z/OS

Text Search: Installation and Administration for the Text Search Engine

Version 1.2

SH12-6716-01

z/OS

Text Search: Installation and Administration for the Text Search Engine

Version 1.2

SH12-6716-01

Note!
Before using this information and the product it supports, be sure to read the general information under "Appendix D. Notices" on page 79.

Second Edition, October 2001

This edition applies to Version 1.2 of z/OS (5694-A01) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this book

The Text Search Engine is part of the IBM z/OS Text Search element. The Program Directory for z/OS Version 1.2, GI10-4001, describes the SMP/E installation. This book describes how to complete the final installation and customization of the Text Search Engine. It also lists the commands you can use for index, document, and thesaurus administration.

Who should read this book

Read this book if you are responsible for setting up and managing the Text Search Engine client/server environment. This book is also for people responsible for the creation and administration of indexes and the administration of documents.

Conventions used in this book

The conventions used in this book are:

- Parameters between **[** and **]** are *optional*, other parameters are required.
- Characters following the **-** (minus) sign are *control characters*.
- Parameters between < and > must be replaced by their actual value.
- Parameters without any additional notation are *keywords* and must be specified as shown.
- z/OS NetQuestion Version 1.x refers to the following FMIDs: HIMN110, HIMN120, and HIMN130.

The Text Search Library

The following books are available for z/OS Text Search:

- z/OS Text Search: Installation and Administration for the Text Search Engine, SH12-6716
- z/OS Text Search: Programming the Text Search Engine, SH12-6717
- z/OS Text Search: NetQuestion Solution, SH12-6718

Chapter 1. Introduction

This chapter introduces you to the Text Search Engine environment and to some of the concepts that are applicable to the Text Search Engine.

The Text Search Engine environment

The Text Search Engine consists of a server component, a client component, and resources, for example, dictionaries and thesaurus files. You can install these components in the following combinations on any machine:

- The client component and resources
- The server component and resources
- Both the client and server components and resources

Figure 1 shows how you might set up your Text Search Engine environment:

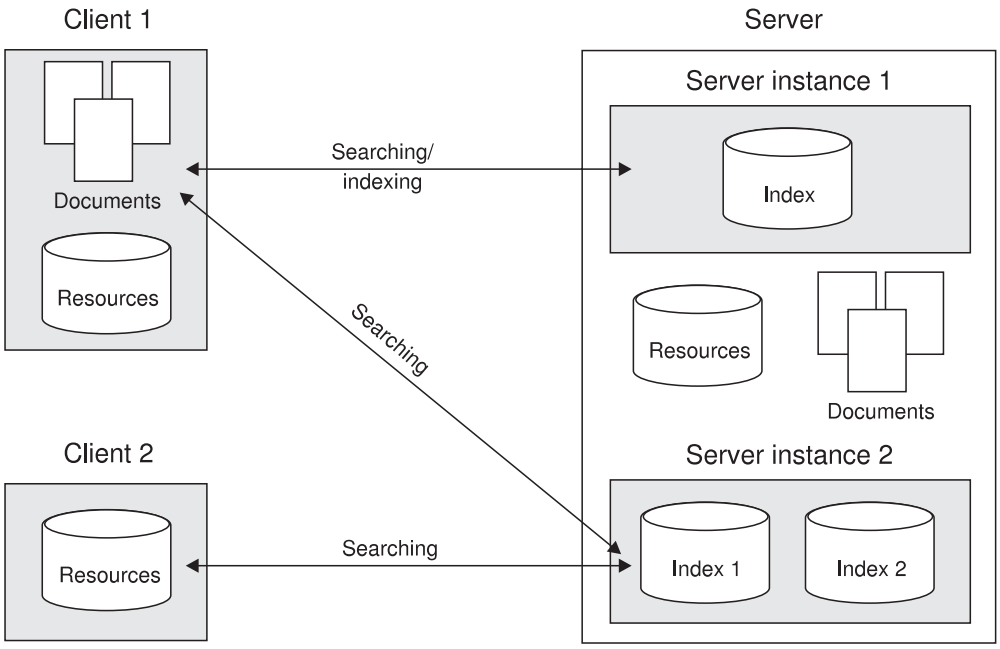

Figure 1. Example of a Text Search Engine environment

Client The client manages access to the Text Search Engine server. For example, it is the interface for building and maintaining indexes, and it provides access to search and result-list handling. It contains a dynamically loadable module for programming text applications and a command-line interface for issuing administration commands that access the API functions. Configuration files on the client define where resources are located, and the communication method used for connecting to the Text Search Engine server. A client can connect to different servers located on different machines.

Server A Text Search Engine server is known as a *search server instance.* Several server instances can be configured on the same machine. A

set of indexes is owned by a server instance. This means that server instances are independent of one another and they can run in parallel. Each server instance:

- Maintains one set of indexes
- Allows for up to 100 parallel processes for client sessions
- Uses one set of configuration files defining resources and the default behavior of the server instance

A server is a daemon process that must be running to be connected to by a client.

Resources Resources include dictionaries, thesaurus files, stop-word lists, and abbreviation files.

Dictionaries support the linguistic processing of documents during indexing and retrieval. The U.S. English dictionary is always installed on every machine. When you install a dictionary, the stop words for that language are also installed.

Thesaurus files can be used during a search for expanding query terms. Two sample thesaurus files are available; one for ngram indexes (imonthes.*) and one for all the other index types (imothes.*).

For a complete list of the resource file names and code pages for each of the supported languages, see "Appendix B. Dictionary files" on page 75.

Client/server communication

The Text Search Engine supports the following communication methods for a client/server environment:

TCP/IP

TCP/IP allows clients on the same or different machines to access the Text Search Engine server. The server starts a daemon process that controls access to the index from each of the clients. It also starts communication processes to accept client requests.

If you have a lot of clients running in parallel and you receive an error message that the server is busy, you should increase the number of client processes running in parallel. To do this, use the `imcfgsv` command to increase the number of running tasks.

Local Local communication allows clients to be connected to a server only if they are on the same machine. The server starts a daemon process that controls access to the index only. A client can connect to only one server at any one time (in one process).

> **Note**
> Local communication is not thread reentrant. Before you can use local communication, ensure that the applications using this mode serialize the client requests within the application.

Text Search Engine concepts

The following are general concepts that apply throughout the Text Search Engine components. These concepts are especially important in a heterogeneous environment with, for example, clients on workstation platforms and the server on z/OS.

Configuration files

There are several files available for configuring the Text Search Engine. These are in a flat-file format and can be changed using any editor available on your system.

Character data

Whenever character data is passed in interfaces, especially across systems or environments, character encoding problems might arise. To overcome this problem, the Text Search Engine has rules for input data.

There are rules for:

- Text Search Engine names (such as index names or server names)
- External names (such as document names)
- Document identifiers (and document group identifiers)
- Document text
- Search terms

These rules are described in *z/OS Text Search: Programming the Text Search Engine*.

Chapter 2. Installing and customizing the Text Search Engine

The IBM Text Search Engine and the NetQuestion Solution are shipped together as the z/OS Text Search element, HIMN230. The Program Directory for z/OS Version 1.1 describes the SMP/E installation of both components. This chapter describes the final installation and customization of the Text Search Engine in the z/OS UNIX (OpenEdition®) after the SMP/E installation.

Check the README file before carrying out the steps described in this chapter. It might contain important information about late changes to the installation and configuration process. If you used the default directories during SMP/E installation, the README file is in */usr/lpp/TextTools/readme*.

Getting started

The z/OS Text Search load library hlq[1].SIMOMOD1 can be accessed in one of the following ways from the z/OS UNIX® environment:

- By adding hlq.SIMOMOD1 to the PARMLIB member (PROGxx) or (LNKLSTxx).
- By adding hlq.SIMOMOD1 to the STEPLIB environment variable

For performance reasons, it is recommended that you add hlq.SIMOMOD1 to one of the PARMLIB members. This also simplifies the customization steps described in this chapter.

Before you start the final installation, ensure that:

- For administration, you have an administration group and a user ID set up. These IDs are usually created during the installation of the IBM HTTP Server and, by default, they are WEBADM and IMWEB respectively.
- The PARMLIB(BPXPRMxx) member includes the following:

  ```
  IPCSEMNSEMS(50)
  IPCSHMMPAGES(2048)
  ```

- Your PATH environment variable contains either "." or "./". Use the command `echo $PATH` to check the setting. To change the PATH setting to include the period:
  1. Enter

     ```
     export PATH=$PATH:.
     ```

  2. Update the system profile (/etc/profile) or your user profile (/u/<user>.profile)
- If you use PROGxx or LNKLSTxx, the corresponding PARMLIB member contains the entry **hlq.SIMOMOD1**.

  The hlq.SIMOMOD1 data set must also be APF-authorized. This is achieved if the IEASYSxx member of the PARMLIB defaults to LINKAUTH=LNKLST.
- If you use the STEPLIB environment variable:
  - You have either RACF™ READ access for the z/OS load library hlq.SIMOMOD1 or RACF EXEC access and there is a PROGRAM profile for all members in hlq.SIMOMOD1.
  - The PARMLIB(BPXPRMxx) member contains an entry for the sanction list, for example, STEPLIBLIST('/system/steplib').

1. High-level qualifier

  - The sanction list, for example, the HFS file '/system/steplib' contains the entry **hlq.SIMOMOD1**.
- If you are migrating from any release of OS/390 Version 2 to Version 2.10, you do not have any data files in directories belonging to NetQuestion 1.x. These directories are **deleted** when you install z/OS Text Search. If you have stored the document lists for indexing documents in these directories, save them to another directory; you can use them for migrating your indexes. For information on migrating from OS/390 NetQuestion Version 1.x, see "Migrating from OS/390 NetQuestion Version 1.x" on page 11.

Installing the Text Search Engine

The z/OS system provides menu screens for finalizing the installation. Online help is available for the installation. The log file, imoinst.log, tracks the installation events.

1. Log on with a user ID that has superuser privilege.

   Tip
   Do not use the `su` command to become a superuser because the permission rights might not be sufficient to complete the installation successfully.

