CIFLIB
C Language
Application Program Interface
Reference Guide

CIFLIB Version 1.12

Based on Dictionary Description Language v. 2.1
July 1996

Shu-Hsin Hsieh
John D. Westbrook
Nucleic Acid Database Project
Department of Chemistry
Rutgers, The State University of New Jersey

Please direct comments on this document to jwest@ndb.rutgers.edu.

CIFLIB - C Language Application Program Interface

Contents

1 Introduction

CIFLIB [1] is a class library that was developed by the Nucleic Acid Database (NDB) Project [2] to provide an application interface to Crystallographic Information File (CIF) [3, 4, 5, 6] data. CIFLIB is designed to completely encapsulate all I/O operations and integrity checking on CIF dictionaries and data files from a calling application. CIFLIB performs the following functions:

- reads and writes CIF format data files and dictionaries.
- provides read and write access for individual data items.
- performs detailed integrity checks on CIF data and dictionaries as defined by the Dictionary Description Language (DDL) 2.1 [7, 8, 9, 10].
- provides utility methods for navigating a CIF schema
- provides a stable callable interface in C, C^{++} and FORTRAN.

Figure 1 illustrates how this software library facilitates using CIF as an interchange format in the automated data processing scheme used by the NDB. As the figure illustrates, CIFLIB provides complete access to the DDL, CIF dictionaries and CIF data files. This library can be used to build wrappers and filters around existing applications which need to access CIF data. Because CIFLIB provides complete access to the dictionary schema, the library can be conveniently used as an in-memory database or as a loader for an external database.

This is the first in a series of documents that will present the features of CIFLIB. In this document, the C language application interface [11] for CIFLIB is described.

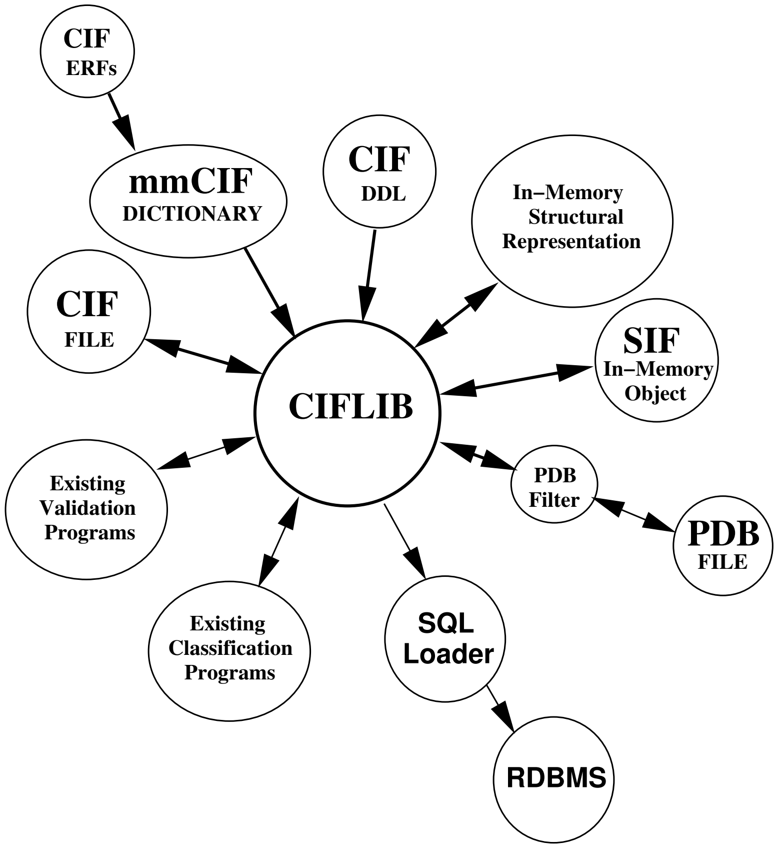

Figure 1: Functional diagram of CIFLIB

2 C Language Interface Description

This section describes the C language application program interface to CIFLIB. The description of this interface has been divided into the following subsections:

- Terminology
- Functions which construct and access CIF schema
- Functions which perform file operations, initialization, and housekeeping
- Functions which operate on or provide information about data blocks
- Functions which operate on or provide information about categories
- Functions which operate on or provide information about category groups
- Functions which operate on or provide information about subcategories
- Functions which operate on or provide information about data items
- Functions which return data item values
- Functions which update data item values
- Various item convenience functions
- Functions which return parent/child relationship information
- Functions which return parent/child values
- Functions which manage error handling and printing
- Missing functionality

2.1 CIFLIB Terminology

In order to ensure uniformity in the description and identification of variables and functions in the CIFLIB C language interface, an attempt has been made to use the following terminology in a consistent manner.

file physical or persistent instance of a collection of data blocks in the I/O subsystem of an operating system.

path the location of a file or directory in the I/O subsystem of an operating system.

data block a named container for CIF definitions and declarations. A data block defines a unit of scope in a CIF dictionary of data file.

definition a named collection of data item declarations encapsulated in a `save_` block.

declaration the instance of a CIF item.

dictionary a CIF data block containing a collection of definitions.

item the basic unit of CIF information composed of a keyword and value pair.

item name the full identifier of CIF item which is the concatenation of the category name, a dot, and the keyword name.

keyword the name of a CIF item within a CIF category.

value the data associated with a CIF item.

category a table of item values with a well defined basis. Key item values define the basis for the category and uniquely identify each tuple of item values in the category.

subcategory a named collection of items within a CIF category.

category group a named collection of CIF categories.

The following terminology has been used when referring to categories and category components:

row a tuple of item values.

column a list of item values.

row index a zero based index identifying the row order.

column index a zero based index identifying the column order.

2.2 CIFLIB File Access Functions

Accessing data in a CIF format using CIFLIB is a multistep process. CIFLIB first reads a DDL file. Although much of CIFLIB is hardwired for DDL 2.1, many DDL attributes act simply as placeholders for information and can be extended without modification to the library. The DDL is also checked against itself using internally coded rules based on DDL 2.1.

Once the DDL is read, a CIF dictionary based on this DDL can be read and checked. This process can be quite time consuming for large dictionaries, so a provision has been made to retain the state of any file which has been checked in an auxiliary file. This auxiliary file will be used in preference to the original file in subsequent file accesses if its modification date is more recent.

Finally, CIF data files are read with respect to a CIF dictionary. In any file access, CIFLIB provides complete access to the data blocks containing the DDL, the CIF dictionary, and any number of blocks containing user data.

The following sections present the set of functions which provide access to files containing CIF DDL, dictionaries, and data files. Initialization and housekeeping functions are also presented.

2.2.1 cifInit

NAME

cifInit

PROTOTYPE

```
#include "ciflib.h"

int cifInit(const char *ddlFilename,
            const int   verify,
            const int   verbose)
```

PURPOSE

cifInit performs all initialization functions for CIFLIB. This function must be called before any other CIFLIB functions. *cifInit* reads the named DDL file and optionally performs data integrity checks on this file with respect to the DDL 2.1 vocabulary. Normally, *cifInit* is called to read the DDL and to build the schema into which a dictionary can be loaded. A CIF dictionary is then read using *cifReadFile* and specifying the DDL as the validation dictionary. Data files are then read using *cifReadFile* and specifying the CIF dictionary as the validation dictionary. The value of the `verify` argument determines if integrity processing is performed in all subsequent read operations. If the `verbose` argument is selected then CIFLIB will generate informational messages describing its internal operations.

RECEIVES

`ddlFileName`	path name of DDL file
`verify`	a non-zero value activates integrity checking
`verbose`	a non-zero value activates verbose output of diagnostic and informational messages from all library functions

RETURN VALUE

Returns 1 if the function could successfully parse the input DDL file, or 0 if an unrecoverable parsing error occurs. Even if the file is successfully parsed, warning and error messages may exist which can be examined using the routines described in Section 2.14. *cifInit* returns the following additional codes:

`CIF_ALLOCATION_FAILURE`	memory allocation error
`CIF_NOT_EXIST`	missing DDL file
`CIF_OPERATION_DISALLOWED`	miscellaneous error
`CIF_OPERATION_PERMISSION_DENIED`	file permission error

REMARKS

See also:	cifFree
	cifReadFile

2.2.2 cifFree

NAME

cifFree

PROTOTYPE

```
#include "ciflib.h"

void cifFree()
```

PURPOSE

cifFree gracefully deallocates all internal CIFLIB data structures. This is the prefered way to close the interaction of a program with CIFLIB.

RECEIVES

`No arguments`

RETURN VALUE

None

REMARKS

See also: cifInit

2.2.3 cifReadFile

NAME

cifReadFile

PROTOTYPE

```
#include "ciflib.h"

int  cifReadFile(const char *fileName,
                 const char *dicDataBlockName)
```

PURPOSE

cifReadFile reads the named file and optionally performs data integrity checks on this file with respect to the named validation dictionary. Normally, *cifInit* is called to read the DDL and build the schema into which a dictionary can be loaded. A CIF dictionary is then read using *cifReadFile* and specifying the DDL as the validation dictionary. Data files are then read using *cifReadFile* specifying as an argument the CIF dictionary to be used for integrity processing. After integrity processing has been performed, the internal representation of the input file is saved in an auxiliary file. Auxiliary files are stored in a concealed directory `.ciflib` within the directory containing the input file. The auxiliary file will be used by CIFLIB in all future accesses of the input file if its modification date is more recent than the input file.

