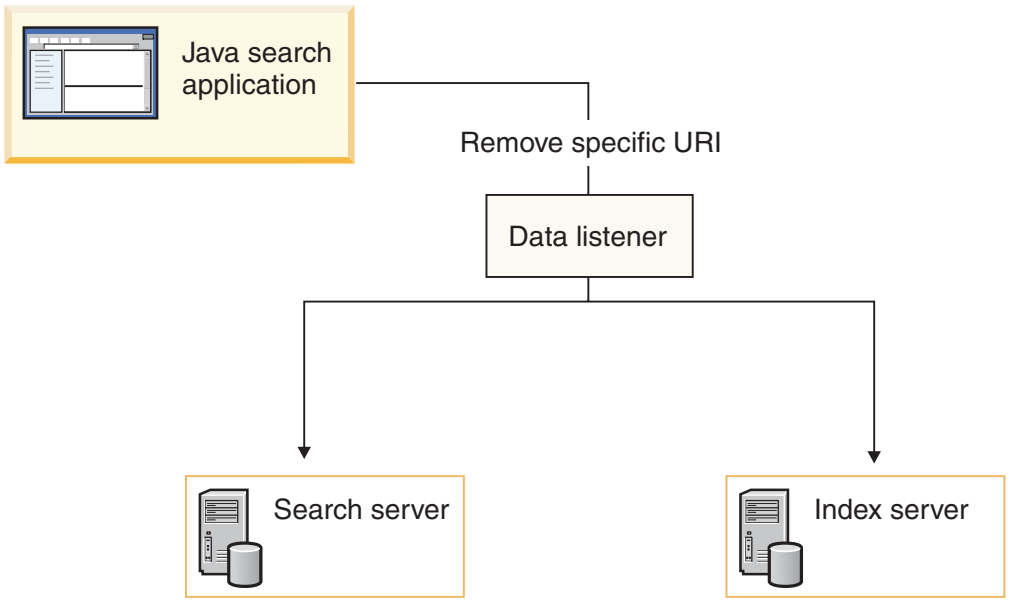

Figure 3. Removing a specific URI

Use the removeURI method to delete the data from the specified collection.

Removing URI patterns from the index

You can use the data listener API to remove a URI pattern. For example, if you submit a remove URI pattern request and specify www.ibm.com/*.html as your URI, the index server will remove the following URLs:

- www.ibm.com/home.html

- www.ibm.com/family.html
- www.ibm.com/pics.html

Note: Use a remove URI pattern request with caution. The removed content cannot be recovered. The content must be added to the index again.

The remove request is sent to the data listener and the URI pattern is removed from the index server as shown in the figure above.

Use the removeURI method to delete the data from the specified collection. The removed URIs are returned in the search results to users until the index server refreshes the data on the search server.

Adding data with the data listener API

There are two ways to push data to the search collections by using the data listener API:

- Adding a URI with content
- Visiting or revisiting a URL

Adding a URI with content

You can use the data listener API to add a URL with its content by including the content parameter in the pushData request. This sends the specified data to the data listener and index server as shown in Figure 4.

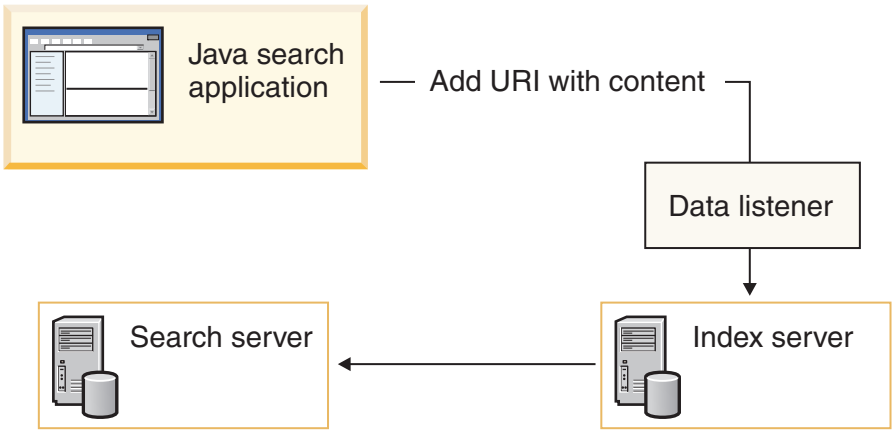

Figure 4. Adding a URI with content

The added URI and its content will be accessible to users when the index server refreshes the data on the search server.

