Cindex™
Application Programming Interface

Version 3.2

(revised Jan 25, 2014)

Jan 2014

Introduction

The Publishers' Edition of Cindex for Windows provides an application programming interface (API) that allows other programs running under Windows to execute certain Cindex commands and retrieve data from Cindex while it is running.

The API is implemented through functions in a dynamic link library (DLL) called by the application program that wishes to communicate with Cindex. The library is available from Indexing Research.

> *NOTE: Version 3.2 of the API is for use only with Publishers' Edition 3.2 and higher. It will fail to make a connection with any earlier version of Cindex.*

Library Functions

The API provides four functions for use by the calling application. These are declared in the header file cindexapi.h. All functions return a result >0 if they succeed, otherwise an error code (less than or equal to 0). The following errors may be returned:

CINAPI_ERR_UNSPECIFIED	
CINAPI_ERR_UNKNOWNCOMMAND	unknown command
CINAPI_ERR_BADARGUMENT	invalid argument
CINAPI_ERR_NUMARGUMENTS	wrong number of arguments to command
CINAPI_ERR_NOSERVER	no connection to server
CINAPI_ERR_NOCONVERSATION	unable to establish a conversation with client. Likely cause is mismatched versions of api and client

INT32 cindex_getversion(void)

Returns the version number of the api dll.

INT32 cindex_connect(void)

This function opens a connection to a running copy of Cindex. It must be called before any other API operations can be undertaken.

INT32 cindex_disconnect(void)

This function closes an open connection previously established with cindex_connect(). It should be called after all API operations are completed.

INT32 cindex_command(char * command, char * arg1, char * arg2)

This function sends a command to be executed by Cindex. Depending on the command, one or two arguments may be optional or required. cindex_command() is used to execute a Cindex operation; it is not used to recover information from Cindex.

A later section of this document describes the available commands and their arguments.

INT32 cindex_getdata(char * command, void *data)

This command obtains information from Cindex. The data are returned in the object pointed to by the pointer *data*.

A later section of this document describes the available options and their arguments.

Using the API

The API is supplied as the dynamic link library **cindexapi.dll**. This may be used by an application program either via load-time dynamic linking or run-time linking. For load-time linking an import library (**cindexapi.lib**) is supplied that can be linked into the application when it is built. The DLL can also be loaded at runtime through the Windows call LoadLibrary.

The first call to the API must be to cindex_connect(), and the last call to cindex_disconnect(). There are no restrictions on calls to cindex_command() or cindex_getdata().

Executing Cindex Commands

The following paragraphs describe each of the commands that can be sent to Cindex to execute some operation via cindex_command().

INT32 cindex_command(char * *command*, char * *arg1*, char * *arg2*)

command	A c string that names a command.
arg1	A pointer to an optional argument, or NULL
arg2	A pointer to an optional argument, or NULL

All permissible command names are defined by symbolic constants in the header file cindexapi.h. You should use these constants (rather than literal strings) when using the API.

Return Value: > 0 if success; < = 0 if error.

cindex_command(CINAPI_CREATE, char *indexname, char *templatename)

Creates a new index *indexname* from (optional) template *templatename* if *templatename* is not NULL. Both *indexname* and *templatename* should specify a path and filename extension.

cindex_command(CINAPI_OPEN, char *filename, char * filetype)

Opens the document *filename* of type *filetype*. *filename,* which must include a filename extension, will be opened from the Cindex current directory unless a path is supplied. *filetype* may be one of the following symbolic constants (defined in cindexapi.h):

CINAPI_OPENINDEX	index
CINAPI_OPENRECORDS	cindex records in .ixml or .arc format
CINAPI_OPENSTYLESHEET	stylesheet

If *filetype* is CINAPI_OPENRECORDS or CINAPI_OPENSTYLESHEET the call will fail unless Cindex already has an index is open. Records in archive (.arc) or xml (.ixml) format are read into the active index. The index record size is increased automatically if necessary.

A stylesheet is applied to the active index.

cindex_command(CINAPI_CLOSE, NULL, NULL)

Closes the active index.

cindex_command(CINAPI_QUIT, NULL, NULL)

Causes Cindex to quit after a subsequent call to cindex_disconnect().

cindex_command(CINAPI_SAVEBACKUP, char *indexname, NULL)

Saves a backup copy of the active index as *indexname*. *indexname* should contain a full path, but not the filename extension ".ucdx".

cindex_command(CINAPI_SAVEAS, char *filename, char *filetype)

Saves the active index as *filename* of type *filetype*. *filename* should *not* include an extension. That is supplied automatically as appropriate for the file type being created. If *filename* does not include a path, the file is saved in the current directory. *filetype* may be one of the following symbolic constants (defined in cindexapi.h):