2. Change the current working directory to the directory that contains the REXX procedures and z/OS UNIX shell scripts by using the command:
   ```
   cd /usr/lpp/TextTools/install
   ```
   Ensure that you have write access to */usr/lpp/TextTools/**.
3. From an z/OS UNIX command line, enter:
   ```
   imoinst
   ```

   Tip
   If you install the Text Search Engine on a driving system and move it later to the target system, you must run imoinst from the driving system

   The Install z/OS Text Search Engine panel is displayed.

```
    REXX Procedure imoinst - Install z/OS Text Search Engine

 Type in selection number to execute or type in ?n (n = selection number)
 to get help and press ENTER.

   0. End
   1. Basic installation
   2. Optional installation steps
   3. PTF installation and information



Please enter your selection:
= = = >
```

4. Select **1. Basic Installation**.
   The Install z/OS Text Search Engine panel is displayed:

```
    REXX Procedure imoinst - Install and Customize IBM Text Search Engine

Type in selection number to execute or type in ?n (n = selection number)
to get help and press ENTER.


    0.  End
    1.  Customize installation and application parameters - file imoparm
    2.  Display the list of all parameters using 'pg imoparm'
    3.  Process final installation steps
    4.  Display final installation log file using 'pg imoisfin.log'
    5.  Set up NetQuestion Solution using 'inqsetup'
    6.  Display activity logging file using 'pg imoinst.log'
    7.  Enter your own shell command

 Please enter your selection:
 ===>
```

5. Select **1. Customize installation and application parameters**.

   The default settings for the installation parameters are:

   ```
   IMODIRPATH=/usr/lpp
   IMOINSTDIR=TextTools
   IMOLOADLIB=IMO.SIMOMOD1
   IMOADMGRP=IMWEB
   IMOADMUSER=WEBADM
   LANG=C
   IMOINSTANCE=inqsrch
   IMOSEARCHSERVICE=SERVER
   INQLANGNO=1
   ```

   To check the current settings of these parameters, select **2. Display the list of all parameters**. If you do not want to use a specific user ID or group for the customization, leave the parameters IMOADMUSER and IMOADMGRP blank.

   If you are working with a STEPLIB environment variable, ensure that the IMOLOADLIB parameter contains the full data set name of the z/OS Text Search load library, before you do any of the following steps.

   Tip

   To change installation parameters at a later date, use menu item **1** to edit the imoparm file. Whenever you change any of these parameters, you must repeat menu item **3** to update your installation.

6. **3. Process final installation steps**.

   To check the results of the final installation, select **4. Display final installation log file**.

All other steps are optional.

Moving z/OS Text Search to another directory

If you want to move z/OS Text Search to another directory, you must do some additional installation steps after the basic installation. This applies, for example, if you are working with driving and target systems.

Tip

You cannot manually copy a customized installation from one directory to another. The customized settings are lost if you do this.

To move z/OS Text Search to another directory:

1. From an z/OS UNIX command line, enter:

   ```
   imoinst
   ```

2. Select **2. Optional installation steps**.

   The Optional Installation Steps panel is displayed.

```
     REXX Procedure imoinst - Optional Installation Steps

Type in selection number to execute or type in ?n (n = selection number)
to get help and press ENTER.

NOTE:
Only Step 1 is required if you want to run/remount the installation in a
directory other than the current directory.
For Step 2, you must first run Step 1 if you want to clone this installation
into other directories.


    0.  End
    1.  Prepare this customization to run/remount in other directories
    2.  Copy a clone of this installation into other directories
    3.  Display logging file for items above


Please enter your selection:
 ===>
```

3. Select **1**.

   The current installation settings are shown.

4. Edit the settings to point to the new directories and when prompted to do so, enter **Y**.

   When this step has finished, the installation runs **only** from the new directory and cannot be run from the original installation directory.

Tip

To enable z/OS Text Search to run from the original installation directories, for example, to install a PTF, edit the settings again to point to the original installation directories.

Copying z/OS Text Search

You can copy your z/OS Text Search installation, for example, if you want to use z/OS Text Search on a new production system, or you are working with driving and target systems. If you copy your installation, you must do some additional installation steps after the basic installation.

Tip

You cannot manually copy a customized installation from one directory to another. The customized settings are lost if you do this.

To copy your z/OS Text Search installation:

1. From an z/OS UNIX command line, enter:

   ```
   imoinst
   ```

2. Select **2. Optional installation steps**.

   The Optional Installation Steps panel is displayed.

   ```
        REXX Procedure imoinst - Optional Installation Steps

   Type in selection number to execute or type in ?n (n = selection number)
   to get help and press ENTER.

   NOTE:
   Only Step 1 is required if you want to run/remount the installation in a
   directory other than the current directory.
   For Step 2, you must first run Step 1 if you want to clone this installation
   into other directories.


       0.  End
       1.  Prepare this customization to run/remount in other directories
       2.  Copy a clone of this installation into other directories
       3.  Display logging file for items above


   Please enter your selection:
    ===>
   ```

3. To specify the location of the copied installation, select **1**.

   The current installation settings are shown.

4. Edit the settings to point to the new directories and when prompted to do so, enter **Y**.

5. To copy the installation, for example, to put a test installation into production, select **2**.

   The updated settings from step 1 are shown.

6. Check that the settings for the new installation are correct. If you want to copy your NetQuestion Solution installation, update the settings for the Web server's server root directory with the new directory information.

7. Close the editor and when prompted to do so, enter **Y** and wait until this step has finished.

   You can now run both the original installation and the copy of the installation.

Customizing the installation

Before you do any of the following steps, ensure that:

- If you are working with PARMLIB members, either (PROGxx) or (LNKLSTxx) contains an entry for hlq.SIMOMOD1.
- If you are working with a STEPLIB environment variable, the IMOLOADLIB parameter contains the full data set name of the z/OS Text Search load library.

To customize the Text Search Engine installation:

1. Log on with a user ID that has superuser privilege.
2. Change the current working directory to the directory that contains the REXX procedures and z/OS UNIX shell scripts by using the command:

   ```
   cd <install path>
   ```

3. From an z/OS UNIX command line, enter:

   ```
   imocust
   ```

Tip

If you install the Text Search Engine and move it later to the target system, you must execute imocust on the target system.

The Customize z/OS Text Search Engine panel is displayed:

```
    REXX Procedure imocust - Customize z/OS Text Search Engine

Type in selection number to execute or type in ?n (n = selection number)
to get help and press ENTER.


   0.  End
   1.  Install Customization Data into the etc directory
   2.  Check/adapt environment settings - file imoexport
   3.  Process installation verification procedure
   4.  Configure local instance and search service using default names
   5.  Erase NetQuestion Version 1.x data files
   6.  Display activity logging file using 'pg imocust.log'
   7.  Enter your own shell command

 Please enter your selection:
 ===>
```

4. Select **1. Install Customization Data into the etc directory**. The default setting for the TextTools Customization directory is `IMOETCPATH=/etc/TextTools`. Close the editor, and when prompted, enter Y.
5. Select **2. Check and adapt environment settings file `imoexport`**.

   If you are working with a STEPLIB environment variable, ensure that the IMOLOADLIB parameter contains the full data set name of the z/OS Text Search load library, before you do any of the following steps.
6. Run the installation verification test by selecting **3. Process installation verification procedure.**

Select **4** to create and configure a local instance and client. If you already have z/OS Text Search installed, you can use your existing instance and client. All other steps are optional.

Continue with the steps described in "Finalizing the installation".

Finalizing the installation

When your installation is complete, add the file `imoexport` from the TextTools customization directory, `/etc/TextTools/imoexport` if you use the default setting, to the user profiles of those users who use the Text Search Engine administration commands. You can check the installation, by using the following commands to start, check, and stop the search service, *inqsrch*:

```
. imoexport
imoss  -start  inqsrch
imoss  -status  inqsrch
imoss  -stop  inqsrch
```

Migrating from releases of OS/390 Text Search

During SMP/E installation, ensure that you specify the existing directories for OS/390 Text Search. This ensures that the installation parameters are kept and that the existing search instances and services with their associated indexes are also kept.

- Migrating from OS/390 Version 2.8 or later

  The search instances and services with their associated indexes are all kept. No migration is required.

  Tip

  When you call `imocust` to customize the Text Search Engine, **do not** carry out step 4. Configure local instance and search service using default names because this will reconfigure your existing installation.

- Migrating from OS/390 Version 2.7 or earlier

  The search instances and services with their associated indexes are all kept. No migration is required. However, if you still have OS/390 NetQuestion Version 1.x installed, you should run step 6 from the imocust menu. This deletes all the files associated with NetQuestion, including the data and work directories.

Migrating from OS/390 NetQuestion Version 1.x

The following describes how to migrate your OS/390 NetQuestion 1.x indexes, applications, and shell scripts to z/OS Text Search:

- Indexes

  Indexes you created using the OS/390 NetQuestion Version 1.x have a different format to those created by z/OS Text Search and therefore they cannot be re-used. You must reindex your documents using z/OS Text Search. If you have copies of the document lists that you used to index the documents with NetQuestion, you can use these when reindexing your documents.

- Applications

  The z/OS Text Search APIs differ slightly from the OS/390 NetQuestion APIs. If you have written applications using the OS/390 NetQuestion APIs, you must rebuild your applications using the z/OS Text Search APIs. The *z/OS Text Search: NetQuestion Solution* contains information to help you convert your applications. If you used the default installation directories, the API files are in the directory *TextTools/TextSearch/samples*.

- Shell scripts

  As with NetQuestion, you can invoke the adminstration functions directly from the command line. However, the commands provided by z/OS Text Search differ from those of OS/390 NetQuestion. If you have built shell scripts using the NetQuestion command-line calls, wrapper code is provided so that you can continue to use the shell scripts with z/OS Text Search. If you used the default installation directories, the wrapper code is in the directory *TextTools/TextSearch/samples/comwrap*. The *z/OS Text Search: NetQuestion Solution* contains information on the differences in the command-line calls.

  If you have shell scripts that check return codes, you might need to modify these when you move to z/OS Text Search because some of the return codes differ from those of z/OS NetQuestion. The file *...include/imoapic.h* lists the return codes for z/OS Text Search.

Chapter 3. Starting and stopping a search server

A Text Search Engine server is known as a *search server instance*. This chapter describes how to start and stop a server instance on the supported platforms.

To start, stop, or query the status of a server instance:

1. Log on to the Text Search Engine server. Ensure that your user ID belongs to the administration group.
2. Ensure that you have added the file imoexport to your user profile. If you have not, change to the *.../bin* subdirectory, and use the `. imoexport` (period blank imoexport) command to set your environment.
3. You can start, stop, or query the status of a server instance from a command prompt using the `imoss` command:
   - To start a server instance, enter:

     ```
     imoss -start <ServerInstanceName>
     ```
   - To stop a server instance, enter:

     ```
     imoss -stop <ServerInstanceName>
     ```
   - To query the status of a server instance, enter:

     ```
     imoss -status <ServerInstanceName>
     ```

   During final installation, the server instance *inqsrch* is created. Use this as the `ServerInstanceName`.