RECEIVES

`fileName`	path name of input file
`dicDataBlockName`	name of the data block containing the validation dictionary

RETURN VALUE

Returns 1 if the function could successfully parse the input file, or 0 if an unrecoverable parsing error occurs. Even if the file is successfully parsed, warning and error messages may exist which can be examined using the routines described in Section 2.14. *cifInit* returns the following additional codes:

`CIF_ALLOCATION_FAILURE`	memory allocation error
`CIF_NOT_EXIST`	missing DDL file
`CIF_OPERATION_DISALLOWED`	miscellaneous error
`CIF_OPERATION_PERMISSION_DENIED`	file permission error

REMARKS

See also: *cifInit*
cifSaveFile
cifCloseFile
cifWriteFile
cifWriteDataBlock

2.2.4 cifSaveFile

NAME

cifSaveFile

PROTOTYPE

```
#include "ciflib.h"

void cifSaveFile(const char *fileName)
```

PURPOSE

cifSaveFile saves the CIFLIB internal representation of the contents of the input file into an auxiliary file. The auxiliary file is stored in a concealed directory (`.ciflib`) in the directory containing the original input file. The auxiliary file will be used in preference to the original file in all future CIFLIB file accesses if its modification date is more recent than the original file. Auxiliary files can be loaded with much less overhead than CIF files and dictionaries, so there is a substantial performance benefit to using these file when possible.

RECEIVES

`fileName`	path name of the source file read with *cifReadFile()*

RETURN VALUE

None

REMARKS

See also: *cifReadFile*
cifCloseFile
cifWriteFile
cifWriteDataBlock

2.2.5 cifCloseFile

NAME

cifCloseFile

PROTOTYPE

```
#include "ciflib.h"

int  cifCloseFile(const char *fileName)
```

PURPOSE

cifCloseFile deletes all of the data blocks that were read from the named file. This effectively deallocates all internal CIFLIB storage associated with reading the named file.

RECEIVES

`fileName`	path name of the source file

RETURN VALUE

Returns 1 for success or 0 for failure.

REMARKS

See also:	*cifSaveFile*
	cifReadFile
	cifWriteFile
	cifWriteDataBlock

2.2.6 cifWriteFile

NAME

cifWriteFile

PROTOTYPE

```
#include "ciflib.h"

int  cifWriteFile(const char *inputFileName,
                  const char *outputFileName,
                  const char  mode,
                  const int   templateFlag,
                  const int   esdFlag,
                  const char *dictionaryName,
                  const char *version)
```

PURPOSE

cifWriteFile writes all of the data blocks which were read from the named input file to the named output file. The `mode` flag determines if the write operation either overwrites or appends the output file. Setting the `templateFlag` causes the descriptions and examples of each data item to be inserted into the output file as comments. This option can be used to create the CIF template files that are used by the NDB project. Setting the `esdFlag` to zero causes precision estimates to be appended to data items that have the DDL attribute *_item_conditions.code* of *esd*; otherwise, the precision estimates are included in the appropriate data items. This choice of possible alias translations is controlled by specifying the name and version of the dictionary in which the desired names are defined.

RECEIVES

`inputFileName`	path name of the source file
`outputFileName`	path name of the output file
`mode`	output mode; 'w' overwrites and 'a' appends.
`templateFlag`	a non-zero value requests output in template format
`esdFlag`	a zero value requests appended esd's
`dictionaryName`	name of the dictionary for alias translation
`version`	dictionary version for alias translation

RETURN VALUE

Returns 1 for success or 0 for failure.

REMARKS

See also: *cifSaveFile*
cifReadFile
cifCloseFile
cifWriteDataBlock

2.2.7 cifWriteDataBlock

NAME

cifWriteDataBlock

PROTOTYPE

```
#include "ciflib.h"

int  cifWriteDataBlock(const char *outputFileName,
                       const char  mode,
                       const char *dataBlockName,
                       const int   templateFlag,
                       const int   esdFlag,
                       const char *dictionaryName,
                       const char *version)
```

PURPOSE

cifWriteDataBlock writes the named data block to the named output file. The `mode` flag determines if the write operation either overwrites or appends the output file. Setting the `templateFlag` causes the descriptions and examples of each data item to be inserted into the output file as comments. This option can be used to create the CIF template files that are used by the NDB project. Setting the `esdFlag` to zero causes precision estimates to be appended to data items that have the DDL attribute *_item_conditions.code* of *esd*; otherwise, the precision estimates are included in the appropriate data items. The choice of possible alias translations is controlled by specifying the name and version of the dictionary in which the desired names are defined.

RECEIVES

`dataBlockName`	name of the source data block
`outputFileName`	path name of the output file
`mode`	output mode; 'w' overwrites and 'a' appends.
`templateFlag`	a non-zero value requests output in template format
`esdFlag`	a zero value requests appended esd's
`dictionaryName`	name of the dictionary for alias translation
`version`	dictionary version for alias translation

RETURN VALUE

Returns 1 for success or 0 for failure.

REMARKS

See also: *cifSaveFile*
cifReadFile
cifCloseFile
cifWriteFile

2.3 CIFLIB Schema Construction Functions

This section includes the set of functions that build and destroy *CifSchema* structures. This structure contains the tabular representation of the data structure within a data block. It contains the names of the categories declared within each data block and the items declared within each category. This structure defines the indices used to identify data blocks, categories, and items in all CIFLIB functions.

The *CifSchema* structure has the following form:

```
#define "ciflib.h"

typedef struct _CifSchema {
  int numCategory;
  CategorySchema *categories;
  char dataBlockName[CIF_MAXSTRLEN];
} CifSchema;

typedef struct _CategorySchema {
  char *category;
  int  numItem;
  ItemSchema *items;
} CategorySchema;


typedef struct _ItemSchema{
  char *item;
  int presentationOrder;
} ItemSchema;
```

2.3.1 cifConstructSchema

NAME

cifConstructSchema

PROTOTYPE

```
#include "ciflib.h"

CifSchema *cifConstructSchema(const int dataBlockIndex)
```

PURPOSE

cifConstructSchema builds the schema for the data block specified by the input argument.

RECEIVES

`dataBlockIndex`	zero-based index of a data block within the current file

RETURN VALUE

A pointer to *CifSchema* is returned if the operation is successful, or `NULL` for failure.

REMARKS

See also: *cifConstructSchemaByAlias*
cifFreeSchema
cifPrintSchema
cifConstructSchemas
cifConstructSchemasByAlias
cifFreeSchemas
cifPrintSchemas

2.3.2 cifConstructSchemaByAlias

NAME

cifConstructSchemaByAlias

PROTOTYPE

```
#include "ciflib.h"

CifSchema *cifConstructSchemaByAlias(const int dataBlockIndex)
```

PURPOSE

cifConstructSchemaByAlias builds the schema for the data block specified by the input argument. The schema structure returned by this function will contain any alias names used in the target data block.

RECEIVES

`dataBlockIndex`	zero-based index of a data block within the current file

RETURN VALUE

A pointer to *CifSchema* is returned if the operation is successful, or `NULL` for failure.

REMARKS

See also:	*cifConstructSchema* *cifFreeSchema* *cifPrintSchema* *cifConstructSchemas* *cifConstructSchemasByAlias* *cifFreeSchemas* *cifPrintSchemas*

2.3.3 cifConstructSchemas

NAME

cifConstructSchemas

PROTOTYPE

```
#include "ciflib.h"

CifSchema **cifConstructSchemas(int *numDataBlock)
```

PURPOSE

cifConstructSchemas builds an array of schemas for each data block in the current active file.

RECEIVES

`numDataBlock`	An integer pointer to hold the number of data blocks in the current file

RETURN VALUE

A pointer to an array of *CifSchema* is returned if the operation is successful, or `NULL` failure. The number of data blocks found by the operation is returned in `numDataBlock`.

REMARKS

See also:	*cifConstructSchema*
	cifConstructSchemaByAlias
	cifFreeSchema
	cifPrintSchema
	cifConstructSchemasByAlias
	cifFreeSchemas
	cifPrintSchemas

2.3.4 cifConstructSchemasByAlias

NAME

cifConstructSchemasByAlias

PROTOTYPE

```
#include "ciflib.h"

CifSchema **cifConstructSchemasByAlias(int *numDataBlock)
```

PURPOSE

cifConstructSchemasByAlias builds an array of schemas for each data block in the current active file. Each schema structure returned by this function will contain any alias names used in the target data block.

RECEIVES

`numDataBlock`	An integer pointer to hold the number of data blocks in the current file

RETURN VALUE

A pointer to an array of *CifSchema* is returned if the operation is successful, or `NULL` for failure. The number of data blocks found by the operation is returned in `numDataBlock`.

REMARKS

See also: *cifConstructSchema*
cifConstructSchemaByAlias
cifFreeSchema
cifPrintSchema
cifConstructSchemas
cifFreeSchemas
cifPrintSchemas

2.3.5 cifFreeSchema

NAME

cifFreeSchema

PROTOTYPE

```
#include "ciflib.h"

void  cifFreeSchema(CifSchema *schema)
```

PURPOSE

cifFreeSchema frees the memory allocated to a *CifSchema* structure.

RECEIVES

`schema`	a pointer to a CifSchema structure

RETURN VALUE

None

REMARKS

See also: *cifConstructSchema*
cifConstructSchemaByAlias
cifPrintSchema
cifConstructSchemas
cifConstructSchemasByAlias
cifFreeSchemas
cifPrintSchemas

2.3.6 cifFreeSchemas

NAME

cifFreeSchemas

PROTOTYPE

```
#include "ciflib.h"

void cifFreeSchemas(CifSchema **schema,
                    const int   numDataBlock)
```

PURPOSE

cifFreeSchemas frees the array of *CifSchema* passed as an input argument.

RECEIVES

`schema`	A pointer to an array of *CifSchema*
`numDataBlock`	An integer which specifies the number of data blocks in the current file

RETURN VALUE

None

REMARKS

See also: *cifConstructSchema*
cifConstructSchemaByAlias
cifFreeSchema
cifPrintSchema
cifConstructSchemas
cifConstructSchemasByAlias
cifPrintSchemas

2.3.7 cifPrintSchema

NAME

cifPrintSchema

PROTOTYPE

```
#include "ciflib.h"

void cifPrintSchema(const CifSchema *schema)
```

PURPOSE

cifPrintSchema prints the contents of a *CifSchema* structure to the standard output stream (stdout).

RECEIVES

`schema`	A pointer to a CifSchema structure.