Visiting or revisiting a URL

You can use the data listener API to add specific URLs and retrieve their data by using the revisitURLs method. This method is valid only for Web content. All other data sources must use the pushData call.

Visit Requests the crawler to retrieve data from a new URL

Revisit
Requests the crawler to refresh data from a URL that is currently in an index

If you add a URL without the associated content, the data listener sends the request to the crawler, which retrieves the content and then sends it to the Index server.

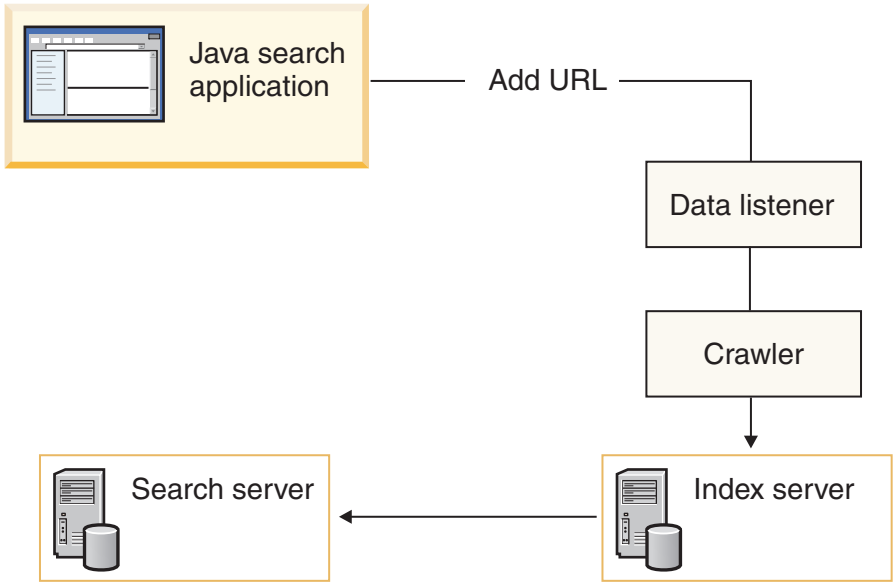

Figure 5. Adding a URL

Push Data to indexer

The data listener API pushes documents to the indexer.

The documents consist of three components:

- URIs
- content
- metadata

Specifying metadata

For all add data requests, you need to specify metadata methods before you issue the push data call. Metadata contains information about the document and can include information such as author, date modified, or date created.

Use the createDataSourceMetadata method to create the metadata. Each metadata has two sections:

- CommonMetadata
- DataSourceSpecificMetadata

Create metadata

Use the createDataSourceMetadata method to create the metadata.

The CommonMetadata section includes all the fields that are common for all data sources. Those fields include:

Datasource
Specifies what the data source is. This field is always field-searchable, and is returned in the search result.

SecurityACL
Specifies what credential tokens the user needs to retrieve this document in a search result. Multiple tokens are separated by comma. If this field is empty or does not exist, the default value is PUBLIC, which means that it is available to everybody.

Date A numeric string that represents the number of seconds since EPOCH (a negative number represents data prior to EPOCH). The date is assumed to be GMT time, not local time. The date can be used for computing the static score of the document. This field is for supporting date based ranking and range search.

StaticScoreRef
A numeric string that can be used for computing the static score of the document. This field is reserved for future use.

HasSeparateContent
A string that indicates if the actual content is empty. If the content is not empty, the attributes will indicate the content type, the character set and language of the content.