There can be serveral server instances running on the same machine. You can start, stop, or query any one of these server instances independently of the others.

Chapter 4. Planning your document indexes

This chapter helps you to plan before you index documents for the first time. It explains:

- Why documents need to be indexed
- Which types of documents can be indexed, and which document libraries and file systems you can use
- How to decide which type of index to use
- How the index type and configuration settings affect performance
- The use of multiple indexes
- How to enable section support

Why documents need to be indexed

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

The retrieval system searches through the index for the terms requested, and finds the names of the documents containing those terms.

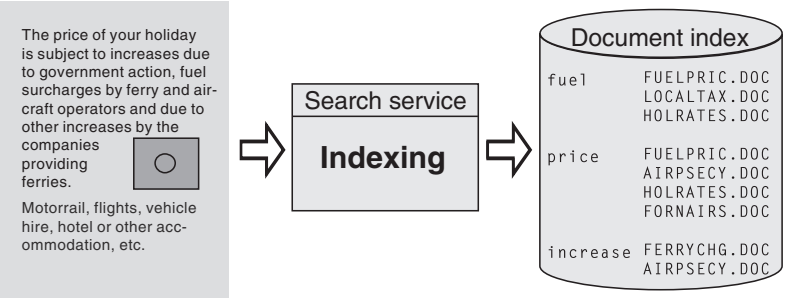

Figure 2. Rapid retrieval by indexing only meaningful terms

A document index contains only relevant information; insignificant information (stop words), such as conjunctions and prepositions, are not indexed. The Text Search Engine uses a list of these stop words to prevent them from being indexed.

Which types of documents and library systems are supported

You can index different document types, including HTML documents, XML documents, and flat files. For a complete list of the document formats that can be indexed, refer to the IMOLSDEF.H header file in the *.../include* subdirectory.

You can index and search for documents in any file system supported by the operating system where the Text Search Engine is installed. For each library system, a separate document index is required because there can be only one Library Service per index.

The documents to be indexed can be located anywhere. However, the library system must ensure unique access to the documents from both the client and the server. This is done using the Library Services interface.

The Text Search Engine delivers Library Services for file-system access:

- On Windows NT when the clients and server are not installed on the same machine, schedule the documents for indexing using UNC names.
- On UNIX platforms, the documents must be located on the server or accessible via NFS/AFS mount points. If the code pages on the clients and the server are not the same, for example, ASCII on the clients and EBCDIC on the server, you must write your own set of Library Services routines to take account of this. One approach to writing the Library Services is to use the Distributed Computing Environment's (DCE) Distributed File Service. This can accommodate both the document access and the differences in the code pages.

For further information on the Library Services supplied with the Text Search Engine, see *z/OS Text Search: Programming the Text Search Engine*.

Using unsupported document formats

If you have documents in a format not supported by the Text Search Engine, you must write a user exit program or command file that converts the documents into another document format. The user exit must be registered in both the server configuration file (IMOSRV.INI) and the client configuration file (IMOCL.INI). Update the USEREXIT option in the [DOCUMENTFORMAT] section with the name of the user exit. For information on the configuration files, see "Appendix A. Configuration files" on page 71.

To call the user exit, use the following syntax:

```
<name_of_executable> -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 the Text Search Engine. 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 you specified using the `imorulix` command.

targetccsid
The code page expected by the Text Search Engine. For workstation platforms the code page is 850 and for z/OS it is 500.

sourceformat
The format of the source file. This is the default format specified using the `imorulix` command.

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

The user exit is called for all document formats above the value of EHW_USER_FORMATS defined in IMOLSDEF.H. To change this behavior, change the options in the [DOCUMENTFORMAT] section in the configuration files.

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 `imomsgix` command to display the error messages.

Deciding which type of document index to use

You can choose one of these index types:

- Linguistic index
- Precise index
- Normalized precise index
- Ngram index

This section summarizes the index types.

Linguistic index

If you choose to create a linguistic index, then, during indexing, linguistic processing is applied while analyzing the documents' text.

For a query, the Text Search Engine then applies the same linguistic processing to the search terms before searching in the document index. The result is that any form of a search term matches any other form occurring in one of the indexed documents.

For example, 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.

The advantage of this type of index is that more documents are likely to be found (increased "recall").

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

The linguistic processing used to index documents for a linguistic index are:

- Word and sentence separation.
- Changing terms to a standard form, in which there are no capital letters, and accented letters, such as ü are changed to a form without accents (normalization). For example, the German term `Grüße` is indexed as `gruesse`.

- Reducing terms to their base form (lemmatization). For example, `bought` is indexed as `buy`, `mice` as `mouse`.
- Word decomposition, where compound words, such as the German `Wetterbericht` (weather report) are indexed not only as `wetterbericht`, but also as `wetter` and `bericht`.
- Stop-word filtering in which only the relevant terms are extracted for indexing. `A report about all animals` is indexed as `report` and `animal`.

Even word fragments (words masked with global characters) are compared to the base forms in the index. For example, a document containing the text `I swum` being searched for all words beginning with `swu` using the word fragment `swu*` is not found, because the index contains only the term `swim`. So the query should be `swi*` instead.

Precise index

In a precise index, linguistic processing is done only to determine word and sentence boundaries. The terms in the documents are indexed in exactly the same form as they occur in the document.

For example, the search term *mouse* matches only the document term *mouse* and not, for example, *MICE* or *Mouse*.

The advantage of this type of index is that the search is more precise (reduced recall); also indexing and retrieval is faster. Because each different form and spelling of every term is indexed, more disk space is needed.

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

- Word and sentence separation
- Sentence-begin processing
- Stop-word filtering

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 only the same form or inflection of a search term is found. Generally, the search in a precise index is case sensitive.

Normalized precise index

The normalized precise index differs from the precise index:

- It is case insensitive (all words are converted to lowercase except words that are written in all uppercase).
- Words in all uppercase are not subject to stop-word filtering (the word UK, for example, is indexed).
- English language search terms can be expanded to include lemma forms using a heuristic algorithm (looking for `house` looks also for `houses`).

Ngram index

Indexing and search in this index type is based on n-grams, that is, limited-length character sequences. There is no linguistic processing involved at all. This technology enables high-performance indexing, and search using both exact and fuzzy matching. This index not only supports English, but it is also optimized for double-byte character set languages: Japanese, Simplified and Traditional Chinese,

and Hanguel (Korean). You can create both case-insensitive ngram indexes (NGRAM) and case-sensitive ngram indexes (NGRAMCS).

Using multiple indexes

Using multiple indexes can bring many benefits:

- Searching in several library systems

  An index is restricted to documents belonging to a particular library system, such as the UNIX file system. If you intend to search in documents that are stored in more than one library system, then you must create at least one index for each library system.

- Searching in a particular subject area

  You can group your documents so that each subject area is indexed separately. By searching in one index, you automatically restrict the search to documents in a particular subject area. You can widen the search to documents in other subject areas by doing a cross-index search.

- Speeding up indexing

  As an index grows larger, it takes longer to index documents and to merge the index. By creating an additional index for new documents, the maintenance time for this index, because it is still small, is relatively short. This procedure is particularly suitable for indexing documents whose content does not change. The indexes could be assigned to documents according to the documents' creation date, such as one index for 1998 documents, another for 1999 documents, and so on.

Enabling 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 or XML 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.

A *document models 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. Use the `imocrdm` command to create document models for the server instance. These models are appended to IMOMODEL.INI.

When you create an index, you can specify a list of models that are to be used by the index for section support:

```
imocrix -s SERVER -x TESTIX -t LING -sections "sample, PLAY" -p /indexes/TESTIX
```

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.

```
imosrch -s SERVER -x TESTIX -section sample/Author -term McDaniel
```

For information on how to include section support in your applications, refer to the sections on EhwCreateIndex and EhwSearch in *z/OS Text Search: Programming the Text Search Engine*.

Comparison of model names is case sensitive, but comparison of section tags can be made case insensitive by using the keyword CASE_IGNORE when defining a document model. For example, if the option is selected, it would not matter whether <title> or <Title> had been used in the title section of HTML documents.

The model file entry will then look like:

```
; list of document models
[MODELS]
modelname = sample, ci
```

Note, that the above model property is invalid for XML document models. By specification, they must use case sensitive names for elements.

Supported characters for document model name, section name and tag are restricted to [a-z,A-Z,0-9].

HTML attributes like CONTENT will be indexed as belonging to the section recognized prior to their occurence - which may be the no-section part of the document.

Searches may retrieve documents containing a combination of CONTENT and some text by specifying an ANDed boolean query (recommended, if not too fuzzy for the document collection), or by using proximity operators like ″same paragraph″. XML attributes are being indexed as text belonging to the section which was defined for the element where they occurred.

Flat files and HTML documents

For flat files, the sections are marked up using HTML-like tags, for example, <title>, <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 are case sensitive and they can contain only A-Z, a-z, and 0-9.

```
;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  - section name identifier
; right - section name tag
[sample]
Title = title
Author = author
Subject = subject
Abstract = abstract
Content = content
[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 specified a list of models when you created the index, the default model is the first in the list. You can change the default model using the `imomodix` command.

XML documents

For section-enabled indexes, XML documents must be correctly structured and contain a root element. The name of the root element must be the same as one of the defined model names and the case must match. The model description in the document models file must be a subset of the document model defined in the DTD (document type definition) file for the document.

The model description must begin with the root element. For each XML element you want to use as a section, you must include its complete hierarchy in the model description. If a section is of type date, this section must be a leaf in the document model tree. Nesting of attribute sections is not supported.

For section enabled indexes, well informed XML documents containing root elements with names matching the name of a valid document model will be indexed according to that document definition.

If no matching document model is found, there will be an error message and the document will not be indexed. For indexes that are not section enabled all, or at least the well formed XML documents will be indexed.