RETURN VALUE

None

REMARKS

See also: *cifConstructSchema*
cifConstructSchemaByAlias
cifFreeSchema
cifConstructSchemas
cifConstructSchemasByAlias
cifFreeSchemas
cifPrintSchemas

2.3.8 cifPrintSchemas

NAME

cifPrintSchemas

PROTOTYPE

```
#include "ciflib.h"

void cifPrintSchemas(CifSchema **schema,
                     const int   numDataBlock)
```

PURPOSE

cifPrintSchemas prints the contents of the array of *CifSchema* on the standard output stream (stdout).

RECEIVES

`schema`	A pointer to an array of *CifSchema*
`numDataBlock`	An integer which specifies the number of data blocks in the current file

RETURN VALUE

None

REMARKS

See also:	*cifConstructSchema*
	cifConstructSchemaByAlias
	cifFreeSchema
	cifPrintSchema
	cifConstructSchemas
	cifConstructSchemasByAlias
	cifFreeSchemas

2.4 CIFLIB Data Block Access Functions

CIF files are divided into sections called data blocks. CIFLIB treats each data block as an independent database loaded into the schema of its associated dictionary. The CIF DDL is at the top of the chain and provides the schema for a CIF dictionary. The CIF dictionary in turn provides the schema for CIF data files. This section presents the collection of access and manipulation functions for data blocks.

2.4.1 cifCountDataBlocks

NAME

cifCountDataBlocks

PROTOTYPE

```
#include "ciflib.h"

int    cifCountDataBlocks()
```

PURPOSE

cifCountDataBlocks counts the current number of data blocks.

RECEIVES

`No Arguments`

RETURN VALUE

The current number of data blocks.

REMARKS

None

2.4.2 cifGetDataBlockName

NAME

cifGetDataBlockName

PROTOTYPE

```
#include "ciflib.h"

char  *cifGetDataBlockName(const int dataBlockIndex)
```

PURPOSE

cifGetDataBlockName gets the name of the data block with the specified index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block

RETURN VALUE

The name of the target data block or a `NULL` value for failure.

REMARKS

See also: *cifGetDataBlockNames*
cifGetDataBlockIndex

2.4.3 cifGetDataBlockNames

NAME

cifGetDataBlockNames

PROTOTYPE

```
#include "ciflib.h"

char **cifGetDataBlockNames(int *numDataBlock)
```

PURPOSE

cifGetDataBlockNames gets an array of data block names and the current number of data blocks.

RECEIVES

`numDataBlock`	integer pointer to hold the number of data blocks

RETURN VALUE

An array of data block names and the number of data blocks or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockName* *cifGetDataBlockIndex*

2.4.4 cifGetDataBlockIndex

NAME

cifGetDataBlockIndex

PROTOTYPE

```
#include "ciflib.h"

int    cifGetDataBlockIndex(const char *dataBlockName)
```

PURPOSE

cifGetDataBlockIndex retrieves the index of the named data block.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

The index of the named data block or the value -1 for failure.

REMARKS

See also: *cifGetDataBlockNames*
cifGetDataBlockName

2.4.5 cifGetDataBlockDictionaryIndex

NAME

cifGetDataBlockDictionaryIndex

PROTOTYPE

```
#include "ciflib.h"

int    cifGetDataBlockDictionaryIndex(const char *dataBlockName)
```

PURPOSE

cifGetDataBlockDictionaryIndex gets the index of the data block containing the dictionary with which the target data block is associated.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

The zero-based index of the dictionary associated with the target data block or the value -1 for failure.

REMARKS

None

2.4.6 cifGetDataBlockFileName

NAME

cifGetDataBlockFileName

PROTOTYPE

```
#include "ciflib.h"

char  *cifGetDataBlockFileName(const char *dataBlockName)
```

PURPOSE

cifGetDataBlockFileName gets the source file name containing the target data block.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

The source file name containing the target data block or a `NULL` value for failure.

REMARKS

See also: *cifGetDataBlockFileNameByIndex*

2.4.7 cifGetDataBlockFileNameByIndex

NAME

cifGetDataBlockFileNameByIndex

PROTOTYPE

```
#include "ciflib.h"

char  *cifGetDataBlockFileNameByIndex(const int dataBlockIndex)
```

PURPOSE

cifGetDataBlockFileNameByIndex gets the source file name containing the target data block.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block

RETURN VALUE

The source file name containing the target data block or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockFileName*

2.4.8 cifGetDataBlockFileOffset

NAME

cifGetDataBlockFileOffset

PROTOTYPE

```
#include "ciflib.h"

long   cifGetDataBlockFileOffset(const char *dataBlockName)
```

PURPOSE

cifGetDataBlockFileOffset retrieves the offset into the source file in which the target data block begins.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

Source file offset or the value -1 for failure.

REMARKS

See also:	*cifGetDataBlockFileOffsetByIndex*

2.4.9 cifGetDataBlockFileOffsetByIndex

NAME

cifGetDataBlockFileOffsetByIndex

PROTOTYPE

```
#include "ciflib.h"

long   cifGetDataBlockFileOffsetByIndex(const int dataBlockIndex)
```

PURPOSE

cifGetDataBlockFileOffsetByIndex retrieves the offset into the source file in which the target data block begins.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block

RETURN VALUE

Source file offset or the value -1 for failure.

REMARKS

See also:	*cifGetDataBlockFileOffset*

2.4.10 cifDeleteDataBlock

NAME

cifDeleteDataBlock

PROTOTYPE

```
#include "ciflib.h"

void   cifDeleteDataBlock(const char *dataBlockName)
```

PURPOSE

cifDeleteDataBlock deletes the target data block.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

None

REMARKS

See also:	*cifDeleteDataBlockByIndex*

2.4.11 cifDeleteDataBlockByIndex

NAME

cifDeleteDataBlockByIndex

PROTOTYPE

```
#include "ciflib.h"

void   cifDeleteDataBlockByIndex(const int dataBlockIndex)
```

PURPOSE

cifDeleteDataBlock deletes the target data block.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block

RETURN VALUE

None

REMARKS

See also:	*cifDeleteDataBlock*

2.5 CIFLIB Category Access Functions

The following sections present the set of functions which provide detailed information about CIF categories.

2.5.1 cifCountCategories

NAME

cifCountCategories

PROTOTYPE

```
#include "ciflib.h"

int cifCountCategories(const char *dataBlockName)
```

PURPOSE

cifCountCategories returns the number of categories in the named data block.

RECEIVES

`dataBlockName`	the target data block name

RETURN VALUE

Returns the number of categories or a value of -1 for failure.

REMARKS

See also:	*cifCountCategoriesByIndex*

2.5.2 cifCountCategoriesByIndex

NAME

cifCountCategoriesByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifCountCategoriesByIndex(const int dataBlockIndex)
```

PURPOSE

cifCountCategoriesByIndex returns the number categories in the data block identified by index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block

RETURN VALUE

Returns the number of categories or a value of -1 for failure.

REMARKS

See also:	*cifCountCategories*

2.5.3 cifGetCategoryName

NAME

cifGetCategoryName

PROTOTYPE

```
#include "ciflib.h"

char *cifGetCategoryName(const char *dataBlockName,
                         const int categoryIndex)
```

PURPOSE

cifGetCategoryName returns the name of the category identified by its index within the named data block.

RECEIVES

`dataBlockName`	the name of the target data block
`categoryIndex`	the zero-based index of the target category

RETURN VALUE

Returns a category name or a `NULL` value for failure.

REMARKS

See also:	*cifGetCategoryNames*
	cifGetCategoryNameByIndex
	cifGetCategoryNamesByIndex
	cifGetCategoryIndex

2.5.4 cifGetCategoryNames

NAME

cifGetCategoryNames

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoryNames(const char *dataBlockName,
                                 int  *numCategory)
```

PURPOSE

cifGetCategoryNames returns an array of category names within the named data block and the number categories.

RECEIVES

`dataBlockName`	the name of the target data block
`numCategory`	integer pointer to hold the number of categories

RETURN VALUE

Returns an array of category names or a `NULL` value for failure. If the operation is successful the number of categories is returned in `numCategory`.

REMARKS

See also:	*cifGetCategoryName*
	cifGetCategoryNameByIndex
	cifGetCategoryNamesByIndex
	cifGetCategoryIndex

2.5.5 cifGetCategoryNameByIndex

NAME

cifGetCategoryNameByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetCategoryName(const int dataBlockIndex,
                         const int categoryIndex)
```

PURPOSE

cifGetCategoryNameByIndex returns the name of the category identified by its index within the data block identified by its index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	the zero-based index of the target category

RETURN VALUE

Returns a category name or a `NULL` value for failure.

REMARKS

See also:	*cifGetCategoryName*
	cifGetCategoryNames
	cifGetCategoryNamesByIndex
	cifGetCategoryIndex

2.5.6 cifGetCategoryNamesByIndex

NAME

cifGetCategoryNamesByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoryNamesByIndex(const int  dataBlockIndex,
                                  int *numCategory)
```

PURPOSE

cifGetCategoryNamesByIndex returns an array of category names and the number categories within the data block identified by its index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`numCategory`	integer pointer to hold the number of categories

RETURN VALUE

Returns an array of category names or a `NULL` value for failure. If the operation is successful the number of categories is returned in `numCategory`.

REMARKS

See also: *cifGetCategoryName*
cifGetCategoryNames
cifGetCategoryNameByIndex
cifGetCategoryIndex

2.5.7 cifGetCategoryIndex

NAME

cifGetCategoryIndex

PROTOTYPE

```
#include "ciflib.h"

int cifGetCategoryIndex(const char *datablockName,
                        const char *categoryName)
```

PURPOSE

cifGetCategoryIndex returns the category index of the target category within the target datablock. Both the category and data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category

RETURN VALUE

Returns a category index or a value of -1 for failure.

