System i

e-business and Web serving
OmniFind Text Search Server for DB2 for i5/OS

V1.1

System i

e-business and Web serving
OmniFind Text Search Server for DB2 for i5/OS

V1.1

Note

Before using this information and the product it supports, read the information in "Notices," on page 63.

This edition applies to version 1, release 1, modification 0 of OmniFind Text Search Server for DB2 for i5/OS (product number 5733-OMF) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Chapter 1. Introduction to IBM Text Search for DB2 for i5/OS

IBM® Text Search for DB2® for i5/OS® allows a DB2 for i5/OS user to issue SQL statements to satisfy familiar text search queries on data that is stored in a DB2 database.

IBM Text Search for DB2 for i5/OS provides a set of administrative stored procedures and 2 built-in functions (CONTAINS and SCORE) which are used to search text indexes created from documents stored in a DB2 table. The administrative stored procedures are used to enable and disable text searching and to create, update, and drop text indexes.

A text index can be created over any CHAR, VARCHAR, CLOB, BLOB, DBCLOB, GRAPHIC, VARGRAPHIC, BINARY, or VARBINARY column which contains plain text, HTML, XML, or many rich document types (such as PDF files). The data is read from the text column and is converted to Unicode (CCSID 1208) before it is indexed.

Text indexes are not normal DB2 indexes. They are not maintained automatically, cannot be journaled, and cannot be backed up using normal backup and restore methods. Text indexes are created and stored on a text search server. By default, the text search server is created on the same i5/OS system as the data stored in the DB2 database. However, a text search server can be created on another server running i5/OS, Linux®, UNIX®, AIX®, or Windows®. The text search server contains a collection of significant terms extracted from each row of the column. A TCP/IP connection is used to communicate with the text search server.

The CONTAINS and SCORE functions are built-in functions which are integrated into DB2.

Overview of the IBM OmniFind Text Search Server for DB2 for i5/OS

DB2 for i5/OS uses the IBM OmniFind™ Text Search Server for DB2 for i5/OS as an indexing and search engine for documents that are stored in a DB2 database.

The IBM OmniFind Text Search Server for DB2 for i5/OS supports multiple collections. A collection contains one text search index and the index specific options for parsing, indexing, and searching.

The IBM OmniFind Text Search Server for DB2 for i5/OS does not have a graphical user interface. However, this text search server provides SQL stored procedures and command-line tools that you can use for common tasks, such as configuring and administering the text search server, creating a synonym dictionary for a collection, and diagnosing problems.

System requirements for enabling IBM Text Search for DB2 for i5/OS

Before you enable IBM Text Search for DB2 for i5/OS, ensure that your system meets all of the operating system and software requirements.

System requirements for installing the IBM OmniFind Text Search Server

Before you install an IBM OmniFind Text Search Server for DB2 for i5/OS, make sure that your system meets all of the hardware, software, and operating system requirements.

When you install IBM OmniFind Text Search Server for DB2 for i5/OS, the installation program creates one text search server for i5/OS. You can install text search servers on remote servers running Linux or

Windows. These servers are part of DB2 Accessories Suite for z/OS® (5655-R14). See the DB2 Accessories Suite for z/OS Web site (http://www.ibm.com/software/data/db2imstools/db2tools/accessories-suite/ ) for information about downloading the suite.

Software requirements

Make sure that your system meets the following software requirements:

- IBM Java™ Runtime 1.5 SR4
- DB2 Universal Java Driver installed and configured on the text search server
- For i5/OS, the following programs must be installed:
  - 5761SS1 Option 30 Qshell
  - 5761SS1 Option 33 i5/OS Portable Application Solutions Environment (i5/OS PASE)
  - 5761SS1 Option 39 International Components for Unicode
  - 5761JV1 Option 11 Java SE 6 32 bit
  - 5761JV1 Option 12 Java SE 6 64 bit
  - 5761SS1 PTFs SI30971, SI30510, and SI30838. The most recent superseding versions of these PTFs should be applied on the system. The Group PTF SF99601 for IBM DB2 for i5/OS should be applied on the system.

You need to meet the previous software requirements before the installation. If a product or PTF is missing, the installation fails. See APAR SE33053 for a listing of the most current installation requirements and fixes.

Chapter 2. Key concepts

Understanding the key concepts about text search functions, such as what document types and languages are supported, will help you to leverage the benefits of IBM Text Search for DB2 for i5/OS.

Text search index creation and updates

Text search index creation is the process of defining and declaring the properties of the text search index. *Text search index update* is the process of adding new data from a DB2 table to the text search index on the IBM OmniFind Text Search Server for DB2 for i5/OS and propagating any changes of the data.

For each text search index that you create, a new collection is created on the IBM OmniFind Text Search Server for DB2 for i5/OS. After initial creation, the text search index contains no data. You add data to the text search index by calling the SYSPROC.SYSTS_UPDATE stored procedure. The first update process adds all of the text documents from the text column to the text search index. This process is known as the *initial update*. The subsequent updates are incremental.

When a text search index is created, the following objects are created or updated:

- The staging table is created in the QSYS2 library.
- The INSERT, DELETE, and UPDATE triggers are added to the base table.
- An SQL view with the name of the text search index is created in the schema of the text search index. This view contains information about the text index. For example, the view can be used to obtain the base table name, the staging table name, and the number of pending changes to the base table that are not yet reflected in the text search index.
- The text search index catalogs (SYSTEXTINDEXES and SYSTEXTCOLUMNS) in the QSYS2 library are updated with a new entry added for the new text search index.

Take the following staging table considerations:

- It is suggested not to perform any DB operation on the staging table except saving and restoring the file, and changing authorities.
- If the authorities are being changed by the user for the base table, the staging table for the base table needs to be changed too.

Take the following base table considerations:

- It is suggested not to remove the DELETE, UPDATE, and INSERT triggers that are added when a text search index is created.
- A drop of the text search index will remove the triggers.
- It is suggested not to alter or remove the rowid, primary key, or unique column that was used as the key in the text search index.
- If the base table is deleted, the text search index will not be deleted from the server. Drop the text search index before the base table is deleted.
- Altering the data column of the base table that results in data truncation might result in false positive matches in the text search index.

Asynchronous indexing and triggers

Because updating a text search index is an extensive operation, the text search index that is maintained on the IBM OmniFind Text Search Server for DB2 for i5/OS is not updated synchronously when the DB2 table is updated.

Instead, changes to the DB2 table column are captured by triggers to a local log table. (Sometimes this log table might be referred to as a staging table.) These triggers automatically store information about new, changed, and deleted documents in a log table. Each log table is associated with one text search index. Applying the contents of the log table to its corresponding text search index is called an *incremental update*.

You can update the text search index that is on the IBM OmniFind Text Search Server for DB2 for i5/OS by calling the SYSPROC.SYSTS_UPDATE stored procedure. You must periodically update the text search index in order for changes to be reflected in queries.

In addition to a manually initiated update by calling SYSPROC.SYSTS_UPDATE, updates can be scheduled to occur automatically by using the UPDATE FREQUENCY clause on the SYSPROC.SYSTS_CREATE procedure when the text index is created.

Supported document formats

The text column data can be plain text, an Hypertext Markup Language (HTML) document, an Extensible Markup Language (XML) document, or any document that is recognized by the search engine.

IBM OmniFind Text Search Server for DB2 for i5/OS parses documents to extract the relevant parts and make those parts searchable. For example, tags and metadata in an HTML document are not indexed. Parsing of the following document formats is supported:

- TEXT: Flat text
- HTML: Hypertext Markup Language
- XML: Extensible Markup Language
- INSO: The IBM OmniFind Text Search Server for DB2 for i5/OS uses filters to detect the format of text documents. The following INSO document formats are supported:
  - XML
  - HTML
  - JustSystems Ichitaro
  - Lotus® 123
  - Lotus Freelance
  - Lotus WordPro
  - Microsoft® Excel
  - Microsoft PowerPoint
  - Microsoft Rich Text Format
  - Microsoft Visio
  - Microsoft Word
  - Microsoft Write
  - Portable Document Format (PDF)
  - Quattro Pro
  - Rich Text RTF
  - StarOffice Calc and OpenOffice Calc

All of the documents in an indexed text column must be of the same format (TEXT, HTML, XML or INSO).

XML data

XML structure in the XML data is indexed in the IBM OmniFind Text Search Server for DB2 for i5/OS after parsing the data through an XML parser. Then, you can use the supported XPath query syntax to retrieve the results.

Supported data types

The data in the text columns that you want to index and search can be either binary data or character data.

The following data types are binary data:

- BINARY
- VARBINARY
- BLOB

In addition, IBM Text Search for DB2 for i5/OS handles the following data types similarly to binary data:

- CHAR FOR BIT DATA
- VARCHAR FOR BIT DATA

The following data types are character data:

- CHAR FOR SBCS DATA or FOR MIXED DATA
- VARCHAR FOR SBCS DATA or FOR MIXED DATA
- CLOB
- DBCLOB
- GRAPHIC
- VARGRAPHIC

If the data is binary data, you can specify the coded character set identifier (CCSID) of the data that is to be used to build the text search index. For character data, DB2 knows the encoding; therefore, if you explicitly specify a CCSID, that specification is ignored.

Text score and synonym support

The IBM OmniFind Text Search Server for DB2 for i5/OS supports the use of synonyms to modify the results of a query, and a text score might be returned as part of the query results.

Text score

A text score is returned as part of the query results. By default, a text score is not returned. A text score is a value between 0 and 1, up to 3 decimal points; for example, 0.000 to 1.000. A text score denotes how closely a result matches the query relative to all of the other documents in the text search index. IBM OmniFind Text Search Server for DB2 for i5/OS composes the text score from various factors, such as the general importance of the search terms (based on the frequency of the terms in each document and offset by the frequency of the terms across all documents) and the proximity of occurrences of the search terms.

Synonym support

You can use synonyms to improve the results for a query. Using synonyms can increase the number of query results by causing more documents to match a query. However, using synonyms might also decrease the precision of a query and make it more difficult to find a small number of documents that match the exact search criteria.

By default, synonyms are not used for a query. To use synonyms for a query, create a synonym dictionary, and add the synonym dictionary to a collection by using the synonym tool.

For more information about synonyms, see "Synonym tool" on page 49.

Linguistic processing

The IBM OmniFind Text Search Server for DB2 for i5/OS provides dictionary packs to support the linguistic processing of documents and queries that are not in English.

As an alternative to dictionary-based word segmentation, the IBM OmniFind Text Search Server for DB2 for i5/OS uses *n-gram segmentation* support for languages such as Chinese, Japanese, and Korean. N-gram segmentation is a method of analysis that considers overlapping sequences of a given number of characters as a single word rather than using blank space to delimit words as in Unicode-based white space segmentation.

If a text document is in one of the supported languages, linguistic processing is carried out when the input is parsed into tokens. For unsupported languages, an error code is returned.

When you search a text search index, a match is indicated that contains linguistic variations of the query terms. The variations of a word depend on the language of the query. In addition to finding variations of query terms, the query language is also used for spelling suggestions.

Supported languages

You can specify that text documents be processed by using a specific language.

You can specify the language for the indexed text data in the SYSPROC.SYSTS_CREATE administration stored procedure. If you set the value to AUTO, the IBM OmniFind Text Search Server for DB2 for i5/OS tries to determine the language. For short documents, automatic detection might be not accurate and is not recommended. The default language for linguistic processing is English (en_US).

The following table shows the five-character language codes for the supported languages.

Table 1. The five-character language codes for the supported languages

Language code	Language
ar_AA	Arabic
cs_CZ	Czech
da_DK	Danish
de_CH	German (Switzerland)
de_DE	German (Germany)
el_GR	Greek
en_AU	English (Australia)
en_GB	English (United Kingdom)
en_US	English (United States)
es_ES	Spanish (Spain)
fi_FI	Finnish
fr_CA	French (Canada)
fr_FR	French (France)
it_IT	Italian
ja_JP	Japanese

Table 1. The five-character language codes for the supported languages (continued)

Language code	Language
ko_KR	Korean
nb_NO	Norwegian Bokmal
nl_NL	Dutch
nn_NO	Norwegian Nynorsk
pl_PL	Polish
pt_BR	Brazilian Portuguese
pt_PT	Portuguese (Portugal)
ru_RU	Russian
sv_SE	Swedish
zh_CN	Simplified Chinese
zh_TW	Traditional Chinese

Linguistic processing for Chinese, Japanese, and Korean documents

You can process documents that are in Chinese, Japanese, or Korean by using dictionary-based segmentation or by using n-gram segmentation.

For a search engine, getting good search results depends in large part on the techniques that are used to process text. After the text is extracted from the document, the first step in text processing is to identify the individual words in the text. Identifying the individual words in the text is referred to as *segmentation*. For many languages, white space (blanks, the end of a line, and certain punctuation) can be used to recognize word boundaries. However, Chinese, Japanese, and Korean do not use white space between characters to separate words, so other techniques must be used.

The IBM OmniFind Text Search Server for DB2 for i5/OS provides the following two methods to support the linguistic processing of Chinese, Japanese, and Korean:

- Dictionary-based word segmentation (also called morphological analysis)
- N-gram segmentation

Dictionary-based word segmentation

Dictionary-based word segmentation uses a language-specific dictionary to identify words in the sequence of characters in the document. This technique provides precise search results, because the dictionaries are used to identify word boundaries. However, dictionary-based word segmentation can miss specific matching results.

N-gram segmentation

N-gram segmentation avoids the problem of identifying word boundaries, and instead indexes overlapping pairs of characters. Because the IBM OmniFind Text Search Server for DB2 for i5/OS uses two characters, this technique is also called bi-gram segmentation.

N-gram segmentation always returns all matching documents that contain the search terms; however, this technique might sometimes return documents that do not match the query.

By default, the IBM OmniFind Text Search Server for DB2 for i5/OS comes with a pre-configured index that uses n-gram segmentation for Chinese, Japanese, and Korean.

To see how both types of linguistic processing work, examine the following text in a document: `election for governor of Kanagawa prefecture`. In Japanese, this text contains eight characters. For this example, the eight characters are represented as A B C D E F G H. A sample query that users might enter could be `election for governor`, which is four characters and are represented as E F G H. (The document text and the sample query share similar characters.)

If you use n-gram segmentation processing:

After the document is indexed, the search engine segments the text `election for governor of Kanagawa prefecture` into the following sets of characters: AB BC CD DE EF FG GH

The sample query `election for governor` is segmented into the following sets of characters: DE EF FG GH. If you search with the sample query `election for governor`, the document is found by the query because the tokens for both the document text and the query appear in the same order.

When you enable n-gram segmentation, you might see more results but possibly less precise results. For example, in Japanese, if you search with the query `Kyoto` and a document in your index contains the text `City of Tokyo`, the query `Kyoto` will return the document with the text `City of Tokyo`. The reason is that `City of Tokyo` and `Kyoto` share two of the same Japanese characters.

If you do not use n-gram segmentation processing:

After the document is indexed, the search engine segments the text `election for governor of Kanagawa prefecture` into the following sets of characters: ABC DEF GH.

The sample query `election for governor` is segmented into the following sets of characters: EF GH. The characters EF do not appear in the tokens of the document text. (Even though the document does not have EF, it does have DEF).

