# CIFPARSE
# A Library of
# Access Tools for mmCIF
# Reference Guide

**CIFPARSE Version 3.01**

**Based on Dictionary Description Language v. 2.1**
**November 1998**

**Olivera Tosic**
**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.

**CIFPARSE - A Library of Access Tools for mmCIF**

# Contents

# 1 Introduction

CIFPARSE is a function library developed to provide simple access tools for mmCIF [1, 2] data files. As shown in Figure 1, CIFPARSE functions do not perform any semantic integrity checks using the DDL [3, 4] or mmCIF [5] dictionaries. This library simply provides functions to read and write mmCIF data files that conform to the subset of STAR [6] syntax rules adopted by mmCIF.

There a variety of circumstances in which it is unnecessary or even to perform data intregity checks on mmCIF data files. For instance, in the situation where two computer program exchange mmCIF data or where a single program must repeatedly access a large database of mmCIF data that is know to be semantically valid. The CIFPARSE library was designed to address the need for simple and efficient tools for exchanging mmCIF data. The CIFPARSE library performs the following functions:

- reads and writes mmCIF format data files.
- provides read and write access for individual data items.
- provides functions for navigating the mmCIF category structure.
- provides a stable callable interface in the C programming language.

This document describes the functional interface to CIFPARSE.

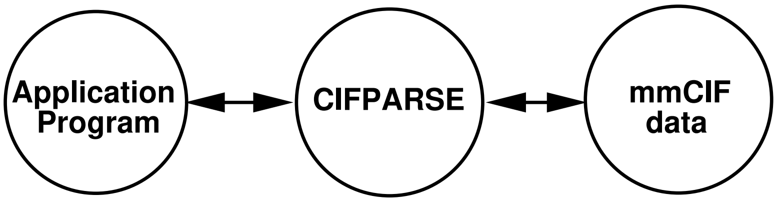


Figure 1: Functional diagram of the CIFPARSE utility library.

# 2 Function Description

### 2.0.1 ndb_cif_read_file

**NAME**

*ndb_cif_read_file*

**PROTOTYPE**

```
#include "ndb_cifparse.h"

int ndb_cif_read_file(CifHandle *cf,
                      FILE *fp);
```

**PURPOSE**

*ndb_cif_read_file* reads an mmCIF data file into the CifHandle data structure specified by *cf*. All interaction with the data file takes place inside this function. Subsequent access to mmCIF data is managed by a set of methods that access memory resident data.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure where the mmCIF data file is loaded |
| `fp` | pointer to the file descriptor for the mmCIF input file |

**RETURN VALUE**

Returns the number of data blocks in the input file or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_write_file*

### 2.0.2 ndb_cif_write_file

**NAME**

*ndb_cif_write_file*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_write_file(CifHandle *cf,
                       FILE *fp);
```

**PURPOSE**

*ndb_cif_write_file* writes all of the data blocks loaded into the CifHande data structure specified by *cf* to the file pointed to by the input file descriptor.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure where the data blocks loaded |
| `fp` | pointer to the file descriptor for the mm-CIF output file |

**RETURN VALUE**

Returns the number of data blocks written or 0 for failure.

**REMARKS**

See also: *ndb_cif_read_file*

### 2.0.3 ndb_cif_read_ddl

**NAME**

*ndb_cif_read_ddl*

**PROTOTYPE**

```
#include "ndb_cifparse.h"

int ndb_cif_read_ddl(CifHandle *ddl,
                     FILE *fp);
```

**PURPOSE**

*ndb_cif_read_ddl* reads a ddl file into the CifHandle data structure specified by *ddl*. All interaction with the data file takes place inside this function. Subsequent access to ddl data is managed by a set of methods that access memory resident data.

**RECEIVES**

| | |
|---|---|
| `ddl` | pointer to the CifHandle data structure where the ddl file is loaded |
| `fp` | pointer to the file descriptor for the ddl input file |

**RETURN VALUE**

Returns the number of data blocks in the input file or 0 for failure.

**REMARKS**

Function does provide new value to the ddl.

See also: *ndb_cif_read_ddl_with_format*
*ndb_cif_write_ddl_by_format*

### 2.0.4 ndb_cif_read_ddl_with_format

**NAME**

*ndb_cif_read_ddl_with_format*

**PROTOTYPE**