REMARKS

None

2.5.8 cifGetCategoryKeys

NAME

cifGetCategoryKeys

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoryKeys(const char *dataBlockName,
                          const char *categoryName,
                                int  *numKey)
```

PURPOSE

cifGetCategoryKeys returns an array of key item names and the number of keys for the named category within the named data block.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`numKey`	an integer pointer to hold the number of keys

RETURN VALUE

Returns an array of key item names or a `NULL` value for failure. If the operation is successful the number of keys is returned in `numKey`.

REMARKS

See also:	*cifGetCategoryKeysByIndex*

2.5.9 cifGetCategoryKeysByIndex

NAME

cifGetCategoryKeysByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoryKeysByInex(const int  dataBlockIndex,
                                const int  categoryIndex,
                                      int *numKey)
```

PURPOSE

cifGetCategoryKeys returns an array of key item names and the number of keys for the target category within the target data block. Both category and data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`numKey`	an integer pointer to hold the number of keys

RETURN VALUE

Returns an array of key item names or a `NULL` value for failure. If the operation is successful the number of keys is returned in `numKey`.

REMARKS

See also: *cifGetCategoryKeys*

2.5.10 cifCountRows

NAME

cifCountRows

PROTOTYPE

```
#include "ciflib.h"

int cifCountRows(const char *dataBlockName,
                 const char *categoryName)
```

PURPOSE

cifCountRows returns the number of rows in the target category within the target data block. Both the category and the data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category

RETURN VALUE

Returns the number of rows or a value of -1 for failure.

REMARKS

See also: *cifCountRowsByIndex*

2.5.11 cifCountRowsByIndex

NAME

cifCountRowsByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifCountRowsByIndex(const int dataBlockIndex,
                        const int categoryIndex)
```

PURPOSE

cifCountRows returns the number of rows in the target category within the target data block. Both the category and the data block are identified by index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category

RETURN VALUE

Returns the number of rows or a value of -1 for failure.

REMARKS

See also:	*cifCountRows*

2.5.12 cifCountColumns

NAME

cifCountColumns

PROTOTYPE

```
#include "ciflib.h"

int cifCountColumns(const char *dataBlockName,
                    const char *categoryName)
```

PURPOSE

cifCountColumns returns the of number columns in the target category within the target data block. Both the category and the data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category

RETURN VALUE

Returns the number of columns or a value of -1 for failure.

REMARKS

See also: *cifCountColumnsByIndex*

2.5.13 cifCountColumnsByIndex

NAME

cifCountColumnsByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifCountColumnsByIndex(const int dataBlockIndex,
                           const int categoryIndex)
```

PURPOSE

cifCountColumns returns the number of columns in the target category within the target data block. Both the category and data block are identified by index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category

RETURN VALUE

Returns the number of columns or a value of -1 for failure.

REMARKS

See also: *cifCountColumns*

2.6 CIFLIB Category Group Access Functions

The following sections present the set of functions that provide detailed information about CIF category groups.

2.6.1 cifGetCategoryGroups

NAME

cifGetCategoryGroups

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoryGroups(const char *dataBlockName,
                            int  *numGroup)
```

PURPOSE

cifGetCategoryGroups returns an array of category group names and the number groups within the named data block.

RECEIVES

`dataBlockName`	name of the target data block
`numGroup`	integer pointer to hold the number of category groups

RETURN VALUE

Returns an array of category group names or a `NULL` value for failure. If the operation is successful the number of groups is returned in `numGroup`.

REMARKS

See also: *cifGetCategoryGroupsByIndex*

2.6.2 cifGetCategoryGroupsByIndex

NAME

cifGetCategoryGroupsByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoryGroupsByIndex(const int  dataBlockIndex,
                                          int *numGroup)
```

PURPOSE

cifGetCategoryGroups returns an array of category group names and the number of groups within the data block identified by its index.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`numGroup`	integer pointer to hold the number of category groups

RETURN VALUE

Returns an array of category group names or a `NULL` value for failure. If the operation is successful the number of groups is returned in `numGroup`.

REMARKS

See also: *cifGetCategoryGroups*

2.6.3 cifGetCategoriesInCategoryGroup

NAME

cifGetCategoriesInCategoryGroup

PROTOTYPE

```
#include "ciflib.h"

char **cifGetCategoriesInCategoryGroup(const char *dataBlockName,
                                       const char *groupName,
                                              int *numCategory)
```

PURPOSE

cifGetCategoriesInCategoryGroup returns an array of category names and the number of categories contained in the target category group within the target data block. Both the data block and category group are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`groupName`	name of the target category group
`numCategory`	integer pointer to hold the returned number of categories

RETURN VALUE

Returns an array of category names or `NULL` for failure. If the operation is successful then the number of categories is returned in `numCategory`.

REMARKS

None

2.7 CIFLIB Subcategory Access Functions

The following sections present the set of functions which provide detailed information about CIF subcategories.

2.7.1 cifGetSubcategories

NAME

cifGetSubcategories

PROTOTYPE

```
#include "ciflib.h"

char **cifGetSubcategories(const char *dataBlockName,
                           const char *categoryName,
                                 int  *numSubcategory)
```

PURPOSE

cifGetSubcategories returns an array of subcategory names and the number of subcategories in the target category within the target data block. The category and data block are identified by name. This function will always return at least one subcategory which is named after the category and contains all of the category items.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`numSubcategory`	integer pointer to hold the returned number of subcategories

RETURN VALUE

Returns an array of subcategory names or a `NULL` value for failure. If the operation is successful then the number of subcategories is returned in `numSubcategory`

REMARKS

None

2.7.2 cifGetItemNamesInSubcategory

NAME

cifGetItemNamesInSubcategory

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemNamesInSubcategory(const char *dataBlockName,
                                    const char *categoryName,
                                    const char *subcategoryName,
                                           int *numItemName)
```

PURPOSE

cifGetItemNamesInSubcategory returns an array of item names and the number of items contained within the target subcategory, category, and data block. The subcategory, category, and data block are identified by name.

RECEIVES

`dataBlockName`	name of target data block
`categoryName`	name of target category
`subcategoryName`	name of target subcategory
`numItemName`	integer pointer to hold the returned number of items

RETURN VALUE

Returns an array of item names or a `NULL` value for failure. If the operation is successful then the number of items is returned in `numItemName`.

REMARKS

See also: *cifGetItemNamesInSubcategoryByIndex*

2.7.3 cifGetItemKeywordsInSubcategory

NAME

cifGetItemKeywordsInSubcategory

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemKeywordsInSubcategory(const char *dataBlockName,
                                       const char *categoryName,
                                       const char *subcategoryName,
                                             int *numKeyword)
```

PURPOSE

cifGetItemKeywordsInSubcategory returns an array of item keywords and the number of item keywords contained within the target subcategory, category and data block. The subcategory, category, and data block are identified by name.

RECEIVES

`dataBlockName`	name of target data block
`categoryName`	name of target category
`subcategoryName`	name of target subcategory
`numKeyword`	integer pointer to hold returned number of item keywords

RETURN VALUE

Returns an array of item keywords or a `NULL` value for failure. If the operation is successful then the number of keywords is returned in `numKeyword`.

REMARKS

None

2.8 CIFLIB Item Access Functions

The following sections present the set of functions which provide detailed information about CIF item names and item alias names.

2.8.1 cifGetItemKeyword

NAME

cifGetItemKeyword

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemKeyword(const char *dataBlockName,
                        const char *categoryName,
                        const int   itemKeywordIndex);
```

PURPOSE

cifGetItemKeyword returns the item keyword identified by its index within the target category within the target datablock. The category and data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeywordIndex`	index of the keyword within the target category

RETURN VALUE

Returns an item keyword or a `NULL` value for failure.

REMARKS

See also: *cifGetItemKeywordByIndex*

2.8.2 cifGetItemKeywordByIndex

NAME

cifGetItemKeywordByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemKeywordByIndex(const int   dataBlockIndex,
                               const int   categoryIndex,
                               const int   itemKeywordIndex);
```

PURPOSE

cifGetItemKeywordByIndex returns the item keyword identified by its index within the target category within the target data block. The category and data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemKeywordIndex`	index of the keyword within the target category

RETURN VALUE

Returns an item keyword or a `NULL` value for failure.

REMARKS

See also: *cifGetItemKeyword*

2.8.3 cifGetItemKeywords

NAME

cifGetItemKeywords

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemKeywords(const char *dataBlockName,
                          const char *categoryName,
                                int  *numKeyword);
```

PURPOSE

cifGetItemKeywords returns an array of item keywords within the target category within the target data block. The category and data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`numKeyword`	integer pointer to hold returned number of keywords

RETURN VALUE

Returns an array of item keywords or a `NULL` value for failure. If the operation is successful, the number of keywords is returned in `numKeyword`.

REMARKS

See also: *cifGetItemKeywordsByIndex*

2.8.4 cifGetItemKeywordsByIndex

NAME

cifGetItemKeywordsByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemKeywordsByIndex(const int  dataBlockIndex,
                                 const int  categoryIndex,
                                       int *numKeyword);
```

PURPOSE

cifGetItemKeywordsByIndex returns an array of item keywords within the target category within the target data block. The category and data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`numKeyword`	integer pointer to hold returned number of keywords

RETURN VALUE

Returns an array of item keywords or a `NULL` value for failure. If the operation is successful the number of keywords is returned in `numKeyword`.

REMARKS

See also: *cifGetItemKeywords*

2.8.5 cifGetItemName

NAME

cifGetItemName

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemName(const char *dataBlockName,
                     const char *categoryName,
                     const int   itemIndex);
```

PURPOSE

cifGetItemName returns the item name identified by its index within the target category within the target data block.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemIndex`	zero-based index of the item within the target category

RETURN VALUE

Returns an item keyword or a `NULL` value for failure.

REMARKS

See also: *cifGetItemNameByIndex*

2.8.6 cifGetItemNameByIndex

NAME

cifGetItemNameByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemNameByIndex(const int dataBlockIndex,
                            const int categoryIndex,
                            const int itemIndex)
```

PURPOSE

cifGetItemNameByIndex returns the item name identified by its index within the target category within the target data block. The category and data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the item within the target category

RETURN VALUE

Returns an item keyword or a `NULL` value for failure.