A model description for XML documents might look as follows:

```
; list of document models
[MODELS]
modelname = LETTER
; sample for XML model definition
; left-hand side  = section name identifier encoding whole path
; right-hand side = section name tags specifying tag for each
:                   element of the path through the tree down to
:                   the specified node. Tag delimiter is /.
[LETTER]
LETTER = LETTER
LETTER/date = LETTER/DATE
LETTER/address = LETTER/ADDRESS
LETTER/address/City = LETTER/ADDRESS/CITY
LETTER/Content = LETTER/CONTENT
LETTER/Content/Greetings = LETTER/CONTENT/GREETINGS
```

An XML document might look as follows. It also shows how the sections that are not defined in the model are indexed.

```
<?xml version="1.0"?>
<!DOCTYPE LETTER SYSTEM "letter.dtd">

<LETTER>
 <HEADER>This tag has been skipped in the definition, to this text will
             be added to the section named LETTER
 </HEADER>
 <DATE>
           01.01.2000  03.02.2000
 </DATE>
 <ADDRESS>
           Text will be added to the section named LETTER/address.
   <CITY>
             Text will added to section named LETTER/address/City.
   </CITY>
 </ADDRESS>
 <CONTENT>
           Text will be added to the section named LETTER/Content.

   <NOSECTION>Text will be added to the section named LETTER/Content
              because NOSECTION is not defined.
   </NOSECTION>
   <GREETINGS>
            Text will be added to section named LETTER/Content/Greetings.
   </GREETINGS>
  </CONTENT>
</LETTER>
```

Performance considerations

The performance of the Text Search Engine during indexing and searching depends on several factors, including the index type and configuration settings for the client (IMOCL.INI) and the server (IMOSRV.INI). Use the information here together with the information in “Chapter 4. Planning your document indexes” on page 15 and “Appendix A. Configuration files” on page 71 to tune the performance of the Text Search Engine.

Space requirements for indexes

The amount of space required for indexes depends on the amount of text contained in the indexed documents. Other information in the documents, such as graphics or formatting information is not relevant to the indexing process. When only a primary index exists, it generally takes up 60-70% of the space required for the text. If both a primary and a secondary index exist, they can take up to 90% of the space required for the text.

During indexing and index reorganization, you need additional disk space for the index directory and the working directory. For the index directory, you need space equivalent to that already used by the index directory. For the working directory, the space requirements depend on the setting of the configuration option UPDATETHRESHOLD. Table 1 shows typical settings for UPDATETHRESHOLD, and the maximum space required by the working directory for linguistic and precise indexes. For example, for the default configuration, UPDATETHRESHOLD=4 000 000, you need a maximum of 115 MB for the working directory.

Note: Larger values for the UPDATETHRESHOLD increase the amount of memory and temporary disk space required. However, this decreases the indexing time.

Table 1. Size of working directories for linguistic and precise indexes

UPDATETHRESHOLD	Max. size of working directory (MB)
1 500 000	50
3 000 000	97
4 000 000	115
6 000 000	180

Time required to index documents

The time required to index documents is influenced by:

- Options in the server and client configuration that determine how fast the index process runs, for example, the amount of memory used, the number of times a reorganization takes place during an index update, and the size of the machine you are using.
- The complexity of the document format, the more complex the format the longer the Text Search Engine needs to parse it. For example, RTF documents take longer to index than plain-text documents.
- The amount of linguistic processing involved, for example, a linguistic index requires more linguistic processing than a precise index.
- The location of the document and the time taken to access it, for example, a file on a local disk can be accessed quicker than a file stored in a database.
- The location of the index. For example, you can save indexing and reorganization time if the index is on the same machine as the Text Search Engine server. You can also save time if the index directory, the working directory for the index, and the documents are all on different drives on the Text Search Engine server machine.
- The size of the documents in the document collection. If you have some knowledge about your document collection, you can set the BUFFERSEGMENTSIZE and the BUFFERSEGMENTCOUNT to other values. For example, if most of your documents are only 10 KB and only a few are up to 100 KB, you can set these options to BUFFERSEGMENTSIZE=10K and BUFFERSEGMENTCOUNT=10.

The fastest index type is the precise index. If the input buffer size is configured so that it is large enough to hold a medium-sized document and a reorganization of the index is not needed, the indexing process can handle approximately 250 MB of text an hour. With the same buffer configuration, an ngram index can handle approximately 180 MB of text an hour.

Retrieval time for a search

The time taken to return a list of documents depends on how the search query is formulated, the more complex the query the longer the search takes. For example, if you set a limit on the number of documents returned, the result will be returned quicker than if no limit is set. Other parameters, such as ranking the search results and including wildcard characters in searches on large indexes, slow the search down.

Merging the secondary index with the primary index

Merging the secondary index with the primary index can give you the following benefits:

- An index update is faster when it does not need to sort words into a large secondary index.
- Deleted documents result in 'holes' in the index. If you add a document that is already indexed, this is treated in the same way as deleting the document and then adding it again—that is, a hole appears. A search on an index without holes tends to be faster.

Do a reorganization if one of more of the following apply:

- The primary index is small compared to the secondary index.
- A lot of documents have been deleted from the index.
- When searches slow down appreciably and you have already turned tracing off.

Depending on values you set for the UPDATETHRESHOLD and UPDATESLICE options in the server configuration file, an internal index reorganization takes place automatically during an index update.

Chapter 5. Building and maintaining indexes

This chapter describes how you might build and maintain indexes using the delivered set of administration commands and Library Services. It shows how you can:

- Build a new index
- Refresh an index
- Merge and compress indexes
- Query the status of an index
- Back up and restore indexes
- Move an index to another location

The commands for building and maintaining indexes are described in full in "Chapter 6. Administration commands" on page 29.

The Text Search Engine application programming interface (API) offers other methods for building and maintaining indexes. For further information on these methods, see *z/OS Text Search: Programming the Text Search Engine*.

Building a new index

Before starting to search for information, you must build an index. Figure 3 shows an overview of the indexing process.

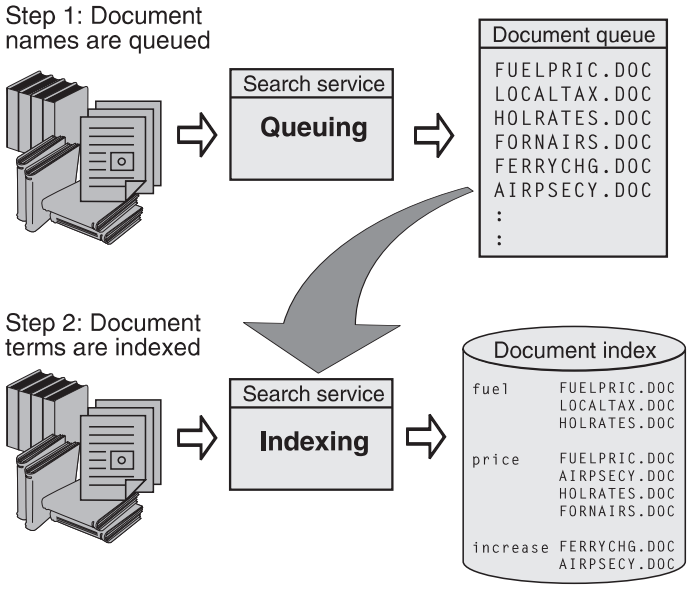

Figure 3. The indexing process

To build a new index, do the following:

1. Create an input file that lists all the files that you want to include in the index. This is called a *document list*.
2. Create an index.
3. Define the rules for the index.
4. Populate the index using the document list.

Creating the document list

The Text Search Engine builds an index of searchable keywords using the content of all the files it finds in the document list. When you create this file, each file name must be on a line by itself (that is, a new-line character must separate the names). In a client/server environment, the document access path given in the document list must be unique. Documents scheduled from the client must be accessible by the server.

For example, if all your documents are located in the same subdirectory, for example, `/usr/local/webdocs`, you can use the following command to create the document list:

```
find /usr/local/webdocs -name "*.html" -type f -print > /tmp/inputlist
```

Creating an index

Create an index using the `imocrix` command. Ensure that you have enough space for the index files and the temporary files created during indexing. For example, files for a linguistic index take up approximately 40% of the space required for the documents and the temporary files require approximately 180% of the space required for the documents.

Defining the rules for the index

The Text Search Engine supports different document formats, code pages, and languages. You can define a set of rules for each index that define default values for each of these characteristics. The rules are used during indexing if any of the values cannot be determined automatically. To specify the rules for an index, use the `imorulix` command.

Populating the index

Building the index is a two-step process; you schedule the documents for indexing, then index the documents.

1. Schedule the documents in the document list for indexing using the `imoqueue` command. To schedule several document lists for the same indexing update, issue an `imoqueue` command for each list.
2. Start the index process for the scheduled documents using the `imoupdix` command.

The index process runs in the background. You can check the status of the index update using the `imostfix` command.

The first time you use `imoupdix` on an index file, a primary index is created. Subsequent calls to `imoupdix` produce secondary indexes.

Refreshing an index

You can refresh an existing index by adding or deleting documents:

1. Create separate document lists for the documents you are adding to the index and the documents you are deleting from it.
2. Schedule the documents for adding or deleting using the `imoqueue` command.
3. Refresh the index using the `imoupdix` command.

You can refresh an index at any time. For optimum performance, however, it is best to avoid periods of peak activity. During a refresh of the index, the current version is always available for searching. When the update is complete, the index is automatically committed for searching.

Reorganizing indexes

When you build a new index with `imocrix`, `imoqueue`, and `imoupdix`, the Text Search Engine creates a set of files that form the primary index. Subsequent calls to `imoqueue` followed by `imoupdix` create additional files that form the secondary index.

As the size of the secondary index grows, the amount of redundant information between the primary and secondary index increases. Eventually, you should merge the primary and the secondary index into one index so that you can recover disk space.

Use the `imoreoix` command to reorganize indexes.

Index reorganizations can also occur automatically during an index update. The value of the options UPDATETHRESHOLD and UPDATESLICE in the configuration file for the server determine when this reorganization takes place. For further information on the configuration options, see "Server configuration file" on page 72.

Querying the status of an index

You can query the status of any Text Search Engine index using the `imostaix` command. The status report shows how many documents are in the primary index, how many are in the secondary index, and how many are in the scheduling queue.

Backing up and restoring index files

If you have system or hardware problems that cause you to lose your index data files, you can either rebuild the indexes or use backed up index data files to restore them. For information on how to build indexes, see "Building a new index" on page 25.

To back up index data files:

1. Determine the index data path for the index you want to backup. All the files belonging to an index are stored in its index data path:
   - Start the server instance if it is not already started (`imoss -start` command).
   - Use the `imostaix` command to determine the index data location.
2. Do one of the following:
   - Stop the server instance (`imoss -stop` command).
   - Use the `imoctrix` command to suspend the index.
3. Copy all the files in the index data directory to the backup medium or location.

4. Do one of the following:
   - Restart the server instance (`imoss -start` command).
   - Use the `imoctrix` command to resume the index.

To restore index data files:

1. Determine the index data path for the index you want to restore.
   - Start the server instance if it is not already started (`imoss -start` command).
   - Use the `imostaix` command to determine the index data location.
2. Stop the server instance; the server instance must be down in order to restore an index (`imoss -stop` command).
3. Copy all backed up files to the index data directory.
4. Restart the server instance (`imoss -start` command).