The document text contains DEF, but the query contains only EF. Therefore, the document is less likely to be found by using the sample query.

When you do not enable n-gram segmentation, you probably receive more precise results but possibly fewer results.

Chapter 3. Installing and configuring text search functions

Populating the QSYS2.SYSTEXTSERVERS table

Installation

IBM Text Search for DB2 for i5/OS is the licensed-program product 5733-OMF from IBM. It is installed after the standard installation procedures for licensed programs on the i5/OS operating system. See the Installing additional licensed programs topic for details about how to install a licensed program. To find this product, enter GO LICPGM from the command line and select option 10 (Display installed licensed programs), and it is displayed under the list of licensed programs.

The QSYS2.SYSTEXTSERVERS table must be populated with information about where the IBM OmniFind Text Search Servers for DB2 for i5/OS are installed. When a default IBM OmniFind Text Search Server for DB2 for i5/OS is installed, the QSYS.SYSTEXTSERVERS table is populated.

If you are using text search servers on a remote i5/OS system, or if you are using non-i5/OS servers such as a Windows or Linux server, you need to populate this table by issuing an SQL INSERT statement.

Creating additional text search servers

If you want to populate the QSYS2.SYSTEXTSERVERS table with additional servers to the default one, follow these steps:

1. Copy the server port number and server name for each text search server to the SERVERPORT column and SERVERNAME column of the QSYS2.SYSTEXTSERVERS table by issuing an SQL INSERT statement.
2. Copy the authentication token from each text search server to the SERVERAUTHTOKEN column of the QSYS2.SYSTEXTSERVERS table by issuing an SQL INSERT statement.

   When the DB2 communicates with a text search server, an authentication token is required. This token is generated on the text search server during the installation.
3. Copy the server key for each text search server to the SERVERMASTERKEY column of QSYS2.SYSTEXTSERVERS table by issuing an SQL INSERT statement.

Example

The following example of an SQL INSERT statement copies the required information for a text search server to the columns in the QSYS2.SYSTEXTSERVERS table:

```
INSERT INTO QSYS2.SYSTEXTSERVERS(SERVERNAME,
                                 SERVERADRINFO,
                                 SERVERPORT,
                                 SERVERTYPE,
                                 SERVERAUTHTOKEN,
                                 SERVERMASTERKEY,
                                 SERVERPATH)
       VALUES('127.0.0.1',
              VARBINARY(X'0000'),
              49200,
              0,
              'AH2X4w==',
              'b1YhcR9O858ArwxLJeIY/Q==',
              '/QOpenSys/QIBM/ProdData/TextSearch/server1/bin/');
```

Starting text search functions

Before you start using the text search functions, call the SYSPROC.SYSTS_START stored procedure. By calling this procedure, you can start all of the production servers that you have defined to be local to the system.

Text search support includes SQL statements that use the CONTAINS function, the SCORE function, and the following administration stored procedures:

- SYSPROC.SYSTS_CREATE
- SYSPROC.SYSTS_UPDATE
- SYSPROC.SYSTS_DROP

Creating a text search index

Call the SYSPROC.SYSTS_CREATE stored procedure to create a text search index.

The DB2 base table must contain a ROWID column, unique key, or primary key.

To create a text search index on an existing DB2 table with a column that contains text:

Call the SYSPROC.SYSTS_CREATE stored procedure.

The text search index is empty until the first time that you update the text search index.

Updating a text search index

Call the SYSPROC.SYSTS_UPDATE stored procedure to update a text search index.

To synchronize the text search index with the contents of the DB2 table:

Call the SYSPROC.SYSTS_UPDATE stored procedure.

Now, you can perform search queries on the text search index. If updates are done to the base table text search column after this update, the search query results will not reflect those changes until the next update is run.

Searching a text search index

An SQL statement with a CONTAINS function or a SCORE function can be issued to search a text search index by using the search argument criteria that is specified.

The user who is performing the text queries on a DB2 table must have the standard privilege set that is required for any form of query, as specified in *DB2 SQL Reference*.

To search a text search index:

Issue an SQL statement with a CONTAINS function or a SCORE function.

To create, update, and search additional text search indexes, repeat those steps for each additional text search index.

Document truncation

The IBM OmniFind Text Search Server for DB2 for i5/OS limits the number of characters that can be indexed for each text document. Sometimes this character limit results in the truncation of large text documents in the text search index.

The default value for the number of Unicode characters that are allowed for each text document is 10 million. For a rich text document, this limit is applied after the document is transformed to plain text.

If a text document is truncated during the parsing stage, you receive a warning that some documents were not processed correctly or completely, and the document is partially indexed. The warning appears in the job log. Text that is in the document after the limit is reached is not indexed and cannot be searched. You might want to remove the document that has been truncated from the text search index to avoid unexpected behavior during search processing. You can remove the document by removing it from the DB2 table, or by changing the value for the document to empty or null.

Chapter 4. Administration stored procedures for text search

A set of administration SQL stored procedures are provided by DB2 for i5/OS to start and stop text search functions and to create, drop, and update text search indexes. These procedures can be called from any SQL interfaces. You cannot call these procedures from an i5/OS command line by using CL commands.

When looking at the system catalogs through ISQL, having your job set to a CCSID other than 65535 will allow you to see the content of the columns.

SYSPROC.SYSTS_START

Call the SYSPROC.SYSTS_START stored procedure to enable DB2 text search functions.

Text search functions include support for SQL queries that use the CONTAINS function, the SCORE function, and the administration stored procedures that are used to maintain text search indexes.

Run the SYSPROC.SYSTS_START stored procedure each time when a server is added or changed in QSYS2.SYSTEXTSERVERS table. If text search functions are not started, DB2 returns SQLCODE -20424 with reason code 4 for the CONTAINS and SCORE functions.

For the text search servers that are contained in the QSYS2.SYSTEXTSERVERS table, TCP/IP names are resolved. Multiple calls of the SYSPROC.SYSTS_START stored procedure are not considered an error because you might need to refresh the address resolution in the QSYS2.SYSTEXTSERVERS table.

If the server is a local server, then this call will start the server if it is not already started. If the server is a remote server, the procedure call will refresh the connection information, but will not actually start the server.

Prerequisites

Before you call the SYSPROC.SYSTS_START stored procedure, verify that the QSYS2.SYSTEXTSERVERS table contains at least one entry.

Authorization

The user ID under which this stored procedure is invoked must have the following privileges:

- *EXECUTE authority on the procedure
- SELECT and UPDATE privilege on the SYSTXTSRVR table.
- *EXECUTE authority on the QSYS2 library of the SYSTXTSRVR file.

Syntax

The schema qualifier is SYSPROC.

Parameter

serverid
Specifies the identifier of the server to be started. The value for *serverid* is an integer that must exist in the SERVERID column of the QSYS2.SYSTEXTSERVERS table. If *serverid* is not specified, the default is to start all servers.

Note: Only the servers that are identified as production servers (the parameter SERVERCLASS is set to 0 in the QSYS2.SYSTEXTSERVERS table) are started if no value is specified for *serverid*. Any test servers must be started by specifying the *serverid* that is associated with the test server.

SYSPROC.SYSTS_STOP

Call the SYSPROC.SYSTS_STOP stored procedure to stop DB2 text search functions. This stored procedure sets the SERVERSTATUS value in the catalog QSYS2.SYSTEXTSERVERS to 1 (stopped).

After this stored procedure runs successfully, SQL queries that use the CONTAINS function, the SCORE function, and administration stored procedures that are used for index maintenance will fail without trying to contact an IBM OmniFind Text Search Server for DB2 for i5/OS.

The stopped status is made persistent in the QSYS2.SYSTEXTSERVERS table, even if errors occur.

After you invoke the SYSPROC.SYSTS_STOP stored procedure, invocations of the SYSPROC.SYSTS_UPDATE stored procedure fail until you invoke the SYSPROC.SYSTS_START stored procedure again. Updates that are made to columns with a text search index continue to be captured by DB2, even while the SYSPROC.SYSTS_STOP stored procedure is running. SQL queries and administration stored procedures that are currently running are not affected.

Authorization

The user ID under which this stored procedure is invoked must have the following privileges:

- *EXECUTE authority on the procedure
- SELECT and UPDATE privileges on the SYSTEXTSERVERS table.
- *EXECUTE authority on the QSYS2 library of the SYSTEXTSERVERS file.

Syntax

The schema qualifier is SYSPROC.

Parameter

serverid
Specifies the identifier of the server to be started. The value for *serverid* is an integer that must exist in the SERVERID column of the QSYS2.SYSTEXTSERVERS table. If *serverid* is not specified, the default is to stop all servers.

SYSPROC.SYSTS_CREATE

Invoke the SYSPROC.SYSTS_CREATE stored procedure to enable a text column for text search indexing, which allows the text search index to be used in SQL queries that contain the CONTAINS function or the SCORE function.

The physical text search index is created on one of the text search servers that is listed in the QSYS2.SYSTEXTSERVERS table. The text search index that is maintained on the IBM OmniFind Text Search Server for DB2 for i5/OS is not updated synchronously when the DB2 table is updated. Instead, records of changes to the DB2 table column are captured by triggers to a staging table.

Tips:

- This stored procedure defines only the text search index. The text search index does not contain any data until after the first invocation of the SYSPROC.SYSTS_UPDATE stored procedure for the new text search index. You should create the text search index after the table is initially populated. By creating the text search index after the table is initially populated, you prevent the firing of change triggers before an initial index update.

Prerequisites

Before you invoke the SYSPROC.SYSTS_CREATE stored procedure, verify the following prerequisites:

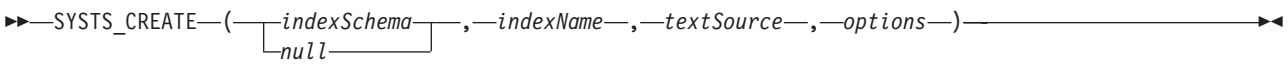
- DB2 text search functionality was started by invoking the SYSPROC.SYSTS_START stored procedure and at least one IBM OmniFind Text Search Server for DB2 is running.
- The table includes a column that is defined as primary key, unique index, or rowid.
- The QSYS2.SYSTEXTSERVERS table contains at least one entry.

Authorization

The user ID under which this stored procedure is invoked must have the following privileges:

- *EXECUTE on the procedure
- *EXECUTE to the library of the table that the Text Index is being created on.
- *EXECUTE and *ADD on the QSYS2 library.
- EXECUTE and *ADD on the library the that Text Index being created on.
- ALTER privilege on the table that the Text Index is being created on.
- INSERT, UPDATE, and DELETE privileges on the table that the Text Index is being created on.
- SELECT and UPDATE privileges on the SYSTEXTSERVERS table.
- WRITE privilege on the SYSTEXTINDEXES and SYSTEXTCOLUMNS tables in QSYS2.

Syntax



The schema qualifier is SYSPROC.

Parameters

indexSchema

Identifies the schema of the text search index. If this parameter is null, the value of the CURRENT SCHEMA special register for the invoker is used and this value must be a valid SQL name.

Note: Enclose names in double quotation marks if the names conflict with SQL keywords or Omnifind keywords that can be used.

The data type of this parameter is VARCHAR(128).

indexName

Identifies the name of the text search index. The name of the text search index with the index schema

uniquely identifies the text search index in the DB2 subsystem. You must specify a non-null value for this parameter and this value must be a valid SQL name.

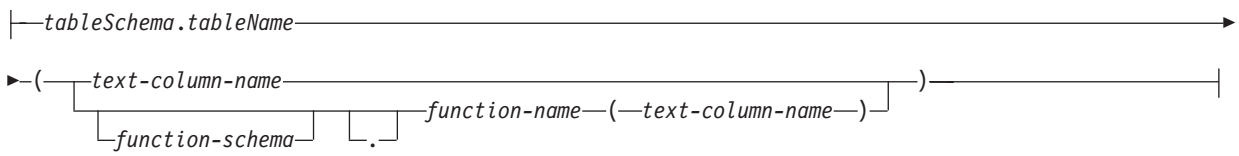
Note: Enclose names in double quotation marks if the names conflict with SQL keywords or Omnifind keywords that can be used.

The data type for this parameter is VARCHAR(128).

textSource
Identifies the table and column specification for the document text source. This parameter can include user-defined functions. You must specify a non-null value for this parameter.

The data type for this parameter is VARCHAR(1024).

textSource:



tableSchema
Identifies the schema of the table that the text search index is created on.

Note: Enclose names in double quotation marks if the names conflict with SQL keywords or Omnifind keywords that can be used.

tableName
Identifies the name of the text table that contains the column that the external text search index is created on.

Notes:

- Views, aliases, and logical files are not supported.
- Enclose names in double quotation marks if the names conflict with SQL keywords or Omnifind keywords that can be used.

text-column-name
Identifies the name of the column that contains the text that is used for creating the text search index. This column must be of type CHAR, CHAR FOR BIT DATA, BINARY, VARCHAR, VARCHAR FOR BIT DATA, VARBINARY, CLOB, DBCLOB, GRAPHIC, or VARGRAPHIC. If the data type of the column is not one of these data types, you can specify an external function that provides access to the text document that you want to index.

Notes:

- Only one text search index is allowed for a column. If a text search index already exists for the column, SQLCODE-20427 is returned.
- Enclose names in double quotation marks if the names conflict with SQL keywords or Omnifind keywords that can be used.

*function-schema***.** *function-name*
Identifies the schema and the name of a built-in or user-defined function that is to be used by IBM Text Search for DB2 for i5/OS to access text documents that are in a column that is not of a supported data type, or that are stored elsewhere. The function has one input parameter for the text column data type (for example, an integer that serves as a foreign key to the document content in another table), and it returns a value of one of the IBM Text Search for DB2 for i5/OS supported data types. The function transforms the text column content to the indexed document content.

Tip: For optimal performance, specify the ALLOW PARALLEL option when the function is created.

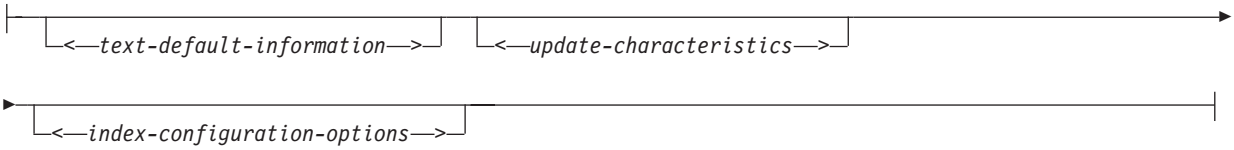
Notes:

- Cast functions and functions with more than one argument are not allowed.
- Enclose names in double quotation marks if the names conflict with SQL keywords or Omnifind keywords that can be used.

options
A character string that specifies the various options that are available for this stored procedure.

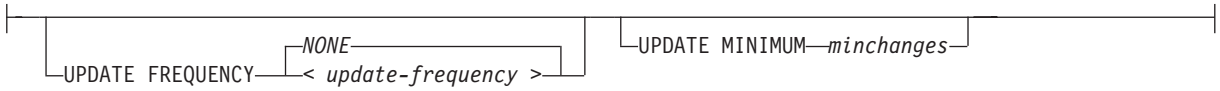
The data type for this parameter is VARCHAR(32000).

options:



text-default-information:

update-characteristics:



index-configuration-options:

text-default-information
Specifies the coded character set identifier that is used when indexing binary text documents, the language that is used when processing documents, and the format of text documents in the column.