REMARKS

See also: *cifGetItemName*

2.8.7 cifGetItemNames

NAME

cifGetItemNames

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemNames(const char *dataBlockName,
                       const char *categoryName,
                            int  *numItem);
```

PURPOSE

cifGetItemNames returns an array of item names within the target category and within the target data block. The category and the data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`numItem`	integer pointer to hold returned number of items

RETURN VALUE

Returns an array of item names or a `NULL` value for failure. If the operation is successful the number of keywords is returned in `numKeyword`.

REMARKS

See also: *cifGetItemNamesByIndex*

2.8.8 cifGetItemNamesByIndex

NAME

cifGetItemNamesByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemNamesByIndex(const int  dataBlockIndex,
                              const int  categoryIndex,
                                    int *numItem);
```

PURPOSE

cifGetItemNamesByIndex returns an array of item names within the target category and within the target data block. The category and the data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`numItem`	integer pointer to hold returned number of items

RETURN VALUE

Returns an array of item names or a `NULL` value for failure. If the operation is successful the number of keywords is returned in `numKeyword`.

REMARKS

See also: *cifGetItemNames*

2.8.9 cifGetItemAliasName

NAME

cifGetItemAliasName

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemAliasName(const char *dataBlockName,
                          const char *categoryName,
                          const char *itemKeyword);
```

PURPOSE

cifGetItemAliasName returns the alias name used to declare the item which is identified by its keyword name within the target category and within the target data block.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target keyword in the the target category

RETURN VALUE

Returns an alias name or a `NULL` value for failure.

REMARKS

See also: *cifGetItemAliasNameByIndex*

2.8.10 cifGetItemAliasNameByIndex

NAME

cifGetItemAliasNameByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemAliasNameByIndex(const int dataBlockIndex,
                                 const int categoryIndex,
                                 const int itemKeywordIndex);
```

PURPOSE

cifGetItemAliasNameByIndex returns the alias name used to declare the item which is identified by its index within the target category within the target data block. The category and data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemKeywordIndex`	zero-based index of the item within the target category

RETURN VALUE

Returns an alias name or a `NULL` value for failure.

REMARKS

See also: *cifGetItemAliasName*

2.8.11 cifGetItemAliasNames

NAME

cifGetItemAliasNames

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemAliasNames(const char *dataBlockName,
                            const char *categoryName,
                                  int  *numItem);
```

PURPOSE

cifGetItemAliasNames returns an array of item names within the target category within the target data block. If an item has been declared using a valid alias name, then that alias name is returned by this function. This behavior differs from *cifGetItemNames* which always returns item names defined in the current dictionary even if those items have been declared using alias names. The category and data block are identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`numItem`	integer pointer to hold returned number of item names

RETURN VALUE

Returns an array of item names or a `NULL` value for failure. If the operation is successful the number of item names is returned in `numItem`.

REMARKS

See also: *cifGetItemAliasNamesByIndex*

2.8.12 cifGetItemAliasNamesByIndex

NAME

cifGetItemAliasNamesByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetItemAliasNamesByIndex(const int  dataBlockIndex,
                                   const int  categoryIndex,
                                         int *numItem);
```

PURPOSE

cifGetItemAliasNamesByIndex returns an array of item alias names within the target category within the target data block. If an item has been declared using a valid alias name, then that alias name is returned by this function. This behavior differs from *cifGetItemNamesByIndex* which always returns item names defined in the current dictionary even if those items have been declared using alias names. The category and data block are identified by their indices.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`numItem`	integer pointer to hold returned number of item names

RETURN VALUE

Returns an array of item alias names or a `NULL` value for failure. If the operation is successful the number of item names is returned in `numItem`.

REMARKS

See also: *cifGetItemAliasNames*

2.9 CIFLIB Item Value Access Functions

The following sections present the set of functions which provide read access to the the values of individual CIF items. These functions also check the integrity of item values with respect the their dictionary definitions.

2.9.1 cifGetItemValue

NAME

cifGetItemValue

PROTOTYPE

```
#include "ciflib.h"

int  cifGetItemValue(      char **value,
                     const char  *dataBlockName,
                     const char  *categoryName,
                     const char  *itemKeyword,
                     const int    rowIndex)
```

PURPOSE

cifGetItemValue gets the address of the string representing the value of an item. The target item is identified by the name of the keyword, category, and data block in which it is defined. The row index is provided to select a particular item value when multiple values exist within the category. This function checks the integrity of the target item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of the string representing the value of the item
`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`rowIndex`	zero-based index of the target row

RETURN VALUE

Returns an integer CIFLIB error code. If the code `CIF_DATA_IS_VALID` is returned, then the value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetItemValueByIndex*
cifGetItemValueByAlias

2.9.2 cifGetItemValueByIndex

NAME

cifGetItemValueByIndex

PROTOTYPE

```
#include "ciflib.h"

int  cifGetItemValueByIndex(      char **value,
                            const int    dataBlockIndex,
                            const int    categoryIndex,
                            const int    itemIndex,
                            const int    rowIndex)
```

PURPOSE

cifGetItemValue gets the address of the string representing the value of an item. The target item is identified by the index of the keyword, category, and data block in which it is defined. The row index is provided to select a particular item value when multiple values exist within the category. This function checks the integrity of the target item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of the string representing the value of the item
`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the target item within the target category
`rowIndex`	zero-based index of the target row

RETURN VALUE

Returns an integer CIFLIB error code. If the code `CIF_DATA_IS_VALID` is returned, then the value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetItemValue*
cifGetItemValueByAlias

2.9.3 cifGetItemValueByAlias

NAME

cifGetItemValueByAlias

PROTOTYPE

```
#include "ciflib.h"

int  cifGetItemValue(      char **value,
                     const int    dataBlockIndex,
                     const char  *aliasName,
                     const int    rowIndex)
```

PURPOSE

cifGetItemValueByAlias gets the address of the string representing the value of an item. The target item is identified by an item alias name and the index of the data block in which it is defined. The row index is provided to select a particular item value when multiple values exist within the category. This function checks the integrity of the target item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of the string representing the value of the item
`dataBlockIndex`	zero-based index of the target data block
`aliasName`	name of the target item within the target category
`rowIndex`	zero-based index of the target row

RETURN VALUE

Returns an integer CIFLIB error code. If the code `CIF_DATA_IS_VALID` is returned, then the value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetItemValue*
cifGetItemValueByIndex

2.9.4 cifGetRow

NAME

cifGetRow

PROTOTYPE

```
#include "ciflib.h"

int  *cifGetRow(      char ***value,
                const char   *dataBlockName,
                const char   *categoryName,
                const int     rowIndex
                      int    *numItem)
```

PURPOSE

cifGetRow gets the address of an array of strings representing the values of each item in the target row of the target table. The table is identified by the name of the category, and by the data block in which it is defined. This function checks the integrity of each item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of an array of strings representing the values of each item
`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`rowIndex`	zero-based index of the target row
`numItem`	address of the integer to hold the number of items in the row

RETURN VALUE

Returns an array of integer CIFLIB error codes. If a code `CIF_DATA_IS_VALID` is returned, then the associated value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetRowByIndex*

2.9.5 cifGetRowByIndex

NAME

cifGetRowByIndex

PROTOTYPE

```
#include "ciflib.h"

int  *cifGetRowByIndex(      char ***value,
                       const int     dataBlockIndex,
                       const int     categoryIndex,
                       const int     rowIndex
                             int    *numItem)
```

PURPOSE

cifGetRowByIndex gets the address of an array of strings representing the values of each item in the target row of the target table. The table is identified by the index of the category, and by the data block in which it is defined. This function checks the integrity of each item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of an array of strings representing the values of each item
`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`rowIndex`	zero-based index of the target row
`numItem`	address of the integer to hold the number of items in the row

RETURN VALUE

Returns an array of integer CIFLIB error codes. If a code `CIF_DATA_IS_VALID` is returned, then the associated value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetRowByIndex*

2.9.6 cifGetColumn

NAME

cifGetColumn

PROTOTYPE

```
#include "ciflib.h"

int  *cifGetColumn(      char  ***value,
                   const char    *dataBlockName,
                   const char    *categoryName,
                   const char    *itemKeyword,
                         int     *numValues)
```

PURPOSE

cifGetColumn gets the address of an array of strings representing the values of each item in the target column of the target table. The table is identified by the name of the category, and by the data block in which it is defined. This function checks the integrity of each item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of an array of strings representing the values of each item
`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`numValues`	address of the integer to hold the number of values in the column

RETURN VALUE

Returns an array of integer CIFLIB error codes. If a code `CIF_DATA_IS_VALID` is returned, then the associated value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetColumnByIndex*

2.9.7 cifGetColumnByIndex

NAME

cifGetColumnByIndex

PROTOTYPE

```
#include "ciflib.h"

int  *cifGetColumnByIndex(      char ***value,
                          const int     dataBlockIndex,
                          const int     categoryIndex,
                          const int     itemIndex
                                int     *numValues)
```

PURPOSE

cifGetColumnByIndex gets the address of an array of strings representing the values of each item in the target column of the target table. The table is identified by the index of the category, and by the data block in which it is defined. This function checks the integrity of each item value with respect to the item's dictionary definition.

RECEIVES

`value`	address of an array of strings representing the values of each item
`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the target item within the target category
`numValues`	address of the integer to hold the number of values in the column

RETURN VALUE

Returns an array of integer CIFLIB error codes. If a code `CIF_DATA_IS_VALID` is returned, then the associated value returned as a string is compliant with its dictionary definition.

REMARKS

See also: *cifGetColumnByIndex*

2.10 CIFLIB Item Value Update Functions

The following sections present the set of functions which provide write access to the the values of individual CIF items. These functions also check the integrity of item values with respect the their dictionary definitions.

2.10.1 cifUpdateItemValue

NAME

cifUpdateItemValue

PROTOTYPE

```
#include "ciflib.h"

int  cifUpdateItemValue(      char *itemValue,
                        const char *dataBlockName,
                        const char *categoryName,
                        const char *itemKeyword,
                        const int   rowIndex)
```

PURPOSE

cifUpdateItemValue updates an item value using the source string representing the value of the item. The target item is identified by the name of the keyword, category, and data block in which the item is defined. The row index selects an existing row in the table. This function checks the integrity of the source item value with respect to the item's dictionary definition.