If you do not want to stop the server instance while you are backing up or restoring an index, you can also:

1. Deregister the index (`imoregix` command).
2. Copy the data files to or from the backup directory.
3. Register the index (`imoregix` command).

Moving an index to another location

If disk space gets low you might want to move an index from one location to another. Or, if you decide to use another machine as the Text Search Engine server, you might want to move existing index files to the new machine.

Tip

You can move an index only between Windows NT servers or UNIX servers. You cannot move an index created on a Windows NT server to a UNIX server, or vice versa.

To move an index:

1. Get information about the parameters used when the index was created:
   - Start the server instance if it is not already started (`imoss -start` command).
   - Use the `imostaix` command to view the index properties.
2. Deregister the index (`imoregix` command).
3. Copy all the files in the index data directory to the backup medium or location.
4. Register the index on the new server instance:
   a. Start the server instance for the new location (`imoss -start` command).
   b. Use the `imoregix` command to register the index. Use the parameters that were returned by the `imostaix` command in step 1. The index data path should be a directory on the new server instance.

You can also use this method to provide indexes on a read-only medium, such as a CD-ROM or use an index on one server instance that was created on a different server instance. However, you must ensure that the index is registered with only one server instance.

Chapter 6. Administration commands

Use this chapter to perform administration tasks for the Text Search Engine using the command interface. There are commands for:

- Index administration
- Document administration
- Document model administration
- Thesaurus administration
- Server administration
- Client administration

The Text Search Engine API offers other methods for building and maintaining indexes. For further information on these methods, see *z/OS Text Search: Programming the Text Search Engine*.

Index administration

You can use the following commands for index administration:

IMOCLRIX Clear index

IMOCRIX Create index

IMOCTRIX Control index: reset, suspend, and resume index

IMODELIX Delete index

IMOLSTIX List all indexes

IMOMSGIX Display indexing messages

IMOREOIX Reorganize index

IMORULIX Maintain indexing rules

IMOSTAIX Display index status

IMOSTFIX Display status of index functions

IMOUPDIX Update index

IMOCLRIX - clear index

Purpose

Use this command to remove all index entries from an existing index on the Text Search Engine server. For further information on clearing indexes, see EhwClearIndex in *z/OS Text Search: Programming the Text Search Engine*.

Tip

Use the imoclrix command with care. The index entries are cleared without you having to confirm their removal.

Syntax

```
imoclrix
        [-h| -H| -?| -copyright] [-quiet]
        -s <search service name>
        -x <index name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index containing the index entries to be removed.

Example

```
imoclrix -s SERVER -x TESTIX
```

IMOCRIX - create index

Purpose

Use this command to create a new empty index on the Text Search Engine server. For further information on creating indexes, see EhwCreateIndex in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imocrix
        [-h| -H| -?| -copyright] [-quiet]
        -s <search service name>
        -x <index name>
        -t <index type>
        -p <index data path>
        [-pw <index work path>]
        [-sections <list of models>]
        [-lsce <client library services>]
        [-lsse <server library services>]
        [-ccsid <code page for ngram index>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the created index. Index names must be unique. Index names can be up to 8 characters long.

index type
The type of index. The available index types are:

LING Linguistic index, that is, words are indexed in their base form.

PREC Precise index, that is, words are indexed in the grammatical form in which they occur in the original text.

NORM
Normalized precise index, that is, words are indexed in the grammatical form in which they occur in the original text but in lowercase, regardless of the case in the original document.

NGRAM
A case-insensitive index is created for both single-byte character set (SBCS) languages and double-byte character set (DBCS) languages. This index type allows both exact searches and fuzzy searches.

NGRAMCS
A case-sensitive index based on ngrams is created for both single-byte

character set (SBCS) languages and double-byte character set (DBCS) languages. This index type allows both exact searches and fuzzy searches.

index data path
The location of the index data files. The directory is created if it does not already exist.

index work path
The location of the working directory. The directory is created if it does not already exist. This parameter is optional. If an index work path is not specified, the work files are stored in the subdirectory *work* of the index data path. To improve performance, store the index work files on a different physical disk to the index data files.

list of models
To enable section support for the index, you must specify a list of document models. Only these models are available to the index. The first model in the list is the default document model. It is used when a document model is not specifed for indexing or searching.

A set of document models is available on the server instance. Use the `imolstdm` command to see all the models available and choose the subset to be used by the new index.

client library services
The name of the client library services loadable module that is used with this index. The name does not include the file extension. If a name is not specified, the library services for the file system is used, for example, IMOLSCFS. The loadable module must be located in a searchable path for dynamically loadable modules.

server library services
The name of the server library services loadable module that is used with this index. The name does not include the file extension. If a name is not specified, the library services for the file system is used, for example, IMOLSSFS. The loadable module must be located in a searchable path for dynamically loadable modules.

code page
The code page to be used for an NGRAM or an NGRAMCS index. The code pages available depend on the server platform. For a full list of the code pages, see CreateIndex in *z/OS Text Search: Programming the Text Search Engine*.

If you do not specify a code page for an NGRAM index, the index is created using code page 500.

Example

```
imocrix -s SERVER -x TESTIX -t LING -p /index/testix
imocrix -s SERVER -x NGRAMIX -t NGRAM -p /index/ngramix -ccsid 942
imocrix -s SERVER -x NGRAMCS -t NGRAMCS -p /index/ngramcs -pw /indexwork/ngramcs
imocrix -s SERVER -x TESTIX -t LING -p /index/lingsec/ -sections "sample, PLAY"
imocrix -s SERVER -x WEBINDEX -t NORM -p /index/webindex -lsse imolssht
```

IMOCTRIX - control index

Purpose

Use this command to:

- Unlock an index that was locked as a result of an error during processing
- Suspend an index for backup
- Resume an index

You cannot specify both suspend and resume in the same call.

For further information on controlling indexes, see EhwSetIndexFunctionStatus in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imoctrix
        [-h| -H| -?| -copyright] [-quiet]
        -s <search service name>
        -x <index name>
        [-reset]
        [-suspend]
        [-resume]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the created index. Index names must be unique.

reset
Resets the specified index name.

suspend
Suspends the specified index.

resume
Resumes the specified index.

Example

```
imoctrix -s SERVER -x TESTIX -suspend
```

IMODELIX - delete index

Purpose

Use this command to delete an index on the Text Search Engine server. For further information on deleting indexes, see EhwDeleteIndex in *z/OS Text Search: Programming the Text Search Engine*.

Tip

Use the `imodelix` command with care. The index is deleted without you having to confirm the deletion.

The subdirectories that were created with the `imocrix` command are not removed with the `imodelix` command.

Syntax

```
imodelix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index to be deleted.

Example

```
imodelix -s SERVER -x TESTIX
```

IMOLSTIX - list indexes

Purpose

Use this command to list all the indexes for a search service. For further information on listing indexes, see EhwListIndexes in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imolstix
        [-h| -H| -?| -copyright] [-quiet]
        -s <search service name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

Example

```
imolstix -s SERVER
```

IMOMSGIX - display indexing messages

Purpose

Use this command to list the messages that were produced during an indexing run. The error messages and the recommended actions to take in the error situation are explained in *z/OS Text Search: Programming the Text Search Engine*.

For further information on displaying indexing messages, see EhwGetIndexingMsgs in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imomsgix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
         [-delete]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

delete
Deletes all indexing messages.

Example

```
imomsgix -s SERVER -x TESTIX
```

IMOREOIX - reorganize index

Purpose

Use this command to reorganize an index. You might want to do this, for example, when the number of documents in the secondary indexes is greater than the number of documents in the primary index or to remove obsolete information from the index. The index information is compressed thus improving storage utilization and performance.

To see the number of documents in the primary and secondary indexes, use the `imostaix` command. To check the progress of an index reorganization, use the `imostfix` command.

For further information on reorganizing indexes, see EhwReorgIndex in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imoreoix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

Example

```
imoreoix -s SERVER -x TESTIX
```

IMORULIX - maintain indexing rules

Purpose

Use this command to specify rules for assigning default values for the document format, language, and the code page. These rules are used during indexing when any of these values cannot be determined automatically.

For further information on maintaining indexing rules, see EhwGetIndexingRules in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imorulix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
         [-dfmt <document format>]
         [-ccsid <code page>]
         [-lang <language>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

document format
Specify one of the following default document formats:

- TDS
- ASCIISECTION
- RTF
- HTML
- XML

Alternatively, you can use a numeric value for the document format. See the file, IMOLSDEF.H, for a list of the numeric values. This file is located:

- On AIX in /usr/TextTools/include
- On Solaris in /opt/TextTools/include
- On Windows NT in >TARGETDIR<\TextTools\include

To use your own document formats, define a value for the document format here. This value must be greater than the value set for EHW_USER_FORMATS in IMOLSDEF.H. This file is located:

- On AIX in /usr/TextTools/include
- On Solaris in /opt/TextTools/include
- On Windows NT in >TARGETDIR<\TextTools\include

Unsupported document formats can be used only with a user exit. For more information on user exits, see "Using unsupported document formats" on page 16.

code page
The default code page. If UCS2 code page is specified, default is to assume big endian layout. Documents may switch endianness of UCS2 by using "\xff \xfe" notation.For a list of the supported code pages, see *z/OS Text Search: Programming the Text Search Engine*.

language
The default document language. You can specify a three-letter language code or a numeric value. For a list of the languages supported and the values that can be used, see the file IMOLANG.H.

Example

To show default indexing rules:

```
imorulix -s SERVER -x TESTIX
```

To set the document format:

```
imorulix -s SERVER -x TESTIX -dfmt TDS
```

To set the code page and the document format:

```
imorulix -s SERVER -x TESTIX -dfmt 14 -ccsid 850
```

To set the document format, code page, and language:

```
imorulix -s SERVER -x TESTIX -dfmt TDS -ccsid 850 -lang ENU
```

IMOSTAIX - display index status

Purpose

Use this command to display general information and status information for an index on the Text Search Engine server.

- General information

  Includes the index type, index name, the library services, the data and work path, and the code page (optional).
- Status information

  The number of documents in the primary index, the secondary index (if it exists), and the input queue. If messages were generated during indexing, the number of messages generated is also shown. To view the messages, use the `imomsgix` command.