CCSID *ccsid*
Specifies the coded character set identifier that is used for a text search index in a column with a binary data type. The default value is 1208 (UTF-8) and is taken from the QSYS2.SYSTEXTDEFAULTS table. All of the CCSIDs that are supported for conversion to UTF-8 by i5/OS conversion services are allowed for this parameter.

This parameter is ignored for a text search index in a column with a non-binary data type because text columns inherit the CCSID from the table specification. The *ccsid* value is ignored when the *format* value is set to INSO.

LANGUAGE *language*
Specifies the language that IBM OmniFind Text Search Server for DB2 for i5/OS uses for the

linguistic processing of text documents. The default value is en_US (English). If you specify the value for this parameter as AUTO, IBM OmniFind Text Search Server for DB2 for i5/OS tries to determine the language.

Important: If the language of the documents is not English, do not use the default value of en_US. Change the value to the language of the documents; otherwise, linguistic processing does not work as expected.

FORMAT *format*

Identifies the format of text documents in the column, such as HTML. The IBM OmniFind Text Search Server for DB2 for i5/OS needs to know the format, or content type, of the text documents that you intend to index and search. If you do not specify the **FORMAT** parameter, the default value is taken from the FORMAT column in the QSYS2.SYSTEXTDEFAULTS table. The supported *format* values are TEXT, HTML, XML, and INSO.

The *format* value INSO lets IBM OmniFind Text Search Server for DB2 for i5/OS determine the format. In this case, the *ccsid* value is ignored. If the IBM OmniFind Text Search Server for DB2 for i5/OS cannot determine the document format, then a document error is noted in the job log during processing by the SYSPROC.SYSTS_UPDATE stored procedure.

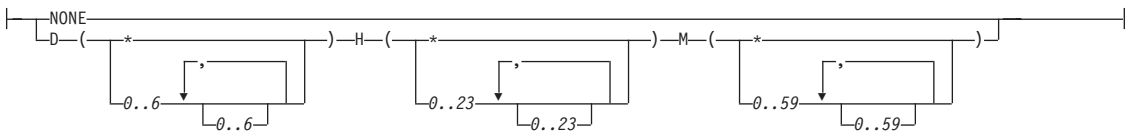
update-characteristics

Specifies the frequency of automatic updates to the text search index and the minimum number of changes to text documents before the text search index is updated incrementally at the specified time.

UPDATE FREQUENCY *update-frequency*

Specifies when to make automatic updates to the text search index. The default value is NONE. The format of the update-frequency option supports two different formats.

update-frequency (Format 1):



NONE

If NONE is specified, then no further index updates are made. The update must be started manually. This option might be useful for a text column in which no further changes are planned.

D Specifies the day or days of the week when the index is updated. An asterisk (*) specifies all days. 0 specifies Sunday.

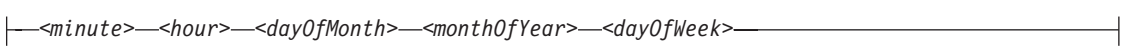
H Specifies the hour or hours when the index is updated. An asterisk (*) specifies all hours.

M Specifies the minute or minutes when the index is updated. An asterisk (*) specifies all minutes.

Example: This example specifies that the index update is to run every 30 minutes.

```
UPDATE FREQUENCY D(*) H(*) M(0,30)
```

update-frequency (Format 2, chronological):



The format of the *update-frequency (chronological)* option is a list of the five values separated by a blank space. The five values represent the minutes, the hours, the days of the month, the months of the year, and the days of the week beginning with Sunday.

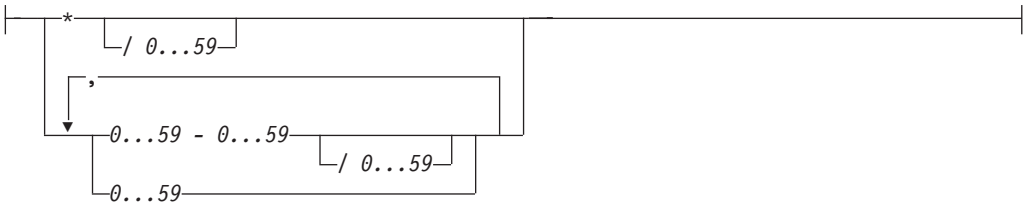
If you specify an interval of values, or an asterisk (*), you can use a forward slash (/) at the end of the defined interval to specify a step value for this interval.

Example: This example specifies that the index update is to run every quarter hour (`0,15,30,45`) on the even hours between 8 a.m. and 6:45 p.m. (`8-18/2` is equivalent to `8,10,12,14,16,18`), from Monday to Friday every month of the year (`* * 1-5`).

```
0,15,30,45 8-18/2 * * 1-5
```

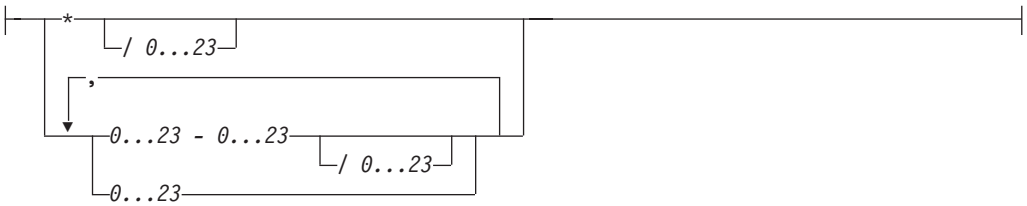
minute Specifies the minutes of the hour when the text search index is to be updated. You can specify an asterisk (*) for an interval of every five minutes, or you can specify an integer from 0 (zero) through 59. You cannot repeat values. The minimum update frequency is five minutes. A value of 1,4,8 is an invalid interval.

update-frequency (minute):



hour Specifies the hours of the day when the text search index is to be updated. You can specify an asterisk (*) for every hour, or you can specify an integer from 0 (zero) through 23. You cannot repeat values.

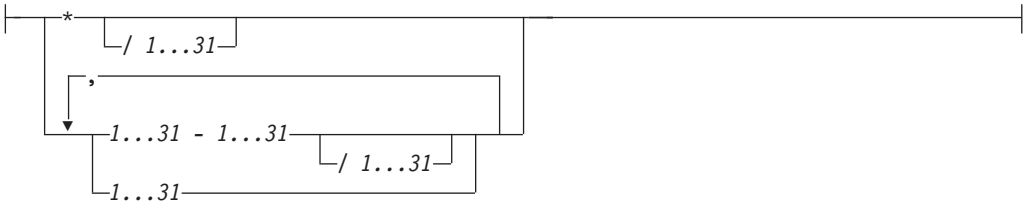
update-frequency (hour):



dayOfMonth

Note: The *dayOfMonth* parameter is allowed, but is ignored by the update scheduler. Specifies the days of the month when the text search index is to be updated. You can specify an asterisk (*) for every day, or you can specify an integer from 1 through 31. You cannot repeat values.

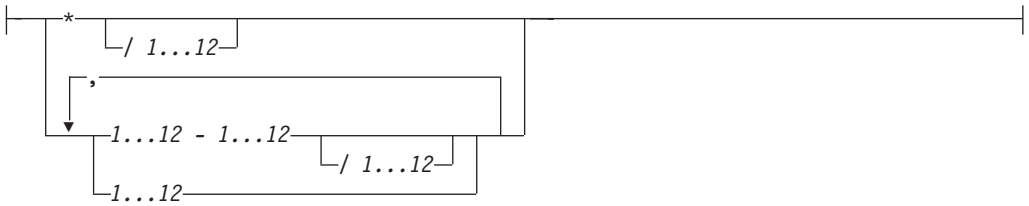
update-frequency (dayOfMonth):



monthOfYear

Note: The *monthOfYear* parameter is allowed, but is ignored by the update scheduler. Specifies the months of the year when the text search index is to be updated. You can specify an asterisk (*) for every month, or you can specify an integer from 1 through 12. You cannot repeat values.

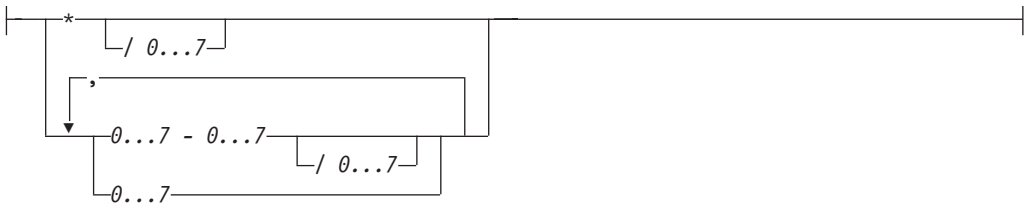
update-frequency (monthOfYear):



dayOfWeek

Specifies the days of the week when the text search index is to be updated. You can specify an asterisk (*) for every day, or you can specify an integer from 0 (zero) through 7. Both 0 and 7 are valid values for Sunday. You cannot repeat values.

update-frequency (dayOfWeek):



UPDATE MINIMUM *minchanges*

Specifies the minimum number of changes that are made to text documents before the text search index is updated incrementally at the time specified in the *update-frequency* option. The value must be an integer between 1 and 2147483647. The default value is taken from the UPDATEMINIMUM column in the QSYS2.SYSTEXTDEFAULTS table.

This option is ignored when you update the text search index, unless you specify the USING UPDATE MINIMUM option in the SYSPROC.SYSTS_UPDATE stored procedure.

index-configuration-options

Specifies additional index-specific values as option value pairs. You must enclose string values in single quotation marks. A single quotation mark within a string value must be represented by two consecutive single quotation marks.

CJKSEGMENTATION

Specifies the segmentation method to use when you index documents for CJK (Chinese, Japanese, Korean) languages. The supported values are MORPHOLOGICAL and NGRAM. If the CJKSEGMENTATION value is not specified, the default value is used. The default value is specified by the DEFAULTNAME value in the the QSYS2.SYSTEXTDEFAULTS table.

COMMENT

Specifies a comment that is stored in the REMARKS column of the QSYS2. SYSTEXTINDEXES administration table and as the description of the IBM OmniFind Text Search Server for DB2 for i5/OS collection.

The value for this option is a string value that is less than or equal to 512 bytes.

IGNOREEMPTYDOCS
Specifies whether to represent empty documents (documents with an empty string or a null value) in the text search index.

The supported values for this option are 0 (zero) and 1. The default value is 1.

If this option is set to 1, empty documents are not represented in the text search index. If you use this option and change the document content to empty, the next incremental update deletes the documents from the text search index.

KEYCOLUMN
Specifies the name of a unique column to be used as the key column in the text index. The key column is used to associate data in the text index to a document/row in the base table. The default if KEYCOLUMN is not specified is to use the ROWID column from the table if one exists, otherwise to use the primary key defined on the table. The specified column must have a primary key constraint or unique index.

SERVER
Specifies the ID of the server to be used to store the text search index. The value is an integer that must exist in the SERVERID column of the QSYS2.SYSTEXTSERVERS catalog. If SERVER is not specified, the default is to choose the server with the fewest text search indexes from the servers in the QSYS2.SYSTEXTSERVERS table where parameter SERVERSTATUS is set to 0 (zero), which means that the server is available.

UPDATEAUTOCOMMIT
Specifies how often a commit operation is performed when fetching documents during an index update. A value of 0 (zero) means that a commit operation occurs only at the end of processing.

The value must be an integer between 0 (zero) and 2147483647. The default value is 100.

If update processing takes a very long time and DB2 active logs are small, consider using the default value. Otherwise, the active log entries of customer transactions that run in parallel to the text search index update cannot be cleared and might lead to a full the active log.

Default values for the *options* parameter

When you install IBM Text Search for DB2 for i5/OS, the QSYS2.SYSTEXTDEFAULTS table is created and populated with default values for the *options* parameter of the SYSPROC.SYSTS_CREATE stored procedure.

The following table lists the options, default values, and descriptions of the options.

Table 2. Default values for the options parameter

Option	Default value	Description
CCSID	1208	Specifies the coded character set identifier that is used when binary text documents are indexed.
CJKSEGMENTATION	NGRAM	Specifies the segmentation method to use when you index documents for CJK (Chinese, Japanese, Korean) languages.
LANGUAGE	en_US	Specifies the language that IBM OmniFind Text Search Server for DB2 for i5/OS uses to process text documents.
FORMAT	TEXT	Identifies the format of text documents in the column. The default format is plain text.
UPDATEFREQUENCY	NONE	Indicates that no automatic updates are scheduled.

Table 2. Default values for the options parameter (continued)

Option	Default value	Description
UPDATEMINIMUM	1	If at least one document changed since the last index update, the SYSPROC.SYSTS_UPDATE stored procedure starts processing.
IGNOREEMPTYDOCS	1	Specifies that empty documents (documents with an empty string or a null value) are not represented in the text search index. The metadata fields for these documents are not available for search.
UPDATEAUTOCOMMIT	100	Specifies how often a commit operation is performed when documents are fetched during an index update. If update processing takes a very long time and DB2 active logs are small, consider using the default value. Otherwise, the active log entries of customer transactions that run in parallel to the text search index update cannot be cleared and might lead to a full active log.
MINIMUMUPDATEINTERVAL	5	Specifies the intervals for the UPDATEFREQUENCY option. Intervals cannot be shorter than five minutes.
USEREXITTHREADS	0	Reserved

SYSPROC.SYSTS_DROP

Invoke the SYSPROC.SYSTS_DROP stored procedure to drop a text search index that was defined by using the SYSPROC.SYSTS_CREATE stored procedure.

Be aware that dropping the table in DB2 does not drop a text search index. You must drop a text search index by using the SYSPROC.SYSTS_DROP stored procedure either before or after dropping the table. The table does not need to exist to invoke this stored procedure; therefore, you can invoke this stored procedure after a table is dropped.

If the text search server cannot be reached, the collection on the server might become orphaned. If that happens, the collection needs to be deleted manually. When the server is available again, use the IBM OmniFind Text Search Server for DB2 for i5/OS administration tool to delete the collection on the server. In Chapter 6, "Administering an IBM OmniFind Text Search Server for DB2 for i5/OS," on page 39, you can find information about the tools to identify orphaned indexes and the stored procedure STSPROC.SYSTS_REMOVE to delete orphaned indexes.

Prerequisites

Before you invoke the SYSPROC.SYSTS_DROP stored procedure, verify the following prerequisites:

- DB2 text search functionality was started by invocation of the SYSPROC.SYSTS_START stored procedure.
- The text search index was created (by invocation of the SYSPROC.SYSTS_CREATE stored procedure).
- Ensure that the following stored procedures are not running for the text search index that you want to drop: SYSPROC.SYSTS_CREATE, SYSPROC.SYSTS_UPDATE, and SYSPROC.SYSTS_DROP.