The following example shows the common metadata fields.

```
<Metadata Language="en">
<CommonMetadata
   Datasource = "DB2",
   CrawlerId="MyCrawlerId",
   DatasourceName="My Display Name For My Data Source",
   Date=GMT,
   StaticScoreRef=8>
   <SecurityACLs>Token1,Token2</SecurityACLs>

   <HasSeparateContent ContentType="application/pdf",
      Charset="iso-933",
      Language="gb">
    YES
   </HasSeparateContent>

</CommonMetadata>
</Metadata>
```

The DataSourceSpecificMetadata section includes those fields that are specific to the data source. Each field can have the following attributes:

FieldName
Indicates the field name that will be used for field search for this field.

Searchable
Indicates whether this field is searchable or not. The default is yes.

FieldSearchable
Indicates whether fielded search is required for this field. The default is yes.

Metadata
Indicates whether the content of this field must be included in the extraFields data in the search results. The default is yes.

ParametricSearch
Indicates whether this field requires parametric search. The default is no. If yes, the data in this field is expected to be in the format that can be converted into a numeric value by the parser.

ResolveConflict

Specifies how to resolve the conflict if more than one field has the same field name. For example, a document might have two author fields, one from metadata and another from the content. Possible options include: MetadataPreferred, ContentPreferred, and Coexist (concatenation).

The following example shows the DataSourceSpecific metadata fields.

```
<Metadata Language="en">
<DatasourceSpecificMetadata>
   <Field FieldName="DatabaseName",
    Searchable="YES",
    FieldSearchable="YES",
    Metadata="YES",
    ResolveConflict="MetadataPreferred>
   myDatabase
   </Field>
   Field FieldName="TableName",
    Searchable="YES",
    FieldSearchable="YES",
    Metadata="YES",
    ResolveConflict="MetadataPreferred>
   myTable
   </Field>
   <Field FieldName="MYID",
    Searchable="YES",
    FieldSearchable="YES",
    Metadata="YES",
    ResolveConflict="MetadataPreferred">
   10
   </Field>
</DataourceSpecificMetadata>
</Metadata>
```

Creating additional metadata

To add more metadata fields to the provided fields, use the addMetaField method. Use this method to add elements to an existing metadata object.

```
public static void addMetaField(DataSourceMetadata metadata,
                                String fieldName,
                                String fieldValue,
                                boolean searchable,
                                boolean partOfResult,
                                boolean fieldSearchable,
                                boolean parametricSearchable)
```

Data listener API calls and methods

Data listener uses the following APIs.

DLDataPusher class

Pushes data to the data listener to remove URIs from collections and revisits URLs.

To call these APIs you must specify the host name and the port of the data listener, and provide the crawler ID and password for authentication.

pushData method

The pushData method is an API function that pushes data to data listener.

Syntax

```
public static DLResponse pushData(String hostname,
                                  int port,
                                  String id,
                                  String pw,
                                  String uri,
                                  String col,
                                  String metadata,
                                  byte[] content)
```

Parameters

hostname
Th host name of the data listener server.

port
The port number of the data listener.

ID The crawler ID.

pw The crawler password.

URI
The URI of the data.

col The ID of the target collection.

metadata
The XML string version of the metadata.

content
The content to be pushed to the data listener.

removeURIs method

The remove URIs method is an API function that removes URIs from the specified collection.

Syntax

```
public static DLResponse removeURIs(String hostname,
                                    int port,
                                    String id,
                                    String pw,
                                    String uris,
                                    String col,
```

Parameters

hostname
The host name of the data listener server.

port
The port number of the data listener.

ID The crawler ID.

pw The crawler password.

URI
The URI of the data.

col The ID of the target collection.

revisitURLs method

Instructs the Web crawler for the specified collection to add or revisit the URIs.

Syntax

```
public static DLResponse removeURIs(String hostname,
                                    int port,
                                    String id,
                                    String pw,
                                    String uris,
                                    String col)
```

Parameters

hostname
The host name of the data listener server.

port
The port number of the data listener.

ID The crawler ID.

pw The crawler password.

URI
The URI or the URI patterns, separated by white space of the data.

col The ID of the target collection.

addMetaField method

The addMetaField method adds an element to a metadata object.