For further information on displaying the index status, see EhwGetIndexStatus in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imostaix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

Example

```
imostaix -s SERVER -x TESTIX
```

IMOSTFIX - display status of index functions

Purpose

Use this command to display status information for the Text Search Engine functions search, scheduling, indexing, and index reorganization. It shows whether each of these functions are enabled, running, or stopped for a particular index. If an index has been stopped, use the `imoctrix` command to reset it. To get information on the error code, see *z/OS Text Search: Programming the Text Search Engine*.

For further information on displaying the status of the index functions, see EhwGetIndexFunctionStatus in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imostfix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

Example

```
imostfix -s SERVER -x TESTIX
```

IMOUPDIX - update index

Purpose

Use this command to start an index update. The indexing process is a background process. Use the `imostfix` command to check the progress of the index update. Use the imostaix command to see messages generated during the index update.

For further information on updating indexes, see EhwUpdateIndex in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imoupdix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
         [-wait <check interval>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

check interval
A value in seconds that determines how often the status of the indexing process is checked. Choose a value that is about half the value of the total expected indexing time. The command returns when the indexing process has finished.

Example

```
imoupdix -s SERVER -x TESTIX
imoupdix -s SERVER -x TESTIX -wait 5
```

Document administration

You can use the following commands for document administration:

IMOQUEUE Add documents to the queue for indexing or deleting from the index, clearing all documents from the document queue

IMOSRCH Search for documents

IMOQUEUE - queue documents

Purpose

Use this command to schedule one or more documents for indexing or deleting. You can also use this command to clear the document queue. For further information on scheduling documents, see EhwScheduleDocument in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imoqueue
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
         -add | -delete | -clear
         [-ascii | -ebcdic]
         [-l <document list file name>]
         [-f <file name>]
         [-v]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

-add
Adds document IDs to the index queue for adding to the index. Use the `-l` or the `-f` parameter to specify the document IDs.

-delete
Adds document IDs to the index queue for deleting from the index. Use the `-l` or the `-f` parameter to specify the document IDs.

-clear
Removes entries from the index queue.

-ascii/-ebcdic
For file-based library services only. Specifies the character set used by the server operating system; ASCII for all workstation operating systems and EBCDIC for z/OS. The document names are converted to the character set supported by the server.

document list file name
The fully-qualified name of the input file containing the list of documents to be indexed. The new index entries are generated from the list of documents. A new-line character **must** separate each document listed in the file.

The document list file name is optional if you specify a **file name**.

Note: The `-l` parameter is tailored to the delivered file-system library services. If your document IDs do not conform to the Text Search Engine rules, you must write your own library services or queuing command. See *z/OS Text Search: Programming the Text Search Engine* for information on the rules for document IDs.

file name
The fully-qualified name of a single document that you want to schedule for indexing.

The file name is optional if you specify a **document list file name**.

Note: The `-f` parameter is tailored to the delivered file-system library services. If your document IDs do not conform to the Text Search Engine rules, you must write your own library services or queuing command.

-v Lists all the scheduled documents (verbose mode).

Example

The doc.lst might look as follows:

```
/home/user/document1.txt
/home/user/document2.txt
/home/user/document3.txt
```

To add documents in the file list to the input queue for indexing:

```
imoqueue -s SERVER -x TESTIX -add -l /home/user/doc.lst
```

To add documents in the file list to the input queue to be deleted from the index:

```
imoqueue -s SERVER -x TESTIX -delete -l /home/user/doc.lst
```

To clear the index queue:

```
imoqueue -s SERVER -x TESTIX -clear
```

To add single documents to the index queue:

```
imoqueue -s SERVER - x TESTIX -add -f /home/user/document1.txt
```

To queue documents stored in MVS™ data sets:

```
imoqueue -s SERVER -x TEXTIX -add -f "//'SYS1.PARMLIB(MEMBER)' "
```

IMOSRCH - search for documents

Purpose

Use this command to search a document index:

- Boolean search is supported with the keywords AND, OR, NOT.
- Free-text search and hybrid search are available.
- Specify subqueries in parentheses ().
- Specify sentence, paragraph, and document proximity with brackets [].

  The phrase must be separated into single search terms. This is done with the ″|″ operator. Proximity for sentence, paragraph, and document is specified by ″[S ... S]″, ″[P ... P]″, or ″[D ... D]″.
- Wildcard searches using * for masking several characters, and ? for masking single characters.

The `imosrch` command provides only limited search capabilities. You can extend these using the API function EhwSearch. For further information on EhwSearch, see *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imosrch
         [-h| -H| -?| -copyright] [-quiet]
        -s <search service name>
        -x <index name>
        [-ccsid <code page>]
        [-lang <language>]
        [-max <maximum number of documents>]
        [-rlim]
        [-ascii | -ebcdic]
        [-rank]
        [-noseq]
        [-section <section name>]
        -term <search query>
        -fterm <free-text search query>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The name of the index.

code page
The code page for the search terms.

Default: Code page 500.

language
The language you want to search in.

Default: ENU

maximum number of documents
The maximum number of documents shown as a result of the search. This value does not affect the number of documents found and reported.

Default: 50 documents

rlim
The document limit. If the option is specified, it is the maximum number of documents that can be processed on the save and displayed to the user in a search result.

For freetext queries, this option must be used if more than 50 documents are to be processed.

-ascii/-ebcdic
For file-based library services only. Specifies the character set used by the server operating system; ASCII for all workstation operating systems and EBCDIC for z/OS. The document names are converted to the character set supported by the server.

rank
If this option is specified, the results are ranked and the rank values are shown in the output.

noseq
Terms in the search query can occur in any sequence in a single sentence in the document text. If this option is not specified, terms in the search query must occur in exactly the same sequence in a single sentence in the document text.

This option cannot be specified for Boolean searches using the NOT operand or for searches on NGRAM indexes.

section name
This restricts the search to the named section of a document. This is only available if section support is enabled for the index.

search query
A boolean search query including the operands AND, OR, NOT. For search terms containing spaces, enclose the term in quotation marks, for example, ″John Smith″. Enclose subqueries in parentheses. You can also include phrases.

If you also specify an `-fterm`, a hybrid search is carried out.

free-text search query
The free-text part of the query. Any stop words in the query are ignored.

If you also specify a `-term`, a hybrid search is carried out.

Example

To search for a term:

```
imosrch -s SERVER -x TESTIX -term computer
```

To search for a phrase:

```
imosrch -s SERVER -x TESTIX -term "Bill Clinton"
```

To do a search with the codepage and language for German and set the maximum number of documents returned:

```
imosrch -s SERVER -x TESTIX -ccsid 273 -language 4841 -max 100 -term Schönaich
```

To do a complex Boolean search with subqueries and ranking:

```
imosrch -s SERVER -x TESTIX -rank -term "((Bill Clinton AND innovation) AND
(NOT government AND NOT state)) OR IBM"
```

To do a free-text search:

```
imosrch -s SERVER -x TESTIX -fterm "big house water"
```

To do a hybrid search with ranking:

```
imosrch -s SERVER -x TESTIX -rank -term "government AND Washington"
        -fterm "law justice"
```

To do a Boolean search with paragraph proximity:

```
imosrch -s SERVER -x TESTIX -term "[P computer | test P]"
```

To search in an index with section support based on the model 'book' for the word 'computer' in the title section:

```
imosrch -s SERVER -x TESTIX -term computer -section book/Title
```

To search an index using wildcard characters:

```
imosrch -s SERVER -x TESTIX -term "gov* AND was?ington"
```

Document model administration

You can use the following commands for document model administration:

IMOCRDM Create a document model

IMODELDM Delete a document model

IMOGETDM Display a document model

IMOLSTDM List document models

IMOMODIX Set default document model

IMOCRDM - create document model

Purpose

Use this command to create a document model and add it to the set defined for the server instance.

For further information on creating document models, see EhwCreateDocumentModel in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imocrdm
        [-h| -H| -?| -copyright] [-quiet]
        -s <search service name>
        [-x <index name>]
        [-docmod <document model name>]
        [-l <ini file with model description>]
        [-p]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
If you specify an index name, the new model is assigned only to this index. It is then available for the next update on this index.

If you do not specify an index name, the model is added to the set of models defined for the server instance. It is then available when you create an index using the `imocrix` command.

document model name
The document model name can contain the characters A-Z, a-z, and 0-9 and be up to 32 characters long. It is case-sensitive.

ini file
The name of an ini file that contains the description of the document model in the following format:

```
[document model name]
sectionName1 = sectionTag1
sectionName2 = sectionTag2
```

The section names are case-sensitive. Ensure that the document model name and the name passed to the command are the same.

To create the model description interactively, use the `-p` parameter.

-p (interactive mode)

Instead of using an ini file to create the model description, you can specify section names and section tags interactively. Press Enter to end the model description.

To create a model description interactively:

```
imocrdm -s SERVER -docmod newmodel -p
```

You are prompted for information about the sections:

```
Enter section name: title
Enter section tag: TITLE
Enter section name: author
Enter section tag: AUTHOR
Enter section name: (leave blank)
Create document model (Y/N): Y
```

Example

```
imocrdm -s SERVER -x TESTIX -docmod newmodel -l /tmp/model.ini
imocrdm -s SERVER -docmod newmodel -l /tmp/model.ini
```

IMODELDM - delete document model

Purpose

Use this command to delete a document model from the set defined for the server instance. The model is still available for searching and updating the indexes it has been assigned to. However, it is not available when you create a new index.

For further information on deleting document models, see EhwDeleteDocumentModel in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imodeldm
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         [-docmod <document model name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

document model name
The name of the document model you want to delete. To see a list of models for the server instance, use the `imolstdm` command.

Example

```
imodeldm -s SERVER -x TESTIX -docmod oldmodel
```

IMOGETDM - display a document model

Purpose

Use this command to display the description of a document model.

For further information on updating indexes, see EhwGetDocumentModel in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imogetdm
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         [-x <index name>]
         -docmod <document model name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

document model name
The name of the document model you want to display. To see a list of models for an index or for the server instance, use the `imolstdm` command.

index name
The name of the index to which the data model is associated. If you do not specify an index name, the document model from the server instance set of model is shown.

Example

```
imogetdm -s SERVER -x TESTIX -docmod newmodel
imogetdm -s SERVER -docmod newmodel
```

IMOLSTDM - list document models

Purpose

Use this command to display a list of models associated with an index or defined for the server instance.

For further information on displaying a list of models, see EhwListDocumentModels in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imolstdm
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         [-x <index name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
If you specify an index name, a list of models for this index are shown. If you do not specify an index name, a list of the models on the server instance is shown.