RECEIVES

`value`	string representing the value of the item
`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`rowIndex`	zero-based index of the target row

RETURN VALUE

Returns an integer CIFLIB error code. If the code `CIF_DATA_IS_VALID` is returned, then the source string representing the item value is compliant with the dictionary definition.

REMARKS

See also:	*cifUpdateItemValueByIndex*
	cifUpdateItemValueByAlias
	cifAddRow
	cifAddRowByIndex

2.10.2 cifUpdateItemValueByIndex

NAME

cifUpdateItemValueByIndex

PROTOTYPE

```
#include "ciflib.h"

int  cifUpdateItemValueByIndex(      char *itemValue,
                               const int   dataBlockIndex,
                               const int   categoryIndex,
                               const int   itemIndex,
                               const int   rowIndex)
```

PURPOSE

cifUpdateItemValueByIndex updates an item value using the source string representing the value of the item. The target item is identified by the index of the keyword, category, and data block in which the item is defined. The row index selects an existing row in the table. This function checks the integrity of the source item value with respect to the item's dictionary definition.

RECEIVES

`value`	string representing the value of the item
`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the target item within the target category
`rowIndex`	zero-based index of the target row

RETURN VALUE

Returns an integer CIFLIB error code. If the code `CIF_DATA_IS_VALID` is returned, then the source string representing the item value is compliant with the dictionary definition.

REMARKS

See also: *cifUpdateItemValue*
cifUpdateItemValueByAlias
cifAddRow
cifAddRowByIndex

2.10.3 cifUpdateItemValueByAlias

NAME

cifUpdateItemValueByAlias

PROTOTYPE

```
#include "ciflib.h"

int  cifUpdateItemValueByAlias(      char *itemValue,
                               const int   dataBlockIndex,
                               const char *aliasName,
                               const int   rowIndex)
```

PURPOSE

cifUpdateItemValueByAlias updates an item value using the source string representing the value of the item. The target item is identified by the item alias name and the data block in which the item is defined. The row index selects an existing row in the table. This function checks the integrity of the source item value with respect to the item's dictionary definition.

RECEIVES

`value`	string representing the value of the item
`dataBlockIndex`	zero-based index of the target data block
`aliasName`	alias name of the target item
`rowIndex`	zero-based index of the target row

RETURN VALUE

Returns an integer CIFLIB error code. If the code `CIF_DATA_IS_VALID` is returned, then the source string representing the item value is compliant with the dictionary definition.

REMARKS

See also: *cifUpdateItemValue*
cifUpdateItemValueByIndex
cifAddRow
cifAddRowByIndex

2.10.4 cifAddRow

NAME

cifAddRow

PROTOTYPE

```
#include "ciflib.h"

int  cifAddRow(const char *dataBlockName,
               const char *categoryName)
```

PURPOSE

cifAddRow appends a row to the target category within the target data block. The data block and category are both identified by name. Once a new row has been created, other CIFLIB update functions can be used to load item values into the row.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category

RETURN VALUE

Returns the index for the new row or the value -1 for failure.

REMARKS

See also:	*cifUpdateItemValue*
	cifUpdateItemValueByIndex
	cifUpdateItemValueByAlias
	cifAddRowByIndex

2.10.5 cifAddRowByIndex

NAME

cifAddRowByIndex

PROTOTYPE

```
#include "ciflib.h"

int  cifAddRowByIndex(const int dataBlockName,
                      const int categoryName)
```

PURPOSE

cifAddRowByIndex appends a row to the target category within the target data block. The data block and category are both identified by index. Once a new row has been created, other CIFLIB update functions can be used to load item values into the row.

RECEIVES

`dataBlockIndex`	index of the target data block
`categoryIndex`	index of the target category

RETURN VALUE

Returns the index for the new row or the value -1 for failure.

REMARKS

See also:	*cifUpdateItemValue*
	cifUpdateItemValueByIndex
	cifUpdateItemValueByAlias
	cifAddRow

2.11 CIFLIB Item Convenience Functions

The following sections present the set of functions which provide convenient access to CIF item attributes.

2.11.1 cifGetItemFileOffset

NAME

cifGetItemFileOffset

PROTOTYPE

```
#include "ciflib.h"

long cifGetItemFileOffset(const int   dicDataBlockIndex,
                          const char *itemName)
```

PURPOSE

cifGetItemFileOffset returns the byte offset into the dictionary file at which the target item is defined in the target data block. The the data block is identified by index and the item is identified by name.

RECEIVES

`dicDataBlockIndex`	zero-based index of the target data block
`itemName`	name of the target item

RETURN VALUE

Returns an integer byte offset or the value -1 for failure.

REMARKS

See also:	*cifGetDataBlockFileOffset*
	cifGetDataBlockFileOffsetByIndex

2.11.2 cifGetItemCaseSensitivity

NAME

cifGetItemCaseSensitivity

PROTOTYPE

```
#include "ciflib.h"

int cifGetItemCaseSensitivity(const char *dataBlockName,
                              const char *categoryName,
                              const char *itemKeyword)
```

PURPOSE

cifGetItemCaseSensitivity returns the case sensitivity of the target item. The target item is identified by the keyword name and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target keyword in the target category

RETURN VALUE

Returns a value of 1 if the item is case sensitive or a value of zero if the item is case insensitive. The functions returns the value -1 for failure.

REMARKS

See also: *cifGetItemCaseSensitivityByIndex*

2.11.3 cifGetItemCaseSensitivityByIndex

NAME

cifGetItemCaseSensitivityByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifGetItemCaseSensitivityByIndex(const int  dataBlockIndex,
                                     const int  categoryIndex,
                                     const int  itemIndex)
```

PURPOSE

cifGetItemCaseSensitivityByIndex returns the case sensitivity of the target item. The target item is identified by its item index and the indices of the category and data block in which it is defined.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the target keyword in the target category

RETURN VALUE

Returns a value of 1 if the item is case sensitive or a value of zero if the item is case insensitive. The functions returns the value -1 for failure.

REMARKS

See also: *cifGetItemCaseSensitivity*

2.11.4 cifGetItemDefaultValue

NAME

cifGetItemDefaultValue

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemDefaultValue(const char *dataBlockName,
                             const char *categoryName,
                             const char *itemKeyword)
```

PURPOSE

cifGetItemDefaultValue returns a character string representing the default value of the target item. The target item is identified by the keyword name and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target keyword in the target category

RETURN VALUE

Returns a character string representing the default value or a `NULL` value for failure.

REMARKS

See also: *cifGetItemDefaultValueByIndex*

2.11.5 cifGetItemDefaultValueByIndex

NAME

cifGetItemDefaultValueByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetItemDefaultValueByIndex(const int  dataBlockIndex,
                                    const int  categoryIndex,
                                    const int  itemIndex)
```

PURPOSE

cifGetItemDefaultValueByIndex returns a character string representing the default value of the target item. The target item is identified by its item index and the indices of the category and data block in which it is defined.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the target keyword in the target category

RETURN VALUE

Returns a character string representing the default value or a `NULL` value for failure.

REMARKS

See also: *cifGetItemDefaultValue*

2.11.6 cifGetItemPrimitiveCode

NAME

cifGetItemPrimitiveCode

PROTOTYPE

```
#include "ciflib.h"

int cifGetItemPrimitiveCode(const char *dataBlockName,
                            const char *categoryName,
                            const char *itemKeyword)
```

PURPOSE

cifGetItemPrimitiveCode returns an integer code describing the primitive data type of the target item. The target item is identified by the keyword name and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target keyword in the target category

RETURN VALUE

Returns `CIF_STRING_VALUE`, `CIF_INTEGER_VALUE`, `CIF_DOUBLE_VALUE`, or the value -1 for failure.

REMARKS

See also: *cifGetItemPrimitiveCodeByIndex*

2.11.7 cifGetItemPrimitiveCodeByIndex

NAME

cifGetItemPrimitiveCodeByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifGetItemPrimitiveCodeByIndex(const int  dataBlockIndex,
                                   const int  categoryIndex,
                                   const int  itemIndex)
```

PURPOSE

cifGetItemPrimitiveCodeByIndex returns an integer code describing the primitive data type of the target item. The target item is identified by its item index and the indices of the category and data block in which it is defined.

RECEIVES

`dataBlockIndex`	zero-based index of the target data block
`categoryIndex`	zero-based index of the target category
`itemIndex`	zero-based index of the target keyword in the target category

RETURN VALUE

Returns `CIF_STRING_VALUE`, `CIF_INTEGER_VALUE`, `CIF_DOUBLE_VALUE`, or the value -1 for failure.

REMARKS

See also: *cifGetItemPrimitiveCode*

2.12 CIFLIB Parent/Child Access Functions

The following sections present the set of functions which provide information about parent/child relationships, and provide access to the parent and child item values. The parent and child relationships returned by the functions in this section span a single generation.

2.12.1 cifGetParentItemNames

NAME

cifGetParentItemNames

PROTOTYPE

```
#include "ciflib.h"

char **cifGetParentItemNames(const char *dataBlockName,
                             const char *categoryName,
                             const char *itemKeyword,
                                   int  *numParents)
```

PURPOSE

cifGetParentItemNames returns an array of parent item names for the the target item. The target item is identified by its keyword and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`numParents`	address of the integer to hold the number of parent items

RETURN VALUE

Returns an array of parent item names or a `NULL` value. The number of parent items is returned in `numParents`.

REMARKS

See also: *cifGetParentIndexList*

2.12.2 cifGetChildItemNames

NAME

cifGetChildItemNames

PROTOTYPE

```
#include "ciflib.h"

char **cifGetChildItemNames(const char *dataBlockName,
                            const char *categoryName,
                            const char *itemKeyword,
                                  int  *numChildren)
```

PURPOSE

cifGetChildItemNames returns an array of child item names for the target item. The target item is identified by its keyword and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`numChildren`	address of the integer to hold the number of child items

RETURN VALUE

Returns an array of child item names or a `NULL` value. The number of child items is returned in `numChildren`.

