



**Text Extender
Administration and Programming**

Version 7



**Text Extender
Administration and Programming**

Note

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

Third Edition (March 2004)

This edition applies to Version 7 of IBM DB2 Universal Database for OS/390 and z/OS, 5675-DB2, and to any subsequent releases until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

This and other books in the DB2 for OS/390 and z/OS library are periodically updated with technical changes. These updates are made available to licensees of the product on CD-ROM and on the Web (currently at www.ibm.com/software/data/db2/os390/library.html). Check these resources to ensure that you are using the most current information.

© Copyright International Business Machines Corporation 1995, 2001. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this book	vii
Who should use this book	vii
How to use this book.	vii
How to read the syntax diagrams	vii
Accessibility	viii
Summary of Changes	ix

Part 1. Guide 1

Chapter 1. An overview of DB2 Text Extender	3
DB2 Text Extender in the DB2 client/server environment	4
Chapter 2. Customization, Configuration and maintenance	7
Customization.	7
Prerequisites	7
Installation files and directories under HFS	7
Checking the Text Search Installation	7
Deciding whether to use Linklist or STEPLIB	8
Running the customization routine	8
Customizing the user's runtime profile	9
Log files written during the final installation steps	10
Customizing DB2 Text Extender to run on more than one DB2 subsystem	10
Workload Manager related tasks	11
DB2-related tasks	11
Troubleshooting	12
Configuration	13
Environment variables	13
Text configuration settings	13
Changing the text configuration	14
Setting up and maintaining a DB2 Text Extender server	15
Creating the DB2 Text Extender instance	15
Starting and stopping a DB2 Text Extender server	15
Backing up and restoring indexes and enabled servers	15
Tracing faults	16
Chapter 3. Getting started	19
A simple example of making text searchable	19
Chapter 4. Planning for your search needs	21
Why text documents need to be indexed	21
Which document formats are supported.	22
HTML documents, special considerations	23
XML documents, special considerations.	23
Using unsupported document formats	23
Languages	25
CCSIDs	25
EBCDIC	25
ASCII	26
DBCS	26
UNICODE.	27
Avoiding code page problems when storing and enabling text.	27
Types of search	29

Linguistic search	30
Precise search	31
Make a fuzzy search or search in DBCS documents	32
Changing the index type	32
Creating one or several text indexes for a table	32
Calculating the size of an index	33
Updating an index	33
Incremental index update	33
Working with structured documents (section support)	34
Flat files and HTML documents	34
Dictionaries, stop-word lists, abbreviation lists, and language parameters	36
Modifying the stop-word and abbreviation files	37
Chapter 5. Making text searchable	39
Preparation before making text searchable	39
Starting the DB2 Text Extender command line processor	40
Command line processor help	41
Connecting to a subsystem	41
Enabling a server	41
Enabling a text column	42
A handle column is added	42
The document information is set	43
A log table is created	43
An index is created	44
Examples	44
Enabling a text column in a large table	45
Enabling text columns of a nonsupported data type	45
Ending the session	46
Chapter 6. How to search	47
Setting the current function path	47
Searching for text	47
Making a query	48
Searching and returning the number of matches found	49
Searching and returning the rank of a found text document	49
Specifying search arguments	49
Searching for several terms	49
Searching with the Boolean operators AND and OR	50
Searching for variations of a term	50
Searching for parts of a term (character masking)	51
Searching for terms that already contain a masking character	51
Searching for terms in any sequence	52
Searching for terms in the same sentence or paragraph	52
Searching for terms in sections of structured documents	52
Searching for synonyms of terms	53
Making a linguistic search	53
Searching with the Boolean operator NOT	54
Fuzzy search	54
Respecting word-phrase boundaries	55
Searching for similar-sounding words	55
Thesaurus search	55
Free-text and hybrid search	56
Refining a previous search	56
Setting and extracting information in handles	58
Setting text information when inserting new text	59
Extracting information from handles	60

Changing information in handles	61
Improving search performance	62
Chapter 7. Administration	63
Maintaining text indexes	63
Updating an index.	63
Resetting the index status.	64
Deleting index events	64
Reorganizing an index	65
Getting useful information	65
Displaying enabled-status information	65
Displaying the settings of the environment variables	66
Displaying the text configuration settings	66
Displaying the status of an index	67
Displaying error events	68
Displaying the index settings	69
Displaying the text settings for a column	69
Working with the DB2 Text Extender catalog view	70
Reversing the text preparation process	71
Disabling a text column.	72
Disabling a server.	72

Part 2. Reference 73

Chapter 8. Text preparation and administration commands for the client	75
CHANGE INDEX SETTINGS.	76
CHANGE TEXT CONFIGURATION	77
DELETE INDEX EVENTS.	79
DISABLE SERVER FOR DB2TEXT	80
DISABLE TEXT COLUMN.	81
ENABLE SERVER FOR DB2TEXT	82
ENABLE TEXT COLUMN	83
GET ENVIRONMENT	89
GET INDEX SETTINGS	90
GET INDEX STATUS	92
GET STATUS	93
GET TEXT CONFIGURATION	94
GET TEXT INFO	95
QUIT	96
REORGANIZE INDEX	97
RESET INDEX STATUS	98
UPDATE INDEX	99
Chapter 9. Administration commands for the server	101
TXSTART	102
TXSTATUS.	103
TXSTOP.	104
IMOTHEsc.	105
IMOTRACE.	106
Chapter 10. Search functions	111
The DB2 Text Extender distinct types	111
A summary of DB2 Text Extender functions	111
CCSID	112
CONTAINS	113
FORMAT.	114

INIT_TEXT_HANDLE	115
LANGUAGE	116
NO_OF_MATCHES	117
RANK	118
REFINE	119
SEARCH_RESULT	120
Chapter 11. Syntax of search arguments	121
Search argument	122
Chapter 12. Linguistic processing for linguistic and precise indexes	133
Linguistic processing when indexing	133
Basic text analysis	134
Reducing terms to their base form (lemmatization)	135
Stop-word filtering	135
Decomposition (splitting compound terms)	135
Linguistic processing for retrieval	136
Synonyms	137
Thesaurus expansion	137
Sound expansion	137
Character and word masking	138
Thesaurus concepts	138
Terms	139
Relations	139
Creating a thesaurus	141
Chapter 13. Configuration files	145
Client configuration file	145
Server configuration file	146
Chapter 14. Return codes	149
Chapter 15. Messages	155
SQL states returned by DB2 Text Extender functions	155
Messages from DB2 Text Extender	158
Chapter 16. Search engine reason codes	169
Chapter 17. Error event reason codes	171
Notices	183
Trademarks.	184
Glossary	187
Index	191

About this book

This book describes how to use DB2 Text Extender to prepare and maintain a DB2[®] UDB server for z/OS[™] for retrieving text data. It also describes how you can use DB2 Text Extender-provided SQL functions to access and manipulate these types of data. By incorporating DB2 Text Extender's functions in your program's SQL statements, you can create powerful and versatile text-retrieval programs.

References in this book to "DB2" refer to DB2 UDB.

Who should use this book

This book is intended for DB2 database administrators who are familiar with DB2 administration concepts, tools, and techniques.

This book is also intended for DB2 application programmers who are familiar with SQL and with one or more programming languages that can be used for DB2 application programs.

How to use this book

This book is structured as follows:

"Part 1. Guide"

This part gives an overview of DB2 Text Extender, describes how to set it up after installation, and discusses planning considerations. It also describes how to prepare and maintain a DB2 server so that you can search for text.

Read this part if you are new to DB2 Text Extender and want to learn how to use the DB2 Text Extender functions to search for text.

"Part 2. Reference"

This part presents reference information for DB2 Text Extender functions, commands, and diagnostic information such as messages and codes.

Read this part if you are familiar with DB2 Text Extender concepts and tasks, but need information about a specific DB2 Text Extender function, command, message, or code.

How to read the syntax diagrams

Throughout this book, syntax is described using the structure defined as follows:

- Read the syntax diagrams from left to right and top to bottom, following the path of the line.
 - The >>-- symbol indicates the beginning of a statement.
 - The --> symbol indicates that the statement syntax is continued on the next line.
 - The >-- symbol indicates that a statement is continued from the previous line.
 - The -->< symbol indicates the end of a statement.
- Required items appear on the horizontal line (the main path).

▶—required item—▶

- Optional items appear below the main path.

About this book



- If you can choose from two or more items, they appear in a stack. If you *must* choose one of the items, one item of the stack appears on the main path.



If choosing none of the items is an option, the entire stack appears below the main path.



A repeat arrow above a stack indicates that you can make more than one choice from the stacked items.



- Keywords appear in uppercase; they must be spelled exactly as shown. Variables appear in lowercase (for example, `srcpath`). They represent user-supplied names or values in the syntax.
- If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Accessibility

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products. The major accessibility features in z/OS products enable users to:

- Use assistive technologies such as screen reader and screen magnifier software
- Operate specific or equivalent features by using only a keyboard
- Customize display attributes such as color, contrast, and font size

Assistive technology products, such as screen readers, function with the z/OS user interfaces. Consult the documentation for the assistive technology products for specific information when you use assistive technology to access these interfaces.

Online documentation is available in the DB2 Information Center, which is an accessible format when used with assistive technologies such as screen reader or screen magnifier software. The DB2 Information Center for z/OS solutions is available at the following Web site:

<http://publib.boulder.ibm.com/infocenter/db2zhelp>.

Summary of Changes

The documentation has been provided for all releases of the OS/390® and z/OS Text Extender. However, there are certain differences between the releases.

- OS/390 release 2.9 (or later) and z/OS releases include Text Search Engine release 4 (FMID HIMN230).

Where differences in functionality occur, these have been highlighted.

Accordingly, all the following changes only refer to OS/390 release 2.9 (or later) and z/OS releases:

- **More supported languages**

The list of supported languages and CCSIDs has been extended.

- **Structured documents**

A more powerful structured document search facility is now available.

- **Search results**

New user-defined search functions are now available in SQL.

Summary of Changes

Part 1. Guide

Chapter 1. An overview of DB2 Text Extender

DB2 Text Extender adds the power of full-text retrieval to SQL queries.

DB2 Text Extender offers DB2 UDB for OS/390 and z/OS users and application programmers a fast, versatile, and intelligent method of searching through text documents. DB2 Text Extender's strength lies in its ability to find not only what you directly ask for, but also word variations and synonyms.

Note

Text Extender for OS/390 and z/OS does not support the DB2 UDB for OS/390 and z/OS data sharing function in a Parallel Sysplex environment.

The other Extenders can search for images, video and voice data, and can handle XML documents and spatial data.

At the heart of DB2 Text Extender is IBM's linguistic search technology described in Chapter 12, "Linguistic processing for linguistic and precise indexes," on page 133. It allows your applications to access and retrieve text documents in a variety of ways. Your applications can:

- Search for documents that contain specific text, synonyms of a word or phrase, or sought-for words in proximity, such as in the same sentence or paragraph.
- Do wildcard searches, using front, middle, and end masking, for word and character masking.
- Search for documents of various languages in various document formats.
- Make a "fuzzy" search for words having a similar spelling as the search term. This is useful for finding words even when they are misspelled.
- Make a free-text search in which the search argument is expressed in natural language.
- Search for words that sound like the search term.
- Use a thesaurus.

You can integrate your text search with business data queries. For example, you can code an SQL query in an application to search for text documents that are created by a specific author, within a range of dates, and that contain a particular word or phrase. Using the DB2 Text Extender programming interface, you can also allow your application users to browse the documents.

By integrating full-text search into DB2 UDB for OS/390 and z/OS's SELECT queries, you have a powerful retrieval function. The following SQL statement shows an example:

```
SELECT * FROM MyTextTable
WHERE version = '2'
AND DB2TX.CONTAINS (
    DB2BOOKS_HANDLE,
    "authorization"
    IN SAME PARAGRAPH AS "table"
    AND SYNONYM FORM OF "delete") = 1
```

DB2TX.CONTAINS is one of several DB2 Text Extender search functions. DB2BOOKS_HANDLE is the name of a handle column referring to column DB2BOOKS that contains the text documents to be searched. The remainder of the

Overview

statement is an example of a search argument that looks for authorization, occurring in the same paragraph as table, and delete, or any of delete's synonyms.

DB2 Text Extender in the DB2 client/server environment

Figure 1 shows how DB2 Text Extender is integrated into the DB2 client/server environment.

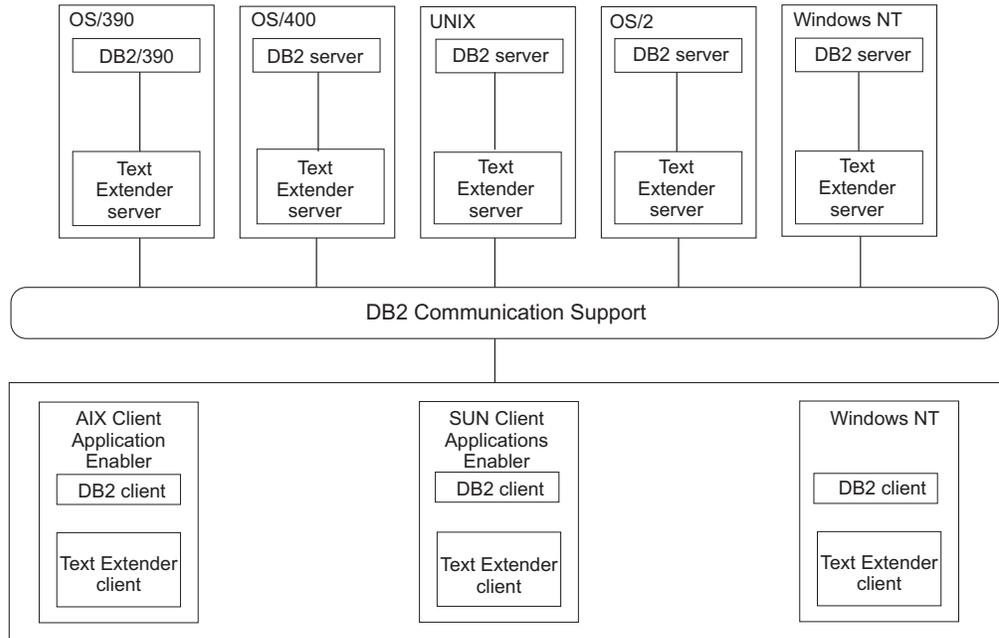


Figure 1. Integration of DB2 Text Extender into the DB2 client/server environment

For a list of the DB2 Communications Support protocols (such as TCP/IP or NETBIOS) for a client, see the *DB2 Quick Beginnings Guide* for the appropriate platform.

The main part of DB2 Text Extender is installed on the same machine as the DB2 server. Only one DB2 Text Extender server instance can be installed with one DB2 server instance.

A DB2 Text Extender installation is flexible and can comprise:

- One or several DB2 Text Extender servers on any of the operating systems shown in Figure 1, where UNIX[®] includes AIX[®], Solaris, and HP-UX workstations.
- AIX, Solaris, HP-UX, OS/2, Windows 98, or Windows 95, Windows NT[®], and Windows 2000 clients with access to one or several remote DB2 Text Extender servers.
- AIX clients containing a local server and having access to remote servers.

Figure 2 on page 5 shows a typical DB2 Text Extender configuration. To run DB2 Text Extender from a client, you must first install a DB2 client and some DB2 Text Extender utilities. These utilities constitute the DB2 Text Extender “client” although it is not a client in the strict sense of the word. The client communicates with the server via the DB2 client connection.

Figure 2 shows the DB2 Text Extender configuration.

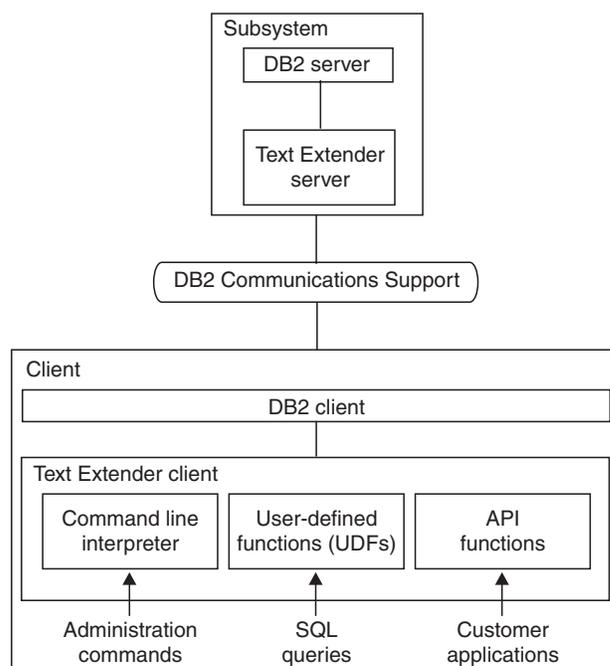


Figure 2. A DB2 Text Extender configuration

DB2 Text Extender has the following main components:

- **A command line interpreter.** Commands are available that let you prepare text in columns for searching, and maintain text indexes.
- **SQL functions.** Functions are available that you can include in SQL queries for searching in text, and finding, for example, the number of times the search term occurs in the text. For clarity, the figure shows the SQL functions on the client because they can be used as part of an SQL query. In fact, they are part of the server installation and are executed there. However, these functions can be used from any DB2 client without the need to install the DB2 Text Extender client.

Tip

The DB2 Text Extender client utilities offer text preparation and administration functions. (These functions are available on the server.) To use these functions, you must install the DB2 Text Extender client.

If you need only the search capability on the client using DB2 UDB for OS/390 and z/OS SQL statements, you do not need to install the DB2 Text Extender client. All communication is handled by DB2 UDB for OS/390 and z/OS and the DB2 Text Extender search engine runs only on the server.

Overview

Chapter 2. Customization, Configuration and maintenance

Customization

After you have completed the SMP/E installation, follow the steps described here to set up DB2 Text Extender so that it runs together with Text Search and DB2. Some of these steps are carried out by a system programmer and some by a database administrator.

Prerequisites

DB2 UDB Text Extender for z/OS requires:

- Workload Manager (WLM) environment as described in the DB2 Program Directory
- IBM Text Search FMID HIMN230, as described in the DB2 Program Directory

The DB2 Text Extender instance owner must have a super user ID and a DB2 SYSADM authority. It must also have its own home directory in the Unix system services on z/OS.

Installation files and directories under HFS

DB2 Text Extender is customized in MVS™ datasets and files within the Hierarchical File System (HFS). During SMP/E installation, you can specify target datasets and HFS directories. By default, the following HFS directories are used:

```
/usr/lpp/db2tx  
/usr/lpp/db2tx/IBM  
/usr/lpp/db2tx/bin  
/usr/lpp/db2tx/ddl  
/usr/lpp/db2tx/include  
/usr/lpp/db2tx/install  
/usr/lpp/db2tx/instance  
/usr/lpp/db2tx/lib  
/usr/lpp/db2tx/nls  
/usr/lpp/db2tx/samples
```

If you want to specify a different set of directories during installation, you can prefix these directories with additional names, like `/u/myname`. For an example, see the SMP/E sample job in the Program Directory. However, you must not omit the default part of the directory names. The fully qualified name becomes, for example, `/u/myname/usr/lpp/db2tx/IBM`. This chapter assumes that you are using the default directory names.

You must customize DB2 Text Extender in the UNIX Shell and HFS directories, because it requires Text Search for z/OS, which provides services only in the UNIX Shell. DB2 Text Extender itself provides its `db2tx` command-line interface only via the UNIX Shell on z/OS.

Checking the Text Search Installation

Before customizing DB2 Text Extender, check the installation of Text Search for z/OS. To do this:

1. Log in as a user with super-user privilege.

customization

Do not use su to become a super-user – your permissions may not be sufficient during customization.

2. Ensure that Text Search is customized and usable. To do this, run the Text Search Engine installation verification program:
 - a. Change your current directory to the Text Search installation directory. The default directory is /usr/lpp/TextTools/install.
 - b. Run the procedure imocust
 - c. Select 2. Customizable Installation
 - d. Select 6. Process installation verification procedure.

The system provides you with installation-related information for Text Search. If you find an installation error, refer to the Text Search documentation.

3. Some of the related files of Text Search may have been installed in directories accessible system-wide, such as imoisinf in /etc. Ensure that you have write access to these files.
4. Make yourself familiar with the concept of an instance in the Text Search environment. DB2 Text Extender refers to it frequently and creates its own instances with a slightly different meaning.

Deciding whether to use Linklist or STEPLIB

There are two ways to access the DB2 Text Extender load library h1q.SDESMOD1 from the UNIX Shell of z/OS environment:

- By adding an entry h1q.SDESMOD1 to your Linklist, that is, the PROGxx or the LNKLSTxx of your PARMLIB
- By adding h1q.SDESMOD1 to the STEPLIB environment variable

The Linklist approach is recommended for performance reasons, and to simplify the customization steps. If you cannot add h1q.SDESMOD1 to the Linklist, then you must use STEPLIB.

If you work with the STEPLIB environment variable ensure all users of DB2 Text Extender have at least one of the following:

- RACF[®] READ access for the z/OS load library h1q.SDESMOD1, which contains all load modules
- RACF EXEC access if there is a PROGRAM profile for the load library members

Running the customization routine

In this step, you run the customization routine descust that runs under the UNIX Shell of z/OS, and can be found in an HFS directory after SMP/E installation.

1. Log in as a user with super-user privilege with at least UPDATE access to the DB2 Text Extender target library datasets h1q.SDES*.

Do not use su to become a super-user – your permissions may not be sufficient during customization (oedit is called, for example).

2. Change your current directory to the DB2 Text Extender installation directory. The default directory is /usr/lpp/db2tx/install.
3. Start the customization procedure menu using the command descust.
4. Press Enter on request until you receive the following menu:

```
REXX Procedure descust - Customize DB2 Text Extender
Type in selection number to execute, or type in ?n (n = selection number)
to get help, and press ENTER.
0. End
1. Customize application parameters - file desparm
```

2. Display the list of all parameters using 'pg desparm'
3. Process final customization steps
4. Display final customization logging file using 'pg destxlnk.log'
5. Check/adapt environment settings file db2txprofile
6. Reset final customization
7. Display activity logging file using 'pg descust.log'
8. Enter your own shell command

Please enter your selection:

====>

5. Run the steps 1 through 5 on the menu. Steps 2 and 4 are optional.

For menu item 1. Customize installation..., the default parameters are:

```
DESDIRPATH=/usr/lpp
DESINSTDIR=db2tx
DESLIB=DES.SDESMOD1
DESTSEINFO=/etc/imoisinf
INSTOWNER=db2txins
INSTOWNERHOMEDIR=/u/db2txins
DESGROUPNAME=<no default, provide a name>
DESLANG=3
APPLENV=          <no default, provide a name>>
DB2SSN=           <no default, provide a name>
DB2LOC=           <no default, provide a name>
DESDATABASE=db2tx
```

For a description of these parameters, select menu item 1.

If menu item 3. Process final installation steps ends with a return code greater than 0, check the installation logging file using menu item 4 for error messages.

If you want to change these parameters after installation, use menu item 1. Customize application... to edit file desparm. You can repeat these steps as often as you wish, but if you change the customization parameters using item 1 you must also rerun items 3 and 5.

If you want to use the automatic indexing feature, make sure that the system path contains an entry with /usr/lpp/db2tx/bin where the executable desxctl resides. Text Extender uses cron jobs to start automatic indexing. The users path will not be checked for such cron jobs.

Customizing the user's runtime profile

When you run the customization routine, you customize an executable profile db2txprofile. This file contains the runtime environment settings that were specified during customization. You can find the profile in the directory /usr/lpp/db2tx/install, or as a logical link in the instance owner's home directory under db2tx <INSTANCEOWNER_HOMEDIR>/db2tx, for example, /u/myname/db2tx.

You will find it useful to code the following call from your .profile to the db2txprofile, so that you always have the correct runtime options set for DB2 Text Extender:

```
# Set DB2 Text Extender environment variables
if [ -f ./db2tx/db2txprofile ]; then
. ./db2tx/db2txprofile
fi
```

If you used the Linklist approach for your customization, comment out the STEPLIB environment settings in the db2txprofile to avoid problems if the loadlib is changed.

If you used the STEPLIB approach for your installation:

customization

- For menu item 1. Customize application... you must ensure that the DESLOADLIB parameter contains the full dataset name of the DB2 Text Extender loadlib before you continue with the next steps.
- Menu item 5. Check/adapt environment settings... updates the STEPLIB environment variable setting in the file db2txprofile.

When you run descust for the first time, the following customized members of datasets are created:

- DESENA1, copied to h1q.SDESSCR1
- TXWLM1, copied to h1q.SDESDB2I
- DSNA0IN1, copied to h1q.SDESDB2I
- DESCSP1, copied to h1q.SDESDB2I
- DESCDB1, copied to h1q.SDESDB2I

The chapters “Workload Manager related tasks” on page 11 and “DB2-related tasks” on page 11 describe how these are used.

Log files written during the final installation steps

These log files are created in the installation directory /usr/lpp/TextTools/install:

- descust.log: Log of menu items processed including error returns
- destx1nk.log: Log of final customization steps (menu item 3)

Keep these files and use them as a reference for IBM Services if you get a bad return code from descust.

Customizing DB2 Text Extender to run on more than one DB2 subsystem

To configure and run DB2 Text Extender on an additional DB2 subsystem simultaneously, you must create an additional set of customized files described above. To do this:

1. Complete the customization of DB2 Text Extender for your first subsystem as described above.
2. Create an additional DB2 Text Extender instance owner user ID with its own home directory.
3. Use the RACF permissions and DB2 grants of the first db2tx user ID analogous for this subsystem and this user ID.
4. Run the descust procedure, menu item 1. The following parameters must be different from the subsystem parameters previously setup, or system problems maybe created.

```
INSTOWNER=      <new instance owner>
INSTOWNERHOMEDIR=<new instance owners home dir>
APPLENV=        <no default, provide new name>
DB2SSN=         <no default, provide new name>
DB2LOC=         <no default, provide new name>
```

A new set of configuration dataset members with a higher number is created:

- DESENA2, copied to h1q.SDESSCR1
- TXWLM2, copied to h1q.SDESDB2I
- DSNA0IN2, copied to h1q.SDESDB2I
- DESCSP2, copied to h1q.SDESDB2I
- DESCDB2, copied to h1q.SDESDB2I

The previous ones are not overwritten. Whenever you repeat these steps and modify the parameters, the last character number is incremented.

If you run the customization procedure descust more than once, and additional members are created, you may have to add additional directory blocks to datasets h1q.SDESDB2I and h1q.SDESSCR1.

Workload Manager related tasks

In this step, you need a user ID with UPDATE access to the proclib dataset to edit the customization information of the started task of your WLM environment.

Configuring the WLM-started task

To configure the WLM-started task:

1. Copy batch job h1q.SDESDB2I(TXWLMn) to the proclib defined in your WLM environment. In TXWLMn, n is 1 for your first DB2 subsystem, 2 for your second subsystem, and so on, as described in “Customizing DB2 Text Extender to run on more than one DB2 subsystem” on page 10
2. Ensure that you have at least WLM read access to the following load libraries:
 - h1q.SIMOMOD1
 - h1q.SDESMOD1
 - h1q.SDSNEXIT
 - h1q.SDSNLOAD

Errors caused by insufficient access rights can only be found in the system log after starting the WLM.

For WLM to find these libraries, they must be defined in one of the following:

- The Linklist
- The STEPLIB section of the WLM job h1q.SDESDB2I(TXWLMn)
Additionally, the libraries must be added to the HFS environment variable STEPLIB which is set in the db2txprofile of the db2tx instance owner.

Activating the WLM application environment

To activate the load library h1q.SDESMOD1, and to make your changes to the configuration of the started task effective, you must refresh the application environment in use for the specified DB2 subsystem.

1. Enter the following z/OS system command:
`/v wlm,applenv=<WLM AppEnv Name>,refresh`

The WLM AppEnv Name is the name of the WLM environment that you specified in the DB2 Text Extender customization procedure descust. It must have been defined previously in the WLM environment. It must also have been defined by your DB2 administrator for use with a specific subsystem.

2. Check the correct startup of this application environment in the system log.

DB2-related tasks

DB2 Text Extender exploits some of the functionality provided by DB2. Perform the following steps to make the necessary database, stored procedures and functions available within a DB2 subsystem.

Creating DB2 Text Extender's own database

DB2 Text Extender needs its own database for storing its own meta data (except text indices). To create the database db2tx use the SPUFI input file in dataset h1q.SDESDB2I(DESCDBn), where n is the last character number incremented by each run of descust for another DB2 subsystem.

Execute this input file using DB2 SPUFI and check that the database has been created successfully.

Defining DB2 Text Extender's own stored procedure

DB2 Text Extender provides most of its functionality via stored procedures on the server. To create the stored procedure DB2TX.DESSRVSP, use the SPUFI input file in dataset h1q.SDESDB2I(DESCSPn), where n is the last character number incremented by each run of descust for another DB2 subsystem.

Execute this input file using DB2 SPUFI in the relevant subsystem, and check that the stored procedure has been created successfully.

Then start the stored procedure DB2TX.DESSRVSP by using the DB2 command in the relevant subsystem:

```
-sta proc(DB2TX.DESSRVSP)
```

Defining DB2 Text Extender's own SQL functions

Start DB2 Text Extender-supplied SQL functions by using the DB2 command in the relevant subsystem:

```
-sta func specific(*.*) via SPUFI
```

Troubleshooting

If you detect errors while performing the customization steps, check the following items:

- Ensure that you are working with a super-user ID with sufficient read- and write-access permissions to the files.
- Check your PATH environment variable using `echo $PATH`. It must contain a `.` or `./` entry. To change it, use the command `export PATH=$PATH:..`
- Ensure that the prerequisites are fulfilled. DB2 Text Extender needs all of the prerequisites for all of its functions. Note that DB2 Text Extender works only with the version of the Text Search Engine specified in the DB2 Program Directory.
- Ensure that the Text Search Engine is customized and usable. To check this, run the Text Search Engine installation verification program:
 1. `cd /usr/lpp/TextTools/install`
 2. Call `descust`
 3. Select "2. Customizable Installation"
 4. Select "6. Process installation verification procedure"
- Ensure that the DB2 ODBC runtime environment is enabled and customized as described in the DB2 UDB ODBC documentation, by using the DSNTIJCL bind sample in SDSNSAMP. Be sure to have issued a `GRANT EXECUTE` either to all users of DB2 Text Extender, or to PUBLIC, for:
 - Each package
 - The DB2 data-access plan
- Ensure that the WLM region size is set to an appropriate value.
- Check the log files described in 2.3 Log files written in final installation steps to get information that could be relevant to the problem.

- If the customization procedure descust fails, you can eliminate file-creation conflicts by removing directory db2tx in the instance owner home directory using the `rm -r` command.
- If you run the customization procedure descust more than once, and additional members are created, it might be necessary to add additional directory blocks to datasets h1q.SDESDB2I and h1q.SDESSCR1
- DB2 Text Extender may create links in system-wide accessible file systems. In some environments, these file systems are mounted read only. Being a super-user does not help in this case. Ensure that you have write access to the related file systems.
- For some of the tasks, such as starting the WLM environment, look at the console system log of your z/OS system to check for successful completion.
- Ensure that you have associated a user ID with your TXWLMn WLM environment address space that has the appropriate RACF permissions.

Configuration

This section describes the DB2 Text Extender environment variables and configuration information. Both of these let you specify default values for many parameters needed by DB2 Text Extender.

Environment variables

The environment variables set the default values of environment parameters for the administration command interface db2tx. To display the current setting of environment variables, use the GET ENVIRONMENT command described in “Displaying the settings of the environment variables” on page 66.

DB2DBDFT	Default server name. The name of the DB2 UDB for OS/390 and z/OS server that is assumed if no server name is specified.
DB2TX_INSTOWNER	DB2 Text Extender instance name. This is the login name of the user that owns the instance.
DB2TX_INSTOWNERHOMEDIR	Instance owner’s home directory.
DB2INSTANCE	Current database manager instance.

The environment variables valid for the process of indexing, stored procedures, background tasks, and user-defined functions are set in h1q.SDESSCR1 (DESENVAn). This member is automatically generated by customization procedure descust.

Text configuration settings

Each database has text configuration settings consisting of:

- Text characteristics
- Index characteristics
- Processing characteristics

These are set when you enable the database for use by DB2 Text Extender. The ENABLE DATABASE command takes either the settings that you specify in the command, or it takes the initial settings described here. You can display and change these default settings; see “Displaying the text configuration settings” on page 66 and “Changing the text configuration” on page 14.

Configuration

The text characteristics

Chapter 4, “Planning for your search needs,” on page 21 describes the document formats, languages, and CCSIDs supported by DB2 Text Extender. Default values for these are required by various commands.

FORMAT	Initial setting: TDS
LANGUAGE	Initial setting: The LANGUAGE that was set for the database
CCSID	Initial setting: The CCSID that was set for the subsystem

The index characteristics

DIRECTORY	Directory to be used to store the index. Initial setting: <i>DB2TX_INSTOWNER/db2tx</i>
INDEXTYPE	Index type to be used. See “Types of search” on page 29 for a description. Initial setting: NGRAM

The processing characteristics

UPDATEINDEX	Setting to determine when the first index update occurs: either immediately during the enabling step, or later according to the update frequency settings (NOUPDATE), or as a result of an explicit UPDATE INDEX command. Initial setting: UPDATE
COMMITCOUNT	Setting to determine after how many insert or update statements DB2 Text Extender issues a DB2 UDB for the z/OS commit statement. See “Enabling a text column in a large table” on page 45. Initial setting: 0

Changing the text configuration

When DB2 Text Extender is first installed, default values are set for the text configuration. To display the current text configuration values, see “Displaying the text configuration settings” on page 66.

To change the text configuration to be used as default values when indexes are created, enter:

```
db2tx CHANGE TEXT CFG USING settings
```

Examples:

To change the default index type and the default index directory for future indexes:

```
DB2TX CHANGE TEXT CONFIGURATION USING  
INDEXTYPE precise  
DIRECTORY /myfs/indexes
```

Setting up and maintaining a DB2 Text Extender server

The next step for a newly installed DB2 Text Extender is to set up the DB2 Text Extender server for use by DB2 Text Extender clients. This is normally done by a DB2 Text Extender server administrator and involves:

1. Creating the DB2 Text Extender instance
2. Starting a DB2 Text Extender server

Other tasks for maintaining a DB2 Text Extender server are:

1. Backing up and restoring DB2 Text Extender indexes and enabled subsystems
2. Tracing faults

See Chapter 9, “Administration commands for the server,” on page 101 for the command syntax.

Creating the DB2 Text Extender instance

Before you can start to work with DB2 Text Extender, you must create the DB2 Text Extender instance, which offers an administration environment to maintain and store the indexes.

You can create one instance of DB2 Text Extender per subsystem. To create an instance, use the descust utility described in “Customization” on page 7.

An instance has to be created only once and remains active until it is dropped.

Starting and stopping a DB2 Text Extender server

Before you can index and search your documents, you have to start the DB2 Text Extender server.

To start the DB2 Text Extender server, in the z/OS UNIX Shell environment, enter:

```
txstart
```

To display the status of the DB2 Text Extender server, enter:

```
txstatus
```

To stop the DB2 Text Extender server, enter:

```
txstop
```

Backing up and restoring indexes and enabled servers

You can back up and restore enabled databases and the text indexes that DB2 Text Extender has created.

To back up:

1. Find out which tables have been enabled by DB2 Text Extender. To do this, enter

```
db2tx GET STATUS
```
2. Find out the names of the index directories used by the database. To do this, enter

```
db2tx GET INDEX SETTINGS table-name
```
3. Ensure that no index update is running, and then stop the DB2 Text Extender server with the command:

Setting up and maintaining a server

TXSTOP

4. Back up the index directories and their subdirectories `index` and `work`.
5. Back up the file `imomastr.dat` which is located in:
`instance_owner_home_directory/db2tx/txins000`
6. Restart the DB2 Text Extender server:

TXSTART

To restore:

1. Stop the DB2 Text Extender server:
TXSTOP
2. Save the existing `imomastr.dat` file.
3. Restore the backup copy of the `imomastr.dat` file.
4. Restore the backup copies of the index directories to the same path as before.
5. Restart the DB2 Text Extender server:

TXSTART

Tracing faults

If you need to report an error to an IBM representative, you may be asked to switch on tracing so that information can be written to a file that can be used for locating the error. Use the trace facility only as directed by an IBM Support Center representative, or by your technical support representative.

System performance is affected when tracing is switched on, so use it only when error conditions are occurring.

To turn tracing on, enter:

```
imotrace on [options]
```

The syntax, and lists of the events and components are given in “IMOTRACE” on page 106. Other options are also described there.

You can filter the trace by specifying a “mask” which causes the trace to accept or reject each trace record on the basis of its ID. The default is to trace everything.

A mask has four parts separated by periods, for example: `2.2-6.1,3.*` where:

- 2** indicates DB2 UDB DB2 Text Extender.
- 2-6** includes only entries with an event ID between 2 and 6.
- 1,3** includes only those events reported by components 1 and 3.
- *** includes all functions of the components.

You can exclude system errors below a certain severity, and you can specify, if the trace buffer becomes full, whether to keep the first or the last records.