Example

```
imolstdm -s SERVER -x TESTIX
imolstdm -s SERVER
```

IMOMODIX - set default document model

Purpose

The default document model for an index is set when the index is created; it is always the first model given in the list of models. Use this command to change or display the default document model used by an index.

If documents are to be indexed for a section-enabled index and they do not contain any information about the document model to be used, they are indexed according to the default document model. If a document model is not specified on a search, the default document model is also used.

For further information on updating indexes, see EhwCreateDocumentModel in *z/OS Text Search: Programming the Text Search Engine*.

Syntax

```
imomodix
         [-h| -H| -?| -copyright] [-quiet]
         -s <search service name>
         -x <index name>
         [-docmod <document model name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

search service name
The search service name specified during creation of the client profile (`imocrcl` command). Each search service is associated with one server instance. Use the `imocfgcl` command to get a list of all the search service names.

index name
The index for which you want to specify a default model.

document model name
The default document model for the index. This must be one of the models specified when the index was created or assigned to the index later using the `imocrdm` command. To the see the models associated with the index, use the `imolstdm` command.

Example

To show the current default model for an index:

```
imomodix -s SERVER -x TESTIX
```

To set the document model, sample, as the default for an index:

```
imomodix -s SERVER -x TESTIX -docmod sample
```

Thesaurus administration

You can use the following commands for thesaurus administration:

IMOTHESC Compile a Text Search Engine thesaurus definition file

IMOTHESN Compile an NGRAM thesaurus definition file

IMOTHESC - compile a thesaurus definition file

Purpose

Use this command to compile a Text Search Engine thesaurus definition file into a binary thesaurus dictionary format. The definition file must be in SGML format. For information on specifying the content of the definition file, see *z/OS Text Search: Programming the Text Search Engine*.

To use the thesaurus dictionary files during a search, do one of the following:

- Move them to the resource directory for the server instance. This is the directory specified by the IMONLPSSRV option in the server configuration file.
- Specify the location of the files in the query.

Syntax

```
imothesc
         [-h| -H| -?| -copyright] [-quiet]
         -ccsid <code page>
         -f <definition file name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

code page
The code page the thesaurus definition file is written in. This must be code page 500.

definition file name
Name of the file containing the thesaurus definition. The file name must contain either the absolute path or the relative path to the file.

The thesaurus dictionary is generated in the same directory as the definition file. It has the same name as the definition file and the extensions th1 through th6.

Tip

Because thesaurus files are overwritten when they have the same names, it is recommended that you use a separate directory for each thesaurus.

Example

```
imothesc -ccsid 500 -f thesaurus/sample.sgm
```

IMOTHESN - compile an NGRAM thesaurus definition file

Purpose

Use this command to compile an NGRAM thesaurus definition file into a binary thesaurus dictionary format. The definition file must be of the format described in *z/OS Text Search: Programming the Text Search Engine* for NGRAM thesauri.

To use the thesaurus dictionary files during a search, do one of the following:

- Move them to the resource directory for the server instance. This is the directory specified by the IMONLPSSRV option in the server configuration file.
- Specify the location of the files in the query.

Syntax

```
imothesn
         [-h| -H| -?| -copyright] [-quiet]
         [-ccsid <code page>]
         [-f <definition file name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

code page
The code page the thesaurus file is written in. For a list of the supported code pages, see EhwCreateIndex in *z/OS Text Search: Programming the Text Search Engine*.

file name
Name of the file containing the thesaurus definition. The file name must contain either the absolute path or the relative path to the file. The file name is restricted to 8+3 characters, the extension is optional.

The thesaurus dictionary is generated in the same directory as the definition file. It has the same name as the definition file and the extensions wdf, wdv, grf, grv, MEY, ROS, NEY, SOS, and lk*n*, where *n* is a digit.

> **Tip**
> Because thesaurus files are overwritten when they have the same names, it is recommended that you use a separate directory for each thesaurus.

Server administration

You can use the following commands for server administration:

IMOADMSV Configure server settings

IMOCFGSV Update the server configuration

IMOCRINS Create a server instance

IMOSS Start server instance

IMOADMSV - update server settings

Purpose

Use this command to update settings for resources used by the server instance, for example, the working directory for temporary files.

Syntax

```
imoadmsv
         [-h| -H| -?| -copyright] [-quiet]
         [-d ]
         [-i <server instance name>]
         [-work <working directory for temporary files>]
         [-r <resource path>]
         [-dtd <path to external dtd files>]
         [-userexit <user exit for format recognition>]
         [-userexitall <on | off>]
         [-fmtrecogn <on | off>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

-d Display current settings.

server instance name
The name of the server instance you want to configure.

working directory
The directory where temporary files are stored. By default, the working directory is a subdirectory of the server instance path set by the environment variable, IMOCONFIGSRV.

resource path
The directory where the resource files, such as dictionaries are located. The server needs the resource path, for example, for stopword handling and other linguistic processing.

path to external dtd files
The directory where the external DTD files needed during indexing of XML files are located. If you do not specify a directory, it is assumed that the DTD files are in the same directory as the XML files or accessible using the path statement specified in the XML files.

user exit
The name of the user exit used to convert unsupported document formats. For information on creating user exits, see "Using unsupported document formats" on page 16.

userexitall
If you specify on, the user exit is used for all document formats, including the supported document formats.

fmtrecogn

If you specify on, each document is analyzed to determine the appropriate parser for the document type.

Processing is faster if you switch off automatic format recognition. If you are not using `userexitall`, documents are parsed according to the default document format set by the `imorulix` command.

Example

```
imoadmsv -i inst1
imoadmsv -i inst1 -dtd /home/dtd -fmtrecogn off
```

IMOCFGSV - update server configuration

Purpose

Use this command to update and display the communication settings for the server instance.

Syntax

To update the server setup:

```
imocfgsv
         [-h| -H| -?| -copyright] [-quiet]
         [-p]
         [-d -i <server instance name>]
         [-c local -i <server instance name>
                  [-t <max tasks>]]

         [-c tcpip -i <server instance name>
                   -n <port number>
                   -m <machine name>
                  [-t <max tasks>]
                  [-k <tasks kept available>]
                  [-l <time limit>]]
```

To set up the server in interactive prompt mode:

```
imocfgsv -p
```

To display the server setup:

```
imocfgsv -d -i <server instance name>
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

local
The clients and the server are on the same machine. The client requests are handled in one daemon process.

tcpip
Defines the TCP/IP protocol for client/server communication.

server instance name
A server instance defines a particular set of indexes, local and remote, that users work with. The name can be up to 8 characters long.

port number
The TCP/IP port address that all clients use when they connect to the search service. This port number must not be assigned or used by any other applications on the system. Your system administrator usually assigns the port number.

machine name
For TCP/IP, this is the host name or IP address of the server on which the Text Search Engine server is installed.

max tasks
The maximum number of tasks that the search service can handle at the same time. Valid values are integers between 1 and 100. Your system performance decreases as the value for this parameter increases. If, however, you set this value too low, some clients might be unable to search at all.

tasks kept available
The number of tasks to be started in advance to handle client requests. Valid values are integers between 1 and 10.

To improve performance, a client making a request should always find a service task ready to handle its request. However, having too many unused service tasks open unnecessarily lowers the search performance. The value specified must be less than or equal to the value specified for the maximum number of tasks (MaxTasks).

time limit
The time (in seconds) that a search service task remains occupied without receiving a request from a client. Valid values are integers between 100 and 99999. After the specified time period has passed, the search service task ends the connection to the client. This sets search service tasks free for other users. The connection is automatically established again when the client performs the next action, provided that a free search service task is available. Note that reconnecting causes a short delay.

Set this value to the longest time that a query can run. If queries are often complex and run simultaneously, set this value to 600 seconds or more. If queries are mostly simple and hardly ever run simultaneously, set it to 100 seconds or more.

Note: If the value is too low, queries that take longer end before they have completed; if the value is too high, free search service tasks might be not available.

Example

```
imocfgsv -c tcpip -i MYINST -n 7777 -m MYHOST
```

IMOCRINS - create server instance

Purpose

Use this command to create and configure a server instance. A configuration file is also created. For further information on the contents of the configuration file, see "Server configuration file" on page 72.

Syntax

```
imocrins
         [-h| -H| -?| -copyright] [-quiet]
         [-c local -i <server instance name>
                   -r <resource path>]

         [-c tcpip -i <server instance name>
                   -r <resource path>
                   -n <port number>
                   -m <machine name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

local
The clients and the server are on the same machine. The client requests are handled in one daemon process.

tcpip
Defines the TCP/IP protocol for client/server communication.

server instance name
A server instance defines a particular set of indexes, local and remote, that users work with. The name can be up to 8 characters long.

resource path
The directory where the dictionaries are located.

port number
The TCP/IP port address that all clients use when they connect to the search service. This port number must not be assigned or used by any other applications on the system. Your system administrator usually assigns the port number.

machine name
For TCP/IP, this is the host name or IP address of the server on which the Text Search Engine server is installed.

Example

```
imocrins -c tcpip -i MYINST -r /usr/TextTools/dict -n 7777 -m MYHOST
```

IMOSS - start/stop server instance

Purpose

Use this command to start, stop, or query the status of a search service.

Syntax

```
imoss
        [-h| -H| -?| -copyright] [-quiet]
        [-start <server instance name>]
        [-stop <server instance name>]
        [-status <server instance name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

server instance name
A server instance defines a particular set of indexes, local and remote, that users work with.

-start
Starts the named server instance.

-stop
Stops the named server instance. On UNIX-based systems, the allocated resources, such as shared memory or semaphores are removed from the system.

-status
Queries the status of the named server instance.

Client administration

You can use the following commands for client administration:

IMOADMCL Configure client settings

IMOCFGCL Update a client profile

IMOCRCL Create a client profile

IMOADMCL - update client configuration

Purpose

Use this command to update settings for resources used by the client, for example, the working directory for temporary files.

Syntax

```
imoadmcl
         [-h| -H| -?| -copyright] [-quiet]
         [-d ]
         [-work <working directory for temporary files>]
         [-r <resource path>]
         [-dtd <path to external dtd files>]
         [-userexit <user exit for format recognition>]
         [-userexitall <on | off>]
         [-fmtrecogn <on | off>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

-d Display current settings.

working directory
The directory where temporary files are stored. By default, the working directory is a subdirectory of the server instance path set by the environment variable, IMOCONFIGCL.

resource path
The directory where the resource files, such as dictionaries are located. The server needs the resource path, for example, for stopword handling and other linguistic processing.

path to external dtd files
The directory where the external DTD files needed for parsing XML documents are located. If you do not specify a directory, it is assumed that the files are in the same directory as the XML files or accessible using the path specified in the XML files. For some API functions, the client also needs access to these files.

user exit
The name of the user exit used to convert unsupported document formats. For information on creating user exits, see "Using unsupported document formats" on page 16.

userexitall
If you specify on, the user exit is used for all document formats, including the supported document formats.

fmtrecogn
If you specify on, each document is analyzed to determine the appropriate parser for the document type.