Syntax

```
public static void addMetaField(DataSourceMetadata metadata,
                                String fieldName,
                                String fieldValue,
                                boolean searchable,
                                boolean partOfResult,
                                boolean fieldSearchable,
                                boolean parametricSearchable)
```

createDataSourceMetadata method

The createDataSourceMetadata method creates a metadata object.

Syntax

```
public static DataSourceMetadata createDataSourceMetadata(String ds,
                                                          String cid,
                                                          String dsName,
                                                          int score,
                                                          Date dt,
                                                          String language,
                                                          String securityACLs,
                                                          String contentType,
                                                          String charSet,
                                                          byte[] content)
```

Sample data listener API applications

The DLSampleClient class gives a few simple examples of how to push data using the data listener API.

The sample applications demonstrate:

- Basic data listener tasks
  - Identification and authentication
  - Pushing data

  - Removing data
- Advanced data listener tasks
  - Preparing metadata
  - Calling the push method
  - Checking the result
  - Repeating the data push

Basic data listener API application

The DLSampleClient class includes an example of basic data listener API tasks.

The basic example shows how to add or revisit URLs for one collection and remove URLs from another collection. The following sample demonstrates:

- Identification and authentication tasks
- Pushing data
- Removing data

Data listener sample

```
public class DLSampleClient {
```

Identification and authentication tasks

To push data to the data listener, you must submit the host name and the port of the server. You also need to provide the crawler ID and password for authentication.

```
static void dataPushExample_1(String hostname, int port, String crawlerID,
 String passwd) {
```

Creating a pusher object

```
 DLDataPusher pusher = new DLDataPusher(hostname, port, crawlerID, passwd);
```

Specifying the URIs to add

Specify either the specific URIs or the URI pattern that you want the crawler to add or revisit.

```
        StringBuffer sb = new StringBuffer();
        sb.append("url1");
        sb.append("\n");
        sb.append("url*pattern1");
        sb.append("\n");

        sb.append("url2");
        sb.append("\n");
        sb.append("url*pattern2");

        String collectionID = "collection2";
        String urls = sb.toString();
```

Directing the data

Specify the target collection of the URLs to be revisited.

```
pusher.revisitURLs(urls, collectionID);
```

Removing data

Specify the target collection to remove the URIs from.

```
collectionID = "collection3";
        pusher.removeURIs(urls, collectionID);

    }
```

DB2 Information Integrator documentation

This topic provides information about the documentation that is available for DB2 Information Integrator.

The tables in the following topics provide the official document title, form number, and location of each PDF book. To order a printed book, you must know either the official book title or the document form number. Titles, file names, and the locations of the DB2 Information Integrator release notes and installation requirements are also provided in the following topics.

Documentation about event publishing function for DB2 Universal Database on z/OS

Documentation about event publishing function for DB2 Universal Database on z/OS

Purpose

Documentation about event publishing function for DB2 Universal Database on z/OS.

Table 3. DB2 Information Integrator documentation about event publishing function for DB2 Universal Database on z/OS

Name	Form number	Location
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web site
Introduction to Replication and Event Publishing	GC18-7567	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about event publishing function for IMS and VSAM on z/OS

Documentation about event publishing function for IMS and VSAM on z/OS

Purpose

Documentation about event publishing function for IMS and VSAM on z/OS.

Table 4. DB2 Information Integrator documentation about event publishing function for IMS and VSAM on z/OS

Name	Form number	Location
Client Guide for Classic Federation and Event Publisher for z/OS	SC18-9160	DB2 Information Integrator Support Web site
Data Mapper Guide for Classic Federation and Event Publisher for z/OS	SC18-9163	DB2 Information Integrator Support Web site
Getting Started with Event Publisher for z/OS	GC18-9186	DB2 Information Integrator Support Web site
Installation Guide for Classic Federation and Event Publisher for z/OS	GC18-9301	DB2 Information Integrator Support Web site
Operations Guide for Event Publisher for z/OS	SC18-9157	DB2 Information Integrator Support Web site
Planning Guide for Event Publisher for z/OS	SC18-9158	DB2 Information Integrator Support Web site
Reference for Classic Federation and Event Publisher for z/OS	SC18-9156	DB2 Information Integrator Support Web site
System Messages for Classic Federation and Event Publisher for z/OS	SC18-9162	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	N/A	DB2 Information Integrator Support Web site