To reproduce the error and write the trace information in binary to a dump file, enter:

```
imotrace dump dump-filename
```

To produce a formatted version of the dump file, enter:

```
imotrace format dump-filename formatted-filename
```

Setting up and maintaining a server

You can also write the trace information directly from shared memory to a formatted file while tracing is switched on:

```
imotrace format > formatted-file
```

After you have written the trace information to a file, turn tracing off using:

```
destrace off
```

Setting up and maintaining a server

Chapter 3. Getting started

Use this chapter to become familiar with the basics of making text searchable. It assumes that you are working with a running DB2 Text Extender system, one that has been installed and configured, and where a DB2 Text Extender instance has been created and started.

Tip

This chapter describes only the basics of making text searchable. Before preparing your own text for searching, read "Preparation before making text searchable" on page 39.

A simple example of making text searchable

1. Enter the following command in the UNIX Shell environment to start the DB2 Text Extender command line processor:

```
db2tx
```

2. **Connect to a subsystem**

Choose a subsystem that contains text that you want to make searchable. To connect to the subsystem, enter:

```
db2tx=>CONNECT TO subsystem
```

The connect is implicit via the environment variable DB2DBDFT.

3. **Enable the subsystem for text search**

To enable the connected subsystem, enter:

```
db2tx=>ENABLE SERVER; FOR DB2TEXT
```

4. **Enable a text column for text search**

Enter the following command to enable DB2 Text Extender to search in text column `mycolumn` in table `db2tx.sample`, and to assign the name `myhandle` to the handle column that this command creates.

```
db2tx=>ENABLE TEXT COLUMN db2tx.sample mycolumn HANDLE myhandle
```

This command creates a text index. Default values are used for the type of documents being indexed and for the index characteristics.

5. **Check the status of the index you are creating**

Enter:

```
db2tx=>GET INDEX STATUS db2tx.sample HANDLE myhandle
```

6. **Leave the DB2 Text Extender command line processor**

Enter:

```
db2tx=>QUIT
```

7. **Search for text**

Now your documents can be searched. Use SPUFI to search for text in your database table. Try this SELECT command which finds all occurrences of `searchterm` in the text that you have just indexed:

```
SELECT COUNT (*)  
FROM sample  
WHERE DB2TX.CONTAINS (myhandle,'searchterm') = 1
```

Getting started

Chapter 4. Planning for your search needs

Before you begin the steps described in Chapter 5, “Making text searchable,” on page 39, you must find out:

- What format and code page your documents have, and what language they are in
- How to avoid code page problems
- What kind of search functionality you will need
- What disk space you will need
- What a text index is, and whether you want a common index for a DB2 table or a separate index for each table column.

This chapter describes why and how to collect this information.

There are several types of index to choose from: linguistic, precise, and Ngram. The choice of index type is significant. For example, if you choose *linguistic* as the index type, you can search for word variations and synonyms of the search term. The index type also affects indexing performance and the size of the index. You can also make use of the search capabilities of more than one index type by creating several indexes, each having a different index type, per text column.

Why text documents need to be indexed

A fast information retrieval system does not sequentially scan through text documents; this would take too long. Instead, it operates on a previously built text index. You can think of a text index as consisting of significant terms extracted from the text documents, each term stored together with information about the document that contains it.

A text index contains only relevant information; insignificant words, such as “and”, “of”, and “which”, are not indexed. (No stop-word filtering is done for Ngram indexes.) DB2 Text Extender uses a list of these words, known as *stop words* to prevent them from being indexed. The retrieval system searches through the index for the terms requested to find which text documents contain those terms.

Tip

If you need to modify the list of stop words, do it only once, and at installation time.

A list of stop words per language is stored in a file that you can modify (see “Modifying the stop-word and abbreviation files” on page 37), but, because there is one file for the whole system, you should change it only once while you are setting up DB2 Text Extender for the first time. If you change the file later, existing indexes will not reflect the change.

As an example, let’s say that some documents contain the name of a weekly magazine called “Now”. If you remove this word from the stop words, it will be indexed and can be found by future searches. However, any indexes created before you removed the stop word will not contain the word “now”, and a search for it will be unsuccessful.

If you do decide to change the stop words, and you want this change to be reflected throughout, you must recreate all your indexes.

Why text documents need to be indexed

Indexing is a two-step process. The first step is to record in a *log table* the text documents that need to be indexed. This occurs automatically through DB2 *triggers* whenever you insert, update, or delete a text document in a column.

The second step is to index the text documents listed in the log table. This may be done periodically. The terms of those documents that were inserted or changed in the column are added to the index. The terms of those documents that were deleted from the column are removed from the index.

Figure 3 shows how to index only significant terms.

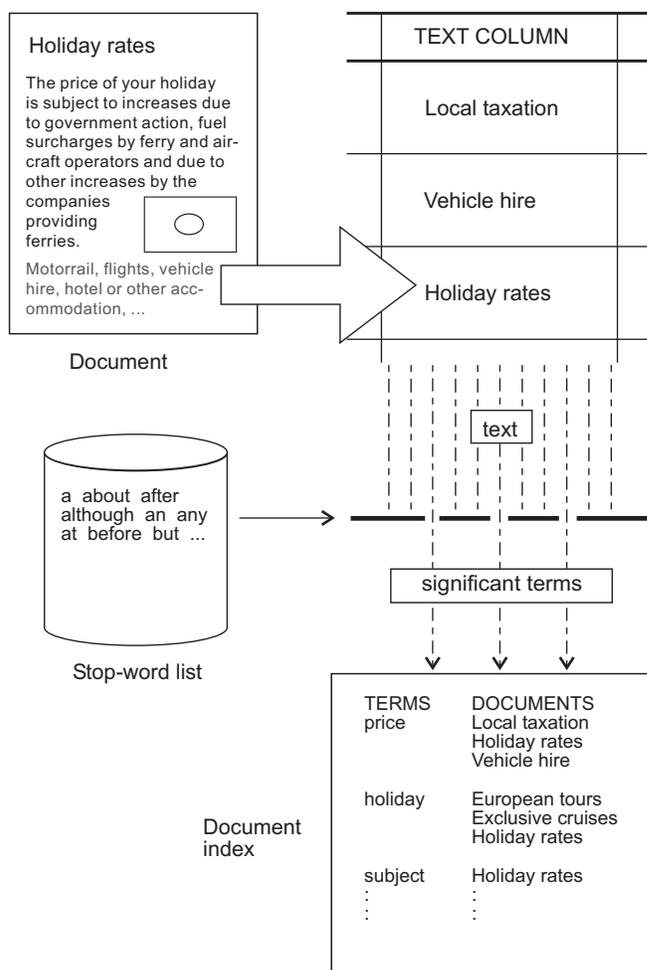


Figure 3. Indexing only significant terms

Which document formats are supported

DB2 Text Extender needs to know the format (or type) of text documents, such as HTML or ASCII, that you intend to search. This information is needed when indexing text documents.

The text document formats supported are:

HTML

Hypertext Markup Language

XML

Extended Markup Language

ASCII_SECTIONS

Structured ASCII containing sections

TDS	Flat ASCII
The following document formats are supported for backward compatibility reasons:	
AMI	AmiPro Architecture Version 4
FFT	IBM Final Form Text: Document Content Architecture
MSWORD	Microsoft Word, Versions 5.0 and 5.5
RFT	IBM Revisable Form Text: Document Content Architecture
RTF	Microsoft Rich Text Format (RTF), Version 1
WP5	WordPerfect (Windows), Versions 5.0, 5.1, and 5.2

HTML documents, special considerations

The treatment of umlauts and diacritical characters in HTML documents depends on the code page of the document:

- For code pages 37, 273, 277, 278, 280, 284, 297, 437, 500, 819, 850, 858, 860, 863, 871, 923, 924, and 1252 the following apply:
 - Entity notation is used for umlauts and special characters, for example, ¨ for ä.
 - Only those characters that have a code point on code page 819 (ASCII) or 500 (EBCDIC) are valid.
 - During indexing, documents containing language-specific characters, such as ä, lead to errors in word recognition if the document's code page is neither 819 nor 500.
 - If you are adding documents to an Ngram index, the index must have been created using code page 819, 500, or UTF8. However, note that UTF8 is only available with OS/390 release 2.9 or higher, and z/OS releases, which include Text Search Engine release 4 (FMID HIMN230).
- For all other single-byte character set code pages, the following apply:
 - Entities are not resolved.
 - Special characters must be written in language-specific code points.

XML documents, special considerations

XML documents should contain an encoding string, which is used during parsing.

If the encoding string is not provided, the default code page is ignored for XML documents.

Note that the XML format is only available with OS/390 release 2.9 or higher, and z/OS releases, which include Text Search Engine release 4 (FMID HIMN230).

Using unsupported document formats

For nonsupported document formats, specify a numeric ID. Valid values are 8192 to 65535. This value is passed as the source format to the user exit that converts the original format to TDS.

If, during indexing, there is a document that is not one of the supported types, DB2 Text Extender provides an exit that writes the document to a disk and calls a

Document formats

program that you provide to extract the text into one of the supported formats. The user exit must be registered in both the server configuration file (imosrv.ini) and the client configuration stream file (imocl.ini).

Update the USEREXIT option in the [DOCUMENTFORMAT] section with the name of the user exit.

To enable the user exit, edit the following ASCII files:

```
$DB2TX_INSTOWNERHOMEDIR/db2tx/imocl.ini  
$DB2TX_INSTOWNERHOMEDIR/db2tx/txinsnn/imosrv.ini
```

by adding the following statements:

```
[DOCUMENTFORMAT]  
USEREXIT=name_of_executable
```

where <name_of_executable> is the name of the user exit. You can specify a fully qualified file name, or, if the user exit is stored in a directory that is in the PATH statement, you can specify only the file name.

To call the user exit use the following syntax:

```
<name_of_user_exit> -sourcefile <sourcefilename>  
-targetfile <targetfilename>  
-sourceccsid <sourceccsid>  
-targetccsid <targetccsid>  
-sourceformat <sourceformat>  
-targetformat <targetformat>
```

sourcefilename

The file to be converted by the user exit program. The file name is fully qualified and is located in the working directory specified either in the client profile or the server instance.

targetfilename

The file containing the output of the user exit. This file is then used for processing by DB2 Text Extender. The file name is fully-qualified and points to the working directory specified either in the client profile or the server instance. The entries in the client profile are used for the API call EhwGetMatches and those in the server instance for the API call EhwUpdateIndex.

sourceccsid

The code page of the source file. This is the default code page.

targetccsid

The code page expected by DB2 Text Extender. The code page is 500.

sourceformat

The format of the source file. This is the default format.

targetformat

The format of the file expected by DB2 Text Extender. Currently, only the flat-file format (TDS) or, for section-enabled indexes, ASCIISECTION are supported.

The user exit must be able to return the following values:

- 0** Format conversion was successful.
- >0** Format conversion was not successful. During indexing, the error messages are written to the document error table. Use the `desmsgix` command to display the error messages.

Languages

DB2 Text Extender also needs to know in which language a document is written so that the correct dictionary can be used for the linguistic processing that occurs. Table 1 on page 36 shows a list of the language parameters that you can specify when you enable a text column or external documents.

CCSIDs

Tip

Before specifying a CCSID when enabling a text column, read “Avoiding code page problems when storing and enabling text” on page 27.

Documents can be indexed if they are in one of the CCSIDs in the table below.

Index type NGRAM only supports CCSID 500, 930, 933, 935, 937, and 939.

Note: CCSIDs 861, 865, and 4946 are not supported by DB2 UDB for OS/390 and z/OS. CCSID 37 is supported only for search. At indexing time, specify CCSID 500. To index documents having these CCSIDs, store the documents in a column with a binary data type (BLOB or FOR BIT DATA).

EBCDIC

37	US, Canadian English
273	Austrian, German
277	Danish, Norwegian
278	Finnish, Swedish
280	Italian
284	Spanish, Latin American
285	UK English
297	French
420	Arabic
424	Hebrew
500	International Latin-1
871	Icelandic
924	Latin-9
933	Korean
935	Simple Chinese
937	Traditional Chinese
1025	Russian
1388	Simple Chinese. Use instead of CCSID 935.
5026	Japanese Katakana
5035	Japanese Latin

CCSIDs

ASCII

437	US English
819	Latin-1
850	Latin-1
858	Latin-1 with Euro symbol
860	Portuguese
861 See note	Icelandic
862	Hebrew
863	Canadian
864	Arabic
865 See note	Danish, Norwegian
866	Russian
915	Russian
916	Hebrew
923	Latin-9 (ISO 8859–15)
1046	Arabic
1089	Arabic
1251	Russian
1252	Latin-1
1255	Hebrew
1256	Arabic

DBCS

878 (identical to RFC 1489)	Cyrillic Russian Internet
932	Japanese, combined SBCS/DBCS
942	Japanese, combined SBCS/DBCS
943	Japanese, combined SBCS/DBCS
5039	Japanese, combined SBCS/DBCS
954	Japanese
949	Korean
970	Korean
1363	Korean
948	Chinese (traditional), combined SBCS/DBCS
950	Chinese (traditional), combined SBCS/DBCS
964	Chinese (traditional), combined SBCS/DBCS
1381	Chinese (simplified), combined SBCS/DBCS
1383	Chinese (simplified), combined SBCS/DBCS
1386	Chinese (simplified), combined SBCS/DBCS

4946 See note	Latin-1 (CP850)
5039	Japanese

UNICODE

The following codes are only available with OS/390 release 2.9 or higher, and z/OS releases, which include Text Search Engine release 4 (FMID HIMN230).

1208	UTF8
13488	UCS2

Avoiding code page problems when storing and enabling text

The following areas have code page settings:

- The active application environment
- Each document
- Each DB2 database
- Each DB2 Text Extender index

When you store documents in a DB2 database column having a character data type, such as VARCHAR and CLOB, DB2 assumes that each document has the same code page defined for the system during installation and converts the document from that code page to the code page of the system. The code page of the system is either already the same as that of the active application environment (no conversion takes place), or it is the code page that you specified when you installed the system and which was different to the application code page. (conversion does take place).

When you store data in a DB2 subsystem in a column having a binary data type, such as BLOB or FOR BIT DATA, DB2 does not convert the data, and the documents retain their original CCSIDs.

shows how DB2 sets the CCSID of a database document.

Figure 4 shows how DB2 sets the CCSID of a database document.

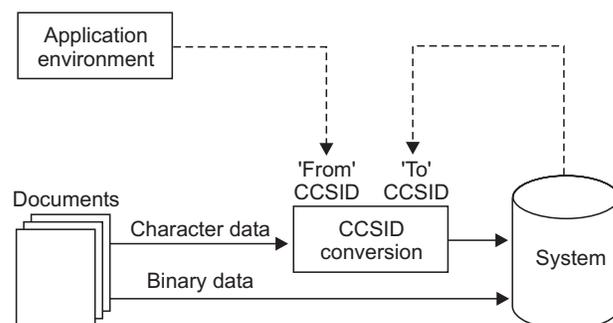


Figure 4. How DB2 sets the CCSID of a system document

When you enable a text column for use by DB2 Text Extender, that is, when you use the ENABLE TEXT COLUMN command to create an index for searching, the code page of the index is set either to the code page of the system (default), or to the current default which can be set using CHANGE TEXT CONFIGURATION command, or to the code page you specified in the ENABLE TEXT COLUMN command.

shows how DB2 Text Extender sets the CCSID of a text index.

Figure 5 shows how DB2 Text Extender sets the CCSID of a text index.

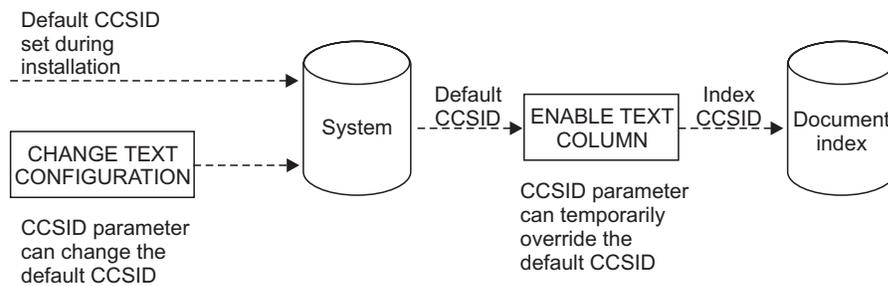


Figure 5. How DB2 Text Extender sets the CCSID of a text index

During a search, the CCSID of the system is used to interpret the CCSID of the search string.

shows how search uses the database CCSID.

Figure 6 shows how search uses the database CCSID.

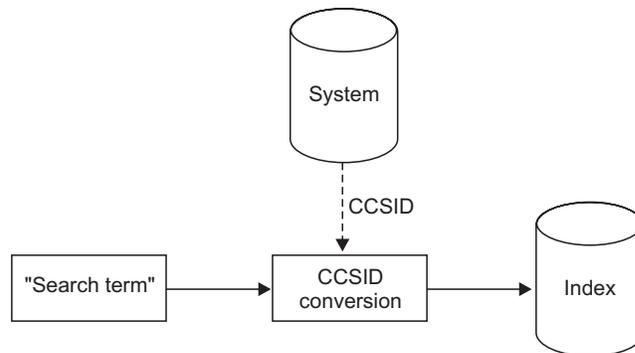


Figure 6. Search uses the system CCSID

Here's how to check the code page settings:

- To check the database code page, use the DB2 command:
`db2 get db cfg`
- To check the default index code page, use the DB2 Text Extender command:
`db2tx get text cfg`

To avoid code page problems it is important to coordinate these code-page settings correctly:

- **Example 1:** Environment 850, Document 850, System 850, Index 850
Correct. DB2 correctly assumes that the document has the same code page as the environment, and, because the system has the same code page as the environment, DB2 makes no conversion, storing the document with code page 850 in the system. When you then enable the text column, and take the default code page setting (850, the code page of the system), the document is indexed correctly into the 850-code page index.
- **Example 2:** Environment 1252, Document 1252, System 850, Index 850

Correct. DB2 correctly assumes that the document has the same code page as the environment, and makes a 1252-to-850 code page conversion when storing the document in the system. When you then enable the text column, and take the default code page setting 850 (the code page of the system), the document is indexed correctly into the 850 code page index.

- **Example 3:** Environment 1252, Document 850, System 850, (Index ANY)

Error. DB2 incorrectly assumes that the document (code page 850) has the same code page as the environment (code page 1252) and makes an incorrect 1252-to-850 code page conversion when storing the document in the system.

- **Example 4:** Environment 1252, Document 850, System 1252, Index 850

Correct. DB2 incorrectly assumes that the document has the same code page as the environment, but, because the environment code page is the same as the system codepage, DB2 makes no conversion, storing the document in the system in code page 850. When you then enable the text column, however, you must specify a document code page of 850 so that the document is correctly indexed into the 850-code page index.

- **Example 5:** Environment 1252, Document 1252, System 850, Index 1252

Potential error. DB2 correctly assumes that the document has the same 1252 code page as the environment and converts the document to code page 850 when storing it in the system. The potential error occurs if you then specify the document's original code page of 1252 when you enable the text column for the 850 code page index. The correct action would be to take the default code page setting 850 (the code page of the system).

Types of search

You can assign one of these index types and various options to a column containing text to be searched: *linguistic*, *precise*, and *Ngram*. You must decide which index type to create before you prepare any such columns for use by DB2 Text Extender. For a more detailed description of how each type of index affects linguistic processing, read Chapter 12, "Linguistic processing for linguistic and precise indexes," on page 133.

Types of search

Summary

- For **searching for linguistic word variations**, use a **linguistic index**.
Finds word variations based on normalization and stemming and on the use of a dictionary; uses the least disk space.
- For **making exact searches**, use a **precise index**.
Finds the term exactly as entered; indexing and search are faster; uses more disk space. If NORMALIZED, searches are case-insensitive.
- For **searching for character variations**, use an **Ngram index**.
Finds words even if spelled incorrectly; if CASE_ENABLED to allow case-sensitive search, the index uses more space, and searches can take longer.
- For **searching in DBCS documents**, use an **Ngram index**.
The only choice for DBCS documents, but can also be used for SBCS documents of type TDS.

DB2 Text Extender offers a wide variety of search options, though not all are available for all index types. See Table 9 on page 127 and Table 10 on page 127 before making your decision about which index type to use.

Note

Not all of the index types are available for languages which are supported by the Text Extender. See the table in “Dictionaries, stop-word lists, abbreviation lists, and language parameters” on page 36 for further information.

Linguistic search

For a linguistic index, linguistic processing is applied while analyzing each document’s text for indexing. This means that words are reduced to their base form before being stored in an index; the term “mice”, for example, is stored in the index as mouse.

For a query against a linguistic index, the same linguistic processing is applied to the search terms before searching in the text index. So, if you search for “mice”, it is reduced to its base form mouse before the search begins. Table 11 on page 133 summarizes how terms are extracted for indexing when you use a linguistic index.

The advantage of this type of index is that any variation of a search term matches any other variation occurring in one of the indexed text documents. The search term mouse matches the document terms “mouse”, “mice”, “MICE” (capital letters), and so on. Similarly, the search term Mice matches the same document terms.

This index type requires the least amount of disk space. However, indexing and searching can take longer than for a precise index.

The types of linguistic processing available depend on the document’s language. Here is a list of the types:

- Word and sentence separation.
- Sentence-begin processing.
- Dehyphenation.

- Normalizing terms to a standard form in which there are no capital letters, and in which accented letters like “ü” are changed to a form without accents. For example, the German word “Tür” (door) is indexed as `tuer`.
- Reducing terms to their base form. For example, “bought” is indexed as `buy`, “mice” as `mouse`.

Tip

Word fragments (words masked by wildcard characters) cannot be reduced to a base form. So, if you search for `swu%`, you will not find the word “swum”, because it is reduced to its base form in the index. To find it, you must search for `swi%`.

Variations of words not having the correct spelling cannot be reduced to a base form. Take, for example, the German word `röstete` which is indexed correctly in its base form, the verb `rösten`. A search term `röstete` or `rösteten` is normalized correctly to the base form `rösten`, and the term is found in the index. A search term `rostete` cannot be normalized to `rösten`, and the term is not found in the index.

- Word decomposition, where compound words like the German “Wetterbericht” (weather report) are indexed not only as `wetterbericht`, but also as `wetter` and `bericht`.
- Stop-word filtering in which irrelevant terms are not indexed. “A report about all animals” is indexed as `report` and `animal`.
- Part-of-speech filtering, which is similar to stop-word filtering; only nouns, verbs, and adjectives are indexed. “I drive my car quickly” is indexed as `drive` and `car`. The words “I” and “my” are removed as stop words, but additionally the adverb “quickly” is removed by part-of-speech filtering.

Precise search

In a precise index, the terms in the text documents are indexed exactly as they occur in the document. For example, the search term `mouse` can find “mouse” but not “mice” and not “Mouse”; the search in a precise index is case-sensitive.

In a query, the same processing is applied to the query terms, which are then compared with the terms found in the index. This means that the terms found are exactly the same as the search term. You can use masking characters to broaden the search; for example, the search term `experiment%` can find “experimental”, “experimented”, and so on.

Table 12 on page 134 gives some examples of how terms are extracted from document text for indexing when you use a precise index.

The advantage of this type of index is that the search is more precise, and indexing and retrieval is faster. Because each different form and spelling of every term is indexed, more disk space is needed than for a linguistic index.

The linguistic processes used to index text documents for a precise index are:

- Word and sentence separation
- Stop-word filtering.

Types of search

Make a fuzzy search or search in DBCS documents

An Ngram index analyzes text by parsing sets of characters. This analysis is not based on a dictionary.

If your text contains DBCS characters, you must use an Ngram index. No other index type supports DBCS characters.

This index type supports “fuzzy” search, meaning that you can find character strings that are similar to the specified search term. For example, a search for Extender finds the mistyped word Extendrrs. You can also specify a required degree of similarity.

Note: Even if you use fuzzy search, the first three characters must match.

To make a case-sensitive search in an Ngram index, it is not enough to specify the PRECISE FORM OF keyword in the query. This is because an Ngram index normally does not distinguish between the case of the characters indexed. You can make an Ngram index case-sensitive, however, by specifying the CASE_ENABLED option when the index is created. Then, in your query, specify the PRECISE FORM OF keyword.

When the CASE_ENABLED option is used, the index needs more space, and searches can take longer.

See “CCSIDs” on page 25 for a list of the CCSIDs supported by Ngram indexes. An Ngram index supports a list of native CCSIDs. For all other CCSIDs, the data is mapped from this CCSID to UTF8.

Although the Ngram index type was designed to be used for indexing DBCS documents, it can also be used for SBCS documents.

Note also that not all of the search syntax options are supported. See the summary of rules and restrictions in Chapter 11, “Syntax of search arguments,” on page 121.

Changing the index type

If you decide that the index type you are using is not suitable, first delete the index by disabling the text column or text table, and then recreate the index by re-enabling the text column or text table.

Creating one or several text indexes for a table

Chapter 5, “Making text searchable,” on page 39 describes how to prepare tables so that you can search in them for text. Before you do this preparation, however, you must decide to create either one text index that is common to all indexed text columns in a table, or several text indexes, one for each indexed text column. A table having a separate index for each text column is known as a *multi-index table*.

Using multiple indexes has these benefits:

- Creating a different index type for each text column
This gives you flexibility in the characteristics that are associated with a text column, such as when its index is periodically updated, and in which directory the index is stored. See “ENABLE TEXT COLUMN” on page 83 for a description of these characteristics.
- Indexing columns at different times

Creating one or several text indexes for a table

Indexing can be a time- and resource-consuming activity. By having a multi-index table, you can spread this activity over a period of time by indexing the columns at different times.

If you do not need the flexibility offered by a multi-index table, a common index makes DB2 Text Extender easier to maintain; when you enable a text table, you set the indexing parameters which are used as default values for all its text columns. Also, if you need to disable the columns you can do it using one command by disabling the text table.

Calculating the size of an index

The disk space you need for an index depends on the size and type of data to be indexed, and on the index type. Text documents written with word processors need less space because much of their content is taken up with control characters. As a guideline, for a linguistic index reserve disk space for about 0.7 times the size of the documents being indexed, then multiply this by 2 to reserve temporary space for reorganizing the index. For an Ngram index you'll need almost twice as much disk space.

If you have several large indexes, you should store them on separate disk devices, especially if you have concurrent access to the indexes during index update or search.

Updating an index

When a text document is added to a system , or when an existing document in a system is changed, the document must be indexed to keep the content of the index synchronized with the content of the system . When a text document is deleted from a system , its terms must be removed from the index.

Information about which documents are new, changed, and deleted is automatically stored by triggers in a log table. The documents listed in the log table are indexed the next time an index update takes place.

The UPDATE INDEX command lets you update an index immediately on request.

Incremental index update

To enable automatic index update, use the operating system crontab utility. If you need more information, refer to the Unix System Services manuals on how to use the cron man pages.

An example of a crontab file is as follows:

```
10,20 8,14,22 * * 0,3
```

```
/u/instanceowner/db2tx/te_update >>  
/tmp/te_update2>&1
```

In the example, the scheduled time, 10,20 8,14,22 * * 0,3, specifies minutes hours * * day.

The te_update script contains the following:

```
export LANG=C DB2INSTANCE=instanceowner export DB2INSTANCE  
LC_CTYPE=C export LC_CTYPE  
DB2TX_NLPS=/u/instanceowner/db2tx/dicts export DB2TX_NLPS
```

Updating an index

```
STEPLIB=SDSN.SDSNEXIT:SDSN.SDSNLOAD:h1q.SIMOMOD1:h1q.  
SDESMOD1 export STEPLIB  
. /u/instanceowner/db2tx/db2txprofile  
db2tx update index your_table_name HANDLE your_handle_name
```

For instanceowner, your_table_name, your_handle_name, and h1q, use your installation setup settings.

Working with structured documents (section support)

Section support allows you to index and search specific sections in a structured document, for example, in the title, author, or description. The documents can be in HTML format or flat-file documents with HTML-like tags. You define the markup tags and their corresponding section names in a *document model*. The document model defines which sections in the documents are indexed and therefore available for searching. The section names are descriptive names used in queries against that section.

To make section support available, you must specify INDEXPROPERTY SECTIONS_ENABLED when you enable the text column that contains the documents.

A *document model file* lists all the defined document models for the server instance. When a server instance is created, a sample document models file, IMOMODEL.INI, is created automatically in the server instance subdirectory. The file is in EBCDIC code page.

The document model information is copied to the index directory. If you change the document models file for the server instance after you create the index, it does not affect the section support for the created index.

A search on an index with section support, for example, to search for McDaniel in the section Author, might look as follows. The section, in this case Author, is always prefixed by the model name.

```
db2 "select count (*)  
    from db2tx.htmltable  
    where db2tx.contains  
        (myhandle,'MODEL myhtmlmodel SECTION (author) "McDaniel") = 1
```

Flat files and HTML documents

For flat files, the sections are marked up using HTML-like tags, such as <title> and <subject>. A document with marked-up sections might look as follows:

```
<title> IBM Dictionary of Computing  
<author> McDaniel, George  
<subject> Computers, Reference, ....
```

A document models file for flat files or HTML documents might look as follows. The model names and section names can contain only A-Z, a-z, and 0-9. Model names are always case sensitive. Section names can be either case sensitive or case insensitive; you specify the setting when you create the model.

```
;list of document models  
;model always starts with 'modelname' and the name of the model  
[MODELS]  
modelname=sample  
modelname=sample2  
modelname=sample3  
  
; a 'sample' document model definition  
; left - logical section name identifier used for searching in a document
```

```

; right - section name tag
[sample]
Title = title
Author = author
Subject = subject
Abstract = abstract
Content = content
Date=publishingdate

[sample2]
Title = title
Author = author
Subject = subject
[sample3]
Title = title
Author = author
Abstract = abstract
Docnum = docnum

```

If a document contains a marked-up section that is not defined in the document model, the contents of the section are included in the previously defined section for indexing and searching. For example, a document contains the following marked-up sections:

```

<title> IBM Dictionary of Computing
<subject> Computers, Reference, ...
<author> McDaniel, George
<abstract> Contains up-to-the-minute coverage of information processing
systems, communication products and facilities, personal computers, and office
systems, as well as the full range of IBM hardware and software products.

```

The document model, book, is defined as:

```

[MODELS]
modelname=book
[book]
Title = title
Author = author
Abstract = abstract

```

The <subject> section is not included in the book document model. When the document is indexed, the contents of the subject section are indexed with the contents of the title section. They are also available for searching within the title section.

If you specify a list of models when you create the index, the default model is the first in the list. You can change the default model using the `imomodix` command.

Note

The availability of document models depends on the OS/390 and z/OS release:

- For OS/390 version 2.8 or less, only a single document model can be associated with documents for indexing.
- For OS/390 version 2.9 or higher, and z/OS releases, multiple document models can be associated.

Dictionaries, stop-word lists, abbreviation lists, and language parameters

Table 1 shows the supported languages and the names of the files that are provided as dictionaries, stop-word lists, and lists of abbreviations. The dictionary files are in binary format and cannot be changed. The stop-word files and abbreviation files (if they exist) are in flat-file format and can be changed. If you change any of these files, ensure that you use the code page for the language.

This table also shows which language parameter you must specify when you enable a text column or external documents. This tells DB2 Text Extender in which language the documents are written so that the correct dictionary can be used for the linguistic processing that occurs.

Table 1. Linguistic functions used for the various languages

Language	File name	LANGUAGE parameter	Code page
Arabic	arabic	ARABIC	420
Brazilian Portuguese	brazil	BRAZILIAN	500
Canadian French	canadien	CAN_FRENCH	500
Catalan	catala	CATALAN	500
Danish	dansk	DANISH	500
Dutch	nederlnd	DUTCH	500
Finnish	suomi	FINNISH	500
French	francais	FRENCH	500
German	deutsch	GERMAN	500
Hebrew	hebrew	HEBREW	424
Icelandic	islensk	ICELANDIC	500
Italian	italiano	ITALIAN	500
Norwegian Bokmal	norbook	BM_NORWEGIAN	500
Norwegian Nynorsk	norntn	NN_NORWEGIAN	500
Portuguese	portugal	PORTUGUESE	500
Russian	russian	RUSSIAN	1025
Spanish	espana	SPANISH	500
Swedish	svensk	SWEDISH	500
Swiss German	dschweiz	SWISS_GERMAN	500
UK English	uk	UK_ENGLISH	500
US English	us	US_ENGLISH	500

(1) The filename of the stop-word file (extension STW) and abbreviation file (extension ABR).

(2) Linguistic processing uses both old and new German spelling.

Dictionaries, stop-word lists, and abbreviation lists

The files are distinguished by their extension.

Table 2. File content and file extension

Content	Extension
Dictionary	DIC
Stop-word list	STW
Abbreviation list	ABR

DB2 Text Extender does not provide linguistic support for DBCS languages. The appropriate LANGUAGE parameters for those languages are listed in the following table:

Table 3. The LANGUAGE parameter values for DBCS languages

Language	LANGUAGE Parameters
Simplified Chinese	S_CHINESE
Traditional Chinese	T_CHINESE
Japanese	JAPANESE
Korean	KOREAN

Modifying the stop-word and abbreviation files

There is one stop-word file and one abbreviation file per language. To understand the implications of editing these files, see “Why text documents need to be indexed” on page 21.

Tip

Before you begin editing one of these files, make a backup copy.

The stop word and abbreviation files are in:

`DB2TX_INSTOWNER/db2tx/dicts`

Remove words and abbreviations that you want to be indexed. Add words that you do not want to be indexed.

Chapter 5. Making text searchable

Chapter 3, “Getting started,” on page 19 helps you become familiar with making text searchable by DB2 Text Extender by walking you through a simple example. This chapter describes making text searchable in more detail, and describes all the aspects that you should consider before you begin.

The steps for making text searchable are:

1. Prepare thoroughly
2. Start the DB2 Text Extender command line processor
3. Connect to a system
4. Enable a system for text search
5. Enable a text table for text search (not required if you are creating one index per text column)
6. Enable a text column for text search

Preparation before making text searchable

Tip

Read this section carefully. It lists the options that you need to know about *before* making your text searchable.

- **Create one index for the whole table?**

You must decide whether to create one index for a whole text table, or a separate index for each text column. “Creating one or several text indexes for a table” on page 32 will help you decide.

- **Know your documents**

When you make documents searchable you must specify their CCSIDs, languages, and the formats of the text. For more information, see Chapter 4, “Planning for your search needs,” on page 21.

- **Decide the type of text indexes you need**

The type of index that you need is determined by the kind of searches you want to make (precise, fuzzy, and so on) and on whether your documents are SBCS or DBCS. You’ll find more information in Chapter 4, “Planning for your search needs,” on page 21.

- **Decide where to store indexes**

When you make documents searchable, DB2 Text Extender creates a text index. You must specify in which directory you want the index to be stored. Be sure that there will be enough disk space (see “Calculating the size of an index” on page 33).

- **Check the default subsystem name**

The default subsystem name in the environment variable DB2DBDFT (see “Environment variables” on page 13) is the name that DB2 Text Extender uses if you do not specify a subsystem name when making text searchable.

- **Set up the text configuration**

The text configuration determines the default settings for the index CCSID, the documents’ language, and the documents’ format, the index type, the tablespace name, and the index directory.

Preparation before making text searchable

You can override these settings when you make text searchable, but it is more convenient to have the defaults set correctly beforehand. The initial text configuration settings when DB2 Text Extender is installed are described in “Text configuration settings” on page 13. To change the installation settings and set up your own default values, use “CHANGE TEXT CONFIGURATION” on page 77.

- **Set up section support**

If you need to restrict searches to a particular section of a document, read “Working with structured documents (section support)” on page 34 to learn how to specify models in the document models file.

- **Modify the stop word and abbreviation lists**

Read “Why text documents need to be indexed” on page 21 and “Modifying the stop-word and abbreviation files” on page 37 to understand the concept of *stop word* lists and abbreviation lists, and decide whether to modify them before you begin indexing.

Once you have collected the information and made the decisions described in “Preparation before making text searchable” on page 39, you are ready to make your text searchable.

Starting the DB2 Text Extender command line processor

Summary

When	Optional. At the beginning of each session.
Command	db2tx
Authorization	Any

You can enter DB2 Text Extender commands at the UNIX system service prompt of the z/OS. The commands are preceded with db2tx.

An alternative to prefixing every DB2 Text Extender command with db2tx is to start the DB2 Text Extender command line processor. This has the advantage that your database connection is not broken after each command; you remain connected. (Without the command line processor you are automatically reconnected each time you execute a db2tx command, but you must ensure that the name of the database you want to connect to is set in the DB2DBDFT environment variable.)

1. Enter the following command in the UNIX Shell environment of z/OS to start the DB2 Text Extender command line processor:

```
db2tx
```

The db2tx prompt is displayed:

```
db2tx=>
```

and all subsequent commands are interpreted as DB2 Text Extender commands.

To leave this mode, enter:

```
db2tx=>quit
```

If you leave out this step, you can issue DB2 Text Extender commands directly from the operating system prompt by prefixing them with db2tx. Here is an example of a command issued from the operating system prompt:

Starting the DB2 Text Extender command line processor

```
=>db2tx enable server for db2text
```

Command line processor help

To display a list of commands, enter:

```
db2tx ?
```

To display the syntax of an individual command, enter:

```
db2tx ? command
```

For example:

```
db2tx ? CHANGE TEXT CONFIGURATION
```

Connecting to a subsystem

Summary

When	Automatic.
Command	None
Authorization	None

Before you can issue further commands in a DB2 Text Extender session, you must be connected to a subsystem. DB2 Text Extender automatically connects you to the default subsystem specified in the DB2DBDFT environment variable. If you want to connect to a different subsystem, you can do so explicitly by using the DB2 Text Extender CONNECT TO command from a DB2 Text Extender workstation client rather than from the z/OS UNIX Shell command line.