Authorization

The user ID under which this stored procedure is invoked must have the following privileges:

- *EXECUTE on the procedure.
- *EXECUTE on the library of the table where the Text Index is being created.
- *EXECUTE and *ADD on the QSYS2 library.
- *EXECUTE and *ADD on the library where the Text Index is being created.
- ALTER privilege on the table where the Text Index is being created.
- INSERT, UPDATE, and DELETE privileges on the table where the Text Index is being created.
- DELETE privilege on the SYSTEXTINDEXES and SYSTEXTCOLUMNS tables in QSYS2.
- *OBJOPR and *OBJEXIST on the staging table and the SQL View.

Syntax

The schema qualifier is SYSPROC.

Parameters

indexSchema
Identifies the schema of the text search index. If this parameter is null, the value of the CURRENT SCHEMA special register for the invoker is used.

The data type of this parameter is VARCHAR(128).

indexName
Identifies the name of the text search index. The name of the text search index together with the index schema uniquely identifies the text search index in the DB2 subsystem. You must specify a non-null value for this parameter.

The data type for this parameter is VARCHAR(128).

SYSPROC.SYSTS_UPDATE

Invoke the SYSPROC.SYSTS_UPDATE stored procedure to update the text search index to reflect the current contents of the text column that the text search index is associated with.

Because updating a text search index is an extensive operation, the text search index that is maintained on the IBM OmniFind Text Search Server for DB2 for i5/OS is not updated synchronously when the DB2 table is updated. Instead, changes to the DB2 table column are captured by a trigger to a staging table. The text search index on the IBM OmniFind Text Search Server for DB2 for i5/OS is updated the next time that you invoke the SYSPROC.SYSTS_UPDATE stored procedure. Therefore, some search requests might not reflect recent updates to the table.

This stored procedure returns only after all of the update processing for the text search index on the IBM OmniFind Text Search Server for DB2 for i5/OS is completed. The duration depends on the number of entries in the staging table at the time the SYSTS_UPDATE was called. During the update process, the text search index remains searchable.

If documents contain errors, you can view them. The error is indicated in the staging table with the TOBEDELETED value set to either `E` (error) or `W` (warning). You can retrieve these records with a query such as the following one:

```
SELECT based_on_columns
FROM based_on_table INNER JOIN QSYS2.stagingtable
ON (QQQ_TEXTSEARCH_KEY(k1, k2, k3, ...) = KEYID)
```

In this case, `based_on_columns` is the column list that you need; `based_on_table` is the user's table; `stagingtable` is the staging table listed in the catalogs for the text search index; and `k1, k2, k3, ...` is the list of key columns in the primary key, row ID, or unique key that is used to build the text search index. You can correct the errors for those documents, and run update again.

Note: If a synonym dictionary has been created for the text search index, this process will remove the dictionary.

Be aware that the IBM OmniFind Text Search Server for DB2 for i5/OS might block the index update process if multiple index updates are being performed at the same time. Only one update is allowed to run at a time per index.

Prerequisites

Before you invoke the SYSPROC.SYSTS_UPDATE stored procedure, verify the following prerequisites:

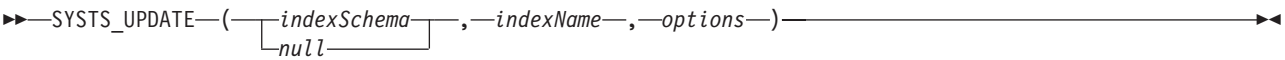
- DB2 text search functionality was started by invocation of the SYSPROC.SYSTS_START stored procedure.
- The text search index was created (by invocation of the SYSPROC.SYSTS_CREATE stored procedure).
- The following stored procedures are not running for the text search index that you want to update: SYSPROC.SYSTS_CREATE, SYSPROC.SYSTS_UPDATE, and SYSPROC.SYSTS_DROP.
- At least one IBM OmniFind Text Search Server for DB2 is running.

Authorization

The user ID under which this stored procedure is invoked must have the following privileges:

- *EXECUTE on the procedure.
- *EXECUTE on the library of the table where the Text Index is being created.
- *EXECUTE on the QSYS2 library.
- *EXECUTE on the library where the Text Index is being created.
- SELECT and UPDATE privileges on the SYSTEXTSERVERS table.
- SELECT, UPDATE, and DELETE privileges on the staging table.
- SELECT privilege on the table where the Text Index is being created.
- WRITE privilege on the SYSTEXTINDEXES table in QSYS2.

Syntax



The schema qualifier is SYSPROC.

Parameters

indexSchema
Identifies the schema of the text search index. If this parameter is null, the value of the CURRENT SCHEMA special register for the invoker is used.

The data type of this parameter is VARCHAR(128).

indexName
Identifies the name of the text search index. The name of the text search index together with the index schema uniquely identifies the full-text index in the DB2 subsystem. You must specify a non-null value for this parameter.

The data type for this parameter is VARCHAR(128).

options
A character string that specifies the option that is available for this stored procedure.

The available option is USING UPDATE MINIMUM. This option uses the USING UPDATE MINIMUM settings that you specified for the SYSPROC.SYSTS_CREATE stored procedure and starts an incremental update only if the specified number of changes was reached. The default is to unconditionally start the update process.

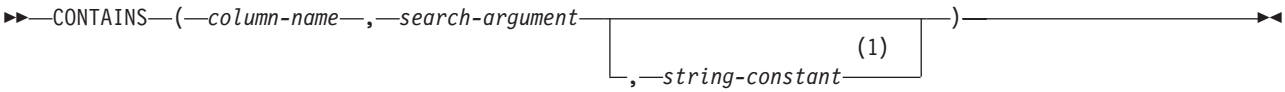
USING UPDATE MINIMUM:

Chapter 5. User-defined functions and search argument syntax

IBM Text Search for DB2 for i5/OS includes support for the CONTAINS function and the SCORE function in an SQL statement to search a text search index using the search argument criteria that you specify.

CONTAINS

The CONTAINS function searches a text search index using criteria that are specified in a search argument and returns a result about whether or not a match was found.



Notes:

1 *string-constant* must conform to the rules for the *search-argument* options.

search-argument-options:

Notes:

1 The same clause must not be specified more than once.

The schema is QSYS2.

column-name
Specifies a qualified or unqualified name of a column that has a text search index that is to be searched. The column must exist in the table or view that is identified in the FROM clause in the statement and the column of the table, or the column of the underlying base table of the view must have an associated text search index (SQLSTATE 38H12). The underlying expression of the column of a view must be a simple column reference to the column of an underlying table, directly or through another nested view.

search-argument
Specifies an expression that returns a value that is a string value that contains the terms to be searched for and must not be all blanks or the empty string (SQLSTATE 38H14). The actual length of the string must not exceed 32704 bytes. This length might be further limited by what is supported by the text search server (SQLSTATE 38H10). The value is converted to Unicode before it is used to search the text search index.

string-constant
Identifies a string constant that specifies the search argument options that are in effect for the function.

The options that can be specified as part of the *search-argument-options* are as follows:

QUERYLANGUAGE = *value*
Specifies the query language. The value can be any of the supported language codes. If the QUERYLANGUAGE option is not specified, the default is the language value of the text search index that is used when this function is invoked. If the language value of the text search index is AUTO, the default value for QUERYLANGUAGE is en_US.

RESULTLIMIT = *value*
Specifies the maximum number of results to be returned from the underlying search engine. The *value* can be an integer value between 1 and 2 147 483 647. If the RESULTLIMIT option is not specified, no result limit is in effect for the query.

This scalar function may not be called for each row of the result table, depending on the plan that the optimizer chooses. This function can be called once for the query to the underlying search engine, and a result set of all of the primary keys that match are returned from the search engine. This result set is then joined to the table containing the column to identify the result rows. In this case, the RESULTLIMIT value acts like a FETCH FIRST ?? ROWS from the underlying text search engine and can be used as an optimization. If the search engine is called for each row of the result because the optimizer determines that is the best plan, then the RESULTLIMIT option has no effect.

SYNONYM = *OFF* **or SYNONYM =** *ON*
Specifies whether to use a synonym dictionary that is associated with the text search index. You can add a synonym dictionary to a collection by using the synonym tool.

OFF OFF is the default.

ON Use the synonym dictionary that is associated with the text search index.

The result of the function is a large integer. If the second argument can be null, the result can be null. If the second argument is null, the result is the null value.

The result is 1 if the document contains a match for the search criteria that are specified in the search argument. Otherwise, the result is 0. The result is also 0 if the column is null. If the search argument is Null, then the result is the null value.

CONTAINS is a nondeterministic function.

Example 1

The following statement finds all of the employees who have ″COBOL″ in their resume. Because COBOL is always in uppercase, delimit the search term in double quotation marks so that only uppercase versions of the word will be matched.

```
SELECT EMPNO
   FROM EMP_RESUME
   WHERE RESUME_FORMAT = 'ascii'
      AND CONTAINS(RESUME, '"COBOL"') = 1
```

Example 2

The search argument does not need to be a string constant. The search argument can be any SQL string expression, including a string contained in a host variable. The following statement searches for the exact term ″ate″ in the COMMENT column.

```
char search_arg[100]; /* input host variable */
...
EXEC SQL DECLARE C3 CURSOR FOR
   SELECT CUSTKEY
   FROM K55ADMIN.CUSTOMERS
   WHERE CONTAINS(COMMENT, :search_arg)= 1
```

```
   ORDER BY CUSTKEY;
strcpy(search_arg, "ate");
EXEC SQL OPEN C3;
...
```

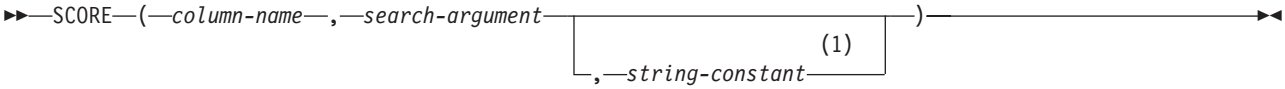
Example 3

The following statement finds 10 students at random who wrote online essays that contain the phrase "fossil fuel" in Spanish, which is "combustible fósil." These students will be invited for a radio interview. Use the synonym dictionary that was created for the associated text search index. Because only 10 students are needed, optimize the query by using the RESULTLIMIT option to limit the number of results from the underlying text search server.

```
SELECT FIRSTNME, LASTNAME
   FROM STUDENT_ESSAYS
   WHERE CONTAINS(TERM_PAPER, 'combustible fósil',
      'QUERYLANGUAGE= es_ES RESULTLIMIT  = 10 SYNONYM=ON') = 1
```

SCORE

The SCORE function searches a text search index using criteria that are specified in a search argument and returns a relevance score that measures how well a document matches the query.



Notes:

1 *string-constant* must conform to the rules for the *search-argument* options.

search-argument-options:

Notes:

1 The same clause must not be specified more than once.

The schema is QSYS2.

column-name
Specifies a qualified or unqualified name of a column that has a text search index that is to be searched. The column must exist in the table or view that is identified in the FROM clause in the statement and the column of the table, or the column of the underlying base table of the view must have an associated text search index (SQLSTATE 38H12). The underlying expression of the column of a view must be a simple column reference to the column of an underlying table, either directly or through another nested view.

search-argument
Specifies an expression that returns a value that is a string value that contains the terms to be searched for and must not be all blanks or the empty string (SQLSTATE 38H14). The actual length of the string must not exceed 32704 bytes. This length might be further limited by what is supported by

the text search server (SQLSTATE 38H10). The value is converted to Unicode before it is used to search the text search index. If the search-argument is null, the result is the null value.

string-constant
Identifies a string constant that specifies the search argument options that are in effect for the function.

The options that can be specified as part of the *search-argument-options* are as follows:

QUERYLANGUAGE = *value*
Specifies the query language. The value can be any of the supported language codes. If the QUERYLANGUAGE option is not specified, the default is the language value of the text search index that is used when this function is invoked. If the language value of the text search index is AUTO, the default value for QUERYLANGUAGE is en_US.

RESULTLIMIT = *value*
Specifies the maximum number of results that are to be returned from the underlying search engine. The *value* can be an integer value between 1 and 2 147 483 647. If the RESULTLIMIT option is not specified, no result limit is in effect for the query.

This scalar function may not be called for each row of the result table, depending on the plan that the optimizer chooses. This function can be called once for the query to the underlying search engine, and a result set of all of the primary keys that match are returned from the search engine. This result set is then joined to the table containing the column to identify the result rows. In this case, the RESULTLIMIT value acts like a FETCH FIRST ?? ROWS from the underlying text search engine and can be used as an optimization. If the search engine is called for each row of the result because the optimizer determines that is the best plan, then the RESULTLIMIT option has no effect.

SYNONYM = *OFF* **or SYNONYM =** *ON*
Specifies whether to use a synonym dictionary that is associated with the text search index. You can add a synonym dictionary to a collection by using the synonym tool.

OFF OFF is the default.

ON Use the synonym dictionary that is associated with the text search index.

The result of the function is a double-precision floating-point number. If the second argument can be null, the result can be null. If the second argument is null, the result is the null value.

The result is greater than 0 but less than 1 if the column contains a match for the search criteria that the search argument specifies. The more frequently a match is found, the larger the result value. If the column does not contain a match, the result is 0. The score is also 0 if the column is null.

SCORE is a nondeterministic function.

Example

The following statement generates a list of employees in the order of how well their resumes matches the query ″programmer AND (java OR cobol)″, along with a relevance value that is normalized between 0 (zero) and 100.

```
SELECT EMPNO, INTEGER(SCORE(RESUME, 'programmer AND
 (java OR cobol)') * 100) AS RELEVANCE
 FROM EMP_RESUME
 WHERE RESUME_FORMAT = 'ascii'
 ORDER BY RELEVANCE DESC
```

Search argument syntax

A search argument is the condition that is specified as part of a search for terms in text documents. It consists of search parameters and one or more search terms. The SQL scalar text search functions that use search arguments are CONTAINS and SCORE.

For any language-specific processing during a search, you can specify a value for the QUERYLANGUAGE parameter as a search argument option. The value can be any of the supported language codes. If the QUERYLANGUAGE parameter is not specified, the default value is the language value of the text search index that is used when this function is invoked. If the language value of the text search index is AUTO, the default value for QUERYLANGUAGE is en_US.

Limitations

You cannot use the CONTAINS and SCORE functions in an SQL constraint or index definition. You can use them in SQL query statements and view definitions under the following restrictions:

- If a view, nested table expression, or common table expression provides a text search column for a CONTAINS or SCORE scalar function, and if the applicable view, nested table expression, or common table expression has a DISTINCT clause on the outermost SELECT statement, then the SELECT list must contain all the corresponding key fields of the text search index. Otherwise, SQL message 38H12 is returned.
- If a view, nested table expression, or common table expression provides a text search column for a CONTAINS or SCORE scalar function, the applicable view, nested table expression, or common table expression cannot have a UNION, an EXCEPT, or an INTERSECT statement at the outermost SELECT level. Otherwise, SQL message 38H12 is returned.
- If a common table expression provides a text search column for a CONTAINS or SCORE scalar function, the common table expression can be referenced again in the entire query only when the reference does not provide a text search column for a CONTAINS or SCORE scalar function. Otherwise, SQL message 38H12 is returned.
- A function cannot be created sourced on the CONTAINS or SCORE scalar functions. Otherwise, SQL message SQL0457 is returned.
- The query can run through the SQL Query Engine (SQE).