Documentation about event publishing and replication function on Linux, UNIX, and Windows

Documentation about event publishing and replication function on Linux, UNIX, and Windows

Purpose

Documentation about event publishing and replication function on Linux, UNIX, and Windows.

Table 5. DB2 Information Integrator documentation about event publishing and replication function on Linux, UNIX, and Windows

Name	Form number	Location
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web site

Table 5. DB2 Information Integrator documentation about event publishing and replication function on Linux, UNIX, and Windows (continued)

Name	Form number	Location
Installation Guide for Linux, UNIX, and Windows	GC18-7036	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Introduction to Replication and Event Publishing	GC18-7567	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Migrating to SQL Replication	N/A	DB2 Information Integrator Support Web site
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
SQL Replication Guide and Reference	SC27-1121	DB2 Information Integrator Support Web site
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web site
Tuning for SQL Replication Performance	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about federated function on Linux, UNIX, and Windows

Documentation about federated function on Linux, UNIX, and Windows

Purpose

Documentation about federated function on Linux, UNIX, and Windows.

Table 6. DB2 Information Integrator documentation about federated function on Linux, UNIX, and Windows

Name	Form number	Location
Application Developer's Guide	SC18-7359	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site

Table 6. DB2 Information Integrator documentation about federated function on Linux, UNIX, and Windows (continued)

Name	Form number	Location
C++ API Reference for Developing Wrappers	SC18-9172	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Data Source Configuration Guide	N/A	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Federated Systems Guide	SC18-7364	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Guide to Configuring the Content Connector for VeniceBridge	N/A	DB2 Information Integrator Support Web site
Installation Guide for Linux, UNIX, and Windows	GC18-7036	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Java API Reference for Developing Wrappers	SC18-9173	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Migration Guide	SC18-7360	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Wrapper Developer's Guide	SC18-9174	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about federated function on z/OS

Documentation about federated function on z/OS

Purpose

Documentation about federated function on z/OS.

Table 7. DB2 Information Integrator documentation about federated function on z/OS

Name	Form number	Location
Client Guide for Classic Federation and Event Publisher for z/OS	SC18-9160	DB2 Information Integrator Support Web site
Data Mapper Guide for Classic Federation and Event Publisher for z/OS	SC18-9163	DB2 Information Integrator Support Web site
Getting Started with Classic Federation for z/OS	GC18-9155	DB2 Information Integrator Support Web site
Installation Guide for Classic Federation and Event Publisher for z/OS	GC18-9301	DB2 Information Integrator Support Web site
Reference for Classic Federation and Event Publisher for z/OS	SC18-9156	DB2 Information Integrator Support Web site
System Messages for Classic Federation and Event Publisher for z/OS	SC18-9162	DB2 Information Integrator Support Web site
Transaction Services Guide for Classic Federation for z/OS	SC18-9161	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	N/A	DB2 Information Integrator Support Web site

Documentation about replication function on z/OS

Documentation about replication function on z/OS

Purpose

Documentation about replication function on z/OS.

Table 8. DB2 Information Integrator documentation about replication function on z/OS

Name	Form number	Location
ASNCLP Program Reference for Replication and Event Publishing	N/A	DB2 Information Integrator Support Web site
Introduction to Replication and Event Publishing	GC18-7567	DB2 Information Integrator Support Web site
Migrating to SQL Replication	N/A	DB2 Information Integrator Support Web site
Replication and Event Publishing Guide and Reference	SC18-7568	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Replication Installation and Customization Guide for z/OS	SC18-9127	DB2 Information Integrator Support Web site
SQL Replication Guide and Reference	SC27-1121	• *DB2 PDF Documentation* CD • DB2 Information Integrator Support Web site
Tuning for Replication and Event Publishing Performance	N/A	DB2 Information Integrator Support Web site

Table 8. DB2 Information Integrator documentation about replication function on z/OS (continued)

Name	Form number	Location
Tuning for SQL Replication Performance	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	N/A	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD

Documentation about enterprise search function on Linux, UNIX, and Windows

Documentation about enterprise search function on Linux, UNIX, and Windows

Purpose

Documentation about enterprise search function on Linux, UNIX, and Windows.