Enabling a server

Summary

When	Once for each server that contains columns of text to be searched in.
Command	db2tx=>ENABLE SERVER FOR DB2TEXT
Authorization	SYSADM, DBADM, CREATIN DB2TX

This command takes no other parameters. It prepares the connected server for use by DB2 Text Extender.

A catalog view, TEXTINDEXES, is created that keeps track of enabled text columns. See “Working with the DB2 Text Extender catalog view” on page 70.

This command creates text configuration information for the database, containing default values for index, text, and processing characteristics. They are described in “Text configuration settings” on page 13.

shows how to enable a database.

Enabling a server

Figure 7 shows how to enable a database.

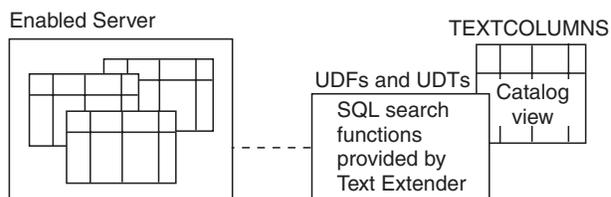


Figure 7. Enabling a server

Once a server has been enabled, it remains so until you disable it.

Tips

If the environment variable `DB2TX_INSTOWNER` is used, it must be set to the name of the instance owner before the server is enabled. This is particularly important for UNIX users because, in UNIX, this variable is set by default.

If you later decide to drop an enabled server, you should first disable it to ensure that the declared DB2 Text Extender functions, the catalog view, and so on, are removed.

Enabling a text column

Summary

When Once for each column that contains text to be searched.
Command `db2tx=>ENABLE TEXT COLUMN ...` (see the examples)
Authorization ALTER, SELECT, UPDATE on the table

Tip

If a setting, such as the index type, should be the same for most text columns, use the text configuration information to specify default settings. See “Text configuration settings” on page 13.

To reverse the changes made by `ENABLE TEXT COLUMN`, use the `DISABLE TEXT COLUMN` command. To disable all enabled text columns in a table, use the `DISABLE TEXT TABLE` command.

When you enable a text column, a handle column is added to the table, the document information (format, language, CCSID) is set, a log table is created, and an index is created.

A handle column is added

During this step, DB2 Text Extender adds to the table a 60-byte VARCHAR handle column – a column that contains handles associated with the text column that is being enabled. Handles contain information about the text in the associated text

column and in the associated external files. This information includes a unique document ID, the document's language, format, and CCSID, and the index name.

Figure 8 shows the structure of the DB2TX.MYTABLE table before enabling.

DB2TX.SAMPLE

DOCID	AUTHOR	SUBJECT	DATE	COMMENT
Data	Data	Data	Data	Text

Figure 8. Structure of the DB2TX.MYTABLE table—before enabling

The column containing text blocks is COMMENT. Before you can search through the text in this column, you must prepare the database and the COMMENT column for use by DB2 Text Extender.

After this preparation step, the DB2TX.MYTABLE table contains an additional column for handles.

Figure 9 shows the structure of the DB2TX.MYTABLE table after enabling.

DB2TX.SAMPLE

DOCID	AUTHOR	SUBJECT	DATE	COMMENT	COMMENTHANDLE
Data	Data	Data	Data	Text	Text handles

Figure 9. Structure of the DB2TX.MYTABLE table—after enabling

Note: When you subsequently search for text, you specify the handle column, not the text column, as the column to be searched.

The document information is set

You specify the type of text documents you typically store in this text column: their format (such as ASCII), their language, and their CCSID. Defaults for this information can be specified in the text configuration settings. See “Text configuration settings” on page 13.

A log table is created

During this step, a log table and a view called LOGIXnnnnnn is created, where *IXnnnnnn* is the index name (available from the catalog view).

If a default tablespace is specified in text configuration, the log table is stored there; otherwise, it is stored in the DB2 system default tablespace. To optimize performance and the use of disk space, you can specify a different tablespace to be used for the log tables.

Triggers are also created that add information to the log table whenever a document in the column is added or changed. This information causes these documents to be indexed the next time indexing takes place.

If errors occur during indexing, such as when a document queued for indexing could not be found, so-called *error events* are added to the log table and can be displayed, as described in “Displaying error events” on page 68.

Enabling a text column

Tip

If you run out of log space in this step, see “Enabling a text column in a large table” on page 45 for possible solutions.

An index is created

If you intend to have a separate index for each text column, that is, you have skipped the step `ENABLE TEXT TABLE`, DB2 Text Extender creates a separate index for the text column during this step. You specify the type of index, how frequently the index is to be updated, and in which directory the index is to be stored. If, on the other hand, you intend to have one index for the whole table, then you have already run `ENABLE TEXT TABLE` and specified the index parameters; they are ignored if you repeat them here.

Use the `UPDATEINDEX` keyword to determine whether the indexing of the text documents in the specified text column begins immediately, or when periodic indexing is next scheduled. If you do not use this keyword, the value specified in the text configuration settings is taken.

Creating indexes of various types for a text column. You can create more than one index for a text column. This can be useful if you want to allow, for example, linguistic and fuzzy search on the same text column, by associating the column with different index types, such as linguistic and Ngram indexes. You do this by running `ENABLE TEXT COLUMN` again, specifying not only the additional type of index to be created, but also a unique handle column name.

Specifying a CCSID. If the subsystem CCSID is different from the CCSID you want, you must specify the CCSID parameter when enabling a text column. You should also change the default CCSID in the text configuration, using this command:

```
db2tx change text cfg
```

Examples

The following example enables text column `COMMENT` in table `DB2TX.MYTABLE`, and assigns the name `COMMENTHANDLE` to the handle column that is created:

```
db2tx ENABLE TEXT COLUMN    db2tx.mytable    comment
                             HANDLE          commenthandle
```

Default values for the text information and for the index characteristics are taken from the text configuration settings.

The next example explicitly sets the values for the type of documents that are in the `COMMENT` column. Default values for the index characteristics are taken from the text configuration settings.

```
db2tx ENABLE TEXT COLUMN    db2tx.mytable    comment
                             HANDLE          commenthandle
                             CCSID          819
                             LANGUAGE       uk_english
                             FORMAT         rft
```

The next example explicitly sets the values for the characteristics of the index that is created for the COMMENT column. The example sets the index type and the index directory. Default values for the text information are taken from the text configuration settings.

```
db2tx ENABLE TEXT COLUMN      db2tx.mytable  comment
      HANDLE      commenthandle
      INDEXTYPE   linguistic
      UPDATEINDEX UPDATE
      DIRECTORY   DB2TX_INSTOWNER/db2tx/indexes
```

Enabling a text column in a large table

If you are working with a table that has a large row length, keep in mind that enabling a text column adds a handle column of type DB2TEXTH (VARCHAR 60). This could be significant if the table is approaching its maximum row length as determined by DB2.

Also when you enable a text column in large table, use the DB2 UDB for OS/390 and z/OS REORGANIZE utility to check whether the table needs to be reorganized. When you enable a large table for the first time, the following steps make indexing faster:

1. Enable the table using the NOUPDATE option. This creates the handles, but does not yet index the documents.
2. Reorganize the table using the DB2 UDB for OS/390 and z/OS REORGANIZE utility.
3. Create the index by running UPDATE INDEX.

When you enable a text column, DB2 Text Extender adds a handle column to the table and initializes the handle values, thereby causing DB2 UDB for OS/390 and z/OS log entries to be written. If there is an unusually large number of log entries to be written, DB2 UDB for OS/390 and z/OS can run out of log space.

To enable large tables, you should use a storage group option STOGROUP that is provided for the ENABLE TEXT COLUMN command. This option lets you specify a storage group for the index that is created internally on the handle column.

Syntax:

```
db2tx enable text column [TABLESPACE [dbname.]tablespace-name]
      [STOGROUP storage-group]
```

where dbname is the name of the database.

Both keywords are optional and the database name is optional. If you do not specify a database name, the tablespace must have been created in the default database. The tablespace and the storage group must have been created previously.

Enabling text columns of a nonsupported data type

Text columns must be CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB to be enabled by DB2 Text Extender. If the documents are in a column of a different type, such as a user-defined distinct type (UDT), you must provide a function that takes the user type as input and provides as output type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB.

Enabling a text column

Use the FUNCTION keyword in ENABLE TEXT COLUMN to specify the name of this function.

Example: You intend to store compressed text in a table.

- Create a user-defined distinct type (UDT) for the text in an SQL session:
db2 CREATE DISTINCT TYPE COMPRESSED_TEXT AS CLOB(1M)
- Create a table and insert the text into it.

Note

Creating a table is outside the parameters of this guide. See the *z/OS DB2: SQL Reference* guide for further information

To enable the text column for use by DB2 Text Extender:

1. Create a user-defined function (UDF) called, for example, UNCOMPRESS, that receives a value of type COMPRESSED_TEXT and returns the corresponding uncompressed text as, for example, a CLOB(10M) value.
2. Enable the text column using the FUNCTION keyword to identify the UNCOMPRESS UDF:

```
db2tx ENABLE TEXT COLUMN MYTABLE text
        FUNCTION uncompress
        HANDLE handle
        ...
```

Ending the session

You have now completed the steps to prepare your text documents to be searched.

If you specified NOUPDATE for the UPDATEINDEX keyword when you enabled the text column, DB2 Text Extender does not index the text immediately, but waits for the next periodic indexing. To update the index now, see “Updating an index” on page 63.

When indexing of the documents has finished, you can begin retrieving information as described in Chapter 6, “How to search,” on page 47.

Tip

Use GET INDEX STATUS to determine when indexing has finished.

To end the DB2 Text Extender command processor, enter:

```
db2tx=>quit
```

Chapter 6. How to search

DB2 Text Extender provides SQL functions that enable you to include text search subqueries in SQL queries. These functions are provided in addition to those normally available in SQL, and are referred to here as DB2 Text Extender functions.

Refer to Chapter 10, “Search functions,” on page 111 for a description of the syntax of these functions.

Before searching, read “Types of search” on page 29, and also use GET INDEX SETTINGS to find out which index type is associated with the text you are searching in. A search can produce different results according to the index type. The index type assumed in the examples in this chapter is linguistic.

This chapter describes:

- Setting the function path to give SQL access to the DB2 Text Extender functions
- Searching for text, using CONTAINS, NO_OF_MATCHES, and RANK
- Specifying search arguments in DB2 Text Extender functions, using examples of CONTAINS
- Refining a previous search, using CONTAINS and REFINE
- Setting and extracting information in handles, using INIT_TEXT_HANDLE, CCSID, FORMAT, and LANGUAGE

Setting the current function path

►► SET CURRENT FUNCTION PATH [=] DB2TX, ... ◄◄

Use the SQL statement SET CURRENT FUNCTION PATH to add DB2TX to your current path names so that SQL can find the DB2 Text Extender functions. If you decide not to do this, you can qualify the function names explicitly by typing, for example, DB2TX.CONTAINS for the CONTAINS function.

The examples in this chapter use the qualified form for DB2 Text Extender functions. You can use the example statements exactly as they are written without having to set the current function path.

Tip

Remember to set the current function path each time you connect to a database. If you have installed Net Search Extender V8.1, you must always explicitly qualify the Text Extender functions with DB2TX.

Searching for text

►► CONTAINS [NO_OF_MATCHES | RANK] (—handle—, —search-argument—) ◄◄

Searching for text

This section describes how to use the SQL functions provided with DB2 Text Extender to search in DB2 databases containing text. It tells you how to:

- Make a query
- Determine how many matches were found in a text document
- Get the rank of a found text document.

Each of these DB2 Text Extender functions searches in the text index for occurrences of the search argument. If there are, say, 100 000 text documents in the table, the CONTAINS, RANK, or NO_OF_MATCHES function is called 100 000 times. But the text index is not searched 100 000 times. Instead, the first time the function is called, an internal list of all the documents containing the search term is created; subsequent calls of the function determine if the document concerned is in the list.

Tip

When you use the DB2 Text Extender functions to search in a table, be sure to pass the handle column to the function, rather than the text column. If you try to search in a text column, SQL responds with a message indicating that the data type is wrong, for example:

```
No function by the name "CONTAINS" having compatible arguments was found in the function path.
```

If you search for text immediately after issuing the ENABLE TEXT TABLE or ENABLE TEXT COLUMN command, an error RC_SE_EMPTY_INDEX can occur which indicates that the index being created by the command does not yet exist. The time taken for an index to be created depends on factors such as the number of documents being indexed, and the performance of the system doing the indexing. It can vary from several minutes to several hours, and should be done when the system is lightly loaded, such as over night.

If this message occurs, try searching again later, or use GET INDEX STATUS to check whether indexing errors have occurred.

Making a query

This example demonstrates how the CONTAINS function searches for text in documents identified by a handle. It returns 1 if the text satisfies the search argument, otherwise it returns 0.

```
db2=>SELECT DATE, SUBJECT
      FROM DB2TX.SAMPLE
      WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"compress"') = 1
```

In this example, you search for the term `compress` in the text referred to by the handles in the column `COMMENTHANDLE`. The handles in the `COMMENTHANDLE` column indicate where the `COMMENT` text is indexed.

Tip

If you have created mixed-case identifiers for tables or columns, that these names must be enclosed in double quotes. For example:

```
db2=>SELECT DATE, SUBJECT
      FROM "db2tx.sample"
      WHERE DB2TX.CONTAINS (COMMENTHANDLE, "compress") = 1
```

If you specify DB2 UDB for OS/390 and z/OS, select statements from an AIX or NT client, the operating system command-line parser removes special characters such as double quotes from the command string, so you must use a backslash to mask these special symbols. For example:

```
WHERE DB2TX.CONTAINS (COMMENTHANDLE, "\"compress\"") = 1
```

Searching and returning the number of matches found

Use the NO_OF_MATCHES function to determine how often the search criteria are found in each text document.

```
CREATE VIEW TEMPTABLE (DATE, SUBJECT, MATCHES)
AS (SELECT DATE, SUBJECT,
          DB2TX.NO_OF_MATCHES (COMMENTHANDLE, "compress")
     FROM DB2TX.SAMPLE);
SELECT * FROM TEMPTABLE
       WHERE MATCHES > 0
```

NO_OF_MATCHES returns an integer value.

Searching and returning the rank of a found text document

RANK is an absolute value that indicates how well the document met the search criteria relative to other found documents. The value indicates the number of matches found in the document in relation to the document's size. You can get the rank of a found document by using the RANK function.

Here is an example:

```
CREATE VIEW TEMPTABLE (DATE, SUBJECT, RANK)
AS (SELECT DATE, SUBJECT,
          DB2TX.RANK (COMMENTHANDLE, "compress")
     FROM DB2TX.SAMPLE);
SELECT *
      FROM TEMPTABLE
      WHERE RANK > 0
      ORDER BY RANK DESC
```

RANK returns a DOUBLE value between 0 and 1.

Specifying search arguments

Search arguments are used in CONTAINS, NO_OF_MATCHES, and RANK. This section uses the CONTAINS function to show different examples of search arguments in DB2 Text Extender functions.

Searching for several terms

You can have more than one term in a search argument. One way to combine several search terms is to connect them together using commas, like this:

Specifying search arguments

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
  ('compress", "compiler", "pack", "zip", "compact")) = 1
```

This form of search argument finds text that contains any of the search terms. In logical terms, the search terms are connected by an OR operator.

Searching with the Boolean operators AND and OR

(See also “Searching with the Boolean operator NOT” on page 54.)

Search terms can be combined with other search terms using the Boolean operators “&” (AND) and “!” (OR). For example:

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
  "compress" ! "compiler") = 1
```

You can combine several terms using Boolean operators:

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
  "compress" ! "compiler" & "DB2") = 1
```

If you use more than one Boolean operator, DB2 Text Extender evaluates them from left to right, but the logical AND operator (&) binds stronger than the logical OR operator (!). For example, if you do not include parentheses,

```
"DB2" & "compiler" ! "support" & "compress"
```

is evaluated as:

```
("DB2" & "compiler") ! ("support" & "compress")
```

So in the following example you must include the parentheses:

```
"DB2" & ("compiler" ! "support") & "compress"
```

If you combine Boolean operators with search terms chained together using the comma separator, like this:

```
("compress", "compiler") & "DB2"
```

the comma is interpreted as a Boolean OR operator, like this:

```
("compress" ! "compiler") & "DB2"
```

Searching for variations of a term

If you are using a **precise** index, DB2 Text Extender searches for the terms exactly as you type them. For example, the term `media` finds only text that contains “media”. Text that contains the singular “medium” is not found.

If you are using a **linguistic** index, DB2 Text Extender searches also for variations of the terms, such as the plural of a noun, or a different tense of a verb.

For example, the term `drive` finds text that contains “drive”, “drives”, “driving”, “drove”, and “driven”.

Searching for parts of a term (character masking)

Masking characters, otherwise known as “wildcard” characters, offer a way to make a search more flexible. They represent optional characters at the front, middle, or end of a search term. They increase the number of text documents found by a search.

Tip

If you use masking characters, you cannot use the SYNONYM FORM OF keyword.

Masking characters are particularly useful for finding variations of terms if you have a precise index. If you have a linguistic index, many of the variations found by using masking characters would be found anyway.

Note that word fragments (words masked by wildcard characters) cannot be reduced to a base form. So, if you search for `pass%`, you will not find the words “passes” or “passed”, because they are reduced to their base form “pass” in the index. To find them, you must search for `pass%`.

DB2 Text Extender uses two masking characters: percent (%) and underscore (_):

- % represents **any number of arbitrary characters**. Here is an example of % used as a masking character at the front of a search term:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE, '%"name'') = 1
```

This search term finds text documents containing, for example, “username”, “file_name”, and “table-name”.

% can also represent a **whole word**: The following example finds text documents containing phrases such as “graphic function” and “query function”.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE, '%" function'') = 1
```

- _ represents **one character** in a search term: The following example finds text documents containing “CLOB” and “BLOB”.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE, '%"_LOB'') = 1
```

Searching for terms that already contain a masking character

If you want to search for a term that contains the “%” character or the “_” character, you must precede the character by a so-called *escape* character, and then identify the escape character using the ESCAPE keyword.

For example, to search for “100% interest”:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             '"100!% interest" ESCAPE "!'') = 1
```

The escape character in this example is “!”.

Specifying search arguments

Searching for terms in any sequence

If you search for “hard disk” as shown in the following example, you find the two terms only if they are adjacent and occur in the sequence shown, regardless of the index type you are using.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE, '"hard disk"') = 1
```

To search for terms in any sequence, as in “data disks and hard drives”, for example, use a comma to separate the terms:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE, '("hard", "disk"') = 1
```

Searching for terms in the same sentence or paragraph

Here is an example of a search argument that finds text documents in which the search terms occur in the same sentence:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             '"compress" IN SAME SENTENCE AS "decompress"') = 1
```

You can also search for more than two words occurring together. In the next example, a search is made for several words occurring in the same paragraph:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             '"compress" IN SAME PARAGRAPH AS "decompress"
                             AND "encryption"') = 1
```

Searching for terms in sections of structured documents

Here is an example of a search argument that finds text documents in which the search term Williams occurs in the subsection author in section play of structured documents. The document structure is specified by model play which is described in a document models file. See “Working with structured documents (section support)” on page 34 for more information.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             'MODEL play SECTIONS (play/author) "williams"') = 1
```

Note

The availability of document models depends on the OS/390 and z/OS release:

- For OS/390 version 2.8 or less (including Text Search Engine release 3 (FMID HIMN210 or 220)), only a single document model can be associated with documents for indexing.
- For OS/390 version 2.9 or higher, and z/OS releases (which include Text Search Engine release 4 (FMID HIMN230)), multiple document models can be associated

Searching for synonyms of terms

For a linguistic index, you can make your searches more flexible by looking not only for the search terms you specify, but also for words having a similar meaning. For example, when you search for the word “book”, it can be useful to search also for its synonyms. To do this, specify:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE, 'SYNONYM FORM OF "book"') = 1
```

When you use SYNONYM FORM OF, it is assumed that the synonyms of the term are connected by a logical OR operator, that is, the search argument is interpreted as:

```
"book" ! "article" ! "volume" ! "manual"
```

The synonyms are in a dictionary that is provided with DB2 Text Extender. The default dictionary used for synonyms is always US_ENGLISH, not the language specified in the text configuration settings.

You can change the dictionary for a particular query by specifying a different language. Here is an example:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             'SYNONYM FORM OF UK_ENGLISH "programme"') = 1
```

Tip

You cannot use the SYNONYM keyword if there are masking characters in a search term, or if NOT is used with the search argument.

Making a linguistic search

DB2 Text Extender offers powerful linguistic processing for making a search based on the search terms that you provide. The linguistic functions are applied when the index is linguistic. The linguistic functions are described in Chapter 12, “Linguistic processing for linguistic and precise indexes,” on page 133.

An example of this is searching for a plural form, such as “utilities”, and finding “utility”. The plural is reduced to its base form *utility*, using an English dictionary, before the search begins.

The English dictionary, however, does not have the information for reducing variations of terms in other languages to their base form. To search for the plural of a term in a different language you must use the dictionary for that language.

If you specify GERMAN, for example, you can search for “geflogen” (flown) and find all variations of its base form “fliegen” (fly)—not only “geflogen”, but also “fliege”, “fliegt”, and so on.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             'STEMMED FORM OF GERMAN "geflogen"') = 1
```

Specifying search arguments

Tip

When searching in documents that are not in U.S. English, specify the language in the search argument *regardless of the default language*.

If you always specify the base form of a search term, rather than a variation of it, you do not need to specify a language.

To understand why, consider what happens when the text in your database is indexed. If you are using a linguistic index, all variations of a term are reduced to their base form before the terms are stored in the index. This means that, although the term “decompress” occurs in the first entry in the COMMENT column, “decompression” may occur in the second entry, the index contains only the base form “decompress” and identifies this term (or its variations) as being in both entries.

Later, if you search for the base form “decompress”, you find all the variations. If, however, you search for a variation like “decompression”, you cannot find it directly. You must specify an appropriate dictionary for the search, so that the variation can first be converted to its base form.

Searching with the Boolean operator NOT

You can use the Boolean operator NOT to exclude particular text documents from the search. For example:

```
("compress", "compiler") & NOT "DB2"
```

Any text documents containing the term “DB2” are excluded from the search for “compress” or “compiler”.

You cannot use the NOT operator in combination with IN SAME SENTENCE AS or IN SAME PARAGRAPH AS described in “Searching for terms in the same sentence or paragraph” on page 52, neither can you use it with SYNONYM FORM OF described in “Searching for synonyms of terms” on page 53.

You can use the NOT operator only with a search-primary, that is, you cannot freely combine the &, !, and NOT operators (see “Search argument syntax” on page 122).

Example of the use of NOT that is **not** allowed:

```
NOT("compress" & "compiler")
```

Allowed is:

```
NOT("compress" , "compiler")
```

Fuzzy search

“Fuzzy” search searches for words that are spelled in a similar way to the search term. It is available for Ngram indexes.

For example:

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
'FUZZY FORM OF 2 "compress"') = 1
```

This search could find an occurrence of the misspelled word `conpress`.

The match level, in the example “2”, specifies the degree of accuracy. Five levels are supported, where level 1 gives the loosest matching of about 20 percent, and level 5 gives the tightest matching of about 90 percent. Use a fuzzy search when the misspellings are possible in the document, as is often the case when the document was created using an Optical Character Recognition device, or phonetic input.

Respecting word-phrase boundaries

“Bound” search has been developed for the Korean language. It ensures that DB2 Text Extender respects word boundaries during the search. For example:

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             'BOUND "korean-expression"') = 1
```

Searching for similar-sounding words

“Sound” search finds words that sound like the search argument. This is useful when documents can contain words that sound alike, but are spelled differently. The German name that is pronounced `my-er`, for example, has several spellings.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             'SOUNDS LIKE "Meyer"') = 1
```

This search could find occurrences of “Meyer”, “Mayer”, and “Maier”.

Thesaurus search

Thesaurus search is another of DB2 Text Extender’s powerful search-term expansion functions. The additional terms searched for are taken from a thesaurus that you build yourself, so you have direct control over them. You search for “database”, for example, and could find terms like “repository” and “DB2”.

This type of search is intended for specific areas of interest in which you make frequent searches; an area in which it is worth the investment in time to build a thesaurus in order to produce significantly more effective search results.

See “Thesaurus concepts” on page 138 for more information and a description of how to build a thesaurus. The example in Figure 13 on page 138 is a small extract from a thesaurus on the subject of databases. It is used in the following examples that demonstrate the syntax for using thesaurus expansion.

This example takes the term “object relational database management system” and expands it, adding all *instances* of this term found in the thesaurus “myterms”. Here, “DB2” is added to the search.

```
SELECT DATE, SUBJECT
       FROM DB2TX.SAMPLE
       WHERE DB2TX.CONTAINS (COMMENTHANDLE,
                             'THESAURUS "myterms"
                             EXPAND "INST"
                             TERM OF "object relational database
                             management system"') = 1
```

The next example takes the term “document management system” and expands it, adding all its *synonyms*. There is one synonym – “library”.

Specifying search arguments

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
    'THESAURUS "myterms"
    EXPAND "SYN"
    TERM OF "document management system"') = 1
```

Free-text and hybrid search

“Free-text search” is a search in which the search term is expressed as free-form text. A phrase or a sentence describes in natural language the subject to be searched for. The sequence of words in a free-text query are not relevant. Furthermore, so-called *lexical affinities*. In retrieval, these are certain pairs of words occurring in a free-text query term, and occurring in the document collection, with a certain minimal frequency and a certain minimal distance. The distance for English documents is five words.

Note that the masking of characters or words is not supported for search strings in a free-text argument.

For example:

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
    'IS ABOUT "everything related to AIX installation"') = 1
```

Hybrid search is a combination of Boolean search and free-text search. For example:

```
SELECT DATE, SUBJECT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
    '"DB2" & IS ABOUT "everything related to
    AIX installation"') = 1
```

Refining a previous search

When a search argument finds too many occurrences, it can often be useful to narrow, or *refine*, the search by combining the initial search argument with a second search argument in a Boolean-AND relationship.

You can refine search results without using the REFINE function, by storing the results in a table and making the next search against this table. However, depending on the number of qualifying terms, this method is less efficient than that of storing the latest search argument and using REFINE.

The following steps show how to make a search, and then refine it using the REFINE function. The REFINE function returns a search argument that is a Boolean-AND combination of its two input parameters. The combined search argument returned by REFINE is a value of type LONG VARCHAR.

1. Create a table for old search arguments.

Create a table PREVIOUS_SEARCHES to hold the search arguments of searches that have already been made.

```
CREATE TABLE PREVIOUS_SEARCHES (STEP INTEGER,
    SEARCHARGUMENT VARCHAR(500)
    SEARCHARGUMENT LONG VARCHAR)
    IN DB2TX.DB2TXTSS CCSID EBCDIC;
```

Figure 10 on page 57 shows a table with previous search arguments.

PREVIOUS_SEARCHES

STEP	SEARCHARGUMENT

Figure 10. The previous search arguments

2. Search for the first search argument.

Search for the word “compress” in the sample table.

```
SELECT COMMENT
FROM DB2TX.SAMPLE
WHERE DB2TX.CONTAINS (COMMENTHANDLE, "compress") = 1
```

Insert the search argument into the PREVIOUS_SEARCHES table for use by further steps.

```
INSERT INTO PREVIOUS_SEARCHES
VALUES (1, "compress")
```

Figure 11 shows a table with the first search argument.

PREVIOUS_SEARCHES

STEP	SEARCHARGUMENT
1	"compress"

Figure 11. The first search argument

3. Refine the search.

Assuming that the search returns too many text documents, refine the search by combining the previous search term with the word “compiler” using the REFINE function.

```
CREATE VIEW LAST_STEP(STEP_MAX)
AS (SELECT MAX(STEP)
FROM PREVIOUS_SEARCHES);
CREATE VIEW LAST_SEARCH(LAST_SEARCH)
AS (SELECT SEARCHARGUMENT
FROM PREVIOUS_SEARCHES, LAST_STEP
WHERE STEP = STEP_MAX);
SELECT COMMENT
FROM DB2TX.SAMPLE, LAST_SEARCH
WHERE DB2TX.CONTAINS (COMMENTHANDLE,
DB2TX.REFINE(LAST_SEARCH, "compiler")) = 1;
```

Insert the refined search argument into the PREVIOUS_SEARCHES table for use by further steps.

```
INSERT INTO PREVIOUS_SEARCHES
SELECT STEP_MAX+1,
CAST(DB2TX.REFINE(LAST_SEARCH, "compiler") AS VARCHAR(500))
FROM LAST_SEARCH, LAST_STEP
```

Figure 12 shows a table with the refined search.

PREVIOUS_SEARCHES

STEP	SEARCHARGUMENT
1	"compress"
2	"compress" & "compiler"

Figure 12. The refined search argument

Refining a previous search

You can repeat this step until the number of text documents found is small enough.

Setting and extracting information in handles

Handles contain the CCSID, format, and language of their text documents. Handles for external files contain additionally a pointer to the external file. These handles are created when you enable a text column or external files.

The DB2 Text Extender functions described here let you set and extract the text information in the handles.

Setting text information when inserting new text

```

▶▶—INIT_TEXT_HANDLE—┌(—format—,—language—)—————▶
                    └(—CCSID—,—format—,—language—,—filename—)┘

```

When you run the ENABLE TEXT COLUMN command to enable a text column that already contains text, you can implicitly set the format and language of the text to the values specified in the text configuration settings. These format and language settings are then stored in the handle. If you want different format and language values, you can specify them explicitly in the ENABLE TEXT COLUMN command.

When you run the ENABLE TEXT FILES command, you can also set the document's CCSID and location.

When you later insert a row containing text, an insert trigger creates a handle and sets the text format and language to the values that were used when the text column was enabled.

To set the format and language to values that are different from these values, use the INIT_TEXT_HANDLE function in the INSERT command. While the row is being inserted, the INIT_TEXT_HANDLE function creates a partially initialized handle that contains the language and format values you specify. The insert trigger then fills in the other values in the handle.

In the following example, INIT_TEXT_HANDLE presets the language and format in an initialized handle. The INSERT command places this handle in the COMMENTHANDLE column.

```

INSERT INTO DB2TX.SAMPLE (DOCID, COMMENT, COMMENTHANDLE)
VALUES ('doc 101',
       'I have installed...',
       DB2TX.INIT_TEXT_HANDLE('AMI', 'GERMAN') )

```

The value returned by INIT_TEXT_HANDLE is type DB2TEXTH.

Setting and extracting information in handles

Extracting information from handles



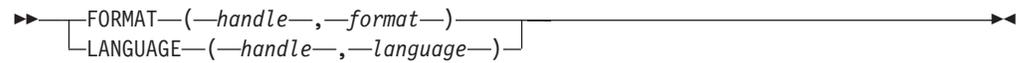
Here is an example of extracting a CCSID from a handle:

```
SELECT DISTINCT DB2TX.CCSID(COMMENTHANDLE)
FROM DB2TX.SAMPLE
```

In the same way, you can extract the format or the language of a text document. Here is an example that illustrates the use of the FORMAT function. It returns the number of ASCII (TDS) documents in the sample table.

```
SELECT COUNT(*)
FROM DB2TX.SAMPLE
WHERE DB2TX.FORMAT(COMMENTHANDLE) = 'TDS'
```

Changing information in handles



The `FORMAT` and `LANGUAGE` functions can also change the corresponding specification in a handle. These functions return the changed handle as a value of type `DB2TEXTH`.

The following example shows how to change the language setting of a text document.

```
UPDATE DB2TX.SAMPLE
  SET COMMENTHANDLE = DB2TX.LANGUAGE(COMMENTHANDLE, 'FRENCH')
  WHERE ...
```

Using the `LANGUAGE` function again, you can see that the change has occurred:

```
SELECT DISTINCT DB2TX.LANGUAGE(COMMENTHANDLE)
  FROM DB2TX.SAMPLE
```

Improving search performance

The SEARCH_RESULT function exploits the DB2 concept of table-valued functions. The function is used in the FROM clause of an SQL statement, and returns an intermediate table with the search result of the specified search string. The syntax of the search string is the same as described in Chapter 11, “Syntax of search arguments,” on page 121. The advantage of this function compared with CONTAINS or RANK is a significant performance improvement when large tables are involved.

The returned table has the following structure:

Column Name	Data type
HANDLE	DB2TX.DB2TEXTH
NUMBER_OF_MATCHES	INTEGER
RANK	DOUBLE

Example:

```
CREATE VIEW REPHANDLE (MYDOCHANDLE)
AS (SELECT DB2TX.DB2TEXTH(PROTOTYPEHANDLE)
FROM db2tx.textcolumns
WHERE TABLESCHEMA = 'DB2TX' AND
TABLENAME = 'SAMPLE' AND
COLUMNNAME = 'COMMENT' AND
HANDLENAME = 'COMMENTHANDLE'
);
SELECT COMMENT
FROM db2tx.sample T1, rephandle T2,
TABLE(DB2TX.SEARCH_RESULT(MYDOCHANDLE, '"COMPRESS"')) T3
where cast(T1.C_H as varchar(60)) =
cast(T3.handle as varchar(60));
```

SELECT NUMBER_OF_MATCHES, RANK, HANDLE causes all three items to be returned, but you can specify them in any combination. You may wish, for example, to omit RANK to avoid the intensive processing associated with it.

If you need only the HANDLE value, you can simply use SELECT COUNT(*).

Chapter 7. Administration

This chapter describes how to maintain text indexes, how to get useful information, and how to reverse the text preparation process.

Maintaining text indexes

These are the index maintenance tasks:

- Updating an index
- Resetting the status of an index
- Deleting index events
- Reorganizing an index

You can run these tasks at any time and in any sequence.

Updating an index

Summary

When When an index must be updated immediately without waiting for periodic indexing to occur. (See “Enabling a text column” on page 42 for information about periodic indexing.)

Command
UPDATE INDEX

Authorization
ALTER, SELECT, UPDATE on the table

This example updates the index for a common-index table:

```
db2tx UPDATE INDEX db2tx.mytable
```

This example updates the index for a column of a multi-index table:

```
db2tx UPDATE INDEX db2tx.mytable HANDLE commenthandle
```

Use this command to update the index immediately, without waiting for the next periodic indexing to take place automatically. This is useful when you have added several text documents to a database and want to search them immediately.

DB2 Text Extender indexes the text documents in this column (or all columns in the table) that have been inserted or changed, and removes from the index the terms from documents that have been deleted. The log table associated with the index contains information about which documents have been inserted, updated, and deleted.

Resetting the index status

Summary

When When an index can no longer be searched or updated.

Command

```
RESET INDEX STATUS
```

Authorization

None

Some situations can occur that prevent you from searching in an index, or from updating it. “Displaying the status of an index” on page 67 describes how to determine if one of these events has occurred. `RESET INDEX STATUS` reactivates the index so that you can use it again.

This example resets the index status for the index of a common-index table:

```
db2tx RESET INDEX STATUS db2tx.mytable
```

The syntax lets you reset the index status for a particular text column. This example resets the index status for the index of a multi-index table column:

```
db2tx RESET INDEX STATUS db2tx.mytable HANDLE commenthandle
```

Deleting index events

Summary

When When you no longer need the messages in an index’s log table.

Command

```
DELETE INDEX EVENTS
```

Authorization

None

If something prevents you from searching in an index, or from updating it, or if a document cannot be indexed, this is known as an indexing *event*. Information about indexing events is stored in the index’s log table. It can help you determine the cause of the problem. When you no longer need these messages, you can delete them.

This example deletes messages from the index of a common-index table:

```
db2tx DELETE INDEX EVENTS db2tx.mytable
```

The syntax also lets you delete indexing events for a particular text column. This example deletes the messages for the index of a multi-index table column:

```
db2tx DELETE INDEX EVENTS db2tx.mytable HANDLE commenthandle
```

Reorganizing an index

Summary

When When GET INDEX STATUS indicates that an index should be manually reorganized.

Command
REORGANIZE INDEX

Authorization
None

If a text column is often updated, searching the index can become inefficient. To make searching efficient again, the index has to be *reorganized*. Although DB2 Text Extender recognizes when an index needs to be reorganized and does so in the background automatically, there may be situations that require an index to be reorganized manually using REORGANIZE INDEX. You can use the command GET INDEX STATUS to find out if an index needs to be reorganized.

Although searches can be made on the index while REORGANIZE INDEX is running, index updates cannot.

This example reorganizes the index of a common-index table:

```
db2tx REORGANIZE INDEX db2tx.mytable
```

This example reorganizes the index of a multi-index table column:

```
db2tx REORGANIZE INDEX db2tx.mytable HANDLE commenthandle
```

Getting useful information

This section describes the commands for displaying information about:

- The enabled status of databases, tables, columns, and files
- The settings of the environment variables
- The text configuration settings
- The index status
- The error events
- The index settings
- The text settings for a column

Displaying enabled-status information

Summary

When When you need information about the enabled status of databases, tables, text columns or external files.