REMARKS

See also: *cifGetChildIndexList*

2.12.3 cifGetParentIndexList

NAME

cifGetParentIndexList

PROTOTYPE

```
#include "ciflib.h"

int cifGetParentIndexList(const char  *dataBlockName,
                          const char  *categoryName,
                          const char  *itemKeyword,
                                int  **parentCategoryIndex,
                                int  **parentItemIndex,
                                int   *numParents)
```

PURPOSE

cifGetParentIndexList retrieves arrays of category and item indices for the the parents of the target item. The target item is identified by its keyword and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`parentCategoryIndex`	address of the integer array to hold parent category indices
`parentItemIndex`	address of the integer array to hold parent item indices
`numParents`	address of the integer to hold the number of parent items

RETURN VALUE

Returns a value of 1 if the operation was successful or a value of 0 for failure. If the operation is successful then the index arrays and number of parents are also returned.

REMARKS

See also: *cifGetParentItemNames*

2.12.4 cifGetChildIndexList

NAME

cifGetChildIndexList

PROTOTYPE

```
#include "ciflib.h"

int cifGetChildIndexList(const char  *dataBlockName,
                         const char  *categoryName,
                         const char  *itemKeyword,
                               int  **childCategoryIndex,
                               int  **childItemIndex,
                               int   *numChildren)
```

PURPOSE

cifGetChildIndexList retrieves arrays of category and item indices for the the children of the target item. The target item is identified by its keyword and the name of the category and data block in which it is defined.

RECEIVES

`dataBlockName`	name of the target data block
`categoryName`	name of the target category
`itemKeyword`	name of the target item within the target category
`childCategoryIndex`	address of the integer array to hold child category indices
`childItemIndex`	address of the integer array to hold child item indices
`numChildren`	address of the integer to hold the number of child items

RETURN VALUE

Returns a value of 1 if the operation was successful or a value of 0 for failure. If the operation is successful then the index arrays and number of children are also returned.

REMARKS

See also: *cifGetChildItemNames*

2.13 CIFLIB Schema Extension Functions

This section includes the set of functions that add new elements to a CIF schema. Functions are provided to add data blocks, to add categories to data blocks, and to add items to categories. These functions simply add a placeholder for the respective schema element. Once a new schema element has been created, other CIFLIB functions can be used to load values into the element.

2.13.1 cifAddDataBlock

NAME

cifAddDataBlock

PROTOTYPE

```
#include "ciflib.h"

int cifAddDataBlock(const char *dataBlockName,
                    const char *dicDataBlockName)
```

PURPOSE

cifAddDataBlock creates a new data block and assigns a validation dictionary to the new block. The data block and the dictionary data block are identified by name.

RECEIVES

`dataBlockName`	name of the new data block
`dicDataBlockName`	name of the data block holding the validation dictionary

RETURN VALUE

Returns the index of the new block or a value of -1 for failure.

REMARKS

See also:	*cifAddCategory*
	cifAddItem

2.13.2 cifAddCategory

NAME

cifAddCategory

PROTOTYPE

```
#include "ciflib.h"

int cifAddCategory(const char *categoryName,
                   const char *dataBlockName)
```

PURPOSE

cifAddCategory creates a new category in a data block. The data block and new category are both identified by name.

RECEIVES

`categoryName`	name of the new category
`dataBlockName`	name of the target data block

RETURN VALUE

Returns the index of the new category or a value of -1 for failure.

REMARKS

See also:	*cifAddDataBlock*
	cifAddItem

2.13.3 cifAddItem

NAME

cifAddItem

PROTOTYPE

```
#include "ciflib.h"

int cifAddItem(const char *itemKeyword,
               const char *categoryName,
               const char *dataBlockName)
```

PURPOSE

cifAddItem creates a new data item in a category. The target data block and category are both identified by name.

RECEIVES

`itemKeyword`	name of the new item within the target category
`categoryName`	name of the target category
`dataBlockName`	name of the target data block

RETURN VALUE

Returns the index of the new item in the target category or a value of -1 for failure.

REMARKS

See also:	*cifAddDataBlock* *cifAddCategory*

2.14 CIFLIB Error Handling and Print Functions

The following sections present the set of functions which provide access to the error codes generated by those CIFLIB functions which perform integrity checking.

The CIFLIB functions which access and update individual item values return only a single error code. Functions providing read access only return the first error encountered in checking the target item. Similarly, functions providing update access only return the first error encountered in the checking process; however, other problems that may be detected are appended to the warning or error lists maintained for each datablock. Higher level functions, like *cifReadFile*, also append their diagnostic codes to internal error and warning lists. A set of functions has been provided to access and refresh these lists. Functions are also provided to translate individual error codes and to print the contents of an entire data block.

2.14.1 cifErrorMessage

NAME

cifErrorMessage

PROTOTYPE

```
#include "ciflib.h"

char *cifErrorMessage(const int errorCode)
```

PURPOSE

cifErrorMessage returns the error message character string associated with a CIFLIB error code.

RECEIVES

`errorCode`	integer error code returned by CIFLIB function

RETURN VALUE

Returns a character string holding the error message or a `NULL` value for failure.

REMARKS

None

2.14.2 cifCountDataBlockErrors

NAME

cifCountDataBlockErrors

PROTOTYPE

```
#include "ciflib.h"

int cifCountDataBlockErrors(const char *dataBlockName)
```

PURPOSE

cifCountDataBlockErrors returns the number of errors in the list of errors for the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

Returns the number of errors or the value of -1 for failure.

REMARKS

See also: *cifCountDataBlockErrorsByIndex*
cifCountDataBlockWarnings
cifCountDataBlockWarningsByIndex

2.14.3 cifCountDataBlockErrorsByIndex

NAME

cifCountDataBlockErrorsByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifCountDataBlockErrorsByIndex(const int dataBlockIndex)
```

PURPOSE

cifCountDataBlockErrorsByIndex returns the number of errors in the list of errors for the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block

RETURN VALUE

Returns the number of errors or the value of -1 for failure.

REMARKS

See also:	*cifCountDataBlockErrorsBy*
	cifCountDataBlockWarnings
	cifCountDataBlockWarningsByIndex

2.14.4 cifCountDataBlockWarnings

NAME

cifCountDataBlockWarnings

PROTOTYPE

```
#include "ciflib.h"

int cifCountDataBlockWarnings(const char *dataBlockName)
```

PURPOSE

cifCountDataBlockWarnings returns the number of warnings in the list of warnings for the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

Returns the number of warnings or the value of -1 for failure.

REMARKS

See also: *cifCountDataBlockWarningsByIndex*
cifCountDataBlockErrors
cifCountDataBlockErrorsByIndex

2.14.5 cifCountDataBlockWarningsByIndex

NAME

cifCountDataBlockWarningsByIndex

PROTOTYPE

```
#include "ciflib.h"

int cifCountDataBlockWarningsByIndex(const int dataBlockIndex)
```

PURPOSE

cifCountDataBlockWarningsByIndex returns the number of warnings in the list of warnings for the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block

RETURN VALUE

Returns the number of warnings or the value of -1 for failure.

REMARKS

See also:	*cifCountDataBlockWarnings*
	cifCountDataBlockErrors
	cifCountDataBlockErrorsByIndex

2.14.6 cifGetDataBlockErrors

NAME

cifGetDataBlockErrors

PROTOTYPE

```
#include "ciflib.h"

char **cifGetDataBlockErrors(const char *dataBlockName,
                             int  *numErrors)
```

PURPOSE

cifGetDataBlockErrors returns the list of errors and the number of errors for the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`numErrors`	address of the integer to hold the number of errors in the target data block

RETURN VALUE

Returns an array of error messages or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockErrorsByIndex*
	cifGetDataBlockWarnings
	cifGetDataBlockWarningsByIndex

2.14.7 cifGetDataBlockErrorsByIndex

NAME

cifGetDataBlockErrorsByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetDataBlockErrorsByIndex(const int  dataBlockIndex,
                                    int *numErrors)
```

PURPOSE

cifGetDataBlockErrorsByIndex returns the list of errors and the number of errors for the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`numErrors`	address of the integer to hold the number of errors in the target data block

RETURN VALUE

Returns an array of error messages or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockErrors*
	cifGetDataBlockWarnings
	cifGetDataBlockWarningsByIndex

2.14.8 cifGetDataBlockWarnings

NAME

cifGetDataBlockWarnings

PROTOTYPE

```
#include "ciflib.h"

char **cifGetDataBlockWarnings(const char *dataBlockName,
                                int  *numWarnings)
```

PURPOSE

cifGetDataBlockWarnings returns the list of warnings and the number of warnings for the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`numWarnings`	address of the integer to hold the number of warnings in the target data block

RETURN VALUE

Returns an array of warning messages or a `NULL` value for failure.

REMARKS

See also: *cifGetDataBlockWarningsByIndex*
cifGetDataBlockErrors
cifGetDataBlockErrorsByIndex

2.14.9 cifGetDataBlockWarningsByIndex

NAME

cifGetDataBlockWarningsByIndex

PROTOTYPE

```
#include "ciflib.h"

char **cifGetDataBlockWarningsByIndex(const int  dataBlockIndex,
                                      int *numWarnings)
```

PURPOSE

cifGetDataBlockWarningsByIndex returns the list of warnings and the number of warnings for the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`numWarnings`	address of the integer to hold the number of warnings in the target data block

RETURN VALUE

Returns an array of warning messages or a `NULL` value for failure.