Simple search

To do a simple keyword search, you can enter one or more query terms (keywords). The search engine returns documents that contain all of those keywords, or variations of the keywords.

For example, if you enter `king`, the search engine returns all documents that contain the word `king` or `kings`. If you enter the query `king lear`, the search engine returns documents that contain the terms `king` and `lear`.

To see more precise results, use more specific keywords. For example, use `French roast coffee` rather than `coffee`, or use `Kauai hiking tours` rather than `Hawaiian vacations`.

If a simple keyword search returns too many documents that are not what you are looking for, you can use operators to refine your search.

Exclusion of terms in a search

Use the minus sign (-) to exclude terms. For example, if you want to find documents with the term `lear`, and you do not want to see documents with `edward`, enter the query `lear -edward`.

The minus sign (-) also applies to a term and its variants. For example, the query `-edward` will exclude documents that contain the word `edward's`.

Phrase search

If you want to ensure that terms appear exactly in the sequence that you typed them in, you can use double quotation marks. For example, if you want to see documents with the term `king lear` exactly, and you do not want matches on related phrases such as `kingly lear` or `king and queen lear`, enter `"king lear"`. The search is not case-sensitive, but term variants are not considered matches.

Wildcard character in a search

The wildcard character (*) helps you find documents when you do not know the full spelling, or if you want to find variations of the term. For example, the query `czech*` returns documents with the terms `czech`, `czechoslovakia`, `czechoslovakian`, `czech republic`, and other possible results.

You can also use the wildcard character in a phrase search. For example, the query `"John * Kennedy"` returns documents with the terms `John Fitzgerald Kennedy` and `John F Kennedy` but not `John Kennedy`. The query `Mi*l Gorbachev` will return `Mikhail Gorbachev`.

Adding a wildcard character to the beginning of a query (for example, `*zech`) might cause the search engine to take longer to return results.

Searches for at least one of the terms

The logical operator OR specifies that at least one of the terms in a query must appear in the returned document. For example, the query `(othello OR otello)` returns documents that contain the term `othello` or `otello`.

You can also use the logical operators AND, OR, and NOT in combinations by using parentheses. For example, the query `cougar OR (jaguar AND NOT car)` returns documents with the terms `cougar` or `jaguar` but not `car`.

You must enter the logical operators AND, OR, and NOT in all uppercase. Use parentheses for grouping.

Simple query examples

Simple queries for the CONTAINS function and the SCORE function search for a single word or multiple words in a text search index.

The search engine ignores white space between characters. The search string must not be empty or contains all blanks (SQLSTATE 38H14).

The following table shows some examples of simple search queries.

Table 3. Simple query examples

Search word types	Examples	Query results
Single word	`king`	Returns all documents that contain the word `king` or `kings`. This query matches different surface forms and is not case-sensitive.
Multiple words	`king lear`	Returns all documents that contain `king` and `lear`. The default operator is the logical operator AND.

The operators AND and + (plus sign) are implicit in every query. For example, the query `King Lear` returns the same results as `King AND Lear` or `King + Lear`.

You must enter the logical operators NOT, AND, and OR in all uppercase.

Advanced search operators

You can use advanced search operators to refine the search results for the CONTAINS function and the SCORE function.

In the following table, the first column describes the operator that you can use in a search query. (You must enter the logical operators NOT, AND, and OR in all uppercase letters.) The second column shows a sample query that you might enter. The third column describes the types of results that you might see from the example query.

Table 4. Advanced search operators and complex query examples

Operators	Examples	Query results
AND	`"King Lear" AND "Othello"` `"King Lear" "Othello"`	Either query returns documents that contain both terms `King Lear` and `Othello`. The AND operator is the default conjunction operator. If no logical operator is between the two terms, the AND operator is used. For example, the query `King Lear` is the same as the query `King AND Lear`.
OR	`"King Lear" OR Lear`	Returns documents that contain either `King Lear` or just `Lear`. The OR operator links the two terms and finds a matching document if either of the terms exist in a document.
NOT	`"King Lear"` `NOT "Norman Lear"`	Returns documents that contain `King Lear` but not `Norman Lear`.

" "
 	
(Exact match)
	First query:
	`"King Lear"`
	
	Second query:
	`"king"`
		The first query returns the exact phrase `King Lear`.
		
		The second query returns only the word `king` and no other forms, such as `kings` or `kingly`.

* (Wildcard character)	`test*` `te*t`	Returns documents that can match possible combinations, such as `test`, `tests`, and `tester`, or `test` and `text`.

^
 	
(Score boost factor)
some word or phrase^number
	First query:
	`"King Lear"^4 "Richard III"`
	
	Second query:
	`title: (software download)^5`
	`pdf viewer -shipping`
		The first query forces documents with the phrase `King Lear` to appear higher in the list of search results.
		
		The second query forces a document titled `software download` to appear higher in the list of results.
		
		Although a boost factor must be positive, the boost factor can be less than 1. For example, 0.2. The boost factor number has no limit.

+ (Includes)	`+Lear King`	Returns all documents that contain `Lear` and `King`, which is the same as the query `Lear AND King`.
- (Excludes)	`"King Lear" -"Lear Jet"`	Returns documents that contain `King Lear` but not `Lear Jet`.

Table 4. Advanced search operators and complex query examples (continued)

Operators	Examples	Query results
( )	`(King OR Lear) AND plays`	Returns documents that contain either `King` or `Lear` and `plays`. The parentheses ensure that `plays` is found and either term `King` or `Lear` is present.
\ (Escape character)	`\(1\+1\)\:2`	Returns documents that contain `(1+1):2`. Use the \ to clear special characters that are part of the query syntax. Special characters are + - && || ! ( ) { } [ ] ^ " ~ * ? : \. If a special character is cleared, the special character is analyzed as part of the query.

Example that uses the CONTAINS function and SCORE function

You can use the CONTAINS function or the SCORE function in the same query to search a text search index and return if and how frequently the text document matches the search argument criteria.

The example in the following table uses data from the base table BOOKS with the columns ISBN (VARCHAR(20)), ABSTRACT (VARCHAR(10000)), and PRICE (INTEGER).

Table 5. The base table BOOKS

ISBN	ABSTRACT	PRICE
i1	"a b c"	7
i2	"a b d"	10
i3	"a e a"	8

You run the following query:

```
SELECT ISBN, SCORE(ABSTRACT,'"b"')
FROM BOOKS
WHERE CONTAINS (ABSTRACT,'"b"') = 1
```

This query returns the following two rows:

```
i1, 0.3
i3, 0.4
```

The score values might differ depending on the content of the text column.

XML search

By using a subset of the XPath language with extensions for text search, XML search allows you to index and search XML documents so that structural elements can be used separately, or combined with free text in queries.

Structural elements are tag names, attribute names, and attribute values.

The following list highlights the key features of XML search:

XML structural search

By including special opaque XML terms in queries, you can search XML documents for structural elements (tag names, attribute names, and attribute values) and text that is scoped by those elements.

XML query tokenization
Free text in XML query terms is tokenized the same way that text in non-XML query terms is tokenized, except that spelling correction, fielded terms, and (nested) opaque terms are not supported. Synonyms, wildcard characters, phrases, and lemmatization are supported.

XML namespaces are disregarded
Namespace prefixes are not retained in the indexing of XML tag and attribute names. XML documents that declare and use namespaces can be indexed and searched, but namespace prefixes are discarded during indexing and removed from XML search queries.

Numeric values
Predicates that compare attribute values to numbers are supported.

Complete match
The = (equal sign) operator with a string argument in a predicate calls for a complete match of all tokens in the string with all tokens in the identified text span. The order is significant.

No UIMA access
Unstructured Information Management Architecture (UIMA) is used for tokenization in XML search, but user-written annotators are not supported.

XML search configuration

You can configure some of the settings that are used for XML search. All of the parser configuration parameters are in the parser_config.xml file in the XML element that defines the parser com.ibm.es.nuvo.parser.xml.XMLParser.

Each parameter is specified by a parameter element, as shown in the following example:

```
<Parameter Name="parameter-name">setting</Parameter>
```

If no parameter elements are present, then the default settings are used. The following table shows the valid parameter names and settings.

Table 6. Valid parameter names and settings for XML search

Parameter name	Setting	Result
handleExternalFiles	ignore	External files are not downloaded. The files are resolved as if they were empty. Parsing is faster, but externally defined entities cannot be indexed. This is the default setting.
	read	External files are downloaded if they are accessible, with a consequent increase in parsing time. If a file is not accessible, an I/O exception is thrown by the parser, and the URI of the affected document is indexed with an error code.
handleSkippedEntities	ignore	Unresolved references are silently ignored during parsing. This is the default setting.
	reject	A document containing an unresolved entity is rejected, and an error entry is stored in the index for its URI.

Table 6. Valid parameter names and settings for XML search (continued)

Parameter name	Setting	Result
titleTagNameList	absent	The only tag name that is used to identify the title content is *title* (not case-sensitive). This is the default setting.
	empty	No tag name is used to identify the title content.
	single tag name	Tag names that match the specified tag name (not case-sensitive) are used to identify document titles.
	comma-separated list	Tag names that match any name in the list (not case-sensitive) are used to identify document titles.

XML search query grammar

A subset of the XPath language, which is defined by an Extended Backus-Naur Form (EBNF) grammar, is supported by the XML search query parser. Queries that do not conform to the supported grammar are rejected by the query parser, which throws an exception.

The EBNF grammar has been simplified in the following ways by:

- Removing the largest-scale structures for specifying iteration and ranges
- Eliminating filter expressions
- Disallowing absolute pathnames in predicate expressions
- Recognizing only one axis (tag) and only in the forward direction

The following table shows the supported grammar in EBNF notation.

Table 7. Supported query grammar in EBNF notation

XMLQuery	::=	QueryPrefix QueryString
QueryPrefix	::=	"@xpath:"
QueryString	::=	"'" PathExpr "'"
PathExpr	::=	RelativePathExpr | "/" RelativePathExpr? | "//" RelativePathExpr
RelativePathExpr	::=	StepExpr (("/" | "//") StepExpr)*
StepExpr	::=	("." | AbbrevForwardStep) Predicate?
AbbrevForwardStep	::=	"@"? (QName | "*")
Predicate	::=	"[" PredicateExpr "]"
PredicateExpr	::=	Expr | PredicateExpr ("and" | "or") PredicateExpr | "(" PredicateExpr ")"
Expr	::=	ComparisonExpr | ContainmentExpr
ComparisonExpr	::=	PathExpr ComparisonOp Literal
ComparisonOp	::=	"=" | "<" | ">" | "!=" | "<=" | ">="
Literal	::=	StringLiteral | NumericLiteral
ContainmentExpr	::=	PathExpr "contains" "(" StringLiteral ")" | PathExpr "excludes" "(" StringLiteral ")"
StringLiteral	::=	"\"" [^"]* "\"" | "'" [^']* "'"

For more information about QName, see http://www.w3.org/TR/REC-xml-names/#NT-QName.

The following information about XML search queries that use XPath notation might not be obvious from the EBNF grammar notations:

- **Names not normalized:** XML tag and attribute names are not normalized when they are indexed. The names are not changed to lowercase, tokenized, or modified in any way. Case is significant in XML tag and attribute names to get a match. Therefore, the strings that are used for XML tag and attribute names in queries must match exactly the names that appear in the source documents.
- **Namespace handling:** Documents with XML namespace prefixes can be indexed, but namespace prefixes are not retained in the index. For example, the tag "<nsdoc:heading ...>" is indexed under "heading" only. Namespace prefixes are also removed from element names in XML search queries. As a result, a query that specifies the tag name "nsdoc:heading" is parsed to produce a query that looks for the tag name "heading."
- **Free text normalization:** Free text in XML documents (text between tags, not inside a tag itself) and attribute values are normalized before indexing by using a UIMA text-analysis engine and the Frost tokenizer. Text in XML search queries (in *contains* or *excludes* operators, or in strings that are surrounded by quotation marks) is normalized, too. Features such as phrases, synonyms, wildcard characters, and lemmas are supported.
- **Operator precedence:** In XML search predicates, containment operators and comparison operators take precedence over logical operators, and all logical operators have the same precedence. Containment operators are *contains* and *excludes*. Comparison operators are =, !=, <, >, <= and >=. Logical operators are ″and″ and ″or.″ You can use parentheses to ensure the desired precedence.
- **Semantics:** In XML search predicates, the comparison operators can be applied only to attribute values and not to tags.

XPath query examples

All valid XPath queries that are sent to the XML parser must be written in a subset of the XPath language using opaque terms. Opaque terms are not parsed by the linguistic query parser.

The query parser recognizes an opaque term by the syntax that is used in the query. For example:

```
@xpath:'query'
```

where *query* is the text shown in the examples in the following table.

Table 8. Examples of valid XPath queries

Query	Description
/	The root node; any document.
/*	Any document with any top-level tag.
//*	Any document with any tag at any level.
/sentences	Any document with a top-level tag called *sentences*.
//sentences	Any document with a tag at any level called *sentences*.
sentences	Any document with a tag at any level called *sentences*.
/sentence/paragraph	Any document with a top-level tag *sentence* having a direct child tag *paragraph*.
/sentence/paragraph/	Any document with a top-level tag *sentence* having a direct child tag *paragraph*.
/book/@author	Any document with a top-level *book* tag having an attribute *author*.
/book//@author	Any document with a top-level *book* tag having a descendant tag at any level with the attribute *author*.

Table 8. Examples of valid XPath queries (continued)

Query	Description
/book[@author contains(″barnes″) and @title = ″the lemon table″]	Any document with a top-level *book* tag with an *author* attribute containing ″barnes″ (normalized) and a *title* attribute that only contains the words ″the,″ ″lemon″ and ″table″ (normalized in that order).
/book[@author contains(″barnes″) and (@title contains(″lemon″) or @title contains(″flaubert″))]	Any document with a top-level *book* tag with the specified *author* attribute and either of the two specified *title* attributes.
/program[. contains(″″″hello, world.″″″)	Any document with a top-level *program* tag containing both the tokens *hello* and *world* (normalized) in that order and in consecutive positions.
/book[paragraph contains(″foo″)]//sentence	Any document with a top-level *book* tag with a direct child tag *paragraph* containing ″foo″ and, referring to the *book* tag, having a descendant tag *sentence* at any level.
/auto[@price < 30000.]	Any document with a top-level *auto* tag having an attribute *price* with a numeric value that is less than 30000.
//microbe[@size < 3.0e-06]	Any document containing a *microbe* tag at any level with a *size* attribute with a value that is less than .000003.

Chapter 6. Administering an IBM OmniFind Text Search Server for DB2 for i5/OS

Administer the IBM OmniFind Text Search Server for DB2 for i5/OS server using the following information.

The SQL catalogs for the text search indexes are in library QSYS2. The catalog names all begin with SYSTXT, for example SYSTXTSRVR. Like the other SQL catalogs in QSYS2, it is your responsibility to ensure a backup copy is saved and available. This can be accomplished in one of two ways:

- The entire library can be saved as part of the SAVLIB command, specifying parameter LIB as either *ALLUSR or *IBM.
- The specific text search catalogs can be saved using the SAVOBJ command, specifying LIB(QSYS2) and OBJ((SYSTXT*)).