Command
GET STATUS

Authorization
None

Enter:

Getting useful information

```
db2tx GET STATUS
```

Here is an example of the output displayed by GET STATUS. It shows the enabled status of the database, and of any enabled tables, text columns, or text files that it contains.

```
Database is enabled for Text Extender
```

```
Table DB2TX.MYTABLE is enabled as a common-index table
```

```
Table DB2TX.MYTABLE is enabled as a common-index table
```

TextColumnName	HandleColumnName
-----	-----
COMMENT	COMMENTHANDLE

```
Table DB2TX.TEST is enabled as a multi-index table
```

TextColumnName	HandleColumnName
-----	-----
ABSTRACT1	ABSTRACT1HANDLE
ABSTRACT2	ABSTRACT2HANDLE

Displaying the settings of the environment variables

Summary

When When you need information about the settings of the environment variables.

Command

```
GET ENVIRONMENT
```

Authorization

```
None
```

Enter:

```
db2tx GET ENVIRONMENT
```

Here is an example of the output displayed by GET ENVIRONMENT. It shows the current settings of the DB2 Text Extender environment variables.

```
Instance name          (DB2TX_INSTOWNER) = user1
Instance directory    (DB2TX_INSTOWNERHOMEDIR) = /usr/instance1
```

Displaying the text configuration settings

Summary

When When you need the default settings for text, index, and process information.

Command

```
GET TEXT CONFIGURATION
```

Authorization

```
None
```

These settings are described in “Text configuration settings” on page 13. To change them, see “Changing the text configuration” on page 14.

To display the text configuration, enter:

```
db2tx GET TEXT CFG
```

Here is an example of the output displayed by GET TEXT CONFIGURATION. It shows the current text configuration settings.

```
Coded character set ID      (CCSID) = 500
Language                   (LANGUAGE) = US_ENGLISH
Format                     (FORMAT) = TDS

Index type                 (INDEXTYPE) = LINGUISTIC
Update frequency          (UPDATEFREQ) = NONE
Index directory            (DIRECTORY) = /user1/db2tx/indexes

Update index option       (UPDATEINDEX) = UPDATE
Commit count              (COMMITCOUNT) = 10 000
Tablespace name           (TABLESPACE) = TXLOG
```

Displaying the status of an index

Summary

When When you need to determine whether an index can be searched or updated.

Command

```
GET INDEX STATUS
```

Authorization

None

Some situations can occur that prevent you from searching in an index, or from updating it. In such situations, messages are stored in the index's log table that can help you determine the cause. So it can be useful to check the status of an index, and whether there are any messages available.

This example displays the index status for the index of a common-index table:

```
db2tx GET INDEX STATUS db2tx.mytable
```

The syntax lets you display the index status for a particular text column. This example gets the index status for the index of a multi-index table column:

```
db2tx GET INDEX STATUS db2tx.mytable HANDLE commenthandle
```

Here is an example of the output displayed by GET INDEX STATUS.

```
Node 0
Search status           = Search available
Update status          = Update available
Reorg status            = started 13.55
Scheduled documents     = 0
Indexed documents      = 187000
Primary index documents = 130000
Secondary index documents = 57000
Error events            = No error events
```

Search status

Indicates whether you can use the specified handle column to search in the index. If search is not available, check the indicated reason code for more information about why the situation occurred, and then use RESET INDEX STATUS to be able to work with the index again. See Chapter 17, "Error event reason codes," on page 171.

Getting useful information

Update status

Indicates whether you can update the index for the specified table or column. If the index update function is not available, check the indicated reason code for more information about why the situation occurred, and then use `RESET INDEX STATUS` to be able to work with the index again.

Reorg status

Indicates whether you can reorganize the index for the specified table or column. If the reorganize function is not available, check the indicated reason code for more information about why the situation occurred. A common reason for reorganization not being available is because the index is currently being updated.

Scheduled documents

Shows the number of documents that are listed in the queue for indexing (or for deleting from the index).

Indexed documents

Shows the number of documents that have already been indexed from the queue of scheduled documents.

Primary index documents

Shows the number of documents in the primary index.

Secondary index documents

Shows the number of documents in the secondary index.

Error events

Shows the number of events that are available in the index's log table. You can display this information as described in "Displaying error events." When you no longer need this information, you can delete it as described in "Deleting index events" on page 64.

Displaying error events

When problems occur during indexing, such as a document scheduled for indexing could not be found, these so-called *error events* are written to the index's log table.

The event return codes are described in Chapter 17, "Error event reason codes," on page 171.

You can access the error events in a view of the log table called `db2tx.LOGIXnnnnnn`, where *IXnnnnnn* is the name of the index, obtainable from the catalog view.

To get the name of the index:

```
DB2 SELECT TABLENAME,  
          HANDLENAME,  
          INDEXNAME  
FROM   DB2TX.TEXTCOLUMNS
```

The error event view has the following layout:

UPDATESTATUS	SMALLINT
EVENTREASON	INTEGER
EVENTMESSAGE	VARCHAR(1024)
UPDATETIME	TIMESTAMP
HANDLE	DB2TEXTH

Here is an example showing how to access the information from the index log:

```
DB2 SELECT EVENTREASON,
          EVENTMESSAGE,
          UPDATETIME,
          HANDLE
FROM DB2TX.LOGIXNNNNNN
```

Displaying the index settings

Summary

When When you need information about the settings of an index.

Command

```
GET INDEX SETTINGS
```

Authorization

None

This example gets the index settings for the index of a common-index table:

```
db2tx GET INDEX SETTINGS db2tx.mytable
```

This example gets the index settings for the index of a multi-index table column:

```
db2tx GET INDEX SETTINGS db2tx.mytable HANDLE commenthandle
```

If the table is enabled as a multi-index table, this command displays the index settings of all enabled text columns in the table.

Here is an example of the output displayed by GET INDEX SETTINGS for a common-index table. The output for a multi-index table shows similar information for each index. The syntax lets you request the index settings for a particular text column.

Current index settings:

```
Index type           (INDEXTYPE) = LINGUISTIC
Update index option  (UPDATEINDEX) = UPDATE
Update frequency     (UPDATEFREQ) = NONE
Node 0
Index directory      (DIRECTORY) = /home/user1/db2tx/indices
```

If the index is split among several nodes, the node information is displayed for the index directory.

Displaying the text settings for a column

Summary

When When you need information about the text settings for a column.

Command

```
GET TEXT INFO
```

Authorization

None

This example gets the text information for the index of a common-index table:

```
db2tx GET TEXT INFO db2tx.mytable
```

This example gets the text information for the index of a multi-index table column:

Getting useful information

```
db2tx GET TEXT INFO db2tx.mytable HANDLE commenthandle
```

The syntax lets you specify a table name and the name of a handle column.

If you specify only a table name in the command, the text information for each enabled column in this table is displayed. If you also specify a handle column name, only the text information for that column is displayed.

Here is an example of what is displayed by this command for a multi-index table:

```
Text information for column ABSTRACT1
      with handle column ABSTRACT1HANDLE:
Coded character set ID (CCSID) = 500
Language              (LANGUAGE) = US_ENGLISH
Format                (FORMAT) = TDS

Text information for column ABSTRACT2
      with handle column ABSTRACT2HANDLE:
Coded character set ID (CCSID) = 500
Language              (LANGUAGE) = US_ENGLISH
Format                (FORMAT) = TDS
```

Working with the DB2 Text Extender catalog view

DB2 Text Extender creates and maintains a catalog view called DB2TX.TEXTINDEXES for each subsystem. It is created when you run ENABLE SERVER. It contains information about the tables and columns that are enabled for DB2 Text Extender.

New entries are created in DB2TX.TEXTINDEXES whenever a table, a column are enabled. Entries are deleted if columns or tables are disabled.

Data in the catalog view is available through normal SQL query facilities. However, you cannot modify the catalog view using normal SQL data manipulation commands. You cannot explicitly create or drop the catalog view. Table 4 shows the contents of the catalog view.

Table 4. DB2 Text Extender catalog view

Column name	Data type	Nullable	Description
TABLESCHEMA	CHAR(8)	No	Schema of the table to which this entry applies.
TABLENAME	VARCHAR(18)	No	Name of the table to which this entry applies.
COLUMNNAME	VARCHAR(18)	Yes	Name of a column that has been enabled within this table. This value is null if the table has been enabled, but no column has been enabled.
HANDLENAME	VARCHAR(18)	Yes	Name of a handle column. This value is null if there is no column enabled in the table TABLESCHEMA.TABLENAME.
INDEXNAME	CHAR(8)	No	Name of the text index created during enabling of the text table or a text column.
LOGTABLE	VARCHAR(18)	No	Name of the log table for the index INDEXNAME. The table DB2TX.LOGTABLE contains information about which text documents are scheduled for the next update of the text index, and error events.
INDEXTYPE	VARCHAR(30)	No	Type of index: LINGUISTIC, PRECISE, NGRAM.
MINIMUM	INTEGER	Yes	For future use.
DAYS	VARCHAR(15)	Yes	For future use.

Table 4. DB2 Text Extender catalog view (continued)

Column name	Data type	Nullable	Description
HOURS	VARCHAR(75)	Yes	For future use.
MINUTES	VARCHAR(185)	Yes	For future use.
INDEXDIRECTORY	VARCHAR(254)	No	Name of the directory where the text index is stored within the file system.
UPDATEONCREATE	VARCHAR(10)	No	The value "update" or "noupdate", whatever has been specified with the UPDATEINDEX option in ENABLE TEXT TABLE or ENABLE TEXT COLUMN, or in the last CHANGE INDEX SETTINGS.
COMMONINDEX	VARCHAR(4)	No	"yes" if the table TABLESCHEMA.TABLENAME is a common-index table. "no" if the table TABLESCHEMA.TABLENAME is a multi-index table.
CCSID	SMALLINT	Yes	CCSID for the text column TEXTCOLUMN specified with the enable text column command. This value is null if TEXTCOLUMN is null.
LANGUAGE	VARCHAR(30)	Yes	The name of the dictionary used when processing text column TEXTCOLUMN. This value is null if TEXTCOLUMN is null.
FORMAT	VARCHAR(30)	Yes	The format specified for text column TEXTCOLUMN. This value is null if TEXTCOLUMN is null.
FUNCTIONSCHEMA	CHAR(8)	Yes	Schema of the access function specified in the ENABLE TEXT COLUMN command using the FUNCTION option. This value is null if no FUNCTION option is specified.
FUNCTIONNAME	VARCHAR(18)	Yes	Name of the access function specified in the ENABLE TEXT COLUMN command using the FUNCTION option. This value is null if no FUNCTION option is specified.
PROTOTYPEHANDLE	VARCHAR(60)	Yes	A handle for use in performance functions. It contains only the index name which is common for the whole text column.
INDEXOPTION	VARCHAR(30)	Yes	Option used when creating the index: CASE_ENABLED.
INDEXPROPERTY	VARCHAR(30)	Yes	Property used when creating the index: SECTIONS_ENABLED.

Reversing the text preparation process

When text is prepared for use by DB2 Text Extender, certain administrative changes are made. This section describes functions that help you to reverse this process.

Reversing the text preparation process

Disabling a text column

Summary

When When you no longer intend to make text searches in a text column.

Command

```
DISABLE TEXT COLUMN
```

Authorization

ALTER, SELECT, UPDATE on the table.

Example:

```
db2tx DISABLE TEXT COLUMN db2tx.mytable HANDLE commenthandle
```

When you disable a text column, the following occurs:

- If this is a multi-index table, that is, the column has its own text index and log table, then the index, the log table, and the log table triggers are deleted.
- If this is a common-index table, that is, there is one index shared by all text columns, then the terms for this column's documents are removed from the common index. If this is the only remaining enabled text column in the table, then the index, the log table, and the log table triggers are deleted.

Disabling a server

Summary

When When you no longer intend to make text searches in this server.

Command

```
DISABLE SERVER for DB2TEXT
```

Authorization

SYSADM or DBADM on the database.

To disable the connected subsystem, enter:

```
db2tx DISABLE SERVER for DB2TEXT
```

When you disable a server, the following objects are deleted:

- The DB2 Text Extender catalog view that was created when the server was enabled
- The declaration of DB2 Text Extender's SQL functions (UDFs), and DB2 Text Extender distinct types (UDTs) for this server
- All indexes related to any of this server's text tables or text columns
- The log tables used to automatically record which text documents are to be indexed, and the triggers used to maintain them

Because handle columns cannot be deleted, and the handle column is of a distinct type, some distinct types are not deleted.

Only the connected server is affected by this call.

Part 2. Reference

Chapter 8. Text preparation and administration commands for the client

Table 5. Text preparation and administrative commands for the client

Command	Purpose	Page
CHANGE INDEX SETTINGS	Changes the characteristics of an index	76
CHANGE TEXT CONFIGURATION	Changes the text configuration settings	77
DELETE INDEX EVENTS	Deletes index events from a log table	79
DISABLE SERVER FOR DB2TEXT	Disables a server from use by DB2 Text Extender	80
DISABLE TEXT COLUMN	Disables a text column from use by DB2 Text Extender, and deletes its associated index	81
ENABLE SERVER FOR DB2TEXT	Prepares a server for use by DB2 Text Extender	82
ENABLE TEXT COLUMN	Prepares a text column for use by DB2 Text Extender and creates an individual text index for the column	83
GET ENVIRONMENT	Displays the current settings of the environment variables	89
GET INDEX SETTINGS	Displays the characteristics of an index	90
GET INDEX STATUS	Displays status information for an index	92
GET STATUS	Displays the enabled status of subsystems, tables, and columns	93
GET TEXT CONFIGURATION	Displays the text configuration settings	94
GET TEXT INFO	Displays the text information for a text column	95
QUIT	Exits from the DB2 Text Extender command line processor mode	96
REORGANIZE INDEX	Reorganizes an index to improve search efficiency	97
RESET INDEX STATUS	Resets the status of an index to allow it to be used again	98
UPDATE INDEX	Updates a text index	99

This chapter describes the syntax of the text preparation and administration commands for the client. Chapter 5, “Making text searchable,” on page 39 and Chapter 7, “Administration,” on page 63 describe how to use these commands.

Before you use these commands, start the DB2 Text Extender command line processor by entering in the UNIX Shell of z/OS, the command `db2tx`. It puts you into an interactive input mode in which all subsequent commands are interpreted as DB2 Text Extender commands.

To leave this mode, enter `QUIT`.

CHANGE INDEX SETTINGS

This command changes the characteristics of an index **after** the subsystem has been enabled.

To changes the default settings that are used when a subsystem is first enabled, use “CHANGE TEXT CONFIGURATION” on page 77.

Command syntax

```
▶▶—CHANGE INDEX SETTINGS—table-name—┬──────────────────────────────────┬──────────▶  
└──HANDLE—handle-column-name──┘
```

Command parameters

table-name

The name of the text table in the connected subsystem that contains the text column whose index update frequency is to be changed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose index update frequency is to be changed. This is required if the text column has its own index, that is, if the index was created using the command ENABLE TEXT COLUMN.

If, however, the index was created using the command ENABLE TEXT TABLE, that is, the table has one text index for all text columns, then this keyword is ignored.

CHANGE TEXT CONFIGURATION command

UPDATEINDEX

A keyword that determines whether the text documents are indexed immediately after the command using this option has completed, without waiting for the next periodic indexing set by UPDATEFREQ. These commands are ENABLE TEXT COLUMN, and ENABLE TEXT FILES.

UPDATE

Indexing of the text documents occurs immediately after the command has completed.

NOUPDATE

Indexing occurs at a time set by the update frequency settings specified either in this command by UPDATEFREQ, or by the text configuration setting.

COMMITCOUNT count

A value from 500 to 1 000 000 indicating the number of inserts or updates after which DB2 UDB for OS/390 and z/OS must issue an intermediate commit statement. This can avoid a situation in which there is insufficient log space when enabling large tables, or columns, or a large number of external files.

CCSID ccsid

The Coded Character Set Identifier to be used when indexing text documents.

For information about CCSIDs that can be supported, see “CCSIDs” on page 25.

LANGUAGE language

The language in which the text is written. This determines which dictionary is to be used when indexing text documents and when searching in text documents. Chapter 12, “Linguistic processing for linguistic and precise indexes,” on page 133 describes how dictionaries are used.

The supported languages are listed in Table 1 on page 36.

FORMAT format

The type of text document stored, such as WordPerfect, or ASCII. DB2 Text Extender needs this information when indexing documents. The document formats supported are listed in “Which document formats are supported” on page 22.

DELETE INDEX EVENTS

This command deletes indexing events from an index's log table for a given handle column or table.

Command syntax

```

▶▶—DELETE INDEX EVENTS—table-name—┬──┬──▶
                                     |
                                     └──HANDLE—handle-column-name—┘
  
```

Command parameters

table-name

The name of the text table in the connected subsystem whose error events are to be deleted from the log table. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose messages are to be deleted from the log table.

Usage notes

If a handle column name is given, the indexing events for only this column are deleted.

You clean up the log tables, you should delete index events after you have checked the reason for the event and possibly removed the source of the error.

DISABLE SERVER FOR DB2TEXT

This command resets any preparation work done by DB2 Text Extender for a server and disables all text tables for use by DB2 Text Extender.

Command syntax

▶▶—DISABLE SERVER FOR DB2TEXT—▶▶

Command parameters

None.

Usage notes

This command resets the connected server so that it can no longer be searched by DB2 Text Extender; that is, it disables all DB2 Text Extender text tables and text columns in the server. This includes all modifications that were made in the server to enable DB2 Text Extender text tables, and text columns. All text columns are reset, all related text indexes are deleted, the DB2 Text Extender catalog view TEXTCOLUMNS in the server is deleted, and all DB2 Text Extender triggers are deleted.

DISABLE TEXT COLUMN

This command disables a text column for use by DB2 Text Extender.

Command syntax

►►—DISABLE TEXT COLUMN—*table-name*—HANDLE—*handle-column-name*—►►

Command parameters

table-name

The name of the text table in the connected subsystem that contains the column to be disabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column to be disabled for use by DB2 Text Extender.

Usage notes

The index is deleted.

The log table used to record changes in the handle column (inserts, updates, and deletions) is deleted.

The triggers that write entries to the log table are deleted.

The handle column is not changed.

Do not run this command more than once simultaneously (from different clients). This could result in a deadlock or a timeout.

ENABLE SERVER FOR DB2TEXT

This command enables the current server to be used by DB2 Text Extender.

Command syntax

▶▶—ENABLE SERVER FOR DB2TEXT—▶▶

Command parameters

None.

Usage notes

This command prepares the connected subsystem for use by DB2 Text Extender. It is a mandatory step before a DB2 Text Extender text table or text column can be enabled in the subsystem. ENABLE SERVER creates a DB2 Text Extender catalog view called DB2TX.TEXTINDEXES, described in “Working with the DB2 Text Extender catalog view” on page 70, and a catalog view called DB2TX.TEXTCOLUMNS.

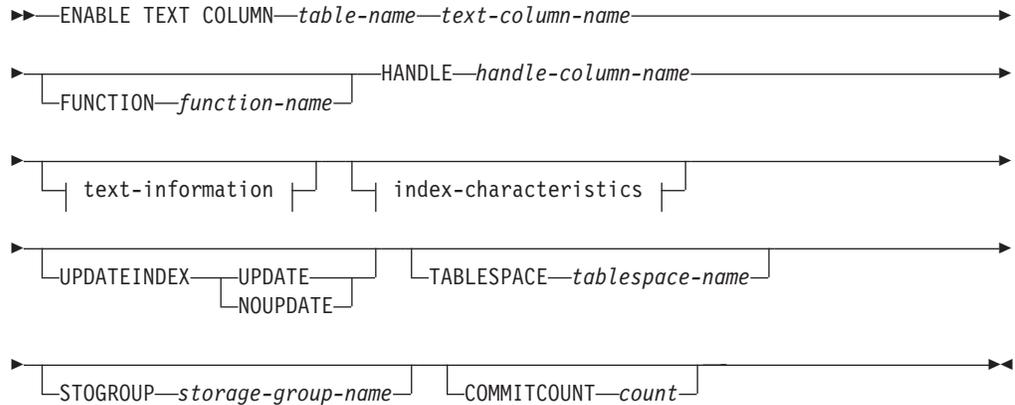
This command also creates text configuration settings, described in “Text configuration settings” on page 13.

Some other work is also done, such as the declaration of DB2 Text Extender distinct types and DB2 Text Extender functions.

ENABLE TEXT COLUMN

This command enables a text column for use by DB2 Text Extender.

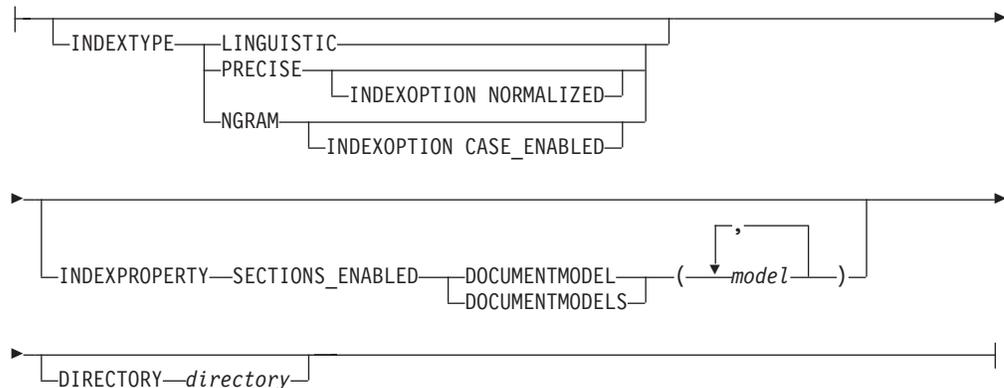
Command syntax



text-information:



index-characteristics:



Command parameters

table-name

The name of the text table in the connected subsystem that contains the column to be enabled. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

text-column-name

The name of the column to be enabled for use by DB2 Text Extender. This column must be of the type CHAR, VARCHAR, LONG VARCHAR, CLOB, DBCLOB, GRAPHIC, VARGRAPHIC, or LONG VARGRAPHIC. If the document type is not one of these, use FUNCTION to convert the document type.

ENABLE TEXT COLUMN command

FUNCTION *function-name*

The name of a user-defined function to be used by DB2 Text Extender to access text documents that are in a column that is not of type CHAR, VARCHAR, LONG VARCHAR, CLOB, DBCLOB, GRAPHIC, VARGRAPHIC, or LONG VARGRAPHIC. See “Enabling text columns of a nonsupported data type” on page 45 for more information.

HANDLE *handle-column-name*

The name of the handle column to be added to the table for use by DB2 Text Extender’s functions.

CCSID *ccsid*

The Coded Character Set Identifier to be used when indexing text documents.

If you specify a CCSID when you enable a text column for an Ngram index, the CCSID must be the same as the CCSID of the subsystem, and the CCSID used during search (the CCSID of the subsystem) must match this CCSID. To find the default CCSID, use:

```
db2tx get text cfg
```

The installation default is the subsystem CCSID.

If the subsystem CCSID is not 500, you must specify the CCSID parameter. In this case, you should also change the default CCSID in the text configuration with the command:

```
db2tx change text cfg
```

If this keyword is not specified, the CCSID specified in the text configuration settings is used. Subsequent changes to the text configuration settings are ignored; the value used is the one that existed at the time the column was enabled, not the one that exists when indexing text documents.

For information about other CCSIDs that can be supported, see “CCSIDs” on page 25.

LANGUAGE *language*

The language in which the text is written. This determines which dictionary is to be used when indexing text documents and when searching in text documents. Chapter 12, “Linguistic processing for linguistic and precise indexes,” on page 133 describes how dictionaries are used.

This keyword specifies the language once for the whole column. You can override this value for individually inserted text documents using the INIT_TEXT_HANDLE function in an INSERT statement.

If this keyword is not specified, the language specified in the text configuration settings is used. Subsequent changes to the text configuration settings are ignored; the value used is the one that existed at the time the column was enabled, not the one that exists when indexing text documents.

The supported languages are listed in Table 1 on page 36.

FORMAT *format*

The type of text document stored, such as WordPerfect, or ASCII. DB2 Text Extender needs this information when indexing documents. The document formats supported are listed in “Which document formats are supported” on page 22.

The document formats supported for structured documents are:

- **ASCII_SECTIONS**
Documents having the format ASCII_SECTIONS cannot contain nested sections. (For information about nested sections, see “Working with structured documents (section support)” on page 34.) A start tag for a section is ended by the next start tag.
- **HTML**
A sample document model file is provided for HTML documents. It contains a subset of the standard HTML definitions, which you can modify. HTML documents cannot contain nested sections.
- **XML**
The processing of XML documents includes Document Type Definition (DTD) evaluation. The model assigned to the document is checked against the DTD. If the tags defined in the document models file are not defined in the DTD, the document is not indexed. If no model has been defined for a recognized DTD, the document will not be indexed. XML documents can contain nested sections.
XML is only available with OS/390® release 2.9 and z/OS, which includes Text Search Engine release 4 (FMID HIMN230).

For these formats, you must specify the structure information in a document model file. See “Working with structured documents (section support)” on page 34. If the format TDS and INDEXPROPERTY SECTION_ENABLED are specified, it is assumed that the document format is ASCII_SECTIONS.

Tags that are not defined in the models file are indexed in the normal way, according to the index type.

This keyword specifies the format once for the whole column. You can override this value for individually inserted text documents using the INIT_TEXT_HANDLE function in an INSERT statement.

If this keyword is not specified, the format specified in the text configuration settings is used. Subsequent changes to the text configuration settings are ignored; the value used is the one that existed at the time the column was enabled, not the one that exists when indexing text documents.

INDEXTYPE

The type of index to be created. For more information, see “Types of search” on page 29.

PRECISE

Terms are indexed and searched for exactly as they occur in the text documents.

LINGUISTIC

Terms are processed linguistically before being indexed. Search terms are also processed linguistically before the search begins.

NGRAM

Terms are indexed by parsing sets of characters rather than by using a dictionary. This index type is mandatory if the documents you are indexing contain DBCS characters, although an Ngram index can also be used for SBCS documents.

If you do not specify the INDEXTYPE keyword, the value in the text configuration settings is used.

ENABLE TEXT COLUMN command

Documents in XML format are not supported for Ngram indexes.

INDEXOPTION

Options to be used when creating the index.

CASE_ENABLED

This option is available **only for Ngram indexes**. Normally, Ngram indexes do not allow a case-sensitive search. By specifying `CASE_ENABLED`, you ensure that documents are indexed such that a case-sensitive search is possible. For more information see “Make a fuzzy search or search in DBCS documents” on page 32.

NORMALIZED

This option is available **only for precise indexes**. A normalized precise index differs from a precise index in that:

- It is not case-sensitive; all words except those in all uppercase are converted to lowercase.
- Words in all uppercase are not subject to stop-word filtering; the abbreviation UK, for example, is indexed.
- English language search terms may be expanded to include lemma forms using a heuristic algorithm, so that a search for house also searches for houses.

INDEXPROPERTY SECTIONS_ENABLED DOCUMENTMODEL(S) model

Properties of a selected index type.

`SECTIONS_ENABLED` specifies that the selected index type can contain information about the document structure.

`DOCUMENTMODEL/DOCUMENTMODELS model` specifies the model or models to be associated as default for the documents to be indexed. A model name must be specified if the index property `SECTIONS_ENABLED` is used. If a list of models is specified, the first model is used as the default model for the index. The default model is used during indexing if the document has no reference to a model, or if no model is specified during search.

Note

The availability of document models depends on the OS/390 and z/OS release:

- For OS/390 version 2.8 or less, (including Text Search Engine release 3 (FMID HIMN210 or 220)), only a single document model can be associated with documents for indexing.
- For OS/390 version 2.9 or higher, and z/OS releases (which include Text Search Engine release 4 (FMID HIMN230)), multiple document models can be associated.

The characters that can be used for the model name are a-z, A-Z, and 0-9.

The specified model name must correspond to a model definition in the model definition file `imodel.ini`. Note that the model name is case sensitive.

To change the model or models associated with an index,

1. Use `DISABLE TEXT COLUMN` to disable the index

2. Use ENABLE TEXT COLUMN to reindex the documents, specifying different document model names.

DIRECTORY directory

The directory path in which the text index is to be stored. The specified path is concatenated with "txinsnnn" where *nnn* is the node number.

This is a directory on the system where the DB2 Text Extender server is running. If the directory does not yet exist, it is created.

If you do not specify the DIRECTORY keyword, the value of the DIRECTORY setting in the text configuration settings is used.

This setting is ignored if it has already been set for the whole table by ENABLE TEXT TABLE.

UPDATEINDEX

A keyword that determines whether the text documents associated with this handle column are indexed immediately after this command has completed, without waiting for the next periodic indexing set by UPDATEFREQ.

UPDATE

Indexing of the text documents occurs immediately after this command has completed.

NOUPDATE

Indexing occurs at a time set by the update frequency settings specified either in this command by UPDATEFREQ, or by the text configuration setting.

If you do not specify this keyword, the value in the text configuration settings is taken.

TABLESPACE tablespace-name

The name of the table space for the index that is created internally on the handle column. The tablespace must have been created previously.

STOGROUP storage-group-name

The name of the storage group for the index that is created internally on the handle column. The storage group must have been created previously.

COMMITCOUNT count

A value from 500 to 1 000 000 indicating the number of inserts or updates after which DB2 UDB for OS/390 and z/OS must issue an intermediate commit statement. This can avoid a situation in which there is insufficient log space when enabling large tables, or columns, or a large number of external files. Using COMMITCOUNT, requires you to set KEEP DYNAMICS(YES) and to rebind the DB2 ODBC packages. See job DSNTIJCL for more detail.

Usage notes

This command adds a handle column to the specified DB2 table. Each handle column is associated with a text column, and is used by DB2 Text Extender's functions.

If this table has not already been enabled to create a common index, an index is created that is associated with this text column.

ENABLE TEXT COLUMN command

Also, a log table is created in the subsystem. The log table is used to record changes to the text column, that is inserts, updates, and deletions. Insert, update, and delete triggers are defined for the text column to keep the log table up to date automatically.

Do not run this command more than once simultaneously (from different clients). This could result in a deadlock or a timeout.

GET ENVIRONMENT

This command displays the settings of the environment variables.

Command syntax

▶▶—GET ENVIRONMENT—————▶▶

Command parameters

None.

Usage notes

These are the environment variables displayed:

DB2DBDFT

Default subsystem name.

DB2TX_INSTOWNER

DB2 Text Extender instance name. Required only if you connect to a UNIX server.

DB2TX_INSTOWNERHOMEDIR

Instance owner's home directory. Required only if you connect to a UNIX server.

GET INDEX SETTINGS

This command displays the settings of an index, showing the following:

- Index type
- Index option (optional)
- Update index option
- Index directory
- Update frequency
- Default model.

Command syntax

```
▶▶ GET INDEX SETTINGS table-name [HANDLE handle-column-name] ▶▶
```

Command parameters

table-name

The name of the text table in the connected subsystem whose index settings are to be displayed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose index settings are to be displayed.

Usage notes

If the table is enabled as a multi-index table, this command displays the index settings of all enabled text columns in the table. If a *handle-column-name* is provided, this command displays the index settings of the specified column.

If the table is a common-index table, the settings of the common index are displayed. If a *handle-column-name* is provided, it is ignored.

If the table or column is enabled with the index property `SECTIONS_ENABLED`, the command `GET INDEX SETTINGS` displays the default model for the index. The default model is the model name you specified during enabling or the first model name in a list of model names.

Note

The availability of default model depends on the OS/390 and z/OS release:

- For OS/390 version 2.8 or less, only a single default document model can be associated with documents for indexing.
- For OS/390 version 2.9 or higher, and z/OS releases, multiple default document models can be associated.

Here is an example:

```
Current index settings:  
Index type          (INDEXTYPE) = LINGUISTIC  
Default model      (DOCUMENTMODEL) = mymodel  
Update index option (UPDATEINDEX) = UPDATE
```

GET INDEX SETTINGS command

```
Update frequency (UPDATEFREQ) = NONE
Node 0
Index directory (DIRECTORY) = /home/user1/db2tx/indices
```

GET INDEX STATUS

This command displays the following index status information for a given handle column or table:

- Whether the search function is available
- Whether the index update function is available
- Whether the reorganize function is available
- The number of scheduled documents
- The number of indexed documents
- The number of indexed documents in the primary index
- The number of indexed documents in the secondary index
- Error events.

Command syntax

```
▶▶ GET INDEX STATUS table-name [HANDLE handle-column-name] ▶▶
```

Command parameters

table-name

The name of the text table in the connected subsystem that contains the text columns whose status is to be displayed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose status is to be displayed.

Usage notes

For a multi-index table, you must specify the name of the handle column.

GET STATUS

This command displays information about the enabled status of subsystems, tables, or text columns.

Command syntax

▶▶—GET STATUS—▶▶

Command parameters

None.

Usage notes

This command displays whether the server is enabled, the names of the enabled text tables in the subsystem, the names of the enabled text columns and their associated handle columns.

GET TEXT CONFIGURATION

This command displays the default settings for the text configuration for the connected subsystem.

To change these default settings, use “CHANGE TEXT CONFIGURATION” on page 77.

Command syntax

▶▶ GET TEXT CONFIGURATION ◀◀
 └─CFG─┘

Command parameters

None.

Usage notes

For an example of the text configuration information, see “Displaying the text configuration settings” on page 66.

GET TEXT INFO

This command displays the text information settings for text columns:

- CCSID
- Language
- Format.

Command syntax

```
▶▶ GET TEXT INFO table-name [HANDLE handle-column-name]
```

Command parameters

table-name

The name of the text table in the connected subsystem that contains the text columns whose text information settings are to be displayed. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose text information settings are to be displayed.

Usage notes

If a handle column name is given, the text information for only this column is displayed.

If a handle column name is not given, the text information for each enabled column in this table is displayed.

QUIT command

QUIT

This command stops the DB2 Text Extender command line processor and returns control to the operating system.

Command syntax

▶▶—QUIT—▶▶

Command parameters

None.

Usage notes

The connection to the subsystem is terminated.

REORGANIZE INDEX

If a text column is often updated, searching the index can become inefficient. To make searching efficient again, the index has to be *reorganized*. Although DB2 Text Extender recognizes when an index needs to be reorganized and does so in the background automatically, there may be situations that require an index to be reorganized manually using REORGANIZE INDEX. You can use the command GET INDEX STATUS to find out if an index needs to be reorganized.

Command syntax

```

▶▶—REORGANIZE INDEX—table-name—┬──────────────────────────────────────────┬▶▶
                                   └─HANDLE—handle-column-name—┘

```

Command parameters

table-name

The name of the text table in the connected subsystem whose index is to be reorganized. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose index is to be reorganized.

Usage notes

For a multi-index table, you must specify a handle column name.

Although searches can be made on the index while REORGANIZE INDEX is running, index updates cannot.

RESET INDEX STATUS

When the index status of a table or column shows Search not available or Update not available, an error has occurred during indexing that prevents you working with the index.

This command resets the index status so that you can continue to work with it. Before resetting the index status, check for any errors that may be logged in the index's log table (see "Displaying error events" on page 68).

Command syntax

```
▶▶—RESET INDEX STATUS—table-name—┬──────────────────────────────────────────┬▶▶  
└─HANDLE—handle-column-name─┘
```

Command parameters

table-name

The name of the text table in the connected subsystem that contains the text columns whose status is to be reset. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

The name of the handle column whose status is to be reset.

Usage notes

For a multi-index table, you must specify a handle column name.

For a common-index table, each enabled column in this table is reset.

UPDATE INDEX

This command starts indexing immediately. It brings the index up to date to reflect the current contents of the text column(s) with which the index is associated.

Command syntax

```

▶▶—UPDATE INDEX—table-name—┐
                                └—HANDLE—handle-column-name—┘
└──────────────────────────────────────────────────────────────────────────────────▶▶
└──COMMITCOUNT—count──┘

```

Command parameters

table-name

The name of the text table in the connected subsystem that contains the text column whose index is to be updated. This can also be the name of a common-index table. The name must include an explicit schema name (qualifier) unless the schema name is the same as your user ID.

HANDLE handle-column-name

If this is a common-index table, the *handle-column-name* is not required and is ignored. The index to be updated is associated with the whole table and not with an individual text column.

If this is a multi-index table, then *handle-column-name* is the name of the handle column whose index is to be updated.

COMMITCOUNT count

A value from 500 to 1 000 000 indicating the number of inserts or updates after which DB2 UDB for OS/390 and z/OS must issue an intermediate commit statement. This can avoid a situation in which there is insufficient log space when enabling large tables, or columns.

UPDATE INDEX command

Chapter 9. Administration commands for the server

This chapter describes the syntax of the administration commands for the server. Server administration consists of tasks you can do to start, stop, and check the status of the DB2 Text Extender server, and to create a sample database and sample a table. "Setting up and maintaining a DB2 Text Extender server" on page 15 describes how to use these commands.

Table 6. Administration commands for the server

Command	Purpose	Page
TXSTART	Starts the DB2 Text Extender services	102
TXSTATUS	Displays the status of the search service	103
TXSTOP	Stops the DB2 Text Extender services	104
IMOTHESC	Compiles a thesaurus definition file	105
IMOTRACE	Produces trace information	106

TXSTART

This command starts the DB2 Text Extender search services.

Command syntax

▶▶—txstart—————▶▶

Command parameters

None.