```
#include "ndb_cifparse.h"

int ndb_cif_read_ddl_with_format(CifHandle *ddl,
                                  FILE *fp,
                                  CifHandle *format);
```

**PURPOSE**

*ndb_cif_read_ddl_with_format* reads a ddl file into the CifHandle data structure specified by *ddl*. Also saves the format of save frames into CifHandle data structure specified by *format* . All interaction with the data file takes place inside this function. Subsequent access to ddl data is managed by a set of methods that access memory resident data.

**RECEIVES**

| | |
|---|---|
| `ddl` | pointer to the CifHandle data structure where the ddl file is loaded |
| `fp` | pointer to the file descriptor for the ddl input file |
| `format` | pointer to the CifHandle structure where the format is loaded |

**RETURN VALUE**

Returns the number of data blocks in the input file or 0 for failure.

**REMARKS**

Function does provide new value to the ddl and format.

See also: *ndb_cif_read_ddl*
*ndb_cif_write_ddl_by_format*

### 2.0.5 ndb_cif_write_ddl_by_format

**NAME**

*ndb_cif_write_ddl_by_format*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_write_ddl_by_format(CifHandle *ddl,
                                FILE *fp
                                CifHandle *format);
```

**PURPOSE**

*ndb_cif_write_ddl_by_format* writes all of the data blocks loaded into the CifHande data structure specified by *ddl* to the file pointed to by the input file descriptor. Format of save frames is defined by input parameter *format*.

**RECEIVES**

| | |
|---|---|
| `ddl` | pointer to the CifHandle data structure where the data is loaded |
| `fp` | pointer to the file descriptor for the mmCIF output file |
| `format` | pointer to the CifHandle data structure where the format is loaded |

**RETURN VALUE**

Returns the number of data blocks written or 0 for failure.

**REMARKS**

See also: *ndb_cif_read_ddl*
*ndb_cif_read_ddl_with_format*

### 2.0.6 ndb_cif_read_dic

**NAME**

*ndb_cif_read_dic*

**PROTOTYPE**

```
#include "ndb_cifparse.h"

int ndb_cif_read_dic(CifHandle *cf,
                     FILE *fp,
                     CifHandle *ddl,
                     FILE *fp_ddl);
```

**PURPOSE**

*ndb_cif_read_dic* reads a dictionary/cif file into the CifHandle data structure specified by *cf* from a file pointed to by file descriptor *fp*. Function first reads ddl/dictionary into the CifHandle data structure specified by *ddl*, reads dictionary/cif file from a file pointed to by file descriptor *fp_ddl* and checks correctness of dictionary/cif file. All interaction with the data file takes place inside this function. Subsequent access to ddl data is managed by a set of methods that access memory resident data.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure where the dictionary/cif file is loaded |
| `fp` | pointer to the file descriptor for the dictionary/cif input file |
| `ddl` | pointer to the CifHandle data structure where the ddl file is loaded |
| `fp_ddl` | pointer to the file descriptor for the ddl input file |

**RETURN VALUE**

Returns the number of data blocks in the input file or 0 for failure.

**REMARKS**

Function does provide new value to the cf and ddl.

See also: *ndb_cif_read_ddl_with_format*
*ndb_cif_write_ddl_by_format*

### 2.0.7 ndb_cif_read_dic_with_format

**NAME**

*ndb_cif_read_dic_with_format*

**PROTOTYPE**

```
#include "ndb_cifparse.h"

int ndb_cif_read_dic_with_format(CifHandle *cf,
                                 FILE *fp,
                                 CifHandle *ddl,
                                 FILE *fp_ddl,
                                 CifHandle *format);
```

**PURPOSE**

*ndb_cif_read_dic_with_format* reads a dictionary/cif file into the CifHandle data structure specified by *ddl* from a file pointed to by file descriptor *fp*. Checks corectness of dictionary/cif file like the *ndb_cif_read_dic*. Additionaly, saves the format of save frames of dictionary/cif file into the CifHandle dat structure *format*. All interaction with the data file takes place inside this function. Subsequent access to ddl data is managed by a set of methods that access memory resident data.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure where the data is loaded |
| `fp` | pointer to the file descriptor for the dictionary/cif input file |
| `ddl` | pointer to the CifHandle data structure where the ddl file is loaded |
| `fp_ddl` | pointer to the file descriptor for the ddl input file |
| `format` | pointer to the CifHandle structur where the format is loaded |

**RETURN VALUE**

Returns the number of data blocks in the input file or 0 for failure.

**REMARKS**

Function does provide new value to the cf, ddl and format.

| | |
|---|---|
| See also: | *ndb_cif_read_dic* |
| | *ndb_cif_write_dic_by_format* |

### 2.0.8 ndb_cif_write_dic_by_format

**NAME**

*ndb_cif_write_dic_by_format*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_write_dic_by_format(CifHandle *dic,
                                FILE *fp
                                CifHandle *ddl,
                                CifHandle *format,
                                int implicit);
```

**PURPOSE**

*ndb_cif_write_dic_by_format* writes all of the data blocks loaded into the CifHande data structure specified by *dic* to the file pointed to by the input file descriptor. The structure of the save frames in the output file depends on the *format*. Also, if the parameter implicit = 1, then all columns are shown. If implicit = 0, then the implicit columns are avoided. Information about imlicitly generated column can be found in the CifHandle data