Saving and restoring a text search index requires saving the text search index and its associated objects. You must save the following objects:

- The text search index (stored in the integrated file system).
- The staging table used as a log file that tracks record changes in the base table (over which the index is built). The staging table is in library QSYS2. Its name begins with QDBTS, for example, `QDBTS00001`.
- The view, which is the database object representing the text index. The view has the same name as the text index.
- The base table over which the index is built.
- The SQL catalogs that store the information to track the index.

Do the following to save the text search indexes:

1. (Recommended) Bring the indexes up to date by first performing update operations (SYSTS_UPDATE) for the text search indexes.
2. Save the base table and view using standard save techniques such as the SAVOBJ command.
3. Save the staging tables that are in QSYS2 using standard save techniques; for example, `SAVOBJ LIB(QSYS2) OBJ(QDBTS*)`.
4. Save the text search index catalogs in QSYS2 as described in previous steps.
5. Save the text search index information in the integrated file system. This is the entire contents of the *config* directory under the text server path. The text server path can be determined by querying the SERVERPATH column of the SYSTXTSRVR catalog for the server of interest. The server path will have the directory bin appended to it, which should be replaced with the config directory. A common save technique is to use the SAV command, though any type of save (for example, zip) should work. Note that this save information is only applicable to text servers running on i5/OS.

For example assume you want to save all text indexes associated with the default text server created by IBM OmniFind Text Search Server for DB2 for i5/OS. You have a table name QGPL/MYDOCS with text index QGPL/MYDOCIX built over it. In this example, the save media are save files. You perform the following steps:

1. Save base table and view:

   ```
   SAVOBJ OBJ(MYDOCS MYDOCIX) LIB(QGPL) DEV(*SAVF) SAVF(QGPL/SAVFMYFILE)
   ```

2. Save all staging tables and the IBM OmniFind Text Search Server for DB2 for i5/OS catalogs from QSYS2:

   ```
   SAVOBJ OBJ(QDBTS* SYSTXT*) LIB(QSYS2) DEV(*SAVF) SAVF(QGPL/SAVFQSYS2)
   ```

3. Using SQL, get the path name of the text server:

```
SELECT SERVERPATH FROM systxtsrvr WHERE serverid=2
```

Note: serverid may be something other than 2. Verify you are querying for the correct server.

4. The SERVERPATH value is /QOpenSys/QIBM/ProdData/TextSearch/server1/bin/. Therefore substitute `config` for `bin/` and do the following SAV:

```
SAV DEV('/QSYS.LIB/QGPL.LIB/SAVIFS.FILE') OBJ(('/QOpenSys/QIBM/ProdData/TextSearch/server1/
config'))
```

The text indexes are now saved in save files QGPL/SAVFMYFILE, QGPL/SAVFQSYS2 and QGPL/SAVIFS, respectively.

Independent ASP considerations for IBM OmniFind Text Search Server for DB2 for i5/OS

Because an independent auxiliary storage pool (ASP) can be switched between multiple systems, there are some additional considerations for administrating a text search index on an ASP.

Text search indexes that are on the independent ASP must be contained in text search servers that have been defined in the independent ASP. You cannot view a text search server that is defined in a different independent ASP group or in the system ASP when the job is connected to the independent ASP.

A local text search server is created during the installation of IBM OmniFind Text Search Server for DB2 for i5/OS. For independent ASPs, a local text server is created by an administrator using the ServerInstance tool (`ServerInstance.sh`) after the independent ASP group is created. After you create a local text search server on the independent ASP, the index data exists on the independent ASP file system and is available if the independent ASP is switched to a different system. The administrator need to do this only once for each independent ASP group.

To create a text search server named `myiasp` on an independent ASP, follow these steps:

1. Vary on the independent ASP with the Work with Configuration Status (WRKCFGSTS) CL command or by using System i® Navigator.
2. Connect to the namespace of the independent ASP group by using the Set Auxiliary Storage Pool Group (SETASPGRP) CL command.
3. Use the `ServerInstance.sh` script to create a text search server.

   Here is the QSH command to use:

```
/QOpenSys/QIBM/ProdData/TextSearch/ServerInstance.sh -create
-servernum 2 -port nnnnn -device myiasp
```

   In the command, *nnnnn* is an available port number for the server to use. This port number must be available for use on any systems that the independent ASP group can be switched to.

After a text search server is defined for the independent ASP group, the administrative stored procedures can be used to start and stop the text search server and to create, drop, and update text search indexes.

Note: Job scheduler entries are added when the independent ASP is varied on for any indexes with scheduled updates that exist in the independent ASP. This allows scheduled updates to continue, even when the independent ASP is switched between systems.

Restrictions of using text search indexes and independent ASPs

- All systems that the independent ASP can be switched between must have IBM OmniFind Text Search Server for DB2 for i5/OS installed, and must be at the same program temporary fix (PTF) levels.
- Do not create text search indexes on an ASP other than the one that the table index is built over.

- The system catalogs SYSTEXTSERVERS, SYSTEXTINDEXES, SYSTEXTDEFAULTS, SYSTEXTCOLUMNS, and SYSTEXTCONFIGURATION do not contain records for indexes and servers that are defined in a different ASP group, including the system ASP. The catalogs contain rows only for indexes and servers that are defined for the independent ASP group that the job is connected to.
- The administrative stored procedures can be used to perform functions only on text search servers and indexes that are defined in the independent ASP group that the job is connected to.

Note: You can use the CONTAINS and SCORE SQL statements when a job is connected to an independent ASP group, even if the column is based on a table that exists on the system ASP.

Starting the IBM OmniFind Text Search Server for DB2 for i5/OS

You can start the IBM OmniFind Text Search Server for DB2 for i5/OS by calling SYSPROC.SYSTS_START.

If you installed the IBM OmniFind Text Search Server for DB2 for i5/OS as a service, the text search server starts automatically each time that the host system starts. However, you can start the server manually even if you installed the IBM OmniFind Text Search Server for DB2 for i5/OS as a service.

To start the IBM OmniFind Text Search Server for DB2 for i5/OS, call SYSPROC.SYSTS_START

```
CALL SYSPROC.SYSTS_START(serverid)
```

You can find the SERVERSTATUS in QSYS2.SYSTEXTSERVERS is set to 0 after you call the procedure. When the server is running, the following jobs are active in the background:

- QJVAEXEC *userx* BCI .0 JVM-com.ibm.es
- QJVAEXEC *userx* BCI .0 PGM-OutsideInP
- QJVAEXEC *userx* BCI .0 PGM-OutsideInP
- QJVAEXEC *userx* BCI .0 PGM-OutsideInP
- QJVAEXEC *userx* BCI .0 PGM-OutsideInP

where *userx* is the user ID of the administrator that called the stored procedure.

It might take some time before all these jobs are active and the text server can be used.

Stopping the IBM OmniFind Text Search Server for DB2 for i5/OS

You can stop the IBM OmniFind Text Search Server for DB2 for i5/OS manually by using the shutdown script that is provided.

If you installed the IBM OmniFind Text Search Server for DB2 for i5/OS as a service, the text search server stops automatically each time that the host system is shut down. However, you can stop the server manually even if you installed the IBM OmniFind Text Search Server for DB2 for i5/OS as a service.

To stop the IBM OmniFind Text Search Server for DB2 for i5/OS:

1. Indicate in the SYSTEXTSERVER catalog that the server is stopped by calling SYSPROC.SYSTS_STOP.
   - To stop all servers: `CALL SYSPROC.SYSTS_STOP()`.
   - To stop a specific server:
     a. Query the server catalog to get the serverid that you want to stop: `SELECT SERVERID,SERVERPORT,SERVERSTATUS,SERVERPATH FROM QSYS2.SYSTEXTSERVERS`

        Note: SERVERPATH identifies the server. SERVERSTATUS indicates where the server is currently active (0) or inactive (1).

b. Call SYSPROC.SYSTS_STOP specifying the numeric serverid of the server you wish to stop: CALL SYSPROC.SYSTS_STOP(*serverid*).

2. (Optional) Stop the server itself by calling the shutdown script. This is done from the qshell environment.

   To shutdown the local server, enter the following command from the command line: `qsh cd /QOpenSys/QIBM/ProdData/TextSearch/server1/bin shutdown.sh`.

   If you stop the server by using the shutdown script, the SERVERSTATUS catalog is not changed to the inactive (1) status. Thus, when the SYSTS_CREATE, SYSTS_UPDATE, and SYSTS_DROP stored procedures are called next time, the server will start automatically.

Administration tools

IBM OmniFind Text Search Server for DB2 for i5/OS provides tools that you can use for common tasks, such as configuring and administering an additional text search server and adding a synonym dictionary to a collection. These tools are shell scripts rather than CL commands. They can be called within the script environment that is started through either the Start QSH (STRQSH) or Start QSH (QSH) CL commands.

These tools do not authenticate user IDs. However, these tools can be run only by a user with valid access to the text search server.

Configuration tool

Use the configuration tool to customize configuration settings after you install IBM OmniFind Text Search Server for DB2 for i5/OS.

To customize most of the configuration settings, you must stop the text search server before running the configuration tool. However, when the server is running, you can run the options to display the current authentication token, the server port, and the current properties of the system.

The configServerAndDB2 tool

The configServerAndDB2 (`configServerAndDB2.sh`) tool is located in integrated-file-system directory /QOpenSys/QIBM/ProdData/TextSearch. This tool can be used to create or modify entries in the DB2 catalog file SYSTEXTSERVERS, and to configure the authentication token or the port number associated with the specific server. The tool modifies or sets the values for SERVERAUTHTOKEN and SERVERPORT in the DB2 catalog file SYSTEXTSERVERS. If you want to create an additional server that runs locally to your system, use the “ServerInstance tool” on page 52 instead.

The configServerAndDB2 (`configServerAndDB2.sh`) tool is called with five parameters:

1. The first parameter is either **generateToken** or **configureHTTPListener**.
2. The second parameter is **-serverPath**.
3. The third parameter is the path to the root node in integrated file system where the information related to the server is stored in a directory such as /QOpenSys/QIBM/ProdData/TextSearch/server2.
4. The fourth and fifth parameters vary depending on the value of the first parameter:
   - If the first parameter is **generateToken**, then the fourth parameter is **-seed** followed by an integer (such as 1) as the fifth parameter
   - If the first parameter is **configureHTTPListener**, then the fourth parameter is **-adminHTTPPort** followed by an integer value that can be used as the socket port for the server as the fifth parameter.

Here are two examples:

- STRQSH
  cd /QOpenSys/QIBM/ProdData/TextSearch
  configServerAndDB2.sh generateToken -serverPath /QOpenSys/QIBM/ProdData/TextSearch/server2 -seed 1
- STRQSH
  cd /QOpenSys/QIBM/ProdData/TextSearch
  configServerAndDB2.sh configureHTTPListener -serverPath /QOpenSys/QIBM/ProdData/TextSearch/server2
  -adminHTTPPort 9997

The configTool script

The configTool.sh script is available for each local server. It is not suggested to use it to modify the server entries. You can use it to print the server information (such as printAll and printToken).

Table 9. Commands to run the configuration tool

On i5/OS
configTool.sh <mandatory_command_option> <mandatory_global_arguments> <optional_global_arguments> <optional_command_options>
On a Linux server
configTool.sh <mandatory_command_option> <mandatory_global_arguments> <optional_global_arguments> <optional_command_options>
On a Windows server
configTool.bat <mandatory_command_option> <mandatory_global_arguments> <optional_global_arguments> <optional_command_options>

Command options

The configuration tool supports the following command options:

configureParams
Specifies the system parameters that you can configure. You can configure the following parameters:

-configPath
Specifies the absolute path to the configuration folder that contains the config.xml file.

-adminHTTPPort
Specifies the administration HTTP port number. If an error occurs, an error code of -3 is returned.

-logPath
Specifies the absolute path to the log directory.

-temDirPath
Specifies the absolute path to the temporary directory.

-numberOfIndexers
Specifies the number of concurrent text search indexing subsystems.

-numberOfTokenizers
Specifies the number of concurrent subsystems that are used for parsing input into tokens.

-maxDocumentSize
Specifies the maximum number of characters that are to be indexed for a document. If an error occurs, an error code of -3 is returned.

-logLevel
Specifies the log level for system messages in the log file. The default level is `informational`. Additional options are `warning` and `severe`.

-maxHeapSize
Starts and ends the heap size in a format that is accepted by the Java Virtual Machine. If an error occurs, an error code of -5 is returned.

printToken
Prints the current authentication token and encryption key.

printAll
Prints all of the current values for the options that you can configure with this tool.

printAdminHTTPPort
Prints the current value for the administration HTTP port.

generateToken
Generates the authentication token.

Global arguments

-configPath
Specifies the absolute path to the configuration folder that contains the config.xml file. This global argument is mandatory.

-locale
Specifies the five-character locale setting for writing messages to the trace file. If you do not specify this setting, the default value, -en_US, is used.

Example

On a Linux server, use the following command to print the current authentication token:

```
configTool.sh printToken -configPath <path> <optional_global_arguments>
```

QDBTS_LISTINXSTS UDTF

An SQL User Defined Table Function (UDTF) is provided to detect orphaned indexes and missing indexes. An index can be orphaned if a SYSTS_DROP stored procedure is called and the server is stopped or not started at the time the procedure is running.

The QDBTS_LISTINXSTS function combines all the integrated-file-system collections and catalog indexes in the current namespace into one table. The function decides which independent auxiliary storage pool (ASP) or *SYSBASE is set and scans the collection directory of each server in the independent ASP or *SYSBASE.

For *SYSBASE, each server directory under `/QOpenSys/QIBM/ProdData/TextSearch` is checked. For independent ASPs, each server directory under `/the ASP number/QOpenSys/QIBM/ProdData/TextSearch` is checked. For example, if the independent ASP number is 67, each server directory under `/67/QOpenSys/QIBM/ProdData/TextSearch` is checked.

For catalog index information, data is obtained from catalog table QSYS2.SYSTEXTINDEXES. If you want to check servers on an independent ASP, issue the Set Auxiliary Storage Pool Group (SETASPGRP) command before this function is called.

If you want to remove possible orphaned indexes from the integrated file system after they are identified, use the SYSPROC.SYSTS_REMOVE stored procedure or the "Administration tool" on page 49 (`adminTool.sh`).

Terms

Orphaned index
A collection (an index) exists in the integrated file system directory of the server, but no corresponding index is recorded in catalog QSYS2.SYSTEXTINDEXES.

Missing index
Index records exist in catalog QSYS2.SYSTEXTINDEXES, but the corresponding collection directory does not exist.

Syntax

```
>>-QDBTS_LISTINXSTS(--null--)--->< 
```

Return format

The QDBTS_LISTINXSTS function returns information of detected indexes in a table. See the following SQL command that is used to create the UDTF.