Usage notes

Run this command:

- While logged on with a user ID in the SM administration group
- Whenever you stop and restart your server system
- After starting DB2.

Before you can index data or search on your data, you have to start the DB2 Text Extender search services.

TXSTATUS

This command displays whether DB2 Text Extender is up and running.

Command syntax

▶▶—txstatus—▶▶

Command parameters

None.

TXSTOP command

TXSTOP

This command stops the DB2 Text Extender services.

Command syntax

▶▶—txstop—————▶▶

Command parameters

None.

Usage notes

This command does not stop DB2.

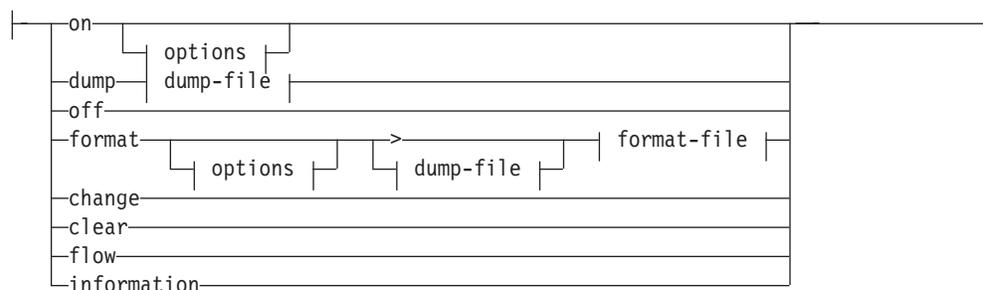
IMOTRACE

This command writes processing information to a trace buffer in shared memory. It can be written in binary from the trace buffer to a file for later formatting when tracing has been switched off, or can be formatted and written to a file while tracing is still on.

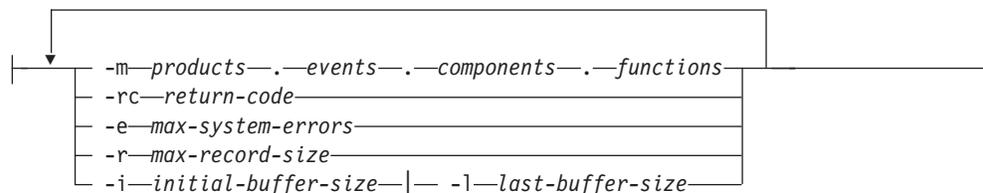
Command syntax

► imotrace | parameters |

parameters:



options:



dump-file:



format-file:



Command parameters

Note:

A -u option is also available with all of the DESTTRACE parameters to display information about the parameter.

on To start the trace facility.

dump | dmp

To write the trace information in binary to a file.

off To stop the trace facility.

format | fmt

To format the binary trace information. You can format the dump file, when tracing is switched off, by specifying the name of the dump file and the name of the file to hold the formatted trace information. To format the trace information directly from the trace buffer while tracing is still switched on, enter: `destrc format > filename.tmp`

change | chg

To change the trace mask, `maxSysErrors`, or `maxRecordSize`.

clear | clr

To clear the trace.

flow | flw

To show control flow of the trace.

information | info | inf

To get information about the trace.

options

To filter the trace information either when turning tracing on (this reduces the shared memory usage), or when formatting the trace information. Unless the trace is very large, it is usually easier to write the full trace information and then filter it during formatting.

-m To add a “mask” to specify which events, components, and functions are to be included in the trace. The default is to trace everything. The mask is in four parts, separated by periods; for example: `2.2-6.1,3.*` You can specify a range using “-” as a separator character, or a list using “,” as a separator character. For example: `2-6` includes only the events whose IDs are in the range from 2 to 6. To include only components 2 *and* 6, specify `2,6`

products

Product ID. The product ID for DB2 Text Extender is “2”. The product ID for TextMiner is “3”.

events The set of event types to be included in the trace:

0	system_error
1	system_error
2	system_error
3	non-fatal_error
4	non-fatal_error
5	api_errcode
7	fnc_errcode
8	trap_error
10	api_entry
11	api_exit
13	api_retcode
15	api_data
30	fnc_entry
31	fnc_exit
33	fnc_retcode
35	fnc_data

IMOTRACE command

components

The components to trace.

The component IDs for DB2 Text Extender are:

- 1 COMMAND_LINE_INTERFACE
- 2 UDF
- 3 STORED_PROCEDURES
- 4 ADMINISTRATION
- 5 INDEX_CONTROL
- 6 LIBRARY_SERVICES
- 7 DES_PARSER
- 8 DES_DEMON
- 9 DES_API
- 10 SERVICES

The component IDs for TextMiner are:

- 1 automachine
- 2 bgproc (background processing)
- 3 cluster
- 4 common
- 5 commsvc (common services)
- 6 communic (communication)
- 7 daemon
- 8 dsclient
- 9 environ (environment)
- 10 glue
- 11 idxcomm (index build, common part)
- 12 libsrv (library services)
- 13 search
- 14 trace
- 15 guru
- 16 indexbld (index build, tm only)
- 17 indexeng (index engine, tm only)
- 18 smsearch
- 19 search engine, tm only)
- 20 tmsearch
- 21 gtrcm (gtr, common part)
- 22 gtrsrch (search, gtr only)
- 23 gtridx (index build, gtr only)

functions

Asterisk (*). The set of functions to trace. Use an asterisk (*) to trace all functions unless directed to do otherwise by the IBM Support Center.

-rc *return-code*

Treat *return-code* as a system error.

-e *max-system-errors*

Integer. To stop the trace after this number of errors. The default is 1 which specifies that when the first system error occurs, all subsequent tracing of lower severity events is suppressed. This is acceptable if you are interested only in the first major error, but you should specify a higher number (such as -e 50) if you want to see the full trace after the initial system error. The trace destination is shared memory.

-r *max-record-size*

Integer. To stop the trace after this number of records have been written to the trace file. The default is 16 KB.

-i *initial-buffer-size*

Integer. To keep this number of records from the beginning of the trace. If **-i** is specified, the default is 16 KB. On a UNIX system, a recommended buffer size is 2 MB.

If neither **-i** nor **-l** are specified, **-l** is the default.

If you specify **-i**, no wrapping occurs; no further trace entries are written if the volume of records exceeds *max-record-size*, even if you clear all trace entries. To get new trace entries written, increase the buffer size, turn the trace off and then on again.

-l *last-buffer-size*

Integer. To keep this number of records from the end of the trace. The default is 16 KB. On a UNIX system, a recommended buffer size is 2 MB.

path The directory where the corresponding file is stored.

dump-filename

The name of the file that contains the binary trace information.

formatted-filename

The name of the file that contains the formatted trace information.

Examples

See “Tracing faults” on page 16.

IMOTRACE command

Chapter 10. Search functions

DB2 Text Extender provides SQL functions to search in text documents stored in DB2 UDB for OS/390 and z/OS, and to work with the results of a search. Some of the functions' parameters are data types known as *distinct types* that are provided with DB2 Text Extender.

This chapter describes the DB2 Text Extender SQL functions and distinct types.

The DB2 Text Extender distinct types

Table 7. The DB2 Text Extender distinct types

Distinct type	Source data type	Comments
DB2TEXTH	VARCHAR(60) FOR BIT DATA	Text handle. A variable-length string containing information needed for indexing a text document stored in a text column. The information in a handle includes a document ID, the name of the server where the text is to be indexed, the name of the index, and information about the text document. Handles are stored in columns that DB2 Text Extender creates and associates with each text column. The functions HANDLE and INIT_TEXT_HANDLE return this data type.

A summary of DB2 Text Extender functions

Table 8. A summary of the DB2 Text Extender functions

Search function	Purpose	Page
CCSID	Returns the CCSID from a handle	112
CONTAINS	Makes a search for text in a particular document	113
INIT_TEXT_HANDLE	Returns a partially initialized handle containing information such as format and language settings	115
LANGUAGE	Returns or changes the language setting in a handle	116
NO_OF_MATCHES	Searches and returns the number of matches found	117
RANK	Searches and returns the rank value of a found text document	118
REFINE	Takes a search argument and a refining search argument and returns a combined search argument	119
SEARCH_RESULT	Returns an intermediate table with the search result of the specified search string	120

Examples of the use of DB2 Text Extender functions are given in Chapter 6, "How to search," on page 47.

CCSID

The CCSID function returns the CCSID (data type SMALLINT) from a handle. This is the CCSID parameter used for indexing the corresponding text document. This is described in “CCSIDs” on page 25. It is set for each text column by the ENABLE TEXT COLUMN command.

Function syntax

►►—CCSID—(*—handle—*)—◄◄

Function parameters

handle

An expression whose result is a value of type DB2TEXTH. It is usually the name of a handle column from which the CCSID setting is to be returned.

CONTAINS

The CONTAINS function searches for text in a particular text document. It returns the INTEGER value 1 if the document contains the text. Otherwise, it returns 0.

Tip

Use the SEARCH_RESULT search function when operating with large tables as this provides a significant performance improvement. If you only use the CONTAINS predicate in your query, the CONTAINS function is called for every row in the table.

Function syntax

►► CONTAINS (—*handle*—, —*search-argument*—) ◀◀

Function parameters

handle

An expression whose result is a value of type DB2TEXTH. It is usually the name of a handle column containing the handles of the text documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See Chapter 11, “Syntax of search arguments,” on page 121.

Note

Use CONTAINS only for small tables. For larger tables, use SEARCH_RESULT.

FORMAT

The FORMAT function does one of the following:

- Returns the document format specified in a handle
- Changes the format specification in a document's handle, and returns the changed handle.

The returned document format is a string of type VARCHAR(30). The returned handle is of type DB2TEXTH.

This is the format parameter used for indexing the corresponding text document. The document formats supported are listed in “Which document formats are supported” on page 22.

Function syntax

►► ⁽¹⁾ FORMAT (—*handle*—) ◀◀

Notes:

- 1 Returns a format value, type VARCHAR(30).

►► ⁽¹⁾ FORMAT (—*handle*—, —*format*—) ◀◀

Notes:

- 1 Returns a handle, type DB2TEXTH.

Function parameters

handle

An expression whose result is a value of type DB2TEXTH. It is usually the name of a handle column from which the format setting is to be returned or set.

format

The new document format setting of data type VARCHAR(30).

If *format* is specified, this document format is set in the handle; in this case, the handle is returned instead of the format setting.

INIT_TEXT_HANDLE

The INIT_TEXT_HANDLE function returns a partially initialized handle that contains preset values for the text's format or language. It can be inserted into a handle column. This is useful when you add a row containing text whose format and language are not the same as those specified in the text configuration settings.

The returned handle is a value of type DB2TEXTH.

Function syntax

```
▶▶—INIT_TEXT_HANDLE—(—format—,—language—)—————▶▶
```

```
▶▶—INIT_TEXT_HANDLE—(—CCSID—,—format—,—language—,—file-name—)—————▶▶
```

Function parameters

format

A string of type VARCHAR(30) specifying the new document format setting. The formats supported are listed in “Which document formats are supported” on page 22.

language

A string of type VARCHAR(30) specifying the new document language setting. The supported languages are listed in Table 1 on page 36.

file-name

A string of type VARCHAR(150) specifying the absolute path and file name of the external file that is to be associated with the handle. To have access to the file in UNIX, the DB2 UDB for OS/390 and z/OS instance owner must be included in the file access permissions. For Windows users, the file access permissions must include the logon user IDs.

LANGUAGE

The LANGUAGE function does one of the following:

- Returns the document language specified in a handle
- Changes the language specification in a document's handle, and returns the changed handle.

The returned document language is a string of type VARCHAR(30). The returned handle is of type DB2TEXTH.

This is the language parameter used for indexing the corresponding text document. The supported languages are listed in Table 1 on page 36.

Function syntax

►► LANGUAGE⁽¹⁾ (—*handle*—) ◀◀

Notes:

- 1 Returns a language value, type VARCHAR(30).

►► LANGUAGE⁽¹⁾ (—*handle*—, —*language*—) ◀◀

Notes:

- 1 Returns a handle, type DB2TEXTH.

Function parameters

handle

An expression whose result is a value of type DB2TEXTH. It is usually the name of a handle column from which the language setting is to be returned or set.

language

The new document language setting of data type VARCHAR(30).

If *language* is specified, this document language is set in the handle; the handle is returned instead of the language setting.

NO_OF_MATCHES

NO_OF_MATCHES can search in text documents and return an INTEGER value indicating how many matches resulted per document.

Function syntax

►►—NO_OF_MATCHES—(*—handle—*,*—search-argument—*)—►►

Function parameters

handle

An expression whose result is a value of type DB2TEXT. It is usually the name of a handle column containing the handles of the text documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See Chapter 11, “Syntax of search arguments,” on page 121.

RANK

RANK can search in text documents and return a rank value for each document found, indicating how well the found document is described by the search argument.

RANK returns an DOUBLE value between 0 and 1. The rank value is absolute, indicating how well the found document satisfies the search criteria in relation to other found documents. The value indicates the number of matches found in the document in relation to the document's size.

Function syntax

►►—RANK—(—*handle*—,—*search-argument*—)—————►►

Function parameters

handle

An expression whose result is a value of type DB2TEXTH. It is usually the name of a handle column containing the handles of the text documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See Chapter 11, "Syntax of search arguments," on page 121.

REFINE

The REFINE function takes two search arguments and returns a combined search argument of type LONG VARCHAR, consisting of the two original search arguments connected by the Boolean operator AND.

Function syntax

►►—REFINE—(—*search-argument*—,—*search-argument*—)—————►◄

Function parameters

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See Chapter 11, “Syntax of search arguments,” on page 121.

The search argument must not contain the search parameters IS ABOUT, THESAURUS, or EXPAND.

SEARCH_RESULT

The SEARCH_RESULT function returns the result of a search in an intermediate table. This function can be used in a FROM clause of an SQL statement.

The returned table has the following structure:

Column Name	Data Type
HANDLE	DB2TX.DB2TEXT
NUMBER_OF_MATCHES	INTEGER
RANK	DOUBLE

Values are generated only for the selected columns of the intermediate table. Select count(*) generates the HANDLE column only. Because the calculation of the rank values consumes a lot of system resources, you should not select the rank value from the intermediate table if the rank value is not required.

This function is faster than CONTAINS or RANK when processing large tables.

Function syntax

►►—SEARCH_RESULT—(—*handle*—,—*search-argument*—)—————►◄

Function parameters

handle

The name of a handle column that corresponds to the column containing the documents to be searched.

search-argument

A string of type LONG VARCHAR containing the terms to be searched for. See Chapter 11, “Syntax of search arguments,” on page 121.

Examples

For an example, refer to “Improving search performance” on page 62.

Chapter 11. Syntax of search arguments

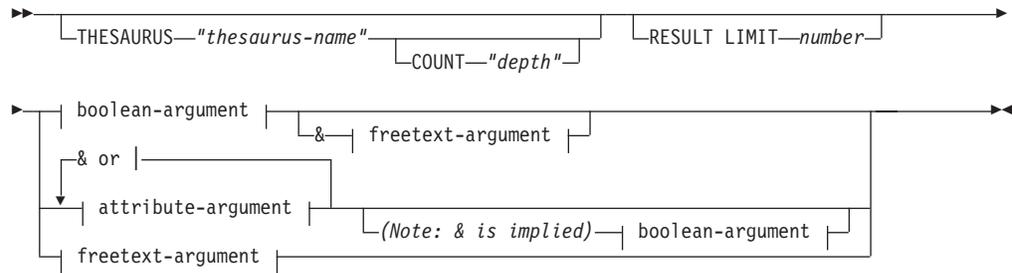
A search argument is the condition that you specify when searching for terms in text documents. It consists of one or several search terms and search parameters.

The DB2 Text Extender functions that use search arguments are:

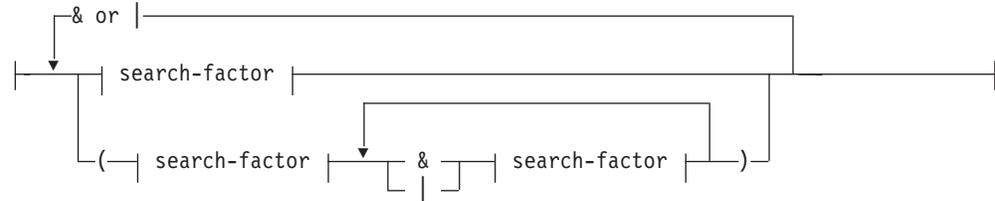
- **CONTAINS**. This function uses a search argument to search for text in a particular text document. It returns the **INTEGER** value 1 if the document contains the text. Otherwise, it returns 0.
- **NO_OF_MATCHES**. This function uses a search argument to search in text documents. It returns an **INTEGER** value indicating how many matches resulted per document.
- **RANK**. This function uses a search argument to search in text documents. It returns a value for each document found, indicating how well the found document is described by the search argument.
- **REFINE**. This function takes two search arguments and returns a combined search argument of type **LONG VARCHAR**, consisting of the two original search arguments connected by the Boolean operator **AND**.
- **SEARCH_RESULT**. This function returns the result of a search in an intermediate table which contains rank, number of matches, and the handle.

Search argument

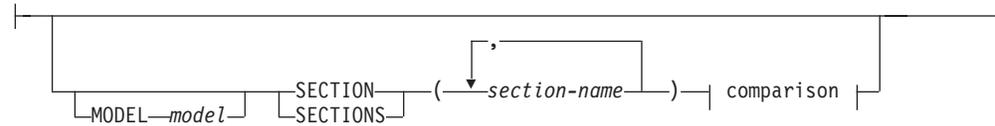
Search argument syntax



boolean-argument:



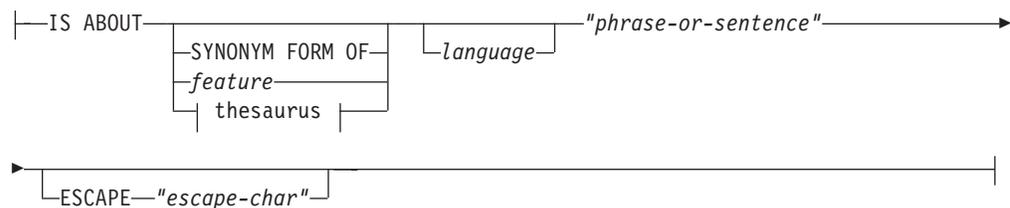
attribute-argument:



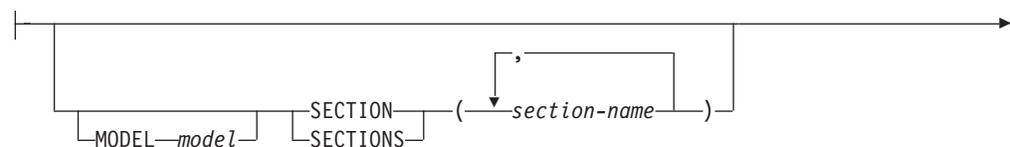
Comparison:

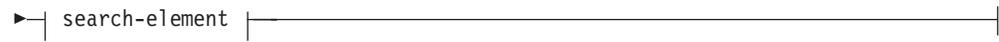


freetext-argument:

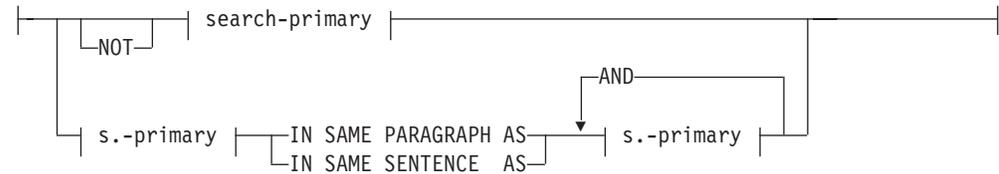


search-factor:





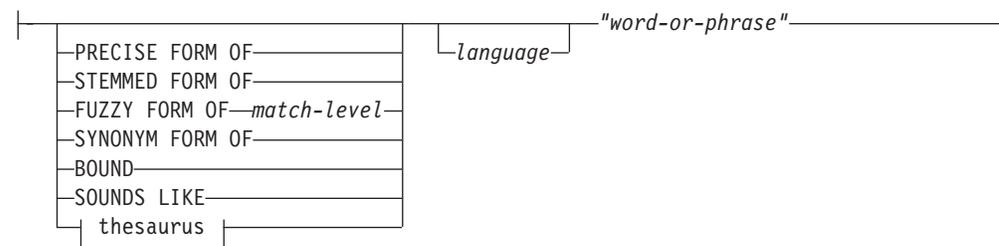
search-element:



search-phrase:



search-atom:



thesaurus (if THESAURUS is specified):



Examples

Examples are given in “Specifying search arguments” on page 49.

Search parameters

IS ABOUT

An option that lets you specify a free-text search argument, that is, a natural-language phrase or sentence that describes the concept to be found. See “Free-text and hybrid search” on page 56.

MODEL model

A keyword used to specify the name of the document model to be used in the search term. The document model describes the structure of documents that contain identifiable sections so that the content of these sections can be searched individually.

Syntax of search arguments

The model name must be specified in a document models file described in “Working with structured documents (section support)” on page 34. The model name can be masked using wildcard characters.

If you do not specify a model, the default model specified during index creation is used.

SECTION(S) *section-name*

A keyword used to specify one or more sections that the search is to be restricted to. The section name must be specified in a model in a document models file, described in “Working with structured documents (section support)” on page 34. A section name can be masked using wildcard characters % and _. Note that for Ngram indexes, the wild card search is only allowed for section names that are not nested.

Restrictions: Searching in nested sections is possible only for documents stored in columns enabled with format XML. For Ngram indexes, only one section name can be searched and XML format is not supported. Note that XML support is only available on OS/390 release 2.9 (or later), and z/OS releases, which include Text Search Engine release 4 (FMID HIMN230).

attribute-value

A value used together with a preceding comparison operator for the attributes listed in the preceding section list. A query requesting an attribute's value to be within a certain range can use two comparison operators within one attribute condition.

A combination of operators using the same kind of comparison, like \geq in the first and $>$ in the second) of the same condition is invalid. Specification of two comparisons with = operator is also invalid.

- = Requests an equality comparison of the attribute in the indexed document with the following attribute value.
- \geq Requests a “greater than or equal to” comparison of the attribute in the indexed document with the following attribute value.
- $>$ Requests a “greater than” comparison of the attribute in the indexed document with the following attribute value.
- \leq Requests a “less than or equal to” comparison of the attribute in the indexed document with the following attribute value.
- $<$ Requests a “less than” comparison of the attribute in the indexed document with the following attribute value.

Restrictions: Searching in nested sections is possible only for documents stored in columns enabled with format XML. For Ngram indexes, only one section name can be searched and XML format is not supported. Note that XML support is only available on OS/390 release 2.9 (or later), and z/OS releases, which include Text Search Engine release 4 (FMID HIMN230).

THESAURUS *thesaurus-name*

A keyword used to specify the name of the thesaurus to be used to expand the search term. The thesaurus name is the file name (without its extension) of a thesaurus that has been compiled using the thesaurus compiler IMOTHESC or IMOTHESN. There are default thesauri desthes and desnthes, stored in the sample directory, where desnthes is an Ngram thesaurus. You can also specify the file's path name. The default path name is the dictionary path.

COUNT *depth*

A keyword used to specify the number of levels (the depth) of terms in the thesaurus that are to be used to expand the search term for the given relation. If you do not specify this keyword, a count of 1 is assumed.

RESULT LIMIT *number*

A keyword used to specify the maximum number of entries to be returned in the result list. *number* is a value from 1 to 32767. If a free-text search is used, the search result list is ranked only with respect to the complete search result list. Otherwise, the limited search result is ranked only from the entries of the list.

EXPAND *relation*

A keyword used to specify the relation, such as INSTANCE, between the search term specified in TERM OF and the thesaurus terms to be used to expand the search term. The relation name must correspond to a relation used in the thesaurus. See “Thesaurus concepts” on page 138.

TERM OF *"word-or-phrase"*

The search term, or multi-word search term, to which other search terms are to be added from the thesaurus.

search-factor

An operand that can be combined with other operands to form a search argument. The evaluation order is from left to right. The logical AND (&) operator binds stronger than the logical OR (|) operator. Example:

```
"passenger" & "vehicle" | "transport" & "public"
```

is evaluated as:

```
("passenger" & "vehicle") | ("transport" & "public")
```

To search for:

```
"passenger" & ("vehicle" | "transport") & "public"
```

you must include the parentheses as shown.

NOT search-primary

An operator that lets you exclude text documents from your search that contain a particular term.

When NOT is used in a search factor, you cannot use the SYNONYM FORM OF keyword.

search-primary IN SAME PARAGRAPH AS search-primary

A keyword that lets you search for a combination of terms occurring in the same paragraph.

The following search argument finds text documents containing the term “traffic” only if the term “air” is in the same paragraph.

```
"traffic" IN SAME PARAGRAPH AS "air"
```

You cannot use the IN SAME PARAGRAPH AS keyword when NOT is used in a search factor.

search-primary IN SAME SENTENCE AS search-primary

A keyword that lets you search for a combination of terms occurring in the same sentence. Similar to IN SAME PARAGRAPH AS.

Syntax of search arguments

AND search-primary

A keyword that lets you combine several search-primaries to be searched for in the same sentence or the same paragraph.

The following search argument searches for “forest”, “rain”, “erosion”, and “land” in the same sentence.

```
"forest" IN SAME SENTENCE AS "rain" AND "erosion" AND "land"
```

search-atom

If you connect a series of search atoms by commas, then a search is successful if a term in any one of the search atoms is found. Each search atom must contain at least a word or a phrase.

The following statement is true if one or more of the search arguments is found.

```
CONTAINS (mytexthandle, '( "text",  
                           "graphic",  
                           "audio",  
                           "video")') = 1
```

PRECISE FORM OF, STEMMED FORM OF, FUZZY FORM OF, SYNONYM FORM OF, BOUND

Table 9 on page 127 shows the options that correspond to the various types of index. For example, for a linguistic index, any of the options are suitable except for PRECISE FORM OF. If you specify PRECISE FORM OF, it is ignored and the default value is taken.

The search term processing is described in more detail in Table 10 on page 127.

Table 9. Linguistic index options

Search atom keyword	Index type				
	Linguistic	Precise	Precise Normalized	Ngram	Ngram case-enabled
PRECISE FORM OF		X	X		O
STEMMED FORM OF	X			O	O
FUZZY FORM OF				O	O
IS ABOUT	O	O	O		
SYNONYM FORM OF	O	O	O		
EXPAND	O	O	O		
SOUNDS LIKE	O	O	O		
IN SAME SENTENCE AS	O	O	O	O	O
IN SAME PARAGRAPH AS	O	O	O	O	O
BOUND				O	O

X=default setting O=function available

Table 10. Search term options for Ngram indexes

Search atom keyword	Search term processing				
	Case		Stemming	Match	
	Sensitive	Insensitive		Exact	Fuzzy
PRECISE FORM OF	when case-enabled	X		X	
STEMMED FORM OF		X	X		
FUZZY FORM OF		X			X

X=default setting

If you use a keyword that is not available for that index type, it is ignored and either the default keyword is used instead, or a message is returned.

PRECISE FORM OF

A keyword that causes the word (or each word in the phrase) following PRECISE FORM OF to be searched for exactly as typed, rather than being first reduced to its stem form. For precise indexes, this form of search is case-sensitive; that is, the use of upper- and lowercase letters is significant. For example, if you search for mouse you do not find “Mouse”.

This is the default option for precise indexes. For a precise normalized index, the default form of search is not case-sensitive. If you specify this keyword for a linguistic index, it is ignored and STEMMED FORM OF is assumed.

STEMMED FORM OF

A keyword that causes the word (or each word in the phrase) following STEMMED FORM OF to be reduced to its word stem before the search is carried out. This form of search is not case-sensitive. For example, if you search for mouse you find “Mouse”.

Syntax of search arguments

The way in which words are reduced to their stem form is language-dependent.

Example: programming computer systems is replaced by program compute system when you use the US-English dictionary, and by programme compute system when you use the UK-English dictionary.

This search phrase can find “programmer computes system”, “program computing systems”, “programming computer system”, and so on.

This is the default option for linguistic indexes. If you specify this keyword for a precise index, it is ignored and PRECISE FORM OF is assumed instead.

FUZZY FORM OF

A keyword for making a “fuzzy” search, which is a search for terms that have a similar spelling to the search term. This is particularly useful when searching in documents that were created by an Optical Character Recognition (OCR) program. Such documents often include misspelled words. For example, the word economy could be recognized by an OCR program as econony.

match-level: An integer from 1 to 5 specifying the degree of similarity, where 5 is more similar than 1.

SYNONYM FORM OF

A keyword that causes the word or phrase following SYNONYM FORM OF to be searched for together with its synonyms. The synonyms are provided by the dictionary specified by *language* or else by the default dictionary.

Synonyms for a phrase are alternative phrases containing all the possible combinations of synonyms that can be obtained by replacing each word of the original phrase by one of its synonyms. The word sequence remains as in the original phrase.

If you specify this keyword for a precise index, it is ignored and PRECISE FORM OF is assumed instead.

You cannot specify this keyword when NOT is used in the search factor, or when the word or phrase to be searched for contains masking characters.

BOUND

A keyword for searching in documents that use the Korean CCSID. It causes the search to respect word phrase boundaries. If *language* is specified, it is ignored; Korean is assumed.

language

A variable that determines which dictionary is used in linguistic processing of text documents during indexing and retrieval. This applies not only to linguistic indexes, but also to precise indexes because these use a dictionary to process stop words.

Linguistic processing includes synonym processing and word-stem processing.

The supported languages are listed in Table 1 on page 36.

Note: When searching in documents that are not in U.S. English, you must specify the language in the search argument *regardless of the default language*.

"word-or-phrase"

A word or phrase to be searched for. The characters that can be used within a word are language-dependent. It is also language-dependent whether words need to be separated by separator characters. For English and most other languages, each word in a phrase must be separated by a blank character.

Precise or linguistic search. DB2 Text Extender can search using either the precise form of the word or phrase, or a variation of it. If you do not specify one of the options in Table 9 on page 127, the default linguistic options are used according to which type of index is being used.

To search for a character string that contains double quotation marks, type the double quotation marks twice. For example, to search for the text "wildcard" character, use:

```
""wildcard"" character"
```

Syntax of search arguments

Masking characters. A word can contain the following masking characters:

_ (underscore)

Represents any single character.

% (percent)

Represents any number of arbitrary characters. If a word consists of a single %, then it represents an optional word of any length.

A word cannot be composed exclusively of masking characters, except when a single % is used to represent an optional word.

If you use a masking character, you cannot use SYNONYM FORM OF, *feature*, or THESAURUS.

You cannot substitute non-alphabetic characters for masking characters if your index type is Ngram.

ESCAPE "*escape-character*"

A character that identifies the next character as one to be searched for and not as one to be used as a masking character.

Example: If *escape-character* is \$, then \$%, \$_, and \$\$ represent %, _, and \$ respectively. Any % and _ characters not preceded by \$ represent masking characters. Also note that the escape character can only be a single character.

Summary of rules and restrictions:

Boolean operations

NOT is not allowed after OR.

FUZZY FORM OF

The first 3 characters must match. Cannot be used if a word in the search atom contains a masking character. Cannot be used in combination with NOT. Can be used only with an Ngram index.

IN SAME PARAGRAPH AS

Cannot be used if NOT is used in a search factor.

IN SAME SENTENCE AS

Cannot be used if NOT is used in a search factor.

Linguistic index

Prevents the use of PRECISE FORM OF. Takes as default STEMMED FORM OF. Masking characters can be used. Searches are case-insensitive.

Masking character

Prevents the use of SYNONYM FORM OF, and THESAURUS.

Ngram index

Masking characters can be used, although not following a non-alphanumeric character. Searches are case-insensitive unless the index is case-enabled and PRECISE FORM OF is used.

NOT Prevents the use of SYNONYM FORM OF, IN SAME PARAGRAPH AS, and IN SAME SENTENCE AS.

PRECISE FORM OF

Ignored for a linguistic index.

Precise index

Prevents the use of **STEMMED FORM OF**, and **SYNONYM FORM OF**. Takes as default **PRECISE FORM OF**. Masking characters can be used. Searches are case-sensitive.

STEMMED FORM OF

Ignored for a precise index, but available for a normalized precise index containing English documents.

SYNONYM FORM OF

Cannot be used if a word in the search atom contains a masking character. Cannot be used in combination with **NOT**. Cannot be used with a precise index.

Syntax of search arguments

Chapter 12. Linguistic processing for linguistic and precise indexes

DB2 Text Extender offers linguistic processing in these areas of retrieval:

- **Indexing.** When DB2 Text Extender analyzes documents to extract the terms to be stored in the text index, the text is processed linguistically to extract the right terms for the index. This is done to make retrieval as simple and as fast as possible.
- **Retrieval.** When DB2 Text Extender searches through the document index to find the documents that contain occurrences of the search terms you have specified, the search terms are also processed linguistically to match them with the indexed terms.

Linguistic processing when indexing

When DB2 Text Extender indexes and retrieves documents, it makes a linguistic analysis of the text. As you can see from the following table, the amount of linguistic processing depends on the index type. For Ngram indexes, no linguistic processing is applied.

The linguistic processing used for indexing documents consists of:

- Basic text analysis
 - Recognizing terms (tokenization)
 - Normalizing terms to a standard form
 - Recognizing sentences
- Reducing terms to their base form
- Stop-word filtering
- Decomposition (splitting compound terms).

Table 11 shows a summary of how terms are indexed when the index type is **linguistic** and no additional index properties have been requested.

Table 11. Term extraction for a linguistic index

Document text	Term in index	Linguistic processing
Mouse Käfer	mouse kaefer	Basic text analysis (normalization)
mice swum	mouse swim	Reduction to base form
system-based Wetterbericht	system-based, system base wetterbericht, wetter bericht	Decomposition
a report on animals	report animal	Stop-word filtering. Stop words are: a, on

By comparison, Table 12 on page 134 shows a summary of how terms are indexed when the index type is **precise**.

Linguistic processing when indexing

Table 12. Term extraction for a precise index

Document text	Term in index	Linguistic processing
Mouse Käfer	Mouse Käfer	No normalization
mice swum	mice swum	No reduction to base form
a report on animals	report animals	Stop-word filtering. Stop words are: a, on
system-based Wetterbericht	system-based Wetterbericht	No decomposition

Basic text analysis

DB2 Text Extender processes basic text analysis without using an electronic dictionary.

Recognizing terms that contain nonalphanumeric characters

When documents are indexed, terms are recognized even when they contain nonalphanumeric characters, for example: "\$14,225.23", "mother-in-law", and "10/22/90".

The following are regarded as part of a term:

- Accents
- Currency signs
- Number separator characters (like "/" or ".")
- The "@" character in e-mail addresses (English only)
- The "+" sign.

Language-specific rules are also used to recognize terms containing:

- Accented prefixes in Roman languages, such as l'aventure in French.
- National formats for dates, time, and numbers.
- Alternatives, such as mission/responsibility, indicated in English using the "/" character.
- Trailing apostrophes in Italian words like securita'. It is usual in typed Italian text, when the character set does not include characters with accents, to type the accent *after* the character; for example, "à" is typed "a'".

Normalizing terms to a standard form

Normalizing reduces mixed-case terms, and terms containing accented or special characters, to a standard form. This is done by default when the index type is linguistic. (In a precise index the case of letters is left unchanged—searches are case-sensitive.)

For example, the term Computer is indexed as computer, the uppercase letter is changed to lowercase. A search for the term computer finds occurrences not only of computer, but also of Computer. The effect of normalization during indexing is that terms are indexed in the same way, regardless of how they are capitalized in the document.

Normalization is applied not only during indexing, but also during retrieval. Uppercase characters in a search term are changed to lowercase before the search is made. When your search term is, for example, Computer, the term used in the search is computer.

Accented and special characters are normalized in a similar way. Any variation of école, such as École, finds école, Ecole, and so on. Bürger finds buerger, Maße finds masse.

If the search term includes masking (wildcard) characters, normalization is done before the masking characters are processed. Example: Bür_er becomes buer_er.

Recognizing sentences

You can search for terms that occur in the same sentence. To make this possible, each document is analyzed during indexing to find out where each sentence ends.

DB2 Text Extender offers one type of sentence-end recognition:

- Universal Unicode Tokenizer for languages other than Arabic and Hebrew.

This is the simpler, but faster method. The tokenizer looks for a period, exclamation or question mark, preceded by a token character, such as a letter, and followed by a blank, tab, or new-line character. To check that this is really the end of a sentence and not just an abbreviation ending in a period, a language-specific list of abbreviations is checked.

Reducing terms to their base form (lemmatization)

In a linguistic index, you can search for mouse, for example, and find mice. Terms are reduced to their base form for indexing; the term mice is indexed as mouse. Later, when you use the search term mouse, the document is found. The document is found also if you search for mice.

The effect is that you find documents containing information about mice, regardless of which variation of the term mouse occurs in the document, or is used as a search term.

In the same way, conjugated verbs are reduced to their infinitive; bought, for example, becomes buy.

Stop-word filtering

Stop words are words such as prepositions and pronouns that occur very frequently in documents, and are therefore not suitable as search terms. Such words are in a stop-word list associated with each dictionary, and are excluded from the indexing process.

Stop word processing is case-insensitive. So a stop word about also excludes the first word in a sentence About. This is The stop word lists, supplied in various languages, can be modified.