Table 9. DB2 Information Integrator documentation about enterprise search function on Linux, UNIX, and Windows

Name	Form number	Location
Administering Enterprise Search	SC18-9283	DB2 Information Integrator Support Web site
Installation Guide for Enterprise Search	GC18-9282	DB2 Information Integrator Support Web site
Programming Guide and API Reference for Enterprise Search	SC18-9284	DB2 Information Integrator Support Web site
Release Notes for Enterprise Search	N/A	DB2 Information Integrator Support Web site

Release notes and installation requirements

Release notes provide information that is specific to the release and fix pack level for your product and include the latest corrections to the documentation for each release. Installation requirements provide information that is specific to the release of your product.

Table 10. DB2 Information Integrator Release Notes and Installation Requirements

Name	File name	Location
Installation Requirements for IBM DB2 Information Integrator Event Publishing Edition, Replication Edition, Standard Edition, Advanced Edition, Advanced Edition Unlimited, Developer Edition, and Replication for z/OS	Prereqs	• The *DB2 Information Integrator* product CD • DB2 Information Integrator Installation Launchpad
Release Notes for IBM DB2 Information Integrator Standard Edition, Advanced Edition, and Replication for z/OS	ReleaseNotes	• In the DB2 Information Center, **Product Overviews** > **Information Integration** > **DB2 Information Integrator overview** > **Problems, workarounds, and documentation updates** • DB2 Information Integrator Installation launchpad • DB2 Information Integrator Support Web site • The *DB2 Information Integrator* product CD
Release Notes for IBM DB2 Information Integrator Event Publisher for IMS for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Event Publisher for VSAM for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for IBM DB2 Information Integrator Classic Federation for z/OS	N/A	DB2 Information Integrator Support Web site
Release Notes for Enterprise Search	N/A	DB2 Information Integrator Support Web site

Viewing release notes and installation requirements

Viewing release notes and installation requirements

Purpose

To view release notes and installation requirements from the CD on Windows operating systems, enter:

x\doc\%*L*

Parameters

x The Windows CD drive letter

%*L*
The locale of the documentation that you want to use, for example, en_US.

Purpose

To view release notes and installation requirements from the CD on UNIX operating systems, enter:

/cdrom/doc/*%L*

Parameters

cdrom
The UNIX mount point of the CD

%L
The locale of the documentation that you want to use, for example, en_US.

Viewing and printing PDF documentation

Viewing and printing PDF documentation

To view and print the DB2 Information Integrator PDF books from the *DB2 PDF Documentation* CD

1. From the root directory of the *DB2 PDF Documentation* CD, open the index.htm file.
2. Click the language that you want to use.
3. Click the link for the document that you want to view.

Accessing DB2 Information Integrator documentation

Accessing DB2 Information Integrator documentation

All DB2 Information Integrator books and release notes are available in PDF files from the DB2 Information Integrator Support Web site at www.ibm.com/software/data/integration/db2ii/support.html.
To access the latest DB2 Information Integrator product documentation, from the DB2 Information Integrator Support Web site, click on the Product Information link, as shown in Figure 6 on page 83.