SQL for LISTINXSTS UDTF

```
CREATE FUNCTION QDBTSLIB.QDBTS_LISTINXSTS()
        RETURNS TABLE(COLLECTIONNAME VARCHAR(255),
                   INDEXID INTEGER,
                   INDEXSCHEMA VARCHAR(128),
                   INDEXNAME VARCHAR(128),
                   SERVERID INTEGER)
   SPECIFIC qdbts_listinxsts
   SCRATCHPAD
   NO FINAL CALL
   LANGUAGE C++
   PARAMETER STYLE DB2SQL
   EXTERNAL NAME 'QDBTSLIB/QDBTSSP(checkIndex)';
```

Examples

- Detect all orphaned indexes:

```
SELECT COLLECTIONNAME, SERVERID
       FROM TABLE(QDBTSLIB.QDBTS_LISTINXSTS()) AS T
       WHERE T.INDEXSCHEMA IS NULL AND T.INDEXNAME IS NULL
```

- Detect all missed indexes:

```
SELECT INDEXSCHEMA, INDEXNAME
       FROM TABLE(QDBTSLIB. QDBTS_LISTINXSTS()) AS T
       WHERE T.COLLECTIONNAME is NULL
```

- Detect orphaned indexes in serverid=2 on the independent ASP iaspXXX:

  CL:

```
SETASPGRP(iaspXXX)
```

  SQL:

```
SELECT T.COLLECTIONNAME, S.SERVERPATH
       FROM TABLE(QDBTSLIB.QDBTS_LISTINXSTS())
            AS T LEFT OUTER JOIN QSYS2.SYSTEXTSERVERS S ON (T.SERVERID = S.SERVERID)
       WHERE T.INDEXSCHMEA IS NULL AND T.INDEXNAME IS NULL AND T.SERVERID = 2
```

Note: If you want to use different DB2 interfaces, you can use different ways to work on an independent ASP. For example, if you use System i Navigator, right-click the database name for the

independent ASP, and run your SQL scripts. The statements are run in the independent ASP namespace without the need for the SETASPGRP command.

SYSPROC.SYSTS_REMOVE

The SYSPROC.SYSTS_REMOVE SQL stored procedure is provided to remove orphaned indexes. The collection-name of the possible orphaned indexes can be identified through the QDBTS_LISTINXSTS User Defined Table Function (UDTF).

Syntax

```
>>-SYSPROC.SYSTS_REMOVE ( collection-name ) -><
```

Parameter

collection-name
Specifies a string literal that identifies the name of the collection to be removed.

Note: This procedure uses the `adminTool.sh` shell script to remove the collection directory. To use this shell script, the server must be in the working state. If the server is not started, this procedure returns an error message.

SQL for SYSTS_REMOVE

```
CREATE PROCEDURE SYSPROC.SYSTS_REMOVE(
       IN COLLECTIONNAME  VARCHAR(255) CCSID 1208)
    EXTERNAL NAME QDBTSLIB.DSN5RMCOLL
    DYNAMIC RESULT SETS 0
    LANGUAGE C++
    PARAMETER STYLE SQL
    PROGRAM TYPE MAIN
    COMMIT ON RETURN NO
    INHERIT SPECIAL REGISTERS;
```

Examples

- To remove orphaned indexes on *SYSBASE, type the following command from any SQL interface:

  ```
  CALL SYSPROC.SYSTS_REMOVE('0_65_2815_2008_06_02_11_58_22_901726')
  ```

  The SYSTS_REMOVE stored procedure checks whether the index information is in catalog table QSYS2.SYSTEXTINDEXES. If it is true, error message DSX_INDEX_EXIST is returned; if not, procedure searches under the `config/collections` directory of server 65. If the collection does not exist, error message DSX_COLLECTION_NOT_FOUND is returned; if the collection exists, procedure calls `adminTool.sh` to remove the collection. After that, the procedure checks the directory again to see whether the collection has been removed. If the collection is not removed, error message DSX_REMOVE_COLLECTION_FAILED is returned.

  Note: When the collection on the text search server is on an independent ASP group, the thread that calls the SYSTS_REMOVE stored procedure must run in the namespace of the independent ASP. To do so, use the Set Auxiliary Storage Pool Group (SETASPGRP) command.

- To remove orphaned indexes in an independent ASP iaspXXX, you can use the following commands:

  CL:

  ```
  SETASPGRP(isapXXX)
  ```

  SQL:

  ```
  CALL SYSPROC.SYSTS_REMOVE(' 33_7_26_2008_06_18_21_28_39_407824')
  ```

Note: If you use System i Navigator, right-click the database name for the independent ASP, and run your SQL scripts.

SYSPROC.SYSTS_VALIDITYCHECK

The SYSPROC.SYSTS_VALIDITYCHECK SQL stored procedure is provided to check valid index items. This stored procedure can fix some items that are not valid if the *autoFix* parameter is specified.

Syntax

```
>>-SYSPROC.SYSTS_VALIDITYCHECK (indexSchema, indexName, autoFix) -><
```

The schema qualifier is SYSPROC.

Parameters

indexSchema
Identifies the schema of the text search index. If this parameter is null, the value of the CURRENT SCHEMA special register for the invoker is used.

The data type of this parameter is VARCHAR(128).

indexName
Identifies the name of the text search index. The name of the text search index with the index schema uniquely identifies the full-text index in the DB2 subsystem. You must specify a value that is not null for this parameter.

The data type for this parameter is VARCHAR(128).

autoFix
Identifies whether automatic fix is required. The value for this parameter can only be 0 or 1. The meanings of these values are described as follows:

0 Only the index validity is checked.

1 Index validity is checked and items that are not valid are fixed.

Note:

If values other than 0 or 1 are specified, they are considered as 0.

The data type for this parameter is INTEGER.

SQL for SYSTS_VALIDITYCHECK

```
CREATE PROCEDURE SYSPROC.SYSTS_VALIDITYCHECK
     (IN INDEXSCHEMA  VARCHAR(128) CCSID 1208,
      IN INDEXNAME  VARCHAR(128) CCSID 1208,
      IN AUTOFIX INTEGER)
    EXTERNAL NAME QDBTSLIB.DSN5VALCHK
    DYNAMIC RESULT SETS 0
    LANGUAGE C
    PARAMETER STYLE SQL
    MODIFIES SQL DATA
    PROGRAM TYPE MAIN
    COMMIT ON RETURN NO
    INHERIT SPECIAL REGISTERS
```

Examples

- To check the validity for an index, type the following command from any SQL interface:

```
CALL SYSPROC.SYSTS_VALIDITYCHECK('indexSchema1','indexName1',0)
```

- To check and fix an index automatically:

```
CALL SYSPROC.SYSTS_VALIDITYCHECK('indexSchema1','indexName1',1)
```

SYSPROC.SYSTS_REPRIMEINDEX

The SYSPROC.SYSTS_REPRIMEINDEX stored procedure is provided to reprime the index and start an initial update. Use this stored procedure when you want to restore data from the base table.

If the data from the base table is restored, the updated content of the base table cannot be indexed while the SYSTS_UPDATE stored procedure is called. In this case, the SYSPROC.SYSTS_REPRIMEINDEX stored procedure can be called to reprime the index.

Note: If a synonym dictionary has been created for the text search index, this process will remove the dictionary.

Syntax

```
>>-SYSPROC.SYSTS_REPRIMEINDEX( indexSchema, indexName, options) -><
```

The schema qualifier is SYSPROC.

Parameters

indexSchema
Identifies the schema of the text search index. If this parameter is null, the value of the CURRENT SCHEMA special register for the invoker is used.

The data type of this parameter is VARCHAR(128).

indexName
Identifies the name of the text search index. The name of the text search index with the index schema uniquely identifies the full-text index in the DB2 subsystem. You must specify a value that is not null for this parameter.

The data type for this parameter is VARCHAR(128).

options
A character string that specifies options that can be added in the future for this stored procedure.

Important: You must specify a null value for the *options* parameter. Otherwise, errors can be generated. Read the "Example" for how to specify the *options* parameter.

SQL for SYSTS_REPRIMEINDEX

```
CREATE PROCEDURE SYSPROC.SYSTS_REPRIMEINDEX(
            IN INDEXSCHEMA  VARCHAR(128) CCSID 1208,
            IN INDEXNAME  VARCHAR(128) CCSID 1208,
            IN OPTIONS VARCHAR(32000) CCSID 1208
    )
    EXTERNAL NAME QDBTSLIB.DSN5RPMIDX
    DYNAMIC RESULT SETS 0
    LANGUAGE C
    PARAMETER STYLE SQL
    MODIFIES SQL DATA
    PROGRAM TYPE MAIN
    COMMIT ON RETURN NO
    INHERIT SPECIAL REGISTERS
```

Example

- To reprime an index from any SQL interface, type the following command from any SQL interface:

```
CALL SYSPROC.SYSTS_REPRIMEINDEX('indexSchema1','indexName1','')
```

Administration tool

You can use the administration tool for advanced administration. The IBM OmniFind Text Search Server for DB2 for i5/OS can be running when you use the administration tool.

You can use the administration tool to do the following tasks:

- Check the status of collections, such as finding out how many documents are present
- Delete orphan collections
- Report the version of the server
- Report all of the collections that are on the text search server

Commands

The command that you issue to run the administration tool depends on what operating system the text search server is installed on and the task that you want to do.

Table 10. Commands to check the status of collections and to delete orphaned collections

On i5/OS (within the QSH interface)	On a Linux server	On a Windows server
adminTool.sh -[delete|status] –collectionName <collection name> -configPath <absolute path to configuration folder>	adminTool.sh -[delete|status] –collectionName <collection name> -configPath <absolute path to configuration folder>	adminTool.bat -[delete|status] –collectionName <collection name> -configPath <absolute path to configuration folder>

Table 11. Commands to display the version of the server and to report all of the collections

On i5/OS (within the QSH interface)	On a Linux server	On a Windows server
adminTool.sh -[version] -configPath <absolute path to configuration folder>	adminTool.sh -[version|reportAll] -configPath <absolute path to configuration folder>	adminTool.bat -[version|reportAll] -configPath <absolute path to configuration folder>

Options

status
Checks the status of the collection.

delete
Specifies that you want to delete the orphaned collection.

version
Displays the version of the server.

reportAll
Reports all of the collections that are on the text search server.

Example

To find out the version of the server, enter the following command on a Linux server:

```
adminTool.sh –version -s <absolute path to server config.xml>
```

When you use a Windows server, a corresponding .bat script is provided.

Synonym tool

Use the synonym tool to add a synonym dictionary. A synonym dictionary can improve the quality of search results.

You can add a synonym dictionary to a collection at any time. The IBM OmniFind Text Search Server for DB2 for i5/OS administrator has the correct authority and privileges to run the synonym tool.

A synonym dictionary consists of synonym groups that you define in an XML file. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<synonymgroups version="1.0">
<synonymgroup>
  <synonym>Paixão</synonym>
  <synonym>amor</synonym>
  <synonym>flor</synonym>
  <synonym>linda</synonym>
</synonymgroup>
<synonymgroup>
  <synonym>worldwide patent tracking system</synonym>
  <synonym>wpts</synonym>
</synonymgroup>
</synonymgroups>
```

Adding a synonym dictionary to a collection

Use the synonym tool to add a synonym dictionary to a specific collection. Specifying synonym groups in a synonym dictionary improves the quality of text search results.

To add a synonym dictionary to a collection, follow these steps:

1. Create a synonym XML file by specifying the synonym groups, as shown in the following example:

   ```
   <?xml version="1.0" encoding="UTF-8"?>
   <synonymgroups version="1.0">
   <synonymgroup>
     <synonym>Paixão</synonym>
     <synonym>amor</synonym>
     <synonym>flor</synonym>
     <synonym>linda</synonym>
   </synonymgroup>
   <synonymgroup>
     <synonym>worldwide patent tracking system</synonym>
     <synonym>wpts</synonym>
   </synonymgroup>
   </synonymgroups>
   ```

2. Copy the synonym XML file to any directory on the text search server.
3. Use the synonym tool to add the synonym dictionary to a collection.

   You can add a synonym dictionary in `append` mode or `replace` mode. If you add a synonym dictionary in `append` mode, the new synonyms are added to the existing synonym dictionary. If you add a synonym dictionary in `replace` mode, the existing synonyms are replaced by the new synonyms that you defined for the text search index.

Option	Description
On i5/OS, enter the following command (within the QSH interface):	`synonymTool.sh importSynonym -synonymFile <absolute path to synonym XML file> -collectionName <collection name> -replace <[true|false]> -configPath <absolute path to configuration folder>`

Option	Description
On a Linux server, enter the following command:	`synonymTool.sh importSynonym -synonymFile <absolute path to synonym XML file> -collectionName <collection name> -replace <[true|false]> -configPath <absolute path to configuration folder>`
On a Windows server, enter the following command:	`synonymTool.bat importSynonym -synonymFile <absolute path to synonym XML file> -collectionName <collection name> -replace <[true|false]> -configPath <absolute path to configuration folder>`

If the format of the XML file is not valid, or if the XML file is empty, an error code is returned.

Removing a synonym dictionary from a collection

Use the script that is provided to remove a synonym dictionary from a collection. The IBM OmniFind Text Search Server for DB2 for i5/OS administrator needs to retrieve the name of the collection from which you want the synonym dictionary to be removed.

Run the script to remove the synonym dictionary from a collection.

Option	Description
On i5/OS, enter the following command (within the QSH interface):	`removeSynonym.sh -collectionName <collection name> -configPath <absolute path to configuration folder>`
On a Linux server, enter the following command:	`removeSynonym.sh -collectionName <collection name> -configPath <absolute path to configuration folder>`
On a Windows server, enter the following command:	`removeSynonym.bat -collectionName <collection name> -configPath <absolute path to configuration folder>`

If a database has several text search indexes, you must complete this task for each of the corresponding collections.

Problem determination

IBM OmniFind Text Search Server for DB2 for i5/OS logs system messages and trace messages to help you determine the source of problems that might occur.

The server logs are located in the <INSTALL_HOME>/log directory. For the default server that is created when IBM OmniFind Text Search Server for DB2 for i5/OS is installed, the directory is `/QOpenSys/QIBM/ProdData/TextSearch/server1/log`. By default, the trace log is turned off, and the system log level is set to `informational`. You can use the configuration tool to change the trace and log level options.

The server logs are rotated by size. The five most recent copies of server logs that are no more than 8 MB are stored. You can view and save the server logs by using the script that is provided. On i5/OS or a Linux server, the script is logformatter.sh. On a Windows server, the script is logformatter.bat.

Options

The script has the following options.

-f logfile
Specifies the server log file that you want to format.

-l locale
Specifies the locale to use when writing the reformatted messages. For example, specify `en_US` for English, or `ja_JP` for Japanese. This value is optional. The default value is en_US.

-o outputfile
Specifies the output file where the reformatted log messages are to be written by using UTF-8 encoding. This value is optional. If you do not specify this option, a standard output file is used.

-? Prints the help message. This value is optional.

-v Specifies the mode for displaying debugging messages. This value is optional.

Viewing and saving server logs

Use the provided scripts to view and save the server logs that can help you to determine the source of problems.

To view and save a server log:

1. To view the server log, run one of the following commands:

Option	Description
On i5/OS (within the QSH environment)	`bin/logformatter.sh -f log/System.0.log`
On a Linux server	`bin/logformatter.sh -f log/System.0.log`
On a Windows server	`bin/logformatter.bat -f log/System.0.log`

2. To save the server log to a file so that you can read the log in a file editor, run one of the following commands:

Option	Description
On i5/OS (within the QSH environment)	`bin/logformatter.sh -f log/System.0.log –o <output filename>`
On a Linux server	`bin/logformatter.sh -f log/System.0.log –o <output filename>`
On a Windows server	`bin/logformatter.bat -f log/System.0.log –o <output filename>`

ServerInstance tool

You can use the ServerInstance tool to create or delete servers on *SYSBASE or an independent auxiliary storage pool (ASP). You can also use the ServerInstance tool to link files from a server to the server where IBM OmniFind Text Search Server for DB2 for i5/OS is installed.

By default, IBM OmniFind Text Search Server for DB2 for i5/OS is installed under directory `/QOpenSys/QIBM/ProdData/TextSearch/server1`. You can use the ServerInstance tool to complete the following tasks before you use it to stop server1 on *SYSBASE:

- Create a new server on *SYSBASE or independent ASPs
- Delete a server on *SYSBASE or independent ASPs
- Link files from a server to server1

Syntax