An Ngram index does not have a stop word list.

Decomposition (splitting compound terms)

Germanic languages, such as German or Dutch, are rich in compound terms, like Versandetiketten, which means mail (Versand) labels (Etiketten). Such compound terms can be split into their components.

Linguistic processing when indexing

For a precise index, compound terms are indexed unchanged as one word. For a linguistic index, compound terms are split during indexing. When you search, compound terms are split if you have a linguistic index.

The components are found if they occur in any sequence in a document as long as they are contained within one sentence. For example, when searching for the German word *Wetterbericht* (weather report), a document containing the phrase *Bericht über das Wetter* (report about the weather) would also be found.

An attempt is made to split a term if:

- The term's language uses compound terms
- The term has a certain minimum length
- The term is not itself an entry in the electronic dictionary—compounds that are commonly used like the German word *Geschäftsbericht* (business report) are in the German dictionary.

If a split is found to be possible, the term's component parts are then reduced to their base form. Here are some examples from Danish, German, and Dutch:

Table 13. The component parts of compound terms

Compound term	Component parts
børsmæglerselskab	børsmæglerselskab børs mægler selskab
Kindersprachen	kindersprache kind sprache
probleemkinderen	probleemkinderen probleemkind kind probleem

Linguistic processing for retrieval

Query processing aims at making search terms weaker so that the recall rate of searches is increased, that is, more relevant documents are found. There are two basic operations on query terms to achieve that goal; they are expansions and reductions. In addition, some search term operations involve both expansion and reduction.

- Expansions take a word or a multi-word term from within a search term and associate it with a set of alternative search terms, each of which may be a multi-word term itself. The source expression and the set of target expressions form a Boolean OR-expression in DB2 Text Extender's query language. As expansions leave the source term unchanged, they are to some extent independent of the index type. The following are expansion operations:
 - Synonym expansion
 - Thesaurus expansion
- Reductions change the search term to a form that is more general than the one specified by the user. Because it changes the search term, reductions are dependent on the index type to ensure that the changed term matches. Therefore, DB2 Text Extender derives reduction information from the type of those indices or index that the query is directed against. The following are reductions:

Lemmatization (see “Reducing terms to their base form (lemmatization)” on page 135)

Normalization (see “Normalizing terms to a standard form” on page 134).

Stop words (see “Stop-word filtering” on page 135).

- Some operations both change the search term and expand it with a set of alternative terms. Due to the inherent reduction, these again depend on information contained in the index. The following operations fall into this class:

Character and word masking

Sound expansion.

Synonyms

Synonyms are semantically related words. Usually, these words have the same word class or classes (such as noun, verb, and so on) as the source term.

Synonyms are obtained from a separate file for each language. They are always returned in base form and, up to a few exceptions, are not multi-word terms. Search term words are always reduced to their base form when looking up synonyms. Here are some examples of a word’s synonyms in three languages:

- English

word:

comment remark statement utterance term expression
communication message assurance guarantee warrant bidding command
charge commandment dictate direction directive injunction instruction
mandate order news advice intelligence tidings gossip buzz cry
hearsay murmur report rumor scuttlebutt tattle tittle-tattle
whispering

- French

mot:

expression parole terme vocable lettre billet missive épître
plaisanterie

- German

Wort:

Vokabel Bezeichnung Benennung Ausdruck Begriff Terminus
Ehrenwort Brocken Bekräftigung Versprechen Zusicherung Gelöbnis
Beteuerung Manneswort Schwur Eid Ausspruch

Thesaurus expansion

A search term can be expanded using thesaurus terms that can be reached through a specific relation. These relations may be hierarchical (such as the “Narrower term” relation), associative (such as a “Related term” relationship), or it may be a synonym relationship. A thesaurus term may be, and often is, a multi-word term.

“Thesaurus concepts” on page 138 describes thesaurus expansion in more detail.

The search term (start term) is not normalized when the thesaurus lookup is done. The words that result from the thesaurus lookup are reduced to their base form according to the index type.

Sound expansion

Sound expansion expands single words through a set of similarly sounding words. It is particularly useful whenever the exact spelling of a term to be searched is not known.

Character and word masking

Masking is a non-linguistic expansion technique, where a regular expression is replaced with the disjunction of all indexed words that satisfy it. Neither a masked expression nor any of its expansions is subject to lemmatization, stop-word extraction, or any of the other expansion techniques. This may have the effect that, for example, an irregular verb form like *swum*, when searched with the masked term *swu**, is matched on a precise index, but not on a linguistic index, where this form has been lemmatized to become *swim*.

If you use word masking, performance can be slow, especially when searching in large indexes.

Thesaurus concepts

A thesaurus is a controlled vocabulary of semantically related terms that usually covers a specific subject area. It can be visualized as a semantic network where each term is represented by a node. If two terms are related to each other, their nodes are connected by a link labeled with the relation name. All terms that are directly related to a given term can be reached by following all connections that leave its node. Further related terms can be reached by iteratively following all connections leaving the nodes reached in the previous step. Figure 13 shows an example of the structure of a very small thesaurus.

Figure 13 displays a thesaurus as a network.

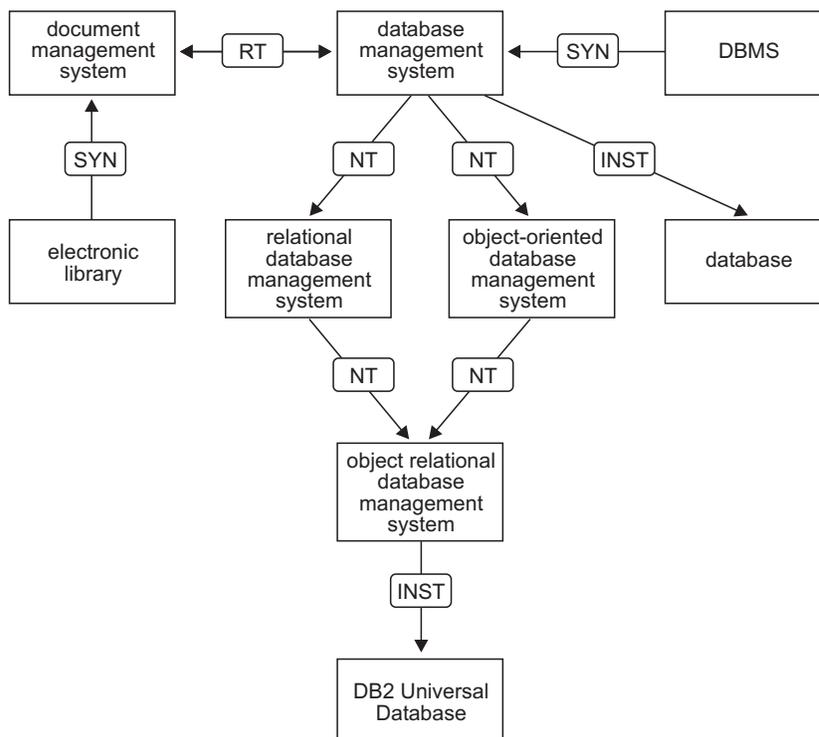


Figure 13. A thesaurus displayed as a network

DB2 Text Extender lets you expand a search term by adding additional terms from a thesaurus that you have previously created. Refer to Chapter 11, “Syntax of search arguments,” on page 121 to find out how to use thesaurus expansion in a query.

To create a thesaurus for using it in a search application requires a thesaurus definition file that has to be compiled into an internal format, the thesaurus dictionary.

The basic components of a thesaurus are “terms” and “relations”.

Terms

A term is a word or expression denoting a concept within the subject domain of the thesaurus. For example, the following could be terms in one or more thesauri:

```
data processing
helicopter
gross national product
```

In a DB2 Text Extender thesaurus, terms are classified as either descriptors or nondescriptors. A *descriptor* is a term in a class of synonyms that is the preferred term for indexing and searching. The other terms in the class are called *nondescriptors*. For example, outline and shape are synonymous, where shape could be the descriptor and outline a nondescriptor.

An Ngram thesaurus does not distinguish between descriptors and nondescriptors.

Relations

A relation is an expression of an association between two terms. Relations have the following properties:

- The *depth* of a relation is the number of levels over which the relation extends. This is specified in the search syntax using the THESDEPTH keyword. Refer to Chapter 11, “Syntax of search arguments,” on page 121 to find out how to use thesaurus expansion in a query.
- The *directionality* of a relation specifies whether the relation is true equally from one term to the other (bidirectional), or in one direction only (unidirectional).

Thesaurus expansion can use every relation defined in the thesaurus. You can also specify the depth of the expansion. This is the maximum number of transitions from a source term to a target term. Note however that the term set may increase exponentially as the depth is incremented.

The following example shows those terms that are newly added as the depth increases.

```
health
```

```
health service, paramedical, medicine, illness
```

```
allergology, virology, veterinary medicine, toxicology, surgery,
stomatology, rheumatology, radiotherapy, psychiatry, preventive
medicine, pathology, odontology, nutrition, nuclear medicine,
neurology, nephrology, medical check up, industrial medicine,
hematology, general medicine, epidemiology, clinical trial,
cardiology, cancerology
```

DB2 Text Extender thesaurus relations

These are the relation types provided by a DB2 Text Extender thesaurus:

- Associative
- Synonymous
- Hierarchical
- Other

Thesaurus concepts

In a DB2 Text Extender thesaurus there are no predefined relations. You can give each relation a name, such as BROADER TERM, which can be a mnemonic abbreviation, such as BT. The common relations used in thesaurus design are:

- BT or BROADER TERM
- NT or NARROWER TERM
- RT or RELATED TERM
- SYN or SYNONYM
- USE
- UF or USE FOR

Associative: An associative relation is a bidirectional relation between descriptors, extending to any depth. It binds two terms that are neither equivalent nor hierarchical, yet are semantically associated to such an extent that the link between them may suggest additional terms for use in indexing or retrieval.

Associative relations are commonly designated as RT (related term). Examples are:

dog RT security
pet RT veterinarian

Synonymous: When a distinction is made between descriptors and nondescriptors, as it is in a DB2 Text Extender thesaurus, the synonymous relation is unidirectional between two terms that have the same or similar meaning. In a class of synonyms, one of the terms is designated as the descriptor. The other terms are then called nondescriptors.

The common designation USE leads from a given nondescriptor to its descriptor. The common designation USE FOR leads from the descriptor to each nondescriptor. For example:

feline USE cat
lawyer UF advocate

Hierarchical: A hierarchical relation is a unidirectional relation between descriptors that states that one of the terms is more specific, or less general, than the other. This difference leads to representation of the terms as a hierarchy, where one term represents a class, and subordinate terms refer to its member parts. For example, the term "mouse" belongs to the class "rodent".

BROADER TERM and NARROWER TERM are hierarchical relations. For example:

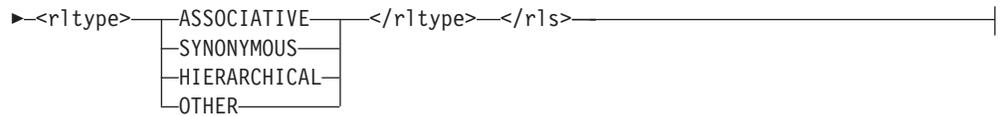
car NT limousine
equine BT horse

Other: A relation of type *other* is the most general. It represents an association that does not easily fall into one of the other categories. A relation of type *other* can be bidirectional or unidirectional, there is no depth restriction, and relations can exist between descriptors and nondescriptors.

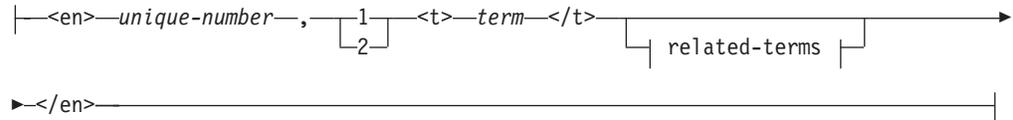
This relation is often used for new terms in a thesaurus until the proper relation with other terms can be determined.

Of course you can define your own bidirectional synonymous relation by using the relation type *associative* for a synonymous relation between descriptors or even with the relation type *other* for a synonymous relation between arbitrary terms.

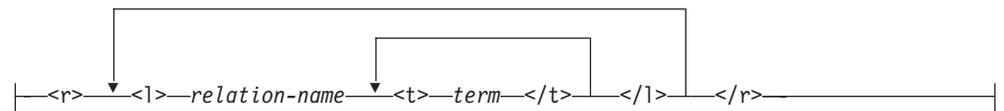
Thesaurus concepts



thesaurus-entry:



related-terms:



relation-name can contain only the characters a-z, A-Z, and 0-9.

Figure 14 on page 143 shows the SGML definition of the thesaurus shown in Figure 13 on page 138.

```

<thesaurus>
<header>
<thname>thesc example thesaurus</thname>
<rldef>

<rls>
<rlname>Related Term</rlname>
<rltype>associative</rltype>
</rls>

<rls>
<rlname>Narrower Term</rlname>
<rltype>hierarchical</rltype>
</rls>

<rls>
<rlname>Instance</rlname>
<rltype>hierarchical</rltype>
</rls>

<rls>
<rlname>Synonym</rlname>
<rltype>synonymous</rltype>
</rls>
</rldef>
</header>

<en> 2, 1
<t>database management system</t>
<r>
  <l>Narrower Term
  <t>oo database management system</t>
  <t>relational database management system</t>
  </l>

  <l>Synonym
  <t>DBMS</t>
  </l>

  <l>Related Term
  <t>document management system</t>
  </l>

  <l>Instance
  <t>database</t>
  </l>
</r>
</en>

```

Figure 14. The definition of a simple thesaurus (Part 1 of 2)

```

<en> 5, 1
<t> relational database management system </t>
<r>
  <l>Narrower Term
  <t>object relational database management system</t>
  </l>
</r>
</en>

<en> 3, 1
<t>object relational database management system</t>
<r>
  <l>Instance
  <t>DB2 Universal Database</t>
  </l>
</r>
</en>

<en> 6, 1
<t>object oriented database management system</t>
<r>
  <l>Narrower Term
  <t>object relational database management system</t>
  </l>
</r>
</en>

<en> 4, 1
<t>document management system</t>
<r>
  <l>Synonym
  <t>library</t>
  </l>
</r>
</en>

<en> 9, 1
<t>library</t>
</en>

<en> 10, 1
<t>DB2 Universal Database</t>
</en>

<en> 11, 1
<t>database</t>
</en>
</thesaurus>

```

Figure 14. The definition of a simple thesaurus (Part 2 of 2)

Chapter 13. Configuration files

This chapter describes the configuration files. These files are automatically generated when a DB2 Text Extender instance is created with the `descust` command, or when a client profile is created with the `descrc1` command. These files are generated in code page 500.

You can edit these files to tune your system, however, ensure that you use the correct code page when editing the files. The section names and the option names are case independent. A semicolon is used as a comment delimiter.

Where the option is a Boolean value, the values `TRUE`, `YES`, `ON`, and `1` are considered true regardless of their case; all other values are considered false.

Note

The client and server configurations in this chapter refer to the configuration of the Text Search Engine application on the z/OS platform.

Client configuration file

File name imocl.ini

Location HFS: DB2TX_INSTOWNERHOMEDIR/db2tx

The updated options become active at the next `StartSession` function.

Table 14. Update options in a client configuration file

Section	Option	Default value	Description
[INSTANCE]	IMOWORKCL		Points to a working directory that is used for temporary files.
	IMONLPSCCL		Points to the resource directory.
	IMODTDPATHSRV		Points to the directory where the DTDs for the XML documents are stored.
	IMODTDPATHCL		Points to the path where DTD files must be stored. Used for XML files.
[BUFFER]	BUFFERSEGMENTSZIE	32 000	The size of the block segments, in bytes, used for buffering.
	BUFFERSEGMENTCOUNT	3	The maximum number of segments used before buffers are swapped to temporary files. A buffer segment is defined by <code>BUFFERSEGMENTSZIE</code> .

Client configuration file

Table 14. Update options in a client configuration file (continued)

Section	Option	Default value	Description
[DOCUMENTFORMAT]	USEREXIT		The name of the user exit used to work with unsupported document formats. Specify either a file name if the user exit is stored in a directory which is part of the PATH statement, or a fully-qualified file name. For further information on the user exit for format conversion, see "Using unsupported document formats" on page 23.
	FORMATRECOGNITION	TRUE	Triggers the format recognition of the document formats. TRUE: format recognition is on FALSE: format recognition is off
	UseExitForAllFormats	FALSE	Determines when the user exit for working with document formats not listed in deslsdef.h is called. You must set a value for USEREXIT. TRUE: the user exit is always called. If this value is set, FORMATRECOGNITION is ignored. FALSE: call the user exit for all document formats above the value of EHW_USER_FORMATS.

Server configuration file

File name imosrv.ini

Location HFS: DB2TX_INSTOWNERHOMEDIR/db2tx/TXINSnnn

The updated options become active the next time the server instance is started.

Table 15. Server configuration file options

Section	Option	Default value	Description
[INSTANCE]	IMOWORKSRV		Points to a working directory that is used for temporary files.
	IMONLPSSRV		Points to the resource directory.
[DAEMON]	MaxMtEntries	30	The maximum number of indexes used in parallel at any one time. Decrease this number if you are short of resources, such as semaphores or shared memories. Available resources are platform dependent and therefore the default values are also platform dependent.

Table 15. Server configuration file options (continued)

Section	Option	Default value	Description
	MaxIndexEntries	1000	The maximum number of indexes used. Decrease this number if you are short of resources, such as shared memories.
[BUFFER]	BUFFERSEGMENTSZIE	32 000	The size of the block segments, in bytes, used for buffering. This is used by EhwUpdate.
	BUFFERSEGMENTCOUNT	3	The maximum number of segments used during the index update process before buffers are swapped to temporary files. Increase this number if your document collections contain large documents.
	BUFFERSORTSIZE	4 000 000	The size of the buffer, in bytes, used for sorting temporary work files.
[DOCUMENTFORMAT]	USEREXIT		The name of the user exit used to work with document formats not listed in deslsdef.h. Specify either a file name if the user exit is stored in a directory which is part of the PATH statement, or a fully-qualified file name. For further information on the user exit for format conversion, see “Using unsupported document formats” on page 23.
	FORMATRECOGNITION	TRUE	Triggers the format recognition of the document formats listed in deslsdef.h. TRUE: format recognition is on FALSE: format recognition is off
	UseExitForAllFormats	FALSE	Determines when the user exit for working with document formats not listed in deslsdef.h is called. You must set a value for USEREXIT. TRUE: the user exit is always called. If this value is set, FORMATRECOGNITION is ignored. FALSE: call the user exit for all document formats above the value of EHW_USER_FORMATS.
[LINGPREC] For all indexes with linguistic or precise as their base type.	UPDATETHRESHOLD	4 000 000	An index update process is split internally into several update and reorganization runs. This value specifies the number of words to be collected in one update step.

Server configuration file

Table 15. Server configuration file options (continued)

Section	Option	Default value	Description
	UPDATESLICE	1	The number of update runs that take place before an internal reorganization process starts. An update run is defined by the UPDATETHRESHOLD.
[NGRAM] For all indexes with DBCS support.	UPDATETHRESHOLD	10 000 000	Total size, in bytes, of documents added to an index during one update run. If the threshold is exceeded, a reorganization process is automatically started.
	UPDATESLICE	10 000	The maximum number of documents in a secondary index. This number is checked after each update run. If the number of documents is greater than this value, a reorganization process is automatically started.

Chapter 14. Return codes

RC_ALLOCATION_ERROR

Explanation: Cannot allocate storage for internal use.

What to do: Ensure that there is sufficient memory available.

RC_FILE_IO_PROBLEM

Explanation: DB2 Text Extender could not read or write a file.

What to do: Check that there is sufficient disk space and memory available at the server. Check that the environment variables and the text configuration settings are set correct.

RC_INVALID_BROWSE_INFO

Explanation: The browse information returned by DesGetSearchResultTable or by DesGetBrowseInfo and used as input for DesStartBrowseSession is not valid.

What to do: Check whether a programming error overrides the browse information.

RC_INVALID_BROWSE_OPTION

Explanation: The browse option in DesGetSearchResultTable is not valid.

What to do: Ensure that the option is BROWSE or NO_BROWSE.

RC_INVALID_MATCH_OPTION

Explanation: The match options used in DesOpenDocument is not valid.

What to do: Check that the option is FAST or EXTENDED.

RC_INVALID_PARAMETER

Explanation: One of the input parameters is incorrect.

What to do: Read the error message returned by DB2 Text Extender to determine the cause.

RC_INVALID_SEARCH_OPTION

Explanation: The search option in DesGetSearchResultTable is not valid.

What to do: Ensure that the option is DES_TEXTHANDLEONLY, DES_RANK, DES_MATCH, or DES_RANKANDMATCH.

RC_INVALID_SESSION

Explanation: The session pointer specified in the current service call is incorrect or obsolete.

What to do: Save any information that can help to find the cause of the error, then end the application.

RC_NO_BROWSE_INFO

Explanation: No browse information is returned by DB2 Text Extender. This is because the search argument resulted in an empty search result. This is not an error.

What to do: No action necessary.

RC_PARSER_INVALID_ESCAPE_CHARACTER

Explanation: The search criteria contains an incorrect escape character. This error is reported if a blank is used as an escape character or if, for one word or phrase, more than one escape character is specified in the search criteria. Example: ESCAPE " " or ESCAPE "#\$".

What to do: Check the syntax of the search argument, and try again.

RC_PARSER_INVALID_USE_OF_ESCAPE_CHAR

Explanation: The escape character syntax in the search criteria cannot be interpreted.

What to do: Check the escape character syntax. For example, if \$ is the specified escape character, the word or phrase can contain only \$\$, \$_ or \$%, where _ and % are the two masking symbols.

RC_PARSER_SYNTAX_ERROR [: position]

Explanation: The search criteria syntax cannot be interpreted. If a position is shown, the position is the character position in the search string where the problem was detected. Sometimes, this is not the exact position, so also check the characters before specified character position.

What to do: Check the syntax of the search argument, by referring to Chapter 11, "Syntax of search arguments," on page 121.

RC_SE_BROWSER_TIME_OUT

Explanation: The browse process was started but did not respond in an acceptable time. DB2 Text Extender then canceled the pending process.

This error can occur when your system does not have enough storage space or is overloaded.

Return codes

What to do: Terminate the browse session by calling DesEndBrowseSession, free allocated storage by calling DesFreeBrowseInfo, and try again.

RC_SE_CAPACITY_LIMIT_EXCEEDED

Explanation: The requested function cannot be processed. There is insufficient memory or disk space.

What to do: End the program and check your system's resources.

RC_SE_COMMUNICATION_PROBLEM

Explanation: Communication with the DB2 Text Extender server failed. The error could be caused by a lack of storage space or by an incorrect installation of DB2 Text Extender.

What to do: Save any information that can help to find the error, then end the application.

RC_SE_CONFLICT_WITH_INDEX_TYPE

Explanation: The linguistic specification of the search term of the query does not correspond to the type of index. For example, PRECISE FORM OF cannot be used with a linguistic index. The default linguistic specification is used as shown in Table 9 on page 127.

What to do: Adapt your application to prevent the specification of query options that conflict with the index type.

RC_SE_DICTIONARY_NOT_FOUND

Explanation: DB2 Text Extender linguistic services cannot find the dictionary files. The query is processed without linguistic support. The dictionary files corresponding to the specified language code(s) are not in the expected path.

What to do: You can continue to make API calls. For UNIX, check that the required dictionary is in the path {DB2TX_INSTOWNERHOMEDIR}/db2tx/dicts. If necessary, install the required dictionary.

RC_SE_DOCMOD_READ_PROBLEM

Explanation: When a DB2 Text Extender instance is created, a document models file called desmodel.ini is put in the instance directory. When you create an index, a desmodel.ini file is also put in the index directory IXnnnnn. This document models file could not be read.

What to do: Check that a document models file exists and that it is in the correct directory.

RC_SE_DOCUMENT_NOT_ACCESSIBLE

Explanation: The requested text document is found, but is currently not accessible.

What to do: Check whether the document is accessed exclusively by another task or user.

RC_SE_DOCUMENT_NOT_FOUND

Explanation: The requested text document was not found. The most likely cause is that a text document has been deleted from storage, but has not yet been removed from the DB2 Text Extender index. This can also occur if you are trying to browse a document identified by a damaged handle.

What to do: In most cases, you can ignore this return code. It will no longer be displayed after the next index update.

If it is persistent, check that your application program is passing the found handle correctly for browsing.

RC_SE_EMPTY_INDEX

Explanation: The DB2 Text Extender index corresponding to the handle column addressed by the search request is empty. Either no text documents have been added to this index or all text documents have been removed from it.

This can occur when a text column has been enabled, but the documents in the column have not yet been indexed. That is, you specified in the ENABLE TEXT COLUMN command to create the index later, at a time determined by the periodic indexing settings.

This can also occur when a text table has been enabled to create an empty common index for all text columns, but none of the text columns has been enabled.

What to do: If ENABLE TEXT TABLE has been used to create an empty common index for all text columns, run ENABLE TEXT COLUMN for at least one of the text columns that contain text to be searched. In this command, you can determine whether the index is created immediately, or at a time determined by the periodic indexing settings.

Run GET INDEX STATUS to check that the index was built successfully.

RC_SE_EMPTY_QUERY

Explanation: The specified search criteria was analyzed and processed linguistically by DB2 Text Extender. Either a programming error caused a query to be made containing no search terms, or all search terms were stop words (words not indexed by DB2 Text Extender) that are removed from a query. The result was no search terms.

What to do: Reword the query. If the problem persists,

check for a programming error.

RC_SE_END_OF_INFORMATION

Explanation: This is not an error. The end of the document has been reached. No further information is available for DesGetMatches.

What to do: Use this return code to end the iterative processing of the document with DesGetMatches.

RC_SE_FUNCTION_DISABLED

Explanation: The requested function called a DB2 Text Extender function that has been prevented by the administrator.

What to do: Ask your administrator for assistance. It may be necessary to stop and restart DB2 Text Extender (txstop/txstart).

RC_SE_FUNCTION_IN_ERROR

Explanation: The requested function has been locked due to an error situation on the DB2 Text Extender server. The API call cannot be processed.

What to do: Check the index status. Check the available space in the index directory. Reset the index status and retry the command.

RC_SE_INCORRECT_HANDLE

Explanation: A handle specified in an input parameter such as *browse session handle* is not valid. It must be a handle that was returned by a previous call and that is not obsolete.

What to do: Save any information that can help to find the cause of the error, then terminate the session by calling DesEndBrowseSession.

Check whether a programming error produced an incorrect handle.

RC_SE_INDEX_DELETED

Explanation: The DB2 Text Extender index being accessed is deleted.

What to do: Contact the DB2 Text Extender administrator to recreate the index.

RC_SE_INDEX_NOT_ACCESSIBLE

Explanation: The DB2 Text Extender index cannot be accessed and the current call cannot be processed.

What to do: Ask the DB2 Text Extender administrator to check the accessibility of the index.

RC_SE_INDEX_SUSPENDED

Explanation: DB2 Text Extender received a request relating to a index that was suspended from another session or from the current session.

What to do: Ask the DB2 Text Extender administrator to check the status of the index.

RC_SE_INSTALLATION_PROBLEM

Explanation: DB2 Text Extender has encountered an installation problem.

What to do: Check the current setting of the environment variables DB2INSTANCE, DB2TX_INSTOWNER, DB2TXINSTOWNERHOMEDIR. Use descfgcl -d and descfgsv -d -i txins000 to check your search service configuration.

RC_SE_IO_PROBLEM

Explanation: An error occurred when the server attempted to open or read one of its index files. This can be due to one of the following:

- An unintentional action by the administrator, such as the deletion of a DB2 Text Extender index file
- Incorrect setting in the text configuration.

What to do: Terminate the application. Check with the administrator that:

- All files of the current DB2 Text Extender index exist
- The text configuration settings are correct.

RC_SE_LS_FUNCTION_FAILED

Explanation: A function that accessed the database to retrieve text documents for browsing failed. Either the database is no longer accessible to the user, or the user is not authorized for the text table.

What to do: Check that the input to the function, such as the user ID, is correct. Check that the database is accessible and that the user is authorized for the task.

RC_SE_LS_NOT_EXECUTABLE

Explanation: A function that is trying to access the database to retrieve text documents for browsing cannot be executed.

What to do: Check that DB2 Text Extender is installed correctly. If the problem persists, contact your IBM representative.

RC_SE_MAX_OUTPUT_SIZE_EXCEEDED

Explanation: An unusually large number of matches have been found. The size of the browse information has exceeded the maximum that can be handled. The request cannot be processed.

Return codes

What to do: Either make the query more specific or ensure that more system memory is available.

RC_SE_MAX_NUMBER_OF_BUSY_INDEXES

Explanation: The requested function has been prevented by the search service, because the maximum number of indexes is currently active.

What to do: Reissue the function call after a short period of time. In general, the problem is only temporary.

RC_SE_NO_DATA

Explanation: This is not an error. No text document matches the search criteria. If you request browse information, no browse information is returned. No storage is allocated for the browse information.

What to do: No action is necessary.

RC_SE_NOT_ENOUGH_MEMORY

Explanation: There is not enough storage space on the client or on the server system. The current request cannot be processed.

What to do: Release storage space and end the application.

RC_SE_PROCESSING_LIMIT_EXCEEDED

Explanation: The current search request exceeded the maximum result size or the maximum processing time specified for your client/server environment. The request was canceled.

What to do: Make the search request more specific. Consider increasing the maximum processing time.

RC_SE_QUERY_TOO_COMPLEX

Explanation: The specified query is too complex.

What to do: Adapt your application to prevent excessive use of masking characters and synonyms.

Excessive use of masking symbols or excessive use of the SYNONYM option can expand a query to a size that cannot be managed by DB2 Text Extender.

RC_SE_REQUEST_IN_PROGRESS

Explanation: A DB2 Text Extender browse API service was called while another browse API request was active for the same session.

What to do: End the session by calling DesEndBrowseSession and free storage by calling DesFreeBrowseInfo.

The DB2 Text Extender browse API does not support concurrent access to the same browse session.

All applications running concurrently in the same process should handle their own browse sessions.

RC_SE_SERVER_BUSY

Explanation: The DB2 Text Extender client cannot currently establish a session with the requested DB2 Text Extender server, or the DB2 Text Extender server communication link was interrupted and cannot be re-established.

The DB2 Text Extender server has been started correctly, but the maximum number of parallel server processes was reached.

What to do: If this is not a temporary problem, change the communication configuration on the server.

RC_SE_SERVER_CONNECTION_LOST

Explanation: The communication between client and server was interrupted and cannot be re-established.

The DB2 Text Extender server task may have been stopped by an administrator or the server workstation may have been shut down.

What to do: Check whether either of these conditions have occurred, and have them corrected.

RC_SE_SERVER_NOT_AVAILABLE

Explanation: The DB2 Text Extender API services could not establish a session with the requested DB2 Text Extender server.

The DB2 Text Extender server may not have been started.

What to do: Check that the DB2 Text Extender server has been started correctly. If the error persists, there is an installation problem.

RC_SE_STOPWORD_IGNORED

Explanation: This informational code is returned when the specified query contained at least one search term consisting only of stop words. The search term was ignored when processing the query.

What to do: You can continue to issue API calls. Avoid using stop words in DB2 Text Extender queries.

RC_SE_UNEXPECTED_ERROR

Explanation: An error occurred that could be caused by incorrect installation of DB2 Text Extender.

What to do: End the application, saving any information that may help to find the cause of the error.

RC_SE_UNKNOWN_INDEX_NAME

Explanation: The name of the text index associated with a text column is part of the handle.

What to do: Ensure that the handle you use as input to DesGetBrowseInfo is correct.

RC_SE_UNKNOWN_SECTION_NAME

Explanation: A specified section name is not part of a model specified in a document models file, or of the default model used.

What to do: Specify a section name that is part of the specified model or the default model.

RC_SE_UNKNOWN_SERVER_NAME

Explanation: The name of DB2 Text Extender server is part of the handle.

What to do: Ensure that the handle you use as input to DesGetBrowseInfo is correct.

RC_SE_WRITE_TO_DISK_ERROR

Explanation: A write error occurred that could be caused by a full disk on the DB2 Text Extender server workstation, or by incorrect installation of DB2 Text Extender.

What to do: End the application, saving any information that may help to find the cause of the error. Check that there is enough disk space available at the server.

RC_SQL_ERROR_WITH_INFO

Explanation: An SQL error occurred. An error message is returned.

What to do: Check the error message returned by DB2 Text Extender for more information, such as the SQL error message, SQLState and native SQL error code.

RC_SQL_ERROR_NO_INFO

Explanation: An SQL error occurred. No error message is returned.

RC_TEXT_COLUMN_NOT_ENABLED

Explanation: The specified handle column is not a column in the table you specified.

What to do: Check whether the handle column name you specified is correct. Ensure that the text column in that table has been enabled.

Return codes

Chapter 15. Messages

This chapter describes the following:

- **SQL states returned by DB2 Text Extender functions:** These messages can be displayed when you use DB2 Text Extender functions.
- **Messages from the DB2TX command line processor:** These messages can be displayed when you enter commands using the command line processor DB2TX. Each message number is prefixed by DES.

SQL states returned by DB2 Text Extender functions

The SQL functions provided by DB2 Text Extender can return error states. Example:

```
SQL0443N User-defined function
"DB2TX.CONTAINS" (specific name "DES5A")
has returned an error SQLSTATE with
diagnostic text "Cannot open message file".
SQLSTATE=38702
```

The messages in this section are arranged by SQLSTATE number.

01H10 **The file *file-name* cannot be opened.**

What to do: Ensure that the file exists, and that the DB2 instance name has the necessary permissions to open it.

01H11 **The text handle is incomplete**

Explanation: An attempt was made to use a handle that has been initialized, but not completed. A partial handle was created using INIT_TEXT_HANDLE containing preset values for the document language and format. However, the handle has not been completed by a trigger.

What to do: Use only handles that have been completed. If the handle concerned is stored in a handle column, enable or reenables its corresponding text column.

01H12 **Search arguments too long. The second argument was ignored.**

Explanation: The REFINE function was used to combine two search arguments, but the combined length of the search arguments is greater than the maximum allowed for a LONG VARCHAR. The REFINE function returns the first search argument instead of a combined one.

What to do: Reduce the length of one or both search arguments, then repeat the query.

01H13 **A search argument contains a stopword.**

Explanation: The specified query contains at least one search term consisting only of stop words. The

search term was ignored when processing the query.

What to do: Avoid using stop words in DB2 Text Extender queries.

01H14 **A language dictionary for linguistic processing is missing.**

Explanation: DB2 Text Extender linguistic services cannot find the dictionary files. The query is processed without linguistic support. The dictionary files corresponding to the specified language code(s) are not in the expected path.

What to do: For UNIX, check that the required dictionary is in the path {DB2TX_INSTOWNERHOMEDIR}/db2tx/dicts. If necessary, install the required dictionary.

01H15 **A linguistic search term specification does not match the index type.**

Explanation: The linguistic specification of the search term of the query does not correspond to the type of index. For example, PRECISE FORM OF should not be used with a linguistic index. The default linguistic specification is used as shown in Table 9 on page 127.

What to do: Adapt your application to prevent the specification of query options that conflict with the index type.

38700 **The Text Extender library is not current.**

Explanation: An attempt was made to use a handle that can be interpreted only by a later version of DB2 Text Extender.

SQL states returned by DB2 Text Extender functions

What to do: Ensure that the path to the current library version is set correctly, and that you have the necessary permissions to access it.

Look in the DB2 catalog view SYSCAT.FUNCTIONS, in the IMPLEMENTATION column, for the function that caused the problem.

38701 *tracefile* **Cannot open this trace file.**

Explanation: An attempt was made to use a trace function that writes to the file DB2TX_TRACEFILE in the directory DB2TX_TRACEDIR. Either the file does not exist, cannot be found, or the necessary permissions for the file are not available.

38702 **Cannot open message file** *message-file*.

Explanation: A situation occurred that caused DB2 Text Extender to attempt to return a message. The file containing the messages either does not exist or cannot be found, or the necessary permissions for the file are not available.

What to do: Ensure that the file exists, that the path is set correctly, and that you have the necessary permissions to open the file.

38704 **The format of the text handle is incorrect.**

Explanation: A handle having an incorrect format was used as an argument for a DB2 Text Extender function.

What to do: Ensure that the handle was not produced by INIT_TEXT_HANDLE.

38705 *udfname* **Incorrect UDF declaration.**

Explanation: The specific name of a DB2 Text Extender function has been changed in the script where the functions are declared. DB2 Text Extender function names can be changed, but not their specific names.

What to do: Check the script DESCVDF.DDL that contains the DB2 Text Extender function declarations, to ensure that the correct names are still being used. Check the names against those in the original distribution media.

38706 *attribute* **Cannot recognize this attribute value.**

Explanation: An attempt was made to set a CCSID, format, or language to an unknown value.

What to do: Refer to Chapter 4, "Planning for your search needs," on page 21 for the correct values.

38707 **The requested function is not yet supported.**

Explanation: The specified function is not yet supported.

What to do: Check the specified function.

38708 *return code*

Explanation: An error occurred while processing the search request.

What to do: Refer to the description of the return code in Chapter 14, "Return codes," on page 149.