CINAPI_SAVEASINDEX	index
CINAPI_SAVEASXMLRECORDS	xml records
CINAPI_SAVEASARCHIVE	Cindex archive
CINAPI_SAVEASRTF	rich text format

filetype may also be a text string that specifies the name of a Cindex tag set. If you specify a tag set, you should not provide a path (Cindex looks in a special place for tag sets) but you should specify the type of the tag set by appending the extension ".cxtg" or ".cstg".

cindex_command(CINAPI_IMPORT, char *filename, NULL)

Imports the contents of file *filename* into the active index. *filename* should contain a path and filename extension.The file is assumed to be in the (now deprecated) '.dat' format produced by Cindex for DOS.

cindex_command(CINAPI_PRINT, char * filepath, NULL)

Prints the active index on the default printer. Cindex prints the whole index, in the format dictated by the current document settings. If *filepath* is not NULL , the printer output is sent to the file named in *filepath*.

cindex_command(CINAPI_SELECTALL, NULL, NULL)

Selects all records in the current view of the active index.

cindex_command(CINAPI_DELETE, NULL, NULL)

Deletes the selected records from the active index (or restores the records if the first selected is already deleted).

cindex_command(CINAPI_COMPRESS, NULL, NULL)

Compresses the active index, removing deleted records, duplicate records, and empty records.

cindex_command(CINAPI_EXPAND, NULL, NULL)

Expands the active index, generating records as necessary so that each contains only a single locator.

cindex_command(CINAPI_SORT, char *sorttype, NULL)

Sorts the active index in the manner dictated by *sortype*. *sorttype* may be one of the following symbolic constants:

CINAPI_SORTALPHA	sort alphabetically by the current sort settings.
CINAPI_SORTPAGE	sort by locator order.
CINAPI_SORTON	turn sorting on
CINAPI_SORTOFF	turn sorting off

cindex_command(CINAPI_VIEW, char *viewtype, NULL)

Sets the current view of the active index to *viewtype*. *viewtype* may be one of the following symbolic constants:

CINAPI_VIEWFORMATTED	formatted
CINAPI_VIEWDRAFT	draft
CINAPI_VIEWALL	all records

cindex_command(CINAPI_GOTO, char *recordspec, NULL)

Sets the display of the active index to show (and select) the record specified by *recordspec*. *recordspec* may be the beginning text of fields of a record (specified in the manner used by the Cindex GoTo command) or a record number.

cindex_command(CINAPI_SETSIZE, char *recordsize, NULL)

Sets the record size of the active index to *recordsize*. *recordsize* is a string representation of a decimal number.

cindex_command(CINAPI_SHOWINDEX, char *indexname, NULL)

Makes index *indexname* the active index. The index must already be open in Cindex. *indexname* must be fully-qualified with a path and a filename extension.

cindex_command(CINAPI_RECONCILE, char * level, char * joinchar)

Reconciles headings in the active index. *level* is a number that specifies the level below which headings are reconciled. *joinchar* is the single character that will be inserted before headings that are pulled up as modifying phrases.

cindex_command(CINAPI_GENERATE, char *indexpath, char * seeonly)

Generates cross-references from a special cross-reference index whose filename is specified in *indexpath*. *seeonly* is a boolean (0 or 1) that specifies whether only *see* references are generated.

cindex_command(CINAPI_ALTER, char * offset, NULL)

Adds the number *offset* (positive or negative) to all locators in the index.

cindex_command(CINAPI_LABEL, char * labelnumber, NULL)

Applies or removes the label *labelnumber* (1-7) to all records visible in the current view. If the first visible record carries no label, or carries a label other than *labelnumber*, the label is applied to all records in the view. If the first record already carries the label *labelnumber*, the label is removed from that record and all others that carry it. If *labelnumber* is 0, labels are removed from all records in the view.

cindex_command(CINAPI_FINDALL, CINDEXFINDREPLACE * findrep, NULL)

Finds, and displays as a temporary group, all records in the active view that meet the search criteria specified in the structure *findrep*. The structure CINDEXFINDREPLACE is defined in cindexapi.h. When passing the pointer to *findrep*, you should cast the pointer to (char*).

cindex_command(CINAPI_REPLACEALL, CINDEXFINDREPLACE * findrep, NULL)

Searches all records in the active view for the target text specified in the structure *findrep*, and replaces each occurrence with the specified replacement. The structure CINDEXFINDREPLACE is defined in cindexapi.h. When passing the pointer to *findrep*, you should cast the pointer to (char*).

cindex_command(CINAPI_ADDRECORD, char *record, NULL)

Adds a new record to the index. The record contents are in the UTF-8 string *record* (terminated by a NULL character) that may contain special formatting codes described in the Appendix to this document.