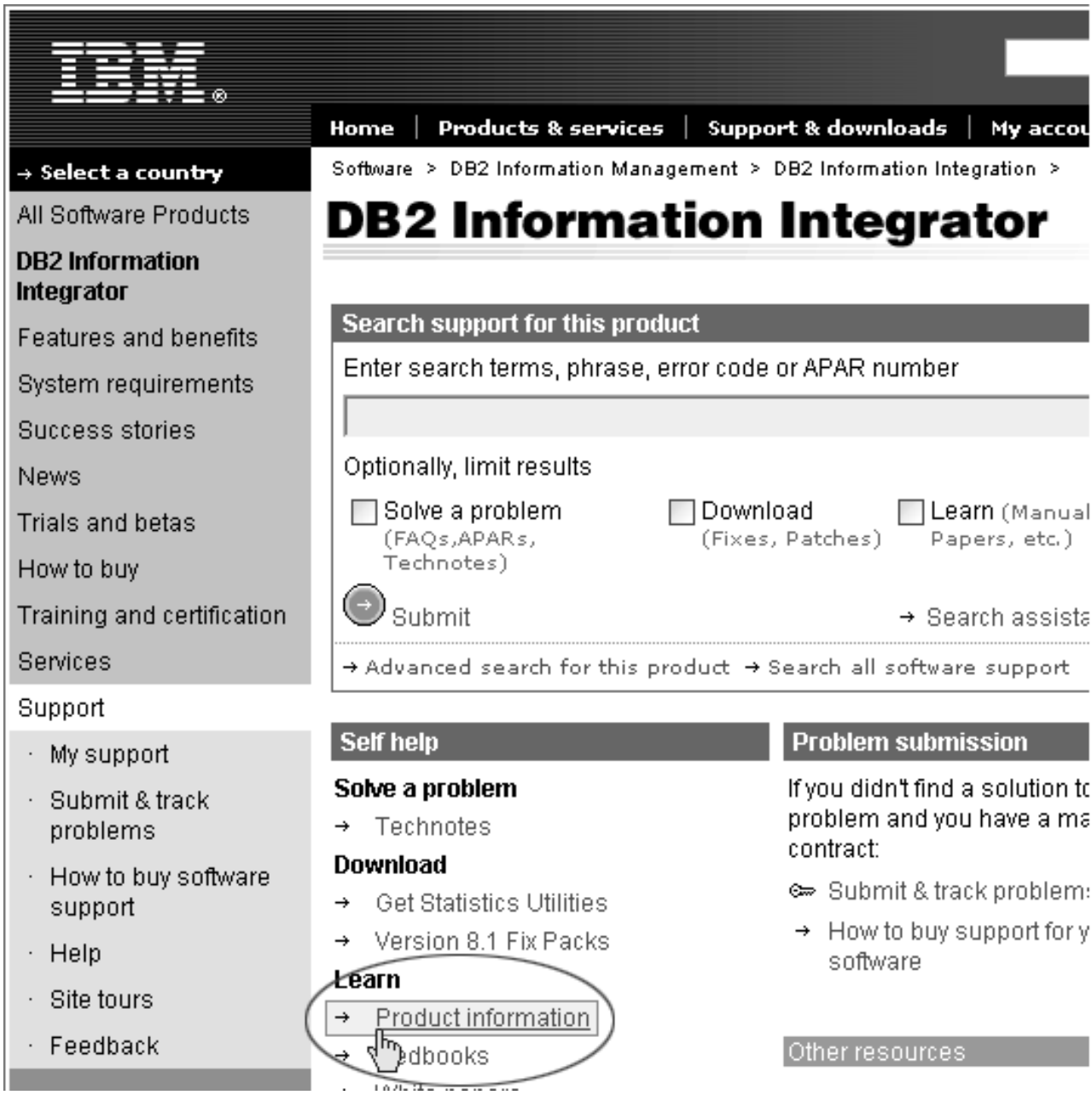

Figure 6. Product information link on the DB2 Information Integrator Support Web site

You can access the latest DB2 Information Integrator documentation, in all supported languages, from the Product Information link:

- DB2 Information Integrator product documentation in PDF files
- Fix pack product documentation, including release notes
- Instructions for downloading and installing the DB2 Information Center for Linux, UNIX, and Windows
- Links to the DB2 Information Center online

The DB2 Information Integrator Support Web site also provides support documentation, IBM Redbooks, white papers, product downloads, links to user groups, and news about DB2 Information Integrator.