38709 **Not enough memory available.**

Explanation: Not enough memory is available to run the DB2 Text Extender function.

What to do: Close any unnecessary applications to free memory, then try again.

38710 *errornumber* **Cannot access the search results.**

Explanation: An error occurred while attempting to read the list of found documents (result list) returned by the search service.

What to do: Try repeating the search. If this is not successful, restart the search service. If the problem persists, report it to your local IBM representative, stating the error number.

38711 **Severe internal error.**

Explanation: A severe error occurred.

What to do: Report the error to your local IBM representative, stating the circumstances under which it occurred.

38712 *indexname* **Incorrect handle in this text index.**

Explanation: A handle has been damaged.

What to do: Use UPDATE INDEX to rebuild the index.

38714 **Shorten DB2TX_INSTOWNERHOMEDIR environment variable.**

Explanation: The name of the home directory of the instance owner must be no longer than 256 characters.

What to do: Use links to reduce the length of the directory name.

SQL states returned by DB2 Text Extender functions

38717 **The specified thesaurus could be found.**

Explanation: The specified thesaurus cannot be found.

What to do: Check the specified thesaurus name.

38718 **The specified relation name could not be found in the thesaurus.**

Explanation: The specified relation does not exist in the specified thesaurus.

What to do: Ensure that the specified relation exists.

38719 **A search processing error occurred.**
Reason code: *rc*.

Explanation: The search could not be made due to the specified reason.

What to do: Try to solve the problem reported by the reason code. If the specified reason is not helpful and no further information is found in the `desdiag.log` file, create a trace and report the information to your local IBM representative.

38720 **A shared memory attach error occurred.**

Explanation: The system is unable to get access to shared memory.

What to do: Check your system configuration and increase shared resources, or check the current shared resource usage (`ipcs`) and clean up resources that are no longer needed.

38721 **A semaphore creation/access error occurred.**

Explanation: The system is unable to create or get access to a semaphore.

What to do: Check your system configuration and increase shared resources, or check the current shared resource usage (`ipcs`) and clean up resources that are no longer needed.

38722 **A search process didn't return.**

Explanation: An error occurred while processing the search request.

What to do: Verify your system configuration `descfgcl` and check if all nodes are up and running.

38723 **The index CCSID and query CCSID do not match.**

Explanation: The database CCSID used for the query string is not the same as the CCSID of the text index.

What to do: Disable the text index and recreate it using the CCSID of the database.

38724 **The section or model name is incorrect.**

Explanation: The specified section or model name in the query is incorrect.

What to do: Check the section or model name.

38726 **A model-file read error occurred.**

Explanation: The model-definition file was not found or cannot be opened.

What to do: Check that the model-definition file exists in the index directory.

Messages from DB2 Text Extender

Each message has a message identifier that consists of a prefix (DES), the message number, and a suffix letter. The suffix letter indicates how serious the occurrence is that produced the message:

- I** Information message
- W** Warning message
- N** Error (or “negative”) message
- C** Critical error message.

DES0001N Incorrect number of arguments for the db2txinstance command.

Explanation: The db2txinstance command needs two arguments.

What to do: Enter the command again with these arguments:

```
db2txinstance instanceName db2InstanceName
```

where *instanceName* is the login name of an existing UNIX user that is being assigned as the owner of this instance, and *db2InstanceName* is the login name of the owner of the corresponding DB2 instance.

DES0002N Invalid instanceName.

Explanation: The specified instance name must be the login name of an existing UNIX user.

What to do: Correct the instance name, or select an existing UNIX user, or create a UNIX user to be the instance owner.

Enter the db2txinstance command again as follows:

```
db2txinstance instanceName
```

where *instanceName* is the login name of the selected UNIX user.

DES0004N The specified instance already exists. The command cannot be processed.

Explanation: The *instanceName* specifies the login name of a UNIX user that is the owner of the instance. This instance owner already has a db2tx directory in the home directory.

What to do: To create the instance, remove the existing instance and then try the command again.

DES0005N The installation message catalog cannot be found.

Explanation: The message catalog required by the installation scripts is missing from the system; it may have been deleted or the database products may have been loaded incorrectly.

What to do: Verify that the db2tx_01_01_0000.client

product option is installed correctly. If there are verification errors, reinstall the option.

DES0015W A linguistic search term specification does not match the index type.

Explanation: The linguistic specification of the search term of the query does not correspond to the type of index. For example, PRECISE FORM OF should not be used with a linguistic index. The default linguistic specification is used as shown in Table 9 on page 127.

What to do: Adapt your application to prevent the specification of query options that conflict with the index type.

DES0016W A language specification is not supported for the current index type.

Explanation: The language you have specified is not supported for the specified index type.

What to do: See the documentation for a list of supported languages for the index type.

DES0017W Feature extraction has not been enabled.

Explanation: You used a feature search argument in your query but the index was not build with index option FEATURE_EXTRACTION.

What to do: The index option FEATURE_EXTRACTION is not supported.

DES0018W option is not supported for the current index type.

Explanation: You requested a search option that is not supported for the current index type and index option.

What to do: Check which index type or index option supports the requested search option. See Table 9 on page 127.

DES0121N Memory could not be allocated (malloc failed).

Explanation: No storage could be reserved for the application.

What to do: Increase the paging space.

DES0333N The client cannot establish a session with the requested server.

Explanation: The DB2 Text Extender client cannot establish a session with the requested server.

What to do: Check that the DB2 Text Extender server has been started. If not, run TXSTART.

DES0377N A text index file I/O problem occurred.

Explanation: DB2 Text Extender cannot access the text index. This can happen if the DIRECTORY setting in the text configuration points to an invalid directory.

What to do: Check the text configuration settings.

DES0500N IBM Text Search Engine (5722DE1, option 3) is not properly installed.

Explanation: IBM Text Search Engine is a prerequisite license program option to DB2 Text Extender. It is not installed on the system.

What to do: Install option 3 of license program 5722DE1, then try again.

DES0700N The node number value *'node'* is not contained in the node group definition.

Explanation: The specified node number is invalid.

What to do: Check the DB2 node number.

DES0701N The node number value *'node'* is out of range.

Explanation: The specified node number is invalid.

What to do: Check the DB2 node number.

DES0704N Format *'format'* requires the specification of an index property.

Explanation: The document format is not compatible with the index type information.

What to do: Specify an index property that is compatible with the document format.

DES0705N The specified document model name *'model'* was not found in the model definition file.

Explanation: The document model name was not found in the model definition file. Note that the model name is case-sensitive.

What to do: Use a model name that is specified in the model definition file.

DES0706N The model definition file cannot be accessed on the DB2 Text Extender server.

Explanation: The model definition file was not found or cannot be opened.

What to do: Check that the model definition file exists.

DES0707N Format *'format'* does not support the specified index property.

Explanation: The document format does not support the index property.

What to do: Specify an index property that is compatible with the document format.

DES0709W The dictionary for the specified language is not installed.

Explanation: DB2 Text Extender cannot find the dictionary files.

What to do: Install, or reinstall the dictionary for the specified language.

DES0710N A null pointer is not allowed for parameter *'parameter'*.

Explanation: No value is specified for *parameter*.

What to do: Specify a value for the parameter.

DES0711N An internal Text Extender error occurred. Diagnosis information: *message*.

Explanation: An internal processing error occurred.

What to do: Check the diagnostic message to solve the problem. If the internal error was not caused by an installation problem, additional information may be in the desdiag.log file or in a created trace file. If this does not help, collect the available information and call your IBM service representative.

Messages from DB2 Text Extender

DES0712N Parameter '*parameter*' is too long.

Explanation: The specified parameter is out of range.

What to do: Specify the parameter using a valid length.

DES0713N The length of parameter '*parameter*' is incorrect: %d1.

Explanation: The specified parameter is out of range.

What to do: Specify the parameter using a valid length.

DES0714N Specify parameter *parameter* either directly or in the configuration table.

Explanation: CCSID, format, or language was not specified, and there is no text configuration setting for this value.

What to do: Either specify the missing parameter directly in the ENABLE TEXT COLUMN command, or set a value in the text configuration settings.

DES0715N Data type *schema.type* is not supported for text data.

Explanation: *schema.type* is the schema name and type name of the text column or the result of an access function. The data type for a text column is not supported by DB2 Text Extender. It must be CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB. If this is not the case, you must provide an access function whose input is the data type of the text column and whose output is VARCHAR, LONG VARCHAR, or LOB.

What to do: If *schema.type* is a text column type, you must register an access function with a result of type VARCHAR, LONG VARCHAR, or LOB. If *schema.type* is the result of an access function, it cannot be used. Provide an access function with a result of the required type.

DES0716N Format *format* is not supported.

Explanation: *format* is a format that is not supported by Text Extender.

What to do: Check the list of supported formats in "Which document formats are supported" on page 22.

DES0717N Language *language* is not supported.

Explanation: *language* is a language that is either not supported by DB2 Text Extender, or the selected language is not supported by the specified index type.

What to do: Check the list of supported languages in Table 1 on page 36.

DES0718N CCSID *ccsid_value* is not supported.

Explanation: You specified an invalid CCSID value.

What to do: See the documentation for a list of supported CCSIDs.

DES0719N A call to the DB2 Text Extender program *program* failed with return code *rc*.

Explanation: An error may have occurred during installation. The return codes are listed in file DES_EXT.H.

What to do: Check if the installation was successful. Check that the environment variables such as DB2TX_INSTOWNER and DB2TX_INSTOWNERHOMEDIR are set correctly.

DES0720N The access function *schema.function* is not registered in the database.

Explanation: The name of the function is either incorrect or has not been registered with the database.

What to do: Check the name of the access function. If it is correct, check that the function is known to the database system. Use the CREATE FUNCTION to register the access function with the database.

DES0721N The database is inconsistent; a DB2 Text Extender catalog view is missing.

Explanation: One of the DB2 Text Extender catalog views is not in the database.

What to do: Use the DISABLE DATABASE command to remove the remaining catalog views, then enter ENABLE DATABASE again. The index data is lost; reindex the text documents.

DES0722N Table *schema.table* is not a base table in the database.

Explanation: Either the table does not exist in the database or it is a result table or a view. A text column must be in a base table before it can be enabled for DB2 Text Extender.

What to do: Ensure that the table name is correct, and that it is a base table.

DES0723N The creation of an index for the handle column *handlecolumnname* in table *schema.table* failed.

Explanation: A text index could not be created for the handle column.

What to do: Use txstatus to check the status of the server. If the services on the server are running correctly, use DISABLE TEXT COLUMN or DISABLE

TEXT TABLE to get a consistent state again. Then enable the text column again using ENABLE TEXT COLUMN or ENABLE TEXT TABLE.

DES0724N An entry in the TextIndices catalog view for the handle column *handlecolumn* in table *schema.table* is missing.

Explanation: The TextIndices catalog view is damaged.

What to do: Use DISABLE TEXT COLUMN or DISABLE TEXT TABLE to get a consistent state again. Then enable the text column using ENABLE TEXT COLUMN or ENABLE TEXT TABLE.

DES0725N Request rejected by server. Licence check failed with message ID '%s1'.

Explanation: No valid license key was found for DB2 Text Extender.

What to do: Request a licence key, install and then retry.

DES0727N Column *column* in table *schema.table* is already enabled.

Explanation: This message can occur if the table has been dropped and then recreated using the same text column, without first disabling the column.

What to do: Disable the column, then try again.

DES0728N Column *column* does not exist in table *schema.table*.

Explanation: You are trying to enable a text column that does not exist.

What to do: Change the table name or the column name, then try again.

DES0729N Handle column *handlecolumn* does not exist in table *schema.table*.

Explanation: You are trying to use a handle column that does not exist.

What to do: Use the GET STATUS command to check if the handle column exists, and that its name has been specified correctly.

DES0730N Table *schema.table* is already enabled as a common-index table.

Explanation: You are trying to enable a table that has already been enabled as a common-index table.

What to do: Either continue without enabling the table, or run the DISABLE TEXT TABLE command to disable the table before enabling it again.

DES0731N Table *schema.table* is not enabled for Text Extender; it cannot be disabled.

Explanation: You are trying to disable a table that has not been enabled.

What to do: Check the table name.

DES0732N The update frequency is incorrect near location *location*; expected was *parameter*.

Explanation: The *parameter* specification for the Update Frequency was not correct.

What to do: Check the update frequency parameter and reenter the command.

DES0733N Table *schema.table* contains an enabled column; it cannot be enabled as a common-index table.

Explanation: This table contains a text column that already has its own index. You cannot create a common index for all the text columns while this individual index exists.

What to do: Use DISABLE TEXT COLUMN to disable the enabled columns, then enter the ENABLE TEXT TABLE command again.

DES0734N Handle column *handlecolumn* belongs to the partial-text table *schema.table*; it cannot be disabled separately.

Explanation: You cannot disable a single text column in a table that was enabled as a partial-text table.

What to do: Disable the complete partial-text table.

DES0736N *handlecolumn* is already a handle column in table *schema.table*.

Explanation: You are trying to use an existing handle column name.

What to do: Reenter the command, using a different name for the handle column.

DES0737N Table *schema.tablename* is enabled as a common-index table with STORAGE option *storage_option*.

Explanation: It is not possible to enable a common index table for external files.

What to do: If you want to enable a table for external files, use a multi-index table.

Messages from DB2 Text Extender

DES0738N Access function *schema.function* has incorrect parameters.

Explanation: The input or output parameters of *schema.function* are incorrect.

- There can be only one input parameter, and it must be of the data type of the text column to be enabled.
- The output parameter must be of type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB.

DES0739W The index update program for table *schema.table*, handle column name *handlecolumn*, could not be started.

Explanation: The program that updates indexes could not be started. An error may have occurred during installation.

What to do: Check if the installation was successful. Check that the environment variables such as DB2TX_INSTOWNER and DB2TX_INSTOWNERHOMEDIR are set correctly.

DES0740N Handle column '*column*' can be reused only for a text column of type *columnntype*.

Explanation: The handle column has already been used for another handle column type.

What to do: Specify a new handle column name.

DES0741N The program or file *parameter* was not found or could not be started.

Explanation: The ENABLE DATABASE or DISABLE DATABASE command could not open the file *parameter*. An error may have occurred during installation.

What to do: Check if the installation was successful.

DES0742N This table uses column indexes. Specify a handle column in the command.

Explanation: The specified table is enabled as multi index table. To work with a specific column specify the related handle column.

What to do: Specify a handle column.

DES0745N The DB2TX instance owner *instance-owner* is not a valid user ID.

Explanation: The environment variable DB2TX_INSTOWNER does not contain a valid user ID.

What to do: Correct the environment variable.

DES0747N The current CCSID is not supported for index type *index_type*.

Explanation: You specified a CCSID that is not supported for the requested index type.

What to do: See the documentation for a list of supported CCSIDs.

DES0748N The commit count value '*commitcount*' is not supported by DB2 Text Extender.

Explanation: The specified commit count value is not supported.

What to do: Specify a valid commit count value.

DES0749N The update index value '*indexvalue*' is not known by DB2 Text Extender.

Explanation: The specified 'update index' value is invalid.

What to do: Specify a valid update index value.

DES0750N The index type value '*indextype*' is not known by DB2 Text Extender.

Explanation: The specified index type value is invalid.

What to do: Specify a valid index type value

DES0751N You do not have the authorization to perform the specified operation.

Explanation: You do not have the required database administrator authorization to do this operation.

What to do: Have this operation done by a database administrator.

DES0756N The database is not enabled for Text Extender.

Explanation: The database must be enabled before this command can be run.

What to do: Run ENABLE DATABASE, then resubmit the command.

DES0763N The update frequency is incorrect.

Explanation: The specified 'update frequency' value is invalid.

What to do: Specify a valid 'update frequency'.

DES0765N The database is already enabled for Text Extender.

Explanation: You are trying to enable a database that is already enabled.

What to do: Either continue without enabling the

database, or use DISABLE DATABASE to disable the database before enabling it again.

DES0766N An action has caused the maximum row size of the table or a temporary table to be exceeded.

Explanation: The ENABLE TEXT COLUMN command adds a handle column to the table. If the table is already large, this can cause the row size of the table to exceed the maximum value of 4005.

The ENABLE TEXT COLUMN command also creates a temporary table whose size is proportional to the number of text columns that are already enabled. If many text columns are already enabled, the size of the temporary table may exceed the maximum value.

What to do: Use the ENABLE TEXT COLUMN only on tables that do not cause this limit to be exceeded.

DES0769W Warning: Index characteristic have been specified but will be ignored. Table 'tablename' is a common-index table.

Explanation: The specified table is a common index table therefore no index characteristics can be specified.

What to do: No action required.

DES0770N The environment variable *env-variable* is not defined.

Explanation: A parameter for a command was not specified and the system tried to read the default value from the environment variable *env-variable*, but this environment variable is not defined.

What to do: Define the required environment variable.

DES0774N The value length *length* for variable '*variable*' is out of range.

Explanation: The value length of the parameter is out of range.

What to do: Specify the parameter using a valid length.

DES0775N The index directory value '*directory*' is incorrect.

Explanation: The index directory value is incorrect, may be the directory length is incorrect.

What to do: Specify a valid index directory value.

DES0776N The table space name '*tablespace*' is incorrect.

Explanation: The specified table space name is incorrect, may be the table space value length is incorrect.

What to do: Specify a valid table space value.

DES0777N The table space *tablespace* is not known by the database management system.

Explanation: The specified table space is not known to the database system.

What to do: Check that the specified table space exists in the database.

DES0778N '*tablespace*' is not a REGULAR table space. It was created with keyword '*keyword*'.

Explanation: The data type for the specified table space is not supported.

What to do: Specify a regular table space.

DES0779I Indexing has been started successfully. To check indexing status use 'GET INDEX STATUS'.

Explanation: The indexing program has started. You can use the 'GET INDEX STATUS' command to check the status of the indexing process.

What to do: Check output of GET INDEX STATUS command.

DES0780I Index reorganization has been started successfully. To check the index status use 'GET INDEX STATUS'.

Explanation: The reorganization program has started. You can use the 'GET INDEX STATUS' command to check the status of the reorganization process.

What to do: Check output of GET INDEX STATUS command.

DES0789W Warning: The partitioning map of the current nodegroup has been updated, please call the TXNCHECK utility.

Explanation: The partitioning map of the current nodegroup has been updated.

What to do: Use the TXNCHECK command.

Messages from DB2 Text Extender

DES0800I The *command* command completed successfully.

Explanation: The specified command completed successfully.

What to do: No action required.

DES0810N Closing quotation mark is missing.

Explanation: A quotation mark has been found, but the second quotation mark is missing.

What to do: Check the syntax of the command and try again.

DES0811N "*token*" is unexpected. Check the index characteristics or the text information.

Explanation: The index characteristics or the text information is incorrect.

What to do: Check the syntax and try again.

DES0812N Table *schema.table* does not exist or is not enabled for DB2 Text Extender.

Explanation: While running the GET command, either the name of a database table is incorrect, or the table does not exist, or it has not yet been enabled.

What to do: If the table name is correct, use GET STATUS to check that it has been enabled. Enable the table and try again.

DES0813N Table *schema.table* does not exist or is not enabled for DB2 Text Extender or does not contain a handle column *column*.

Explanation: While running the GET command, no entries for the handle column are found in the table. If the table exists, it is not enabled or it does not contain a handle column.

What to do: If the table name is correct, use GET STATUS to check that it has been enabled. Enable the table and try again.

DES0814N Table *tablename* does not exist or is not enabled for DB2 Text Extender or no text column is enabled within this table.

Explanation: The specified table is not enabled for DB2 Text Extender.

What to do: Enable the table for DB2 Text Extender.

DES0815N Empty quotes "" found. A name is expected inside the quotes.

Explanation: Two consecutive quotation marks were found with no text between them.

What to do: Check the syntax and try again.

DES0816N The word "*token*" is unexpected. Use one of the keywords *keyword* or *keyword*.

Explanation: An unexpected token was found.

What to do: In the command, use one of the keywords given in the message.

DES0817N "*token*" is unexpected. Use the keyword *keyword*.

Explanation: An unexpected token was found.

What to do: In the command, use the keyword given in the message.

DES0818N Unexpected end of command. The keyword *keyword* is expected.

Explanation: A keyword is missing.

What to do: In the command, use the keyword given in the message.

DES0819N Unexpected end of command. One of the following keywords is expected: *keyword* or *keyword*.

Explanation: A keyword is missing.

What to do: In the command, use one of the keywords given in the message.

DES0820N Index option *index_option* is not supported for index type *index_type*.

Explanation: You specified an index option that is not supported for the given index type.

What to do: See the documentation for supported index options for the given index type.

DES0821N The name "*token*" is too long. Only *nn* characters are allowed for *variable* names.

Explanation: A name is too long.

What to do: Specify a name having an acceptable length.

DES0822N The command contains an unrecognized token "*token*". End of command is expected.

Explanation: End of command found, but a keyword is expected.

What to do: Check the syntax of the command and try again.

DES0823N A table name is expected following "*schema*".

Explanation: A table name or a function name is missing after the ".".

What to do: Check the syntax of the command and try again.

DES0824N Unexpected end of command; a keyword is required.

Explanation: The keyword in the message is missing from the syntax.

What to do: Check the syntax of the command and try again.

DES0825N '*alias*' is not a known database alias name. Only *nn* characters are allowed.

Explanation: The specified database alias name is unknown to the database system.

What to do: Check that the specified alias name is valid.

DES0826N Database alias *alias* must not be in quotation marks.

Explanation: The name in the message has been interpreted as a database alias. It must not be in quotation marks.

What to do: Check the syntax of the command and try again.

DES0827N The CCSID "*ccsid*" is not supported.

Explanation: The CCSID is not one of those supported by DB2 Text Extender.

What to do: Refer to the documentation for a list of the supported CCSIDs.

DES0829N The user name *userid* must not be in quotation marks.

Explanation: You entered a user name in quotation marks.

What to do: Remove the quotation marks.

DES0830N Parameter "*parameter*" in the *enable/disable* DATABASE command not recognized. End of command expected.

Explanation: The commands ENABLE DATABASE and DISABLE DATABASE do not take parameters.

What to do: Enter the command again without parameters.

DES0831N Unexpected end of command. The table name is missing.

Explanation: The command requires a table name.

What to do: Enter the appropriate table name.

DES0832N Unexpected end of command. The database name is missing.

Explanation: The command requires a database name.

What to do: Enter the appropriate database name.

DES0833N Unexpected end of command. The column name is missing.

Explanation: The command requires a column name.

What to do: Enter the appropriate column name.

DES0899N Unknown DB2TX command: *command*.

Explanation: The specified command is not supported by DB2 Text Extender

What to do: Type **db2tx ?** to get a list of the commands.

DES0971N Index directory can be specified once without node specification or multiple times with node specification.

Explanation: The specification of the index directory/ies is not correct.

What to do: Check the specification of the index directory. You can specify one index directory without a node specification or multiple index directories with node specification.

DES0972N The node specification is not correct. An unsigned digit value is expected.

Explanation: A non digit value is specified for the node number.

What to do: Specify an unsigned digit value for the node number.

Messages from DB2 Text Extender

DES0973N The node specification is not correct. One or more unexpected characters found before right parenthesis.

Explanation: The node specification is syntactical incorrect.

What to do: Check the syntax of the node specification and try again.

DES0974N The sequence of the node numbers in a TO clause is not correct (second node smaller than first node).

Explanation: The node specification is syntactical incorrect.

What to do: Check the syntax of the node specification and try again.

DES0975N The syntax for the specification of the node information is not correct.

Explanation: The node specification is syntactical incorrect.

What to do: Check the syntax of the node specification and try again.

DES0976N The node information is incomplete.

Explanation: The node specification is incomplete - some information is missing.

What to do: Check the syntax of the node specification and try again.

DES0977N The node information is incomplete. The left parenthesis is missing.

Explanation: The node specification is incomplete - left parenthesis is missing.

What to do: Correct the node specification and try again.

DES0998N The document model name length is incorrect.

Explanation: The length of the 'document model name' value is incorrect.

What to do: Check the model name value and try again.

DES0999N The syntax for the specification of the document model(s) is not correct.

Explanation: The specification of the model name(s) is syntactical incorrect.

What to do: Check the syntax of the model(s) specification and try again.

DES5250E Unable to create the Text Extender instance directory.

Explanation: The DB2 Text Extender directory /QIBM/userData/DB2Extenders/Text/instance cannot be created.

What to do: Check that you have the required authority to create the directory, and then try again.

DES5251I The Text extender instance already exists.

Explanation: The instance cannot be created as it already exists.

What to do: Use the existing instance, or drop and recreate the instance.

DES5252E Unable to drop the Text Extender instance.

Explanation: The instance directory /QIBM/UserData/DB2Extenders/Text/instance does not exist. The instance cannot be dropped.

What to do: Check if the instance has already been dropped by running TXSTATUS. If it has been dropped, you may wish to recreate the instance.

DES5253I The Text extender instance does not exist.

Explanation: You tried to drop an instance that does not exist.

DES9994N A Text Search Engine error occurred. Reason code: reason-code

Explanation: The text search engine used by DB2 Text Extender raised an error.

What to do: Check the search engine reason code in Chapter 16, "Search engine reason codes," on page 169. If the reported reason does not help to solve the problem additional information may be in the desdiag.log file or in a created trace file. If this does not help collect the available information and call your IBM service representative.

DES9995N A Text Extender error occurred. Message text: message-text

Explanation: A DB2 Text Extender error occurred.

What to do: Use the message provided by DB2 Text Extender to solve the problem. If the reported message does not help to solve the problem additional information may be in the desdiag.log file or in a created trace file. If this does not help collect the available information and call your IBM service representative.

DES9996N An internal DB2 Text Extender error occurred. Reason code: *reason_code*

Explanation: An internal processing error occurred.

What to do: Check that the DB2 Text Extender installation has been completed successfully. If yes, note the reason code and call your IBM service representative. If the error is an SQL error, no additional information is displayed. If the reason codes is:

- 104: Internal error. Check the contents of table db2tx."Environment" and especially the value of DB2TX_INSTOWNERHOMEDIR.

DES9997N An SQL error occurred. SqlState: *state*
QL Error code: *rc*; **SqlErrorMessage:** *message*

Explanation: An SQL error occurred.

What to do: Take action on the SQL error message that is displayed with the message.

DES9998N An SQL error occurred. No further information is available.

DES9999N No corresponding error message.

Explanation: An internal processing error occurred.

What to do: Check the diagnostic message to solve the problem. If no installation problem causes the internal error additional information may be in the desdiag.log file or in a created trace file. If this does not help collect the available information and call your IBM service representative.

Messages from DB2 Text Extender

Chapter 16. Search engine reason codes

This chapter lists the reason codes that are returned by DB2 Text Extender's search engine.

Table 16. Search engine reason codes

Reason codes	Values
RC_DONE	0
RC_CONTINUATION_MODE_ENTERED	1
RC_END_OF_INFORMATION	2
RC_EMPTY_LIST	3
RC_MORE_INFORMATION	4
RC_INDEX_GROUP_SEARCH_ERROR	7
RC_INDEX_SPECIFIC_ERROR	8
RC_DICTIONARY_NOT_FOUND	9
RC_PROCESSING_LIMIT_EXCEEDED	12
RC_UNKNOWN_SERVER_NAME	16
RC_INCORRECT_AUTHENTICATION	17
RC_DATASTREAM_SYNTAX_ERROR	18
RC_QUERY_SCOPE_TOO_COMPLEX	20
RC_QUERY_TOO_COMPLEX	22
RC_MEMBER_OF_INDEX_GROUP	23
RC_UNKNOWN_INDEX_NAME	24
RC_INCORRECT_HANDLE	25
RC_INDEX_NOT_MEMBER_OF_GROUP	26
RC_UNKNOWN_SESSION_POINTER	27
RC_UNKNOWN_COMMUNICATION_TYPE	29
RC_UNKNOWN_SERVER_INFORMATION	30
RC_INVALID_MASKING_SYMBOL	31
RC_UNEXPECTED_ERROR	32
RC_SERVER_NOT_AVAILABLE	33
RC_INDEX_ALREADY_OPENED	35
RC_MAX_NUMBER_OF_OPEN_INDEXES	36
RC_MAX_NUMBER_OF_RESULTS	37
RC_CCS_NOT_SUPPORTED	41
RC_LANGUAGE_NOT_SUPPORTED	42
RC_CONFLICT_WITH_INDEX_TYPE	43
RC_MAX_INPUT_SIZE_EXCEEDED	46
RC_SERVER_BUSY	47
RC_SERVER_CONNECTION_LOST	48
RC_SERVER_IN_ERROR	49
RC_REQUEST_IN_PROGRESS	50
RC_UNKNOWN_INDEX_TYPE	51
RC_INCORRECT_INDEX_NAME	52
RC_INCORRECT_LS_EXECUTABLES	53
RC_INCORRECT_LIBRARY_ID	54
RC_INDEX_ALREADY_EXISTS	55
RC_MAX_NUMBER_OF_INDEXES	56
RC_INCORRECT_LOCATION	57
RC_LOCATION_IN_USE	58
RC_UNKNOWN_CONDITION	59

Search engine reason codes

Table 16. Search engine reason codes (continued)

Reason codes	Values
RC_INDEX_DELETED	60
RC_INDEX_SUSPENDED	61
RC_INDEX_NOT_ACCESSIBLE	62
RC_MAX_NUMBER_OF_BUSY_INDEXES	63
RC_CONFLICTING_TASK_RUNNING	64
RC_NOT_ENOUGH_MEMORY	65
RC_MAX_OUTPUT_SIZE_EXCEEDED	68
RC_COMMUNICATION_PROBLEM	70
RC_NO_ACTION_TAKEN	71
RC_EMPTY_INDEX	72
RC_EMPTY_QUERY	73
RC_INSTALLATION_PROBLEM	74
RC_FUNCTION_DISABLED	75
RC_FUNCTION_IN_ERROR	76
RC_IO_PROBLEM	77
RC_WRITE_TO_DISK_ERROR	78
RC_SERVER_VERSION_NOT_CURRENT	79
RC_FUNCTION_NOT_SUPPORTED	80
RC_RESULT_ALREADY_RANKED	81
RC_RESULT_VIEW_EXISTS	82
RC_INDEX_NOT_OPEN	83
RC_NO_RANKING_DATA_AVAILABLE	84
RC_LINGUISTIC_SERVICE_FAILED	85
RC_THESAURUS_PROBLEM	86
RC_INVALID_IDENTIFIER	88
RC_DOCUMENT_MODEL_ALREADY_EXISTS	89
RC_UNKNOWN_DOCUMENT_SECTION_NAME	90
RC_DOCMOD_READ_PROBLEM	91
RC_UNKNOWN_DOCUMENT_MODEL_NAME	92
RC_SECTION_NAME_ALREADY_EXISTS	94
RC_SECTION_TAG_ALREADY_EXISTS	95
RC_MAX_NUMBER_OF_TASKS	96
RC_LS_NOT_EXECUTABLE	97
RC_LS_FUNCTION_FAILED	98
RC_CAPACITY_LIMIT_EXCEEDED	99
RC_DOCUMENT_NOT_ACCESSIBLE	100
RC_DOCUMENT_CURR_NOT_ACCESSIBLE	101
RC_DOCUMENT_NOT_TO_INDEX	102
RC_DOCUMENT_NOT_FOUND	103
RC_DOCUMENT_IN_ERROR	104
RC_DOCUMENT_NOT_SUPPORTED	105
RC_CROSSIDX_SEARCH_NOT_ALLOWED	110
RC_DOCUMENT_GROUP_NOT_FOUND	111
RC_INVALID_ATTRIBUTE_VALUE	112
RC_INVALID_SECTION_TYPE	113
RC_INCORRECT_RELEVANCE_VALUE	120
RC_NO_RAT_EXPANSION	130
RC_DOCUMENT_NOT_IN_VIEW	131

Chapter 17. Error event reason codes

This chapter lists the error events that can occur when DB2 Text Extender indexes documents. This can occur, for example, when:

- Documents cannot be indexed
- Documents are indexed, but a problem occurs
- A language dictionary cannot be found.

Tip

If a reason code is not documented:

1. Check that there is enough disk space.
2. Collect all the error information that is available:
 - desdiag.log file
 - Event message
3. Call your IBM service representative.

1 Out of storage. The server ran out of memory. Reduce the workload.

64 The index process is still running, or the index is still reorganizing. This reason code is for information only.

116

Datastream syntax error

280

The document has not been indexed. One of the index files could not be opened.

281

The document has not been indexed. One of the index files could not be read.

441

The document has not been indexed. This message occurs for Ngram indexes only. The document's code page is different from the one the index was created with. This may happen for HTML and XML documents if the index was not created in UTF8.

500

The document has not been indexed. The Library Services could not be loaded. Check that the DLL is available and that the resource path is valid.

501

The document has not been indexed. Lib_Init in Library Services failed On Flat File systems: DIT file not found or not on a valid directory, or DIT contents not correct.

502

The document has not been indexed. An error has occurred while reading the document content in library service LIB_read_doc_content.

503

The document has not been indexed. An error occurred in library service LIB_access_doc.

Error event reason codes

- 504**
The document has not been indexed. The library service LIB_doc_index_status returned an error.
- 505**
Close document failed. The library service LIB_close_doc returned an error.
- 506**
End Library Services failed The library service LIB_end returned an error.
- 507**
The library service call LIB_read_doc_content failed with an unexpected return code.
- 508**
The library service call LIB_close_doc returned a RC_TERMINATION error.
- 545**
The document has not been indexed. One of the temporary index files could not be opened.
- 546**
The document has not been indexed. One of the temporary index files could not be closed.
- 548**
Internal error. Send the information in the diagnosis log to your IBM representative.
- 549, 550**
Out of storage (alloc failed). The server ran out of memory. Reduce the workload.
- 551-564**
The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.
- 565**
Out of storage (alloc failed). The server ran out of memory. Reduce the workload.
- 566-587**
The document has not been indexed. One of the index files could not be opened, read, written to or closed.
- 588-590**
Internal error. Send the information in the diagnosis log to your IBM representative.
- 591-604**
The document has not been indexed. One of the index files could not be opened, read, written to or closed.
- 605**
Out of storage (alloc failed). The server ran out of memory. Reduce the workload.
- 606-623**
The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

624

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

625-631

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

632

One of the temporary files created during indexing could not be opened with write access. Check the access rights.

633

One of the temporary files created during indexing could not be closed.

634

One of the temporary files created during indexing could not be written. Check that the index working directory has enough disk space.

635

One of the temporary files created during indexing could not be read.

636

One of the temporary files created during indexing could not be opened with read access. Check the access rights.

659

One of the temporary files created during indexing could not be opened.

660

One of the temporary files created during indexing could not be written.

661

One of the temporary files created during indexing could not be closed.

662

One of the temporary files created during indexing could not be opened.

663

One of the temporary files created during indexing could not be written.

664

One of the temporary files created during indexing could not be closed.

665

One of the temporary files created during indexing could not be opened.

667

One of the temporary files created during indexing could not be written.

668-669

The document was not indexed. There was a matching problem with section tags encountered in the document versus those defined in document models file.

670-672

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

673

Out of storage (alloc failed). The server ran out of memory. Reduce the workload.

Error event reason codes

674

Internal error, send the information in the diagnosis log to your IBM representative.

675-687

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

688, 690

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in the configuration file.

689

Internal error. Send the information in the diagnosis log to your IBM representative.

691-695

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

696-707

Internal error, send the information in the diagnosis log to your IBM representative.

708

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in configuration file.

709-718

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

719-721

Internal error, send the information in the diagnosis log to your IBM representative.

722, 729

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in configuration file.

730, 732, 733, 735-738

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk for the index and that access rights are correct.

731, 739-742, 744-746, 749, 755-758, 760-761, 767

Internal error, send the information in the diagnosis log to your IBM representative.

743, 748, 750-754, 759, 765-766, 768-770

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

747, 763, 764

Out of storage (alloc failed). The server ran out of memory. Reduce the workload. Try smaller values in configuration file.

815

There are two possible causes:

- One of the resource files needed to support the language used for the document causing the failure was not found
- The language requested by that document is not supported by DB2 Text Extender

831

The document has not been indexed. No text has been found. The document length is 0 bytes.

860

File open error. Either some dictionaries or the thesaurus files could not be found. Check your resource path for the dictionary files. If you have specified path information for your thesaurus files during search, check the location and file name.

861

The document has not been indexed. Tokenization of the text found no valid tokens. Check the documents content for validity, with respect to supported languages and contained words. This error can be caused by a document containing only stopwords.

954-956

Internal error. Send the information in the diagnosis log to your IBM representative.

957-967

The document has not been indexed. One of the index files could not be opened, read, written to or closed. Check that there is enough space on the disk used for the index and that access rights are correct.

1000

An error occurred during file open. Check access rights.

1001

An error occurred during file append. Check access rights.

1002

An error occurred during file read. Maybe the file is corrupted.

1003

An error occurred during file write. Check disk space and access rights.

1005

An error occurred during file read (positioning within file). Maybe the file is corrupted.

1006

An error occurred during rename of temporary file. Check the access rights.

1007

An error occurred during file create. Check access rights.

1008

An error occurred during file compression. Check the access rights.

1009

An error occurred during file close. Maybe the file is corrupted.

1010

The specified index name is already in use. Use another index name.

1011

The specified path is already in use. Use another location.