REMARKS

See also: *cifGetDataBlockWarnings*
cifGetDataBlockErrors
cifGetDataBlockErrorsByIndex

2.14.10 cifGetDataBlockError

NAME

cifGetDataBlockError

PROTOTYPE

```
#include "ciflib.h"

char *cifGetDataBlockError(const char *dataBlockName,
                           const  int  errorIndex)
```

PURPOSE

cifGetDataBlockError returns the error message for the error identified by its index in the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`errorIndex`	index of the target error in the target data block

RETURN VALUE

Returns an error message or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockErrorByIndex* *cifGetDataBlockWarning* *cifGetDataBlockWarningByIndex*

2.14.11 cifGetDataBlockErrorByIndex

NAME

cifGetDataBlockErrorByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetDataBlockErrorByIndex(const  int  dataBlockIndex,
                                  const  int  errorIndex)
```

PURPOSE

cifGetDataBlockErrorByIndex returns the error message for the error identified by its index in the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`errorIndex`	index of the target error in the target data block

RETURN VALUE

Returns an error message or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockError* *cifGetDataBlockWarning* *cifGetDataBlockWarningByIndex*

2.14.12 cifGetDataBlockWarning

NAME

cifGetDataBlockWarning

PROTOTYPE

```
#include "ciflib.h"

char *cifGetDataBlockWarning(const char *dataBlockName,
                             const  int  warningIndex)
```

PURPOSE

cifGetDataBlockWarning returns the warning message for the warning identified by its index in the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`warningIndex`	index of the target warning in the target data block

RETURN VALUE

Returns an warning message or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockWarningByIndex*
	cifGetDataBlockError
	cifGetDataBlockErrorByIndex

2.14.13 cifGetDataBlockWarningByIndex

NAME

cifGetDataBlockWarningByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifGetDataBlockWarningByIndex(const  int  dataBlockIndex,
                                    const  int  warningIndex)
```

PURPOSE

cifGetDataBlockWarningByIndex returns the warning message for the warning identified by its index in the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`warningIndex`	index of the target warning in the target data block

RETURN VALUE

Returns an warning message or a `NULL` value for failure.

REMARKS

See also:	*cifGetDataBlockWarning* *cifGetDataBlockError* *cifGetDataBlockErrorByIndex*

2.14.14 cifFreeDataBlockError

NAME

cifFreeDataBlockError

PROTOTYPE

```
#include "ciflib.h"

void  cifFreeDataBlockError(const char *dataBlockName,
                            const  int  errorIndex)
```

PURPOSE

cifFreeDataBlockError frees/removes the error message for the error identified by its index in the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`errorIndex`	index of the target error in the target data block

RETURN VALUE

None

REMARKS

See also:	*cifFreeDataBlockErrorByIndex*
	cifFreeDataBlockWarning
	cifFreeDataBlockWarningByIndex

2.14.15 cifFreeDataBlockErrorByIndex

NAME

cifFreeDataBlockErrorByIndex

PROTOTYPE

```
#include "ciflib.h"

void cifFreeDataBlockErrorByIndex(const  int  dataBlockIndex,
                                  const  int  errorIndex)
```

PURPOSE

cifFreeDataBlockErrorByIndex frees/removes the error message for the error identified by its index in the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`errorIndex`	index of the target error in the target data block

RETURN VALUE

None

REMARKS

See also: *cifFreeDataBlockError*
cifFreeDataBlockWarning
cifFreeDataBlockWarningByIndex

2.14.16 cifFreeDataBlockWarning

NAME

cifFreeDataBlockWarning

PROTOTYPE

```
#include "ciflib.h"

void cifFreeDataBlockWarning(const char *dataBlockName,
                             const  int  warningIndex)
```

PURPOSE

cifFreeDataBlockWarning frees/removes the warning message for the warning identified by its index in the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block
`warningIndex`	index of the target warning in the target data block

RETURN VALUE

None

REMARKS

See also:	*cifFreeDataBlockWarningByIndex*
	cifFreeDataBlockError
	cifFreeDataBlockErrorByIndex

2.14.17 cifFreeDataBlockWarningByIndex

NAME

cifFreeDataBlockWarningByIndex

PROTOTYPE

```
#include "ciflib.h"

char *cifFreeDataBlockWarningByIndex(const  int  dataBlockIndex,
                                     const  int  warningIndex)
```

PURPOSE

cifFreeDataBlockWarningByIndex frees/removes the warning message for the warning identified by its index in the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`warningIndex`	index of the target warning in the target data block

RETURN VALUE

None

REMARKS

See also: *cifFreeDataBlockWarning*
cifFreeDataBlockError
cifFreeDataBlockErrorByIndex

2.14.18 cifFreeDataBlockErrors

NAME

cifFreeDataBlockErrors

PROTOTYPE

```
#include "ciflib.h"

void cifFreeDataBlockErrors(const char *dataBlockName)
```

PURPOSE

cifFreeDataBlockErrors frees/removes the list of errors for the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

None

REMARKS

See also:	*cifFreeDataBlockErrorsByIndex* *cifFreeDataBlockWarnings* *cifFreeDataBlockWarningsByIndex*

2.14.19 cifFreeDataBlockErrorsByIndex

NAME

cifFreeDataBlockErrorsByIndex

PROTOTYPE

```
#include "ciflib.h"

void cifFreeDataBlockErrorsByIndex(const int  dataBlockIndex)
```

PURPOSE

cifFreeDataBlockErrorsByIndex frees/removes the list of errors for the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block

RETURN VALUE

None

REMARKS

See also: *cifFreeDataBlockErrors*
cifFreeDataBlockWarnings
cifFreeDataBlockWarningsByIndex

2.14.20 cifFreeDataBlockWarnings

NAME

cifFreeDataBlockWarnings

PROTOTYPE

```
#include "ciflib.h"

void cifFreeDataBlockWarnings(const char *dataBlockName)
```

PURPOSE

cifFreeDataBlockWarnings frees/removes the list of warnings for the target data block. The data block is identified by name.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

None

REMARKS

See also: *cifFreeDataBlockWarningsByIndex*
cifFreeDataBlockErrors
cifFreeDataBlockErrorsByIndex

2.14.21 cifFreeDataBlockWarningsByIndex

NAME

cifFreeDataBlockWarningsByIndex

PROTOTYPE

```
#include "ciflib.h"

void cifFreeDataBlockWarningsByIndex(const int  dataBlockIndex)
```

PURPOSE

cifFreeDataBlockWarningsByIndex frees/removes the list of warnings for the target data block. The data block is identified by index.

RECEIVES

`dataBlockIndex`	index of the target data block
`numWarnings`	address of the integer to hold the number of warnings in the target data block

RETURN VALUE

None

REMARKS

See also:	*cifFreeDataBlockWarnings*
	cifFreeDataBlockErrors
	cifFreeDataBlockErrorsByIndex

2.14.22 cifPrintDataBlock

NAME

cifPrintDataBlock

PROTOTYPE

```
#include "ciflib.h"

void cifPrintDataBlock(const char *dataBlockName)
```

PURPOSE

cifPrintDataBlock prints the contents of a data block to the output stream (stdout). This function is provided primarily for debugging purposes.

RECEIVES

`dataBlockName`	name of the target data block

RETURN VALUE

None

REMARKS

None

2.14.23 cifPrintDataBlockByIndex

NAME

cifPrintDataBlockByIndex

PROTOTYPE

```
#include "ciflib.h"

void cifPrintDataBlockByIndex(const int dataBlockIndex)
```

PURPOSE

cifPrintDataBlockByIndex prints the contents of a data block to the output stream (stdout). This function is provided primarily for debugging purposes.

RECEIVES

`dataBlockIndex`	index of the target data block

RETURN VALUE

None

REMARKS

None

2.15 Missing Functionality and Documentation

There are a number of issues that have not been included in this version of the interface description. These known ommissions, which will be incorporated in future releases of the documentation, include:

- a detailed description of the parsing rules used by CIFLIB
- a complete description of the CIFLIB error codes and the CIFLIB integrity checking process.
- a systematic treatment of the propagation of updates through a CIF schema.
- merge functions
- query functions

References

[1] H. M. Berman and J. D. Westbrook. Now the Nucleic Acid Database Uses CIF. In P. E. Bourne, editor, *Proceedings fo the First Macromolecular CIF Tools Workshop*, page 65, Tarrytown, New York, 1993. National Science Foundation.

[2] H. M. Berman, W. K. Olson, D. L. Beveridge, J. D. Westbrook, A. Gelben, T. Demeny, S. Hsieh, A. R. Srinivasan, and B. Schneider. Nucleic Acid Database: A comprehensive Relational Database of Three-Dimensional Structures of Nucleic Acids. *Biophys. J.*, 63:751, 1992.

[3] S. R. Hall. The STAR File: A new format for electronic data transfer and archiving. *J. Chem. Inf. Comput. Sci.*, 31, 1991.

[4] S. R. Hall, F. H. Allen, and I. D. Brown. A new standard archive file for crystallography. *Acta Crystallogr.*, A47, 1991.

[5] P. M. D. Fitzgerald, H. M. Berman, P. E. Bourne, and K. Watenpaugh. *The Macromolecular CIF dictionary.* ACA Annual Meeting, Albuquerque, New Mexico, 1993.

[6] P. Fitzgerald, H. M. Berman, P. Bourne, B. McMahon, K. Watenpaugh, and J. D. Westbrook. *The Macromolecular Crystallographic Information File Dictionary.* IUCR, http://ndbserver.rutgers.edu/mmcif, 1995.

[7] H. M. Berman and J. D. Westbrook. A Gentle Introduction to one Working Alternative DDL for Macromolecular Structure. In S. D. Wodak, editor, *European Macromolecular Crystallographic Information (mmCIF) Workshop*, Free University of Brussels, 1994. European Commission.

[8] J. D. Westbrook. DDL 2.0 and a Library of Supporting Tools. Montreal, 1995. ACA, American Crystallographic Association Annual Meeting.

[9] J. D. Westbrook. *A Dictionary Description Language for Structure Macromolecular.* Rutgers University, 1994. (http://ndbserver.rutgers.edu/ddl).

[10] J. D. Westbrook and S. R. Hall. A dictionary description language for macromolecular structure. *J. Chem. Inf. Comput. Sci.*, 1995. to be submitted.

[11] John D. Westbrook, Shu-Hsin Hsieh, and P. M. D. Fitzgerald. CIFLIB: An Application Program Interface to CIF Dictionaries and Data Files. *J. Appl. Cryst.*, 1996. in press.