Obtaining data from Cindex

The following paragraphs describe each of the commands that can be sent to Cindex to obtain data via cindex_getdata().

INT32 cindex_getdata(char * *command*, void * *data*)

command	A c string that specifies a command.
data	A pointer to the data to be filled in by Cindex

Return Value: the size of the object, or < = 0 on error.

All permissible command names are defined by symbolic constants in the header file cindexapi.h. You should use these constants (rather than literal strings) when using the API.

Each of the available commands assumes data of a specified type (defined in cindexapi.h). In making the call to cindex_getdata() you should pass a pointer to that data object, so that Cindex can fill the object. It is your responsibility to ensure that you have allocated sufficient memory to contain the data. If you call cindex_getdata() with *data* set to NULL, Cindex will not fill the object but will return the required size in bytes.

cindex_getdata(CINAPI_GETRECORDINFO, CINDEXRECORDINFO * rinfo)

Returns information about the structure of records in the index. The structure CINDEXRECORDINFO is defined in cindexapi.h.

cindex_getdata(CINAPI_GETSORTINFO, CINDEXSORTINFO* sinfo)

Returns information about the manner in which the index is sorted. The structure CINDEXSORTINFO is defined in cindexapi.h.

cindex_getdata(CINAPI_GETFONTINFO, CINDEXFONTINFO* finfo)

Returns information about the fonts used in the index. The structure CINDEXFONTINFO is defined in cindexapi.h.

cindex_getdata(CINAPI_GETVERSIONINFO, CINDEXVERSIONINFO* vinfo)

Returns information about the version of Cindex that is running. The structure CINDEXVERSIONINFO is defined in cindexapi.h.

cindex_getdata(CINAPI_GETSELECTEDRECORD, char *record)

Returns the contents of the selected record (or the first of multiple selected records) in the active view of the index . The record contents are contained in a UTF-8 string (terminated by a NULL character) that may contain special formatting characters described in the Appendix to this document. To recover succeeding records in the index, use the command CINAPI_GETNEXTRECORD (below).

cindex_getdata(CINAPI_GETNEXTRECORD, char *record)

Returns the contents of the next record in the active view of the index. To retrieve the contents of a series of records, first call CINAPI_GETSELECTEDRECORD (above) then call CINAPI_GETNEXTRECORD to advance the selection and obtain the contents of the next record.

Appendix: Format of imported/exported records

In import/export format the text of a record is contained in a NULL-terminated UTF-8 string, with each field of the record separated from the next by a 'tab' character. The text of the record may include special formatting characters described below. A record may contain up to sixteen fields, depending on the number permitted in the index that contains the record. The record must contain at least two fields, a main heading and a field containing locators (page references). The last field is always the locator field.

A record string may contain additional information appended to the last (locator) field, but before the terminating NULL character. This 'extended' information encodes the following:

The record status (labeled, generated or deleted).

The time at which the record was last modified (in seconds since Jan 1 1970).

The four-character identifier of the user who last modified the record.

The extended information begins with the character 0x13, and has the following format:

0x13#*time_in_second*s *userid*

# is a single character that encodes record status:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
special_last	TRUE		label2	label1	generated	label0	deleted

The three bits label0, label1, label2 when aggregated define a 3 bit field that represents the value of the label on the record.

The special_last bit indicates that the record came from an index with a required last field. The bit is set only if the special field contained text.

If any status bit is set, Bit 6 is TRUE and # resolves to an uppercase letter. Otherwise # is a space (0x20).

*time_in_second*s is an ASCII decimal number representing the time (in seconds since Jan 1 1970) at which the record was last modified. *time_in_second*s follows Δ without any additional space.

userid is a four-character user identifier. It is separated from *time_in_second*s by a single intervening space (0x20).

Text Styles and Fonts

The text fields of a record can contain formatting codes that identify type styles and fonts. A formatting code is defined by two characters, the first of which always has the value 0x1A (if introducing a type style) or 0x19 (if introducing a font). The second character contains the style or font code. If the character identifies a style or styles it has the following format.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	Turn off	subscript	superscript	smallcaps	underline	italics	boldface

Bits 0 through 5 identify the style(s) to be turned on or turned off. The state of Bit 6 determines whether the style(s) is to be turned on or off. When TRUE, the styles are turned off. Bit 7 is always FALSE.

If the character identifies a font it has the following format.

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
			id 4	id 3	id 2	id 1	id 0

Bits 7 and 5 are always FALSE. Bits 0 through 4 encode one of 32 possible font identifiers. Font 0 is always the Cindex default font; fonts 1 through 31 can represent any fonts. The mapping of font identifiers to particular fonts is handled through the Cindex font table, to which you have access via the API call CINAPI_GETFONTINFO.