Error event reason codes

1012

The same path is used for data and working directory. Use another location.

1013

The specified index name is invalid. Index names must be uppercase or digits and not longer than 8 characters.

1014

An error occurred during file copy. Check access rights and disk space.

1017

The index name is unknown. Check correct spelling.

1019

An error occurred during file deletion. Check access rights. This error message can occur as a "secondary error" - look up the diagnosis file to see if a preceding error entry gives more information.

1020

General file error. Check access rights.

1070-1074

The document has not been indexed. The code page specified is either invalid generally, or is invalid for the index being accessed.

1085

The document has not been indexed. An error occurred when reading the index queue.

1086

The document has not been indexed. The index queue is empty.

1116-1117

The document has not been indexed. Information from the server instance initialization file could not be processed. Make sure entries in the initialization file are valid, and that the file is accessible to the application.

1129

No document has been indexed. Starting the background processing failed.

1158

An error occurred while renaming a file. Check access rights and disk space.

1162

The index files of an Ngram index may be corrupted.

1163, 1164

The document has not been indexed due to an unexpected error.

1165

The document has not been indexed due to an unexpected end-of-file condition.

1176

No more documents can be indexed for Ngram index. There is an overflow condition for document numbers (overflow of long). If there were many deletions or repeated updates of the same document, try a call to EhwReorg to solve the problem. If not, consider using a second index for new documents.

1177

The document has not been indexed. It was considered too big by the Ngram indexer.

1189

The document has not been indexed. There was a problem with boundary sequence (Korean-language specific).

1198-1200

The document has not been indexed. There was a problem with index access. The index may be corrupted.

1201

The document has not been indexed. The document code page could not be converted to the index-specific code page. This error is for Ngram indexes in UTF8 code page only.

1202

The document has not been indexed. The document code page could not be converted to the index-specific code page due to invalid data in the document. This error is for Ngram indexes in UTF8 code page only.

1500-1505

The document-analysis component has problems. It could either not be initialized (check LIBPATH and content of configuration file) or failed due to internal problems. See the diagnosis file for more information.

1904

The document has not been indexed. There is a problem with accessing the document model for a section-enabled index. Check access rights and for the existence of the file.

2000

The document has not been indexed. The document type is not supported. Library service Lib_access_doc returned an invalid document type.

2001

The document has not been indexed. An incorrect sequence of fields has been detected in the document's data stream.

2002

The document has not been indexed. An incorrectly structured field has been detected in the document's data stream.

2003

The document has not been indexed. Only one text section is allowed for a document in DB2 Text Extender text format.

2005

The document has not been indexed. A language specified in the document's data stream is not supported.

2006

The document has not been indexed. A CCSID specified in the document's data stream is not supported.

2007

The expected document format given by the library or by the default rule is not correct. The document header is incorrect for the format. Check if the default rule is a document with a special document header, and change if the rule is not correct.

2008

The document was not indexed because it could not be accessed.

Error event reason codes

2009

The document was not indexed because it was in use and could not be accessed.

2010

The document has not been indexed. The specified CCSID is not correct.

2011

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. End-of-page must be the last control in the body text of the document.

2012

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. A structured field contains an incorrect length specification.

2013

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. An incorrect control has been detected in the document.

2014

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. An incorrect multi-byte control or structured field has been detected in the document.

2015

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. Duplicate document parameters have been found.

2016

The document has not been indexed because it is not a valid IBM DCA RFT or FFT document. An empty text unit has been found.

2018

Either the document is in a format that is not supported, or there is an "exclude" entry in the DIT for the document's extension. Check that the document has a extension that allows it to be indexed.

2020

The document has not been indexed. It is neither a WordPerfect document nor a WordPerfect file.

2021

The document has not been indexed. It is a WordPerfect file but not a WordPerfect document.

2022

The document has not been indexed. This version of WordPerfect is not supported.

2023

The document has not been indexed. It is an encrypted WordPerfect file. Store the document without encryption.

2026

An END_TXT occurred in a footnote or an endnote. Check the WordPerfect file, it may be damaged.

2028

The parser returned non-document text. Check the file content, especially with respect to format-specific words. Check whether the document format is supported. If automatic format recognition fails, ensure that the correct parser is called.

2030

The document has not been indexed. Either it is not a Microsoft Word file or it is a version of Word that is not supported.

2031

The document has not been indexed. Unexpected end-of-file has been detected in a Microsoft Word document.

2032

The document has not been indexed. An incorrect control has been detected in a Microsoft Word document.

2033

The document has not been indexed. It was saved in *complex* format with the *fastsave* option. Save it with the *fastsave* option off.

2034

The document has not been indexed. A required field-end mark is missing in a Microsoft Word document.

2035

The document is encrypted. Store the document in Microsoft Word without encryption.

2036

This is a Word for Macintosh document; it cannot be processed. Store the document in Word for Windows format.

2037

This Word document contains embedded OLE objects.

2040

The document has not been indexed because it is not a valid ECTF file.

2041

The document has not been indexed. It contains an .SO LEN control that is not followed by a number.

2042

The document has not been indexed. It contains an .SO LEN control that is followed by an incorrect number. The number must be between 1 and 79.

2043

The document has not been indexed. Only one .SO DOC control is allowed. Save each ECTF document in a separate file.

2044

The document has not been indexed. An .SO HDE control must be followed by begin and end tags.

2046

The document has not been indexed. The document contains text before the .SO DOC control.

2047

The document has not been indexed. The document contains text before an .SO PID control.

2048

The document has not been indexed. An end tag is missing after a begin tag.

2050

The document has not been indexed. Incorrect tags have been detected following an .SO HDE control.

Error event reason codes

- 2051**
The document has not been indexed. End-of-line has been detected after an .SO control.
- 2052**
The document has not been indexed. Unexpected end-of-text has been detected.
- 2060**
The document has not been indexed. Either it is not an AmiPro document or it is a version of AmiPro that is not supported.
- 2061**
The document has not been indexed. The length of a control in an AmiPro document is too long.
- 2062**
The document has not been indexed. This version of AmiPro is not supported. Only AmiPro Architecture Version 4 is supported.
- 2063**
AmiPro Style Sheets have not been indexed.
- 2064**
The document has not been indexed. An incorrect character set has been detected. Only Lotus Character Set 82 (Windows ANSI) is supported.
- 2065**
The document has not been indexed. Unexpected end-of-file has been detected in an AmiPro document.
- 2072**
The document cannot be scanned because it is encrypted.
- 2073**
The document format is inconsistent.
- 2074**
The document has the "bad file" flag bit set.
- 2080**
The document has not been indexed. Either it is not an RTF document or it is a version of RTF that is not supported.
- 2081**
The document has not been indexed. An RTF control word has been detected that is too long.
- 2083**
The document has not been indexed. Macintosh code page is not supported.
- 2084**
The document has not been indexed. It is an RTF document, but this RTF version is not supported. Only RTF Version 1 is supported.
- 2090**
The document has not been indexed. It is an HTML document, which contains a tag considered too long by the parser.
- 2093**
The document has not been indexed. It is an XML document, which was rejected by the XML parser.

2100

The document is damaged or unreadable for some other reason. A new common parser could correct the problem.

2101

The document cannot be indexed because it is empty or it contains no text. Check whether the document contains only graphics.

2102

The document cannot be indexed because it is either password-protected or encrypted.

2105

The document type is known, but the filter is not available.

2106

The document cannot be indexed because it is empty.

2107

The document cannot be indexed because it cannot be opened. Check document access.

2112

The document cannot be indexed because it is an executable file.

2113

The document cannot be indexed because it is compressed.

2114

The document cannot be indexed because it is a graphic. If the graphic document format returns an acceptable piece of text, then request to include this document format in the indexing process.

2120

The output file of the user exit does not exist or is not accessible. A new common parser version could correct the problem.

2121

The output file cannot be opened for read or it is empty. A new common parser version could correct the problem.

2122

Attempting to use a user-exit output file, but no file name has been given or set in the object.

2130

The user exit program could not be run. Check if the executable can be found in the path set by the PATH environment variable. Create a trace and dump to get additional information about the environment (errno) return codes.

2131

The user exit program failed with a bad return code. Create a trace and dump to get additional information about the environment (errno) return codes.

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 give you any license to these patents. You can send license inquiries, in writing, to:

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

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

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

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

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

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

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 illustrates 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.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States, other countries, or both.

AIX
DB2
IBM
MVS
OS/390
RACF
z/OS

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

Microsoft, Windows, Windows NT, and the Windows logo 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.

Glossary

This glossary defines many of the terms and abbreviations used in this manual. If you do not find the term you are looking for, refer to the index or to the *Dictionary of Computing*, New York: McGraw-Hill, 1994.

A

access function. A user-provided function that converts the data type of text stored in a column to a type that can be processed by DB2 Text Extender.

API. Application programming interface.

application programming interface (API). A general-purpose interface between application programs and the DB2 Text Extender information retrieval services.

B

Boolean search. A search in which one or more search terms are combined using Boolean operators.

bound search. A search in Korean documents that respects word boundaries.

browse. To view text displayed on a computer monitor.

browser. A DB2 Text Extender function that enables you to display text on a computer monitor.

C

catalog view. A view of a system table created by DB2 Text Extender for administration purposes. A catalog view contains information about the tables and columns that have been enabled for use by DB2 Text Extender.

CCSID. Coded Character Set Identifier.

code page. An assignment of graphic characters and control function meanings to all code points. For example, assignment of characters and meanings to 256 code points for an 8-bit code.

command line processor. A program called DB2TX that:

- Allows you to enter DB2 Text Extender commands
- Processes the commands
- Displays the result.

common-index table. A DB2 table whose text columns share a common text index. See also *multi-index table*.

count. A keyword used to specify the number of levels (the depth) of terms in the thesaurus that are to be used to expand the search term for the given relation.

D

data stream. Information returned by an API function, comprising text (at least one paragraph) containing the term searched for, and information for highlighting the found term in that text.

DB2 Extender. One of a group of programs that let you store and retrieve data types beyond the traditional numeric and character data, such as image, audio, and video data, and complex documents.

DBCS. Double-byte character support.

dictionary. A collection of language-related linguistic information that DB2 Text Extender uses during text analysis, indexing, retrieval, and highlighting of documents in a particular language.

disable. To restore a subsystem, a text table, or a text column, to its condition before it was enabled for DB2 Text Extender by removing the items created during the enabling process.

distinct type. See *user-defined distinct type*.

document. See *text document*.

document handle. See *handle*.

document model. The definition of the structure of a document in terms of the sections that it contains. A document model makes DB2 Text Extender aware of the sections within documents when indexing. A document model lists the markup tags that identify the sections. For each tag you can specify a descriptive section name for use in queries against that section. You can specify one or more document models in a document models file.

E

enable. To prepare a subsystem, a text table, or a text column, for use by DB2 Text Extender.

environment variable. A variable used to provide defaults for values for the DB2 Text Extender environment.

environment profile. A script provided with DB2 Text Extender containing settings for *environment variables*.

escape character. A character indicating that the subsequent character is not to be interpreted as a *masking character*.

expand. The action of adding to a search term additional terms derived from a thesaurus.

extended matching. A process involving the use of a *dictionary* to highlight terms that are not obvious matches of the search term.

extender. See *DB2 Extender*.

external file. A text document in the form of a file stored in the operating system's file system, rather than in the form of a cell in a table under the control of DB2.

F

file handle. See *handle*.

format. The type of a document, such as ASCII, or WordPerfect.

free-text search. A search in which the search term is expressed as free-form text – a phrase or a sentence describing in natural language the subject to be searched for.

function. See *access function*.

fuzzy search. A search that can find words whose spelling is similar to that of the search term.

H

handle. A binary value that identifies a text document. It includes:

- A document ID
- The name and location of the associated index
- The document's *text information*
- If the document is located in an external file not under the control of DB2, the path and name of the file.

A handle is created for each text document in a text column when that column is *enabled* for use by DB2 Text Extender.

highlighting information. See *data stream*.

hybrid search. A combined *Boolean search* and *free-text search*.

I

index. To extract significant terms from text, and store them in a *text index*.

index characteristics. Properties of a *text index* determining:

- The directory where the index is stored
- The index type
- The frequency with which the index is updated

When the first index update is to occur.

index type. A characteristic of a *text index* determining whether it contains exact or linguistic forms of document terms. See *precise index*, *linguistic index*, and *Ngram index*.

initialized handle. A *handle*, prepared in advance, containing only the text format, or the text language, or both.

instance. A logical DB2 Text Extender environment. You can have several instances of DB2 Text Extender on the same workstation, but only one instance for each DB2 instance. You can use these instances to:

- Separate the development environment from the production environment

- Restrict sensitive information to a particular group of people.

instance variable. A variable used to provide a default value for the name of the *instance* owner, or the name of the instance owner's home directory.

L

language. The name of a *dictionary* to be used when *indexing*, *searching* and *browsing*.

linguistic index. A *text index* containing terms that have been reduced to their base form by linguistic processing. "Mice", for example, would be indexed as "mouse". See also *precise index* and *Ngram index*.

log table. A table created by DB2 Text Extender containing information about which text documents are to be indexed. *Triggers* are used to store this information in a log table whenever a document in an enabled text column is added, changed, or deleted.

M

masking character. A character used to represent optional characters at the front, middle, and end of a search term. Masking characters are normally used for finding variations of a term in a precise index.

match. The occurrence of a search term in a text document.

multi-index table. A DB2 table whose text columns have individual *text indexes*. See also *common-index table*.

N

Ngram index. A *text index* that supports DBCS documents and fuzzy search of SBCS documents. See also *linguistic index* and *precise index*.

nodegroup. A named subset of one or more database partition servers. *node* assigned to a physically separate machine. See also *logical node*.

O

occurrence. Synonym for *match*.

P

physical node. A *node* assigned to a physically separate machine. See also *logical node*.

precise index. A *text index* containing terms exactly as they occur in the text document from which they were extracted. See also *linguistic index* and *Ngram index*.

profile. See *environment profile*.

R

rank. An absolute value of type DOUBLE between 0 and 1 that indicates how well a document meets the search criteria relative to the other found documents. The value indicates the number of matches found in the document in relation to the document's size.

refine. To add the search criteria from a previous search to other search criteria to reduce the number of *matches*.

retrieve. To find a text document using a search argument in one of DB2 Text Extender's search functions.

S

SBCS. Single-byte character support.

search argument. The conditions specified when making a search, consisting of one or several search terms, and search parameters.

shell profile. See *environment profile*.

stop word. A common word, such as "before", in a *text document* that is to be excluded from the *text index*, and ignored if included in a *search argument*.

T

text column. A column containing *text documents*.

text configuration. Default settings for index, text, and processing values.

text document. Text of type CHAR, GRAPHIC, VARGRAPHIC, LONG VARGRAPHIC, DBCLOB, VARCHAR, LONG VARCHAR, or CLOB, stored in a DB2 table.

text index. A collection of significant terms extracted from text documents. Each term is associated with the document from which it was extracted. A significant improvement in search time is achieved by searching in the index rather than in the documents themselves. See also *precise index* and *linguistic index*.

text information. Properties of a *text document* describing:

The *CCSID*

The *format*

The *language*.

text table. A DB2 table containing *text columns*.

tracing. The action of storing information in a file that can later be used in finding the cause of an error.

trigger. A mechanism that automatically adds information about documents that need to be indexed to a *log table* whenever a document is added, changed, or deleted from a text column.

U

UDF. User-defined function.

UDT. User-defined distinct type.

user-defined distinct type (UDT). A data type created by a user of DB2, in contrast to a data type provided by DB2 such as LONG VARCHAR.

user-defined function (UDF). An SQL function created by a user of DB2, in contrast to an SQL function provided by DB2. DB2 Text Extender provides administration and search functions, such as CONTAINS, in the form of UDFs.

W

wildcard character. See *masking character*.

WLM. Work Load Manager

Index

Special characters

- | (OR) operator in search argument
 - how to use 50
 - search argument syntax 125
- & (AND) operator in search argument
 - how to use 50
 - search argument syntax 125

A

- abbreviations
 - editing an abbreviation file 37
 - lists of 36
- access function
 - description 45
 - in ENABLE TEXT COLUMN 84
- administration
 - abbreviation file, editing 37
 - backup and restore 15
 - CHANGE INDEX SETTINGS command 76
 - CHANGE TEXT CONFIGURATION command 77
 - changing the text configuration 14
 - command line processor 75
 - command summary, client 75
 - command summary, server 101
 - compiling a thesaurus definition file 105
 - creating a DB2 Text Extender instance 15
 - DB2TX command 75
 - DELETE INDEX EVENTS 79
 - DELETE INDEX EVENTS command 79
 - deleting index events 64
 - DESTRACE command 106
 - DISABLE SERVER FOR DB2TEXT command 80
 - DISABLE TEXT COLUMN command 81
 - disabling a server 72
 - disabling a text column 72
 - displaying the index settings 69
 - displaying the index status 67
 - displaying the server status 15
 - displaying the status of database, table, and column 65
 - displaying the text information settings 69
 - dropping a DB2 Text Extender instance 15
 - environment variables 13
 - GET ENVIRONMENT command 89
 - GET INDEX SETTINGS command 90
 - GET INDEX STATUS command 92
 - GET STATUS command 93
 - GET TEXT CONFIGURATION command 94
 - GET TEXT INFO command 95
 - IMOTHESC command 105
 - maintaining text indexes 63
 - modifying stop-word and abbreviation files 37
 - QUIT command 96
 - REORGANIZE INDEX command 97
 - reorganizing an index 65
 - RESET INDEX STATUS command 98

- administration (*continued*)
 - resetting the index status 64
 - starting the DB2 Text Extender server 102
 - status information, getting 65
 - stop-word file, modifying 37
 - stopping the DB2 Text Extender server 104
 - summary of commands, client 75
 - summary of commands, server 101
 - tracing faults 16
 - TXSTART command 102
 - TXSTATUS command 103
 - TXSTOP command 104
 - UPDATE INDEX command 99
 - updating an index immediately 63
- AmiPro, document format 22
- analysis of text
 - for indexing 134
- AND
 - Boolean operator 50
 - keyword in search argument 126
- application programming interface (API)
 - messages 155
 - return codes 149
- ASCII, document format 22

B

- backup and restore 15
- base form, reducing terms to 135
- basic text analysis
 - for indexing terms 134
 - normalization 134
 - of terms containing nonalphanumeric characters 134
 - sentence recognition 135
- Boolean operators
 - & (AND) and ! (OR) 50
 - NOT 54
- Boolean search argument 125
- BOUND keyword 126
- bound search, example 55

C

- CASE_ENABLED keyword
 - in ENABLE TEXT COLUMN 86
- catalog view
 - content 70
 - creating 41
 - deleting 72
- CCSID
 - avoiding code page problems 27
 - default in text configuration 14
 - description 25
 - extracting from a handle 60
 - function 112
 - GET TEXT INFO command 95

CCSID (*continued*)
 in CHANGE TEXT CONFIGURATION 78
 in ENABLE TEXT COLUMN 84
 initializing in handles 115
 list of 25

CHANGE INDEX SETTINGS command
 syntax 76

CHANGE TEXT CONFIGURATION command
 syntax 77
 using 14

character masking 138

client/server environment 4

code page
 See CCSID

coded character set identifier
 See CCSID

column
 DISABLE TEXT COLUMN command 81
 disabling 72
 ENABLE TEXT COLUMN command 83
 enabling 42
 enabling for various index types 44
 enabling in a large table 45

command line processor
 DB2TX command 75
 help for 41
 QUIT command 96
 starting 40

commands
 CHANGE INDEX SETTINGS 76
 CHANGE TEXT CONFIGURATION 77
 DB2TX 75
 DELETE INDEX EVENTS 79
 DESTRACE 106
 DISABLE SERVER FOR DB2TEXT 80
 DISABLE TEXT COLUMN 81
 ENABLE SERVER FOR DB2TEXT 82
 ENABLE TEXT COLUMN 83
 GET ENVIRONMENT 89
 GET INDEX SETTINGS 90
 GET INDEX STATUS 92
 GET STATUS 93
 GET TEXT CONFIGURATION 94
 GET TEXT INFO 95
 IMOTHEC 105
 QUIT 96
 REORGANIZE INDEX 97
 RESET INDEX STATUS 98
 summary, client commands 75
 summary, server commands 101
 TXSTART 102
 TXSTATUS 103
 TXSTOP 104
 UPDATE INDEX 99

COMMITCOUNT configuration parameter
 default in text configuration settings 14
 description 45
 in CHANGE TEXT CONFIGURATION 78
 in ENABLE TEXT COLUMN 87
 in ENABLE TEXT TABLE 99

common-index table
 description 32, 33

compiling a thesaurus definition file 105

compound terms, splitting 135

configuration 13

configuration files 145

configuration table
 CHANGE TEXT CONFIGURATION command 77
 creating 41
 displaying 66
 GET TEXT CONFIGURATION command 94

connecting to a database
 how to 41

CONTAINS function
 example 48
 syntax 113

COUNT keyword 125

D

data types of text documents
 function for converting 45
 supported 84

DB2 Extenders
 example of use 3

DB2DBDFT, environment variable 13

DB2TEXTH distinct type 111

DB2TX_ environment variables
 description 13
 displaying 66

DB2TX_INSTANCE, environment variable 13

DB2TX_INSTOWNER, environment variable 13

DB2TX_INSTOWNERHOMEDIR, environment variable 13

DB2TX, command line processor
 syntax 75
 using 40

DB2TX.SAMPLE table
 deleting 72

DBCS documents, searching in 32

decomposition of compound terms 135

DELETE INDEX EVENTS command
 example 64
 syntax 79

depth of terms in a thesaurus, specifying 125

DESCL.INI 145

DESMODEL.INI 34

DESSRV.INI 146

DESTRACE
 syntax 106
 using 16

dictionary file names 36

directory for index
 GET INDEX SETTINGS command 90

DIRECTORY keyword
 default in text configuration settings 14
 displaying the current setting 69
 in CHANGE TEXT CONFIGURATION 77
 in ENABLE TEXT COLUMN 87

DISABLE SERVER FOR DB2TEXT command
 syntax 80

- DISABLE SERVER FOR DB2TEXT command
 - (continued)
 - using 72
- DISABLE TEXT COLUMN command
 - syntax 81
 - using 72
- disk space for indexes 33
- distinct types 111
- document
 - CCSID 25
 - converting data types 45
 - converting format 23
 - displaying the settings 69
 - format in CHANGE TEXT CONFIGURATION 78
 - format in ENABLE TEXT COLUMN 84
 - format, description 22
 - formats supported 22
 - GET TEXT INFO command 95
 - indexing 21
 - information about 69
 - language 25
 - structure 34
 - supported data types 84
- document model
 - attribute value in search syntax 124
 - description 34
 - MODEL keyword in search syntax 123
 - modifying the document models file 34
 - SECTION keyword in search syntax 124
- document models file, contents 34
- dropping an instance
 - how to 15

E

- ENABLE SERVER FOR DB2TEXT command
 - syntax 82
 - using 41
- ENABLE TEXT COLUMN command
 - syntax 83
 - using 42
- environment variables 13
 - description 13
 - displaying 66
 - GET ENVIRONMENT command 89
- environment, client/server 4
- error events
 - DELETE INDEX EVENTS 79
 - deleting 64
 - displaying 68
 - GET INDEX STATUS command 92
 - reason codes 171
 - recording 43
- escape character
 - syntax 130
 - using 51
- event reason codes 171
- EXPAND keyword 125
- Extenders
 - example of use 3

- external files
 - changing path/name in handle 61
 - extracting path/name from a handle 60
 - initializing handles 115

F

- fault finding 16
- FFT, document format 22
- FILE function
 - example 60
- file handle
 - See handle
- files outside DB2 UDB for OS/390 and z/OS tables
 - See external files
- flat ASCII, document format 22
- flat-file documents, section support 34
- FORMAT function
 - example 60
 - syntax 114
- format of text documents 22
 - changing in handle 61
 - converting nonsupported 23
 - default in text configuration 14
 - description 22
 - extracting from a handle 60
 - FORMAT function 114
 - FORMAT keyword 78, 84
 - GET TEXT INFO command 95
 - in CHANGE TEXT CONFIGURATION 78
 - in ENABLE TEXT COLUMN 84
 - initializing in handles 115
 - list of supported 22
- free-text search
 - example 56
- function
 - for converting data types 45
 - search functions 47
 - SET CURRENT FUNCTION PATH statement 47
 - setting the path for DB2 Text Extender functions 47
- FUNCTION keyword
 - description 45
 - in ENABLE TEXT COLUMN 84
- functions
 - CCSID 112
 - CONTAINS 113
 - description 47
 - FORMAT 114
 - function path 47
 - improving search performance 62
 - INIT_TEXT_HANDLE 115
 - LANGUAGE 116
 - NUMBER_OF_MATCHES 117
 - overview 111
 - RANK 118
 - reference 111
 - REFINE 119
 - refining a previous search 56
 - SEARCH_RESULT 120
 - searching for text 47
 - setting and extracting information in handles 58

functions (*continued*)
 specifying search arguments 49
 SQL states returned by 155
FUZZY FORM OF keyword 126
fuzzy search 30
fuzzy search, example 54

G

GET ENVIRONMENT command
 example and output 66
 syntax 89
GET INDEX SETTINGS command
 example and output 69
 syntax 90
GET INDEX STATUS command
 example and output 67
 syntax 92
GET STATUS command
 example and output 65
 syntax 93
GET TEXT CONFIGURATION command
 example and output 66
 syntax 94
GET TEXT INFO command
 example and output 69
 syntax 95
getting started 19

H

handle
 CCSID function 112
 changing format and language 61
 distinct type DB2TEXTH 111
 extracting CCSID, format, and language 60
 FORMAT function 114
 initializing 59
 LANGUAGE function 116
 setting and extracting information in 58
 using lists to improve performance 62
HANDLE function
 using 62
HANDLE_LIST function
 using 62
help for commands 41
hit
 See match
HTML documents, section support 34
HTML structured documents 34
HTML, document format 22
hybrid search, example 56

I

IN SAME PARAGRAPH AS keyword 125
IN SAME SENTENCE AS keyword 125
index
 backup and restore 15
 CASE_ENABLED option 32
 CHANGE INDEX SETTINGS command 76

index (*continued*)
 CHANGE TEXT CONFIGURATION command 77
 changing the index type 32
 changing the text configuration 14
 common-index table 32
 creating various types for a text column 44
 default type in text configuration settings 14
 displaying the current settings 69
 GET INDEX SETTINGS command 90
 GET INDEX STATUS command 92
 GET TEXT CONFIGURATION command 94
 immediate index update 63
 INDEXOPTION in ENABLE TEXT COLUMN 86
 INDEXTYPE in CHANGE TEXT CONFIGURATION 77
 INDEXTYPE in ENABLE TEXT COLUMN 85
 linguistic 30
 maintaining 63
 multiple, using 32
 Ngram 32
 overview 21
 planning 21
 precise 31
 reorganizing 65
 size calculation 33
 TABLESPACE in CHANGE TEXT CONFIGURATION 77
 types of 29
 UPDATE INDEX command 99
index characteristics
 defaults in text configuration settings 14
 displaying 69
 in ENABLE TEXT COLUMN 83
index status, displaying
 example and output 67
 syntax 92
index status, resetting
 example 64
 syntax 98
index type, changing
 changing 32
 creating various types for a text column 44
indexing events
 reason codes 171
indexing events, deleting
 example 64
 syntax 79
indexing, linguistic processing 133
INDEXOPTION keyword
 in CHANGE TEXT CONFIGURATION 77
 in ENABLE TEXT COLUMN 86
INDEXPROPERTY keyword
 in ENABLE TEXT COLUMN 86
INDEXTYPE keyword
 in CHANGE TEXT CONFIGURATION 77
 in ENABLE TEXT COLUMN 85
information about text documents
 CCSID 25
 displaying the current setting 69
 format 22
 GET TEXT INFO command 95

- information about text documents (*continued*)
 - language 25
 - types of 22
- INIT_TEXT_HANDLE function
 - example 59
 - syntax 115
- initializing a handle
 - how to 59
 - INIT_TEXT_HANDLE function 115
- instances
 - creating 15
 - dropping 15

L

- LANGUAGE function
 - example 60
 - syntax 116
- LANGUAGE keyword 78, 84
- language of text documents
 - changing in handle 61
 - default in text configuration 14
 - description 25
 - extracting from a handle 60
 - GET TEXT INFO command 95
 - in a search argument 53
 - initializing in handles 115
 - LANGUAGE function 116
 - list of 25
- language parameters, list of 36
- large tables, enabling 45
- linguistic index
 - description 30
 - search option defaults 126
- linguistic processing
 - basic text analysis 134
 - character masking 138
 - description 133
 - for retrieval 136
 - masking 138
 - reducing terms to base form 135
 - sound expansion 137
 - splitting compound terms 135
 - stop-word filtering 135
 - synonyms 137
 - when indexing 133
 - word masking 138
- log space, running out of 45
- log table
 - assigning to a tablespace 44
 - creating 43
 - description 22
 - extracting error events 68
- LOGPRIMARY, LOGSECOND, and LOGFILSIZ
 - parameters in DB2 UDB for OS/390 and z/OS 45

M

- masking
 - in a search term 51
 - linguistic processing 138

- match
 - in a search result 49
 - NUMBER_OF_MATCHES function 117
- messages 155
- Microsoft, document format 22
- multiple indexes, using 32

N

- national language support 27
- Ngram index
 - CASE_ENABLED option 32
 - description 32
 - search option defaults 126
- node
 - nodegroups and tablespaces 44
- normalization of terms 134
- NORMALIZED keyword
 - in ENABLE TEXT COLUMN 86
- NOT
 - Boolean operator 54
 - keyword in search argument 125
- NUMBER_OF_MATCHES function, syntax 117

O

- occurrences of a search term 117
- OR Boolean operator 50
- overview of DB2 Text Extender 3

P

- performance, improving 62
- PRECISE FORM OF keyword 126
- precise index
 - description 31
 - search option defaults 126
- precise search 31
- processing characteristics
 - defaults in text configuration settings 14

Q

- QUIT command
 - syntax 96
 - using 46

R

- rank
 - in a search result 49
- RANK function
 - example 49
 - syntax 118
- reason codes from the search engine 169
- recognizing sentences 135
- reducing terms to base form 135
- REFINE function
 - example 56
 - syntax 119

- refining a previous search 56
- REORGANIZE INDEX command
 - example 65
 - syntax 97
- RESET INDEX STATUS command
 - example 64
 - syntax 98
- restrictions for search arguments 130
- RESULT LIMIT keyword 125
- retrieval, linguistic processing for 136
- return codes 149
- rules for search arguments 130

S

- sample table
 - deleting 72
- search argument
 - | (OR) operator 125
 - & (AND) operator 125
 - AND keyword 126
 - attribute value 124
 - BOUND keyword 126
 - bound search 55
 - COUNT keyword 125
 - description 121
 - EXPAND keyword 125
 - free-text search 56
 - FUZZY FORM OF keyword 126
 - fuzzy search 54, 126
 - hybrid search 56
 - IN SAME PARAGRAPH AS 125
 - IN SAME SENTENCE AS 125
 - MODEL keyword 123
 - NOT keyword 125
 - PRECISE FORM OF keyword 126
 - RESULT LIMIT keyword 125
 - searching for parts of a term 51
 - searching for several terms 49
 - searching for similar-sounding words 55
 - searching for synonyms 53
 - searching for terms in any sequence 52
 - searching for terms in document sections 52
 - searching for terms in the same paragraph 52
 - searching for terms in the same sentence 52
 - searching for terms in various languages 53
 - searching for variations of a term 50
 - searching with & and ! 50
 - searching with NOT 54
 - SECTION keyword 124
 - specifying 49
 - STEMMED FORM OF keyword 126
 - summary of rules and restrictions 130
 - SYNONYM FORM OF keyword 126
 - syntax 122
 - TERM OF keyword 125
 - THESAURUS keyword 124
 - thesaurus search 55
 - using masking characters 51
- search engine reason codes 169

- search status, displaying
 - example and output 67
 - syntax 92
- search status, resetting
 - example 64
 - syntax 98
- SEARCH_RESULT function
 - example 62
 - syntax 120
- searching for text
 - getting the number of matches found 49
 - getting the rank of a found document 49
 - improving performance 62
 - making a query 48
 - overview 47
 - REFINE function 119
 - refining a previous search 56
 - SEARCH_RESULT function 120
 - syntax 122
- sections in documents
 - attribute value in search syntax 124
 - DESMODEL.INI 34
 - document models file, contents 34
 - enabling section support 34
 - flat-file documents 34
 - HTML documents 34
 - MODEL keyword in search syntax 123
 - search example 52
 - SECTION keyword in search syntax 124
- sentence recognition 135
- sentence separation 31
- server
 - backup and restore 15
 - connecting to 41
 - DESTRACE command 106
 - DISABLE SERVER FOR DB2TEXT command 80
 - disabling 72
 - displaying the status 103
 - ENABLE SERVER FOR DB2TEXT command 82
 - enabling 41
 - GET STATUS command 93
 - setting up and maintaining 15
 - starting 102
 - status information, displaying 65
 - stopping 104
 - tracing faults 16
 - TXSTART command 102
 - TXSTATUS command 103
 - TXSTOP command 104
- SET CURRENT FUNCTION PATH statement 47
- shell profiles 13
- sounds expansion
 - description 137
 - example 55
- space requirements for indexes 33
- SQL data types
 - See data types of text documents
- SQL states returned by DB2 Text Extender
 - functions 155
- starting the DB2 Text Extender server 102

- status of an index
 - displaying 67
 - displaying the current status 92
 - resetting 64
 - resetting after an error 64
- status of the DB2 Text Extender server 103
- STEMMED FORM OF keyword 126
- stop words
 - as a part of basic text analysis 135
 - description 21
 - editing a stop-word file 37
 - lists of 36
- stopping the DB2 Text Extender server 104
- structure of documents
 - attribute value in search syntax 124
 - enabling section support 34
 - MODEL keyword in search syntax 123
 - search example 52
 - SECTION keyword in search syntax 124
- synonyms
 - description 137
 - in a search argument 53
 - SYNONYM FORM OF keyword 126

T

- table
 - See text table
- tablespace 43
- tablespaces and nodegroups 44
- TERM OF keyword 125
- text characteristics
 - CCSID 25
 - defaults in text configuration 14
 - format 22
 - in ENABLE TEXT COLUMN 83
 - language 25
- text column
 - See column
- text configuration settings
 - changing 14
 - displaying 66
 - installation defaults 13
- text document
 - See document
- text handle
 - See handle
- text index
 - See index
- text table
 - backup and restore 15
 - enabling a column in a large table 45
- TEXTINDEXES catalog view
 - content 70
 - creating 41
 - deleting 72
- thesaurus search
 - compiling a thesaurus definition file 105
 - concepts 138
 - creating a thesaurus 141
 - example 55

- thesaurus search (*continued*)
 - IMOTHESC command 105
 - syntax 124
 - THESAURUS keyword 124
- TMTHESC command
 - syntax 105
- tracing faults
 - DESTRACE command 106
 - setting up 16
- triggers
 - creating 43
 - description 22
- TXICRT command
 - creating a DB2 Text Extender instance 15
- TXSTART command
 - syntax 102
 - using 15
- TXSTATUS command
 - syntax 103
 - using 15
- TXSTOP command
 - syntax 104
 - using 15
- types of text index
 - CASE_ENABLED option 32
 - CHANGE TEXT CONFIGURATION command 77
 - default in text configuration settings 14
 - GET INDEX SETTINGS command 90
 - INDEXTYPE in CHANGE TEXT CONFIGURATION 77
 - INDEXTYPE in ENABLE TEXT COLUMN 85
 - linguistic 30
 - Ngram 32
 - precise 31
 - search option defaults 126

U

- UDTs 111
- update frequency
 - GET INDEX SETTINGS command 90
- UPDATE INDEX command
 - example 63
 - syntax 99
- update status, displaying
 - example and output 67
 - syntax 92
- update status, resetting
 - example 64
 - syntax 98
- UPDATEINDEX keyword
 - default in text configuration settings 14
 - displaying the current setting 69
 - GET INDEX SETTINGS command 90
 - in CHANGE TEXT CONFIGURATION 78
 - in ENABLE TEXT COLUMN 87
- user exit, document format conversion 23
- user-defined functions
 - See functions

V

variables

- description of environment variables 13
- displaying environment variables 66
- GET ENVIRONMENT command 89

W

wild-card characters

- in a search term 51
 - word masking 138
- word separation 31
- WordPerfect, document format 22

Readers' Comments — We'd Like to Hear from You

DB2 Universal Database for OS/390 and z/OS
Text Extender
Administration and Programming
Version 7

Publication No. SC26-9948-02

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	<input type="checkbox"/>				

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	<input type="checkbox"/>				
Complete	<input type="checkbox"/>				
Easy to find	<input type="checkbox"/>				
Easy to understand	<input type="checkbox"/>				
Well organized	<input type="checkbox"/>				
Applicable to your tasks	<input type="checkbox"/>				

Please tell us how we can improve this book:

Thank you for your responses. May we contact you? Yes No

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

Name

Address

Company or Organization

Phone No.



Fold and Tape

Please do not staple

Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept 0446
Schoenaicher Strasse 220
71032 Boeblingen
Germany

Fold and Tape

Please do not staple

Fold and Tape



Program Number: 5675-DB2

SC26-9948-02