```
ServerInstance.sh –[create|delete|relink]
–servernum <server number>
(-port <port>)
(-device <device name>)
```

Command options

create
Creates a new server.

delete
Deletes a server.

relink
Links files from a server to server1.

Note: You do not need this option after you have program temporary fix (PTF) SI31548 installed on your system. The system automatically processes the linking operation if you have this PTF installed.

Parameters

servernum
Specifies the server number. For example, when a server with server number 3 is created, the directory of the server is `/QOpenSys/QIBM/ProdData/TextSearch/server3`.

port
Specifies the port of the server. This parameter is needed only when you create a new server.

device
Specifies the name of the independent ASP. This parameter is needed only when the operation is completed on the independent ASP.

Examples

- To create a server with server number 2 and port number 50000 on *SYSBASE:
  ```
  ServerInstance.sh -create -servernum 2 -port 50000
  ```
- To create a new server with server number 3 and port number 50001 on independent ASP iasp1:
  ```
  ServerInstance.sh -create -servernum 3 -port 50001 -device iasp1
  ```
- To delete a server with server number 2 on *SYSBASE:
  ```
  ServerInstance.sh -delete -servernum 2
  ```
- To delete a server with server number 3 on independent ASP iasp1:
  ```
  ServerInstance.sh -delete -servernum 3 -device iasp1
  ```
- To link files from a server to server number 2 on *SYSBASE:
  ```
  ServerInstance.sh -relink -servernum 2
  ```
- To link files from a server to server number 3 on independent ASP iasp1:
  ```
  ServerInstance.sh -relink -servernum 3 -device iasp1
  ```

Chapter 7. Text search administration tables

To facilitate text search support, specific administration tables exist in QSYS2.

QSYS2.SYSTEXTDEFAULTS administration table

The QSYS2.SYSTEXTDEFAULTS administration table provides the default parameters and values. The QSYS2.SYSTEXTDEFAULTS administration table is created when you install IBM Text Search for DB2 for i5/OS.

The following table shows the contents of the QSYS2.SYSTEXTDEFAULTS administration table.

Table 12. Contents of the QSYS2.SYSTEXTDEFAULTS administration table

Column name	Data type	Nullable?	Description
NAME	VARCHAR(30)	No	Name of a default parameter for the database for text search.
VALUE	VARCHAR(512)	No	Value for the default parameter for text search.
TYPE	INTEGER	No	Reserved.

QSYS2.SYSTEXTINDEXES administration table

The QSYS2.SYSTEXTINDEXES administration table provides information about each text search index, such as the name and the schema name of the text search index and the name of the associated collection on the text search server.

The following table shows the contents of the QSYS2.SYSTEXTINDEXES administration table. The unique key for this table is the INDEXSCHEMA column in conjunction with the INDEXNAME column. The primary key is the INDEXID column.

Table 13. Contents of the QSYS2.SYSTEXTINDEXES administration table

Column name	Data type	Nullable?	Description
INDEXID	INTEGER	No	Uniquely generated index ID for the text search index.
INDEXSCHEMA	VARCHAR(128)	No	Schema name for the text search index.
INDEXNAME	VARCHAR(128)	No	Unqualified name of the text search index.
TABLESCHEMA	VARCHAR(128)	No	Schema name of the base table.
TABLENAME	VARCHAR(128)	No	Unqualified name of the base table.
TABLEIASP	SMALLINT	No	Independent ASP of the base table.
COLLECTIONNAME	VARCHAR(255)	No	Name of the associated collection on the text search server.
SERVERID	INTEGER	No	The server ID for the text search index.
TAKEOVERSERVERID	INTEGER	Yes	Reserved for future use.
TAKEOVERSERVERPULSE	TIMESTAMP	Yes	Reserved for future use.
SEARCHARGS	VARBINARY(1024)	Yes	Reserved for future use.

Table 13. Contents of the QSYS2.SYSTEXTINDEXES administration table (continued)

Column name	Data type	Nullable?	Description
ALIASSCHEMA	VARCHAR(128)	No	The alias for the schema of the base table that was used in the SYSPROC.SYSTS_CREATE stored procedure. If no alias is used, this value is identical to TABLESCHEMA.
ALIASNAME	VARCHAR(128)	No	The alias for the name of the base table that was used in the SYSPROC.SYSTS_CREATE stored procedure. If no alias is used, this value is identical to TABLENAME.
STAGINGTABLENAME	VARCHAR(128)	Yes	The name of the log table for the text search index.
EVENTTABLENAME	VARCHAR(128)	No	The name of the event table for the text search index.
OFINDEXTABLENAME	VARCHAR(128)	No	The name of the table for the text search index on the IBM OmniFind Text Search Server for DB2 for i5/OS.
UPDATEMINIMUM	INTEGER	No	Minimum number of entries in the log table before an incremental update of the text search index is performed.
UPDATEFREQUENCY	VARCHAR(512)	No	The update frequency for the text search index as specified by the SYSPROC.SYSTS_CREATE stored procedure.
UPDATEMODE	INTEGER	No	Indicates the update mode of the text search index. The integer 0 (zero) indicates the initial update of the text search index. A value of 1 indicates subsequent, incremental updates.
REORGANIZATIONMODE	INTEGER	No	Indicates the reorganization mode of the text search index.
CREATETIME	TIMESTAMP	No	The time that the text search index was created.
LASTUPDATETIME	TIMESTAMP	Yes	The time that the text search index was last updated.
LASTUPDATESTATUS	CHAR	Yes	Indicates the internal status for optimizing the clean-up process after an initial or incremental update of the text search index.
SCHEDULERTASKID	INTEGER	Yes	Reserved for future use.
EXPRESSIONLISTS	CLOB(32K)	Yes	Reserved for future use.
EXPRESSIONNUMBERS	VARBINARY(32)	Yes	Reserved for future use.
USEREXITFUNCTION	VARCHAR(18)	Yes	Reserved for future use.
REMARKS	VARCHAR(2000)	Yes	Remarks made in the **COMMENTS** option of the **index-configuration-options** parameter of the SYSPROC.SYSTS_CREATE stored procedure.

QSYS2.SYSTEXTCOLUMNS administration table

The QSYS2.SYSTEXTCOLUMNS administration table provides information about the text columns for a text search index, such as the index ID for the text search index, the name of the text columns, and the schema name of the base table.

The following table shows the contents of the QSYS2.SYSTEXTCOLUMNS administration table. The primary key for this table is the INDEXID column in conjunction with the COLUMNNAME column. The foreign key is the INDEXID column.

Table 14. Contents of the QSYS2.SYSTEXTCOLUMNS administration table

Column name	Data type	Nullable?	Description
INDEXID	INTEGER	No	Uniquely generated index ID for the text search index.
COLUMNNAME	VARCHAR(128)	No	Unqualified name of the text column.
TABLESCHEMA	VARCHAR(128)	No	Schema name of the base table.
TABLENAME	VARCHAR(128)	No	Unqualified name of the base table.
LANGUAGE	VARCHAR(5)	No	The language that the text search server uses for the linguistic processing of text documents. The default value is en_US (English).
FUNCTIONSCHEMA	VARCHAR(128)	Yes	The schema of a user-defined function that is to be used by IBM Text Search for DB2 for i5/OS to access text documents that are in a column that is not of a supported data type, or that are stored elsewhere.
FUNCTIONNAME	VARCHAR(18)	Yes	The name of a user-defined function that is to be used by IBM Text Search for DB2 for i5/OS to access text documents that are in a column that is not of a supported data type, or that are stored elsewhere.
CCSID	INTEGER	No	The coded character set identifier that is used for a text search index on a column with a binary data type.
FORMAT	VARCHAR(30)	No	The format of text documents in the column. The supported format values are TEXT, HTML, XML, and INSO.
KEYCOLUMNCOUNT	INTEGER	No	The count of key columns for the text search index.
KEYCOLUMNNAMES	VARCHAR(1200)	No	The key column names for the text search index.

QSYS2.SYSTEXTSERVERS administration table

The QSYS2.SYSTEXTSERVERS administration table stores information about where the text search servers are installed.

The following table shows the contents of the QSYS2.SYSTEXTSERVERS administration table. The unique key for this table is the SERVERNAME column in conjunction with the SERVERPORT column. The primary key is the SERVERID column.

Table 15. Contents of the QSYS2.SYSTEXTSERVERS administration table

Column name	Data type	Nullable?	Description
SERVERID	INTEGER	No	Uniquely generated ID for the text search server.
SERVERNAME	VARCHAR(128)	No	The host name or IP address of the text search server.
SERVERADRINFO	VARBINARY(3000)	Yes	The internal representation of the SERVERNAME and SERVERPORT as determined by the SYSPROC.SYSTS_START stored procedure.
SERVERPORT	INTEGER	No	The port number for the text search server.
SERVERPATH	VARCHAR(512)	No	The server path for the text search server.
SERVERTYPE	INTEGER	No	The server type for the text search server. The value 0 (zero) indicates an i5/OS text search server. The value 1 indicates a Linux text search server. The value 2 indicates a Windows text search server.
SERVERAUTHTOKEN	VARCHAR(256)	No	The authentication token for the text search server.
SERVERMASTERKEY	VARCHAR(36)	No	The server key for the text search server.
SERVERCLASS	INTEGER	No	The server class for the text search server. The value 0 (zero) indicates a production server, available for automatic selection. the value 9 indicates a test server, never allocated automatically.
SERVERSTATUS	INTEGER	No	Indicates whether the server can be used as a text search server to create new text search indexes. The default value is 0 (zero), which means that the server can be used.

QSYS2.SYSTEXTCONFIGURATION administration table

The QSYS2.SYSTEXTCONFIGURATION administration table contains information about the configuration parameters for the text search index as passed by the SYSPROC.SYSTS_CREATE stored procedure.

The following table shows the contents of the QSYS2.SYSTEXTCONFIGURATION administration table. The primary key is the INDEXID column in conjunction with the PARAMETER column. The foreign key is the INDEXID column.

Table 16. Contents of the QSYS2.SYSTEXTCONFIGURATION administration table

Column name	Data type	Nullable?	Description
INDEXID	INTEGER	No	Uniquely generated index ID for the text search index.
PARAMETER	VARCHAR(30)	No	Parameters that are specified for the text search index in the SYSPROC.SYSTS_CREATE stored procedure.
VALUE	VARCHAR(512)	No	Values for the specified parameters.

QSYS2.SYSTEXTSERVERHISTORY administration table

The QSYS2.SYSTEXTSERVERHISTORY administration table is an auxiliary table that records a history of used servers for the SYSPROC.SYSTS_DROP stored procedure.

The following table shows the contents of the QSYS2.SYSTEXTSERVERHISTORY administration table. The unique key for this table is the INDEXID column in conjunction with the SERVERID column. The foreign key is the INDEXID column.

Table 17. Contents of the QSYS2.SYSTEXTSERVERHISTORY administration table

Column name	Data type	Nullable?	Description
INDEXID	INTEGER	No	The index ID for a created text search index.
SERVERID	INTEGER	No	The server ID where a text search index needs to be dropped on SYSPROC.SYSTS_DROP.

Chapter 8. Messages and codes

This section contains information about the messages and SQL return codes for IBM Text Search for DB2 for i5/OS. The messages are listed in numeric sequence.

SQLCODE-20423 Error occurred during text search processing.

Explanation:

An error occurred during the text search processing of a CONTAINS or SCORE function. The error happened on server *server* using text search index *index-name* for reason code *reason-code*. Text describing the problem is:*text*.

server: The host name or IP address and port of the text search server where the error was encountered.

index-name: The name of the index used in the text search processing. **Note:** Include the schema and a period with the index name in a single token.

reason-code: The reason code returned from the IBM OmniFind Text Search Server for DB2 for i5/OS.

text: The text returned from the IBM OmniFind Text Search Server for DB2 for i5/OS.

System action: The statement cannot be processed.

User response:

Contact your system administrator to check that the IBM OmniFind Text Search Server for DB2 for i5/OS is successfully installed.

SQLSTATE: 38H10

SQLCODE-20424 Text search support is not available for reason *reason-code*.

Explanation:

A problem with one of the text search administrative tables was detected. The reason code is *reason-code*.

1 One of the text search administration tables was not found (QSYS2.SYSTEXTINDEXES, QSYS2.SYSTEXTCOLUMNS, or QSYS2.SYSTEXTSERVERS).

3 The Text Search support is not installed.

4 The STATUS column in QSYS2.SYSTEXTSERVERS table has a value of 1, indicating that the support for the text search is stopped.

7 No IBM OmniFind Text Search Server for DB2 for i5/OSs have been defined.

System action: The statement cannot be processed.

User response:

Contact your system administrator to make sure that support for text searching is successfully set up on your system.

SQLSTATE: 38H11

SQLCODE-20425 Text search not allowed for column *column-name*.

Explanation: A CONTAINS or SCORE text search function specified column *column-name* in table *table-name* in table-schema. A text index does not exist for this column so text search processing cannot be performed.

System action: The statement cannot be processed.

User response:

Verify that the column and table are registered to the IBM OmniFind Text Search Server for DB2 for i5/OS.

SQLSTATE: 38H12

SQLCODE-20426 Conflicting text search administration procedure is already running.

Explanation: A conflicting text search administrative procedure such as update is already running on this index.

System action: The statement cannot be processed.

User response:

Invoke the administration stored procedure again after the currently running stored procedure completes.

SQLSTATE: 38H13

SQLCODE-20427 Error occurred during text search administrative procedure.

Explanation: An error occurred during a text search administrative procedure. The reason code is *reason-code*. The text returned is: *text*. The error text describes the problem.

System action: The CALL statement fails with this SQLCODE.

User response:

Fix the problem that is indicated by *error* and invoke the administrative stored procedure again.

SQLSTATE: 38H14

Appendix. Notices

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

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

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

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

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

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

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

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web 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 which has been exchanged, should contact:

IBM Corporation

Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
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, IBM License Agreement for Machine Code, 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.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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.

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

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

This OmniFind Text Search Server for DB2 for i5/OS publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of OmniFind Text Search Server for DB2 for i5/OS.

Trademarks

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

AIX
DB2
i5/OS
IBM
IBM (logo)
Lotus
OmniFind
System i
z/OS

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

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

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

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

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

Terms and conditions

Permissions for the use of these publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED ″AS-IS″ AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Printed in USA