Processing is faster if you switch off automatic format recognition. If you are not using `userexitall`, documents are parsed according to the default document format set by the `imorulix` command.

Example

```
imoadmcl -dtd /home/dtd -fmtrecogn off
```

IMOCFGCL - update client profile

Purpose

Use this command to update the client profile. You can also use it to display, update, or erase a specific client setup.

Syntax

To configure the client communication:

```
imocfgcl
         [-h| -H| -?| -copyright] [-quiet]
         [-p]
         [-e -s <search service name>]
         [-c local -s <search service name>
                   -i <server instance name>]

         [-c tcpip -s <search service name>
                   -n <port number>
                   -m <machine name>]
```

To set up the client in interactive prompt mode:

```
imocfgcl -p
```

To delete client access permissions:

```
imocfgcl -e -s <search service name>
```

To display the current client setup:

```
imocfgcl -d
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

local
The clients and the server are on the same machine. The client requests are handled in one daemon process.

tcpip
Defines the TCP/IP protocol for client/server communication. The server must also be configured for TCP/IP communication.

search service name
The search service name can be up to 8 characters long. It is used to establish a connection to the server.

server instance name
A server instance defines a particular set of indexes, local and remote, that users work with. The name can be up to 8 characters long.

port number
The TCP/IP port address that all clients use when they connect to the search service. This is the same port number as used for the server configuration.

machine name
For TCP/IP, this is the host name or IP address of the server on which the Text Search Engine server is installed.

Example

```
imocfgcl -c tcpip -s SERVER -n 7777 -m MYHOST
```

IMOCRCL - create client profile

Purpose

Use this command to create and configure a client profile. A configuration file is also created. For further information on the contents of the configuration file, see "Client configuration file" on page 71.

Syntax

```
imocrcl
        [-h| -H| -?| -copyright] [-quiet]
        [-c local -s <search service name>
                  -i <server instance name>
                  -r <resource path>]

        [-c tcpip -s <search service name>
                  -r <resource path>
                  -n <port number>
                  -m <machine name>]
```

Parameters

-h, -H, or -?
Help information. If one of these options is specified, all other options are ignored.

-copyright
Returns the internal build number of the product. Use this number when reporting problems with the product.

-quiet
Output information is not displayed on the screen.

local
The clients and the server are on the same machine. The client requests are handled in one daemon process.

tcpip
Defines the TCP/IP protocol for client/server communication.

search service name
The search service name can be up to 8 characters long. Each search service is associated with one server instance. You can have several search services in one client profile.

server instance name
A server instance defines a particular set of indexes, local and remote, that users work with. The name can be up to 8 characters long.

resource path
The directory where the dictionaries are located.

port number
The TCP/IP port address that all clients use when they connect to the search service. Your system administrator assigns it.

machine name
For TCP/IP, this is the host name or IP address of the server on which the Text Search Engine server is installed.

Example

```
imocrcl -c tcpip -s SERVER -r /usr/TextTools/dict -n 7777 -m MYHOST
```

Appendix A. Configuration files

This appendix describes the configuration files. These files are automatically generated when a server instance is created with the `imocrins` command or when a client profile is created with the `imocrcl` command. On workstation platforms, these files are generated in code page 819, on z/OS they 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.

Client configuration file

File name IMOCL.INI

Location The directory to which the environment variable IMOCONFIGCL points.

The updated options become active at the next StartSession function. For further information on the StartSession function, see the *z/OS Text Search: Programming the Text Search Engine*.

Section	Option	Default value	Description
[INSTANCE]	IMOWORKCL		Points to a working directory that is used for temporary files.
	IMONLPSCL		Points to the resource directory.
[BUFFER]	BUFFERSEGMENTSIZE	32 000	The size of the block segments, in bytes, used for buffering. This is used by EhwGetMatches.
	BUFFERSEGMENTCOUNT	3	The maximum number of segments used before buffers are swapped to temporary files. A buffer segment is defined by BUFFERSEGMENTSIZE.
[DOCUMENTFORMAT]	USEREXIT		The name of the user exit used to work with document formats not listed in imolsdef.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 16.

Section	Option	Default value	Description
	FORMATRECOGNITION	TRUE	Triggers the format recognition of the document formats listed in imolsdef.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 imolsdef.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 The directory to which the environment variable IMOCONFIGSRV points + the server instance name. This file exists only once per server instance.

The updated options become active the next time the server instance is started.

Table 2. 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.
	MaxIndexEntries	1000	The maximum number of indexes used. Decrease this number if you are short of resources, such as shared memories.
[BUFFER]	BUFFERSEGMENTSIZE	32 000	The size of the block segments, in bytes, used for buffering. This is used by EhwUpdate.

Table 2. Server configuration file options (continued)

Section	Option	Default value	Description
	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	20 000 000	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 imolsdef.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 16.
	FORMATRECOGNITION	TRUE	Triggers the format recognition of the document formats listed in imolsdef.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 imolsdef.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.
	UPDATESLICE	1	The number of update runs that take place before an internal reorganization process starts. An update run is defined by the UPDATETHRESHOLD.

Table 2. Server configuration file options (continued)

Section	Option	Default value	Description
[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.

Appendix B. Dictionary files

This appendix lists the supported languages and the corresponding file names used for the dictionary files (*.dic), stop-word files (*.stw), and abbreviation files (*.abr). 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 correct code page for the language as shown in Table 3.

Table 3. Dictionary files

Language	File name	Code page	
		Host	Workstation
Arabic	arabic	420	864
Brazilian Portuguese	brasil	500	850
Canadian French	canadien	500	850
Catalan	catalan	500	850
Danish	dansk	500	850
Dutch	nederlnd	500	850
Finnish	suomi	500	850
French	francais	500	850
German	deutsch	500	850
Hebrew	hebrew	424	862
Icelandic	islensk	500	850
Italian	italiano	500	850
Norwegian	norbok	500	850
Norwegian	nornyn	500	850
Portuguese	portugal	500	850
Russian	russian	1025	866
Spanish	espana	500	850
Swedish	svensk	500	850
Swiss German	dschweiz	500	850
Thai	thai	838	874
U.K. English	uk	500	850
U.S. English	us	500	850

Appendix C. Handling errors

The Text Search Engine has different levels of error handling. These levels are:

- Return codes from the API

  The API and the command-line utilities that use the API always give a return code on completion. These return codes are listed in the *z/OS Text Search: Programming the Text Search Engine* together with an explanation of the possible causes of the error and the recommended actions to take in the error situation. If you need more information on the cause of the error, look in the log file imodiag.log, which is located in the server instance path.

- Error codes for a group of functions

  There are groups of functions for searching, scheduling, updating, and reorganizing. Status information is maintained for each group. If an error occurs in one of the groups, the group is locked and is no longer available for use; all other function groups of that index are still available. To make the functions available again, you must unlock the function group using the `imoctrix` command.

  For example, suppose an update operation on an index fails because the process runs out of disk space. The index is locked for further update operations, but it is still searchable. In this case, a search operates on all documents that were present in the index before the failed update operation.

  The `imostfix` command lists the status of the function groups. You can use this information to help determine the cause of an error. This might lead you to check the directories where the index files reside for available space or for correct permissions. (The directories and files must be set to provide read and write access for the Text Search Engine administrative user.)

  After handling the cause of the error (perhaps by resetting the permissions, or providing more disk space), use the `imoctrix` command to unlock the locked function groups. The `imoctrix` command always re-enables all function groups simultaneously.

  The error codes returned by the function groups are listed in the *z/OS Text Search: Programming the Text Search Engine* together with an explanation of the possible causes of the error and the recommended actions to take in the error situation. If you need more information on the cause of the error, look in the log file imodiag.log, which is located in the server instance path.

- Trace

  The Text Search Engine also has a trace facility (`imotrace`) for following the internal program flow. If an error occurs that you cannot solve by one of the other methods, you can enable tracing to get more detailed information about the error situation. For reporting errors to your IBM representative, it is recommended that you enable the trace facility and re-create the error situation.

IMOTRACE - enable trace facility

Purpose

Use this command to get detailed information on error situations. The trace facility records the activity of all active instances on the server. Information about the program flow is written to an internal buffer. You can dump the contents of the buffer onto the screen. Use the output from the trace facility when you report problems or errors to your IBM representative. You can use `imotrace` on both the client and the server. By default, the trace facility is not enabled.

Syntax

```
imotrace
          (clr| fmt| off| on|)
```

Parameters

clr Clears the internal trace buffer.

fmt Formats the trace output onto the screen.

off Turns tracing off.

on Turns tracing on.

Example

```
imotrace on
imolstix -s smss
...
imotrace fmt > /tmp/trace.out
imotrace off
```

Appendix D. 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 10505-1785
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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

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

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

IBM Deutschland
Informationssysteme GmbH
Department 3982
Pascalstrasse 100
70569 Stuttgart
Germany

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.

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

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.

Trademarks

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

- IBM
- MVS
- OpenEdition
- OS/390
- z/OS

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.

Index

M

N

P

Q

R

S

T

U

X

Readers' Comments — We'd Like to Hear from You

z/OS
Text Search: Installation and Administration for the Text Search Engine
Version 1.2

Publication No. SH12-6716-01

Overall, how satisfied are you with the information in this book?

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Overall satisfaction	☐	☐	☐	☐	☐

How satisfied are you that the information in this book is:

	Very Satisfied	Satisfied	Neutral	Dissatisfied	Very Dissatisfied
Accurate	☐	☐	☐	☐	☐
Complete	☐	☐	☐	☐	☐
Easy to find	☐	☐	☐	☐	☐
Easy to understand	☐	☐	☐	☐	☐
Well organized	☐	☐	☐	☐	☐
Applicable to your tasks	☐	☐	☐	☐	☐

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. _____

Readers' Comments — We'd Like to Hear from You
SH12-6716-01

Cut or Fold
Along Line

Fold and Tape | **Please do not staple** | Fold and Tape

PLACE
POSTAGE
STAMP
HERE

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Schoenaicher Str. 220
71032 Boeblingen
Germany

Fold and Tape | **Please do not staple** | Fold and Tape

SH12-6716-01

Cut or Fold
Along Line

Program Number: 5694-A01

SH12-6716-01