Accessibility

Accessibility features help users with physical disabilities, such as restricted mobility or limited vision, to use software products successfully. The following list specifies the major accessibility features in DB2® Version 8 products:

- All DB2 functionality is available using the keyboard for navigation instead of the mouse. For more information, see "Keyboard input and navigation."
- You can customize the size and color of the fonts on DB2 user interfaces. For more information, see "Accessible display."
- DB2 products support accessibility applications that use the Java™ Accessibility API. For more information, see "Compatibility with assistive technologies" on page 86.
- DB2 documentation is provided in an accessible format. For more information, see "Accessible documentation" on page 86.

Keyboard input and navigation

Keyboard focus

Keyboard focus

In UNIX® operating systems, the area of the active window where your keystrokes will have an effect is highlighted.

Keyboard input

Keyboard input

You can operate the DB2 tools using only the keyboard. You can use keys or key combinations to perform operations that can also be done using a mouse. Standard operating system keystrokes are used for standard operating system operations.

For more information about using keys or key combinations to perform operations, see Keyboard shortcuts and accelerators: Common GUI help.

Keyboard navigation

Keyboard navigation

You can navigate the DB2 tools user interface using keys or key combinations.

For more information about using keys or key combinations to navigate the DB2 Tools, see Keyboard shortcuts and accelerators: Common GUI help.

Accessible display

Accessible display

Purpose

Accessible display

Font settings

Font settings

You can select the color, size, and font for the text in menus and dialog windows, using the Tools Settings notebook.

For more information about specifying font settings, see Changing the fonts for menus and text: Common GUI help.

Non-dependence on color

Non-dependence on color

You do not need to distinguish between colors to use any of the functions in this product.

Compatibility with assistive technologies

Compatibility with assistive technologies

The DB2 tools interfaces support the Java Accessibility API, which enables you to use screen readers and other assistive technologies with DB2 products.

Accessible documentation

Accessible documentation

Documentation for DB2 is provided in XHTML 1.0 format, which is viewable in most Web browsers. XHTML allows you to view documentation according to the display preferences set in your browser. It also allows you to use screen readers and other assistive technologies.

Syntax diagrams are provided in dotted decimal format. This format is available only if you are accessing the online documentation using a screen-reader.

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 all 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 give 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/region 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/region 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 sites. The materials at those Web sites are not part of the materials for this IBM product, and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information that has been exchanged, should contact:

IBM Corporation J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems, and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements, or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious, and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs, in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

Outside In (®) Viewer Technology, © 1992-2004 Stellent, Chicago, IL., Inc. All Rights Reserved.

Trademarks

This topic lists IBM trademarks and certain non-IBM trademarks.

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM
AIX, AIX 5L
DB2
DB2 Universal Database
Domino
Informix
Lotus
Lotus Notes
Notes
OmniFind
WebSphere
z/OS

The following terms are trademarks or registered trademarks of other companies:

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

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

To locate an IBM office in your country or region, see the IBM Directory of Worldwide Contacts on the Web at www.ibm.com/planetwide.

Obtaining product information

Information about DB2 Information Integrator is available by telephone or on the Web.

Information about DB2 Information Integrator is available by telephone or on the Web. The phone numbers provided here are valid in the United States.

1. To order products or to obtain general information: 1-800-IBM-CALL (1-800-426-2255)
2. To order publications: 1-800-879-2755
3. Visit the Web at www.ibm.com/software/data/integration/db2ii/support.html.

This site contains the latest information about:

- The technical library
- Ordering books
- Client downloads
- Newsgroups
- Fix packs
- News
- Links to Web resources

Providing comments on the documentation

Please send any comments that you have about this book or other DB2 Information Integrator documentation.

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 Information Integrator documentation.You can use any of the following methods to provide comments:

1. Send your comments using the online readers' comment form at www.ibm.com/software/data/rcf.
2. Send your comments by e-mail to comments@us.ibm.com. Include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a title, a table number, or a page number).

Index

T

U

Printed in USA

SC18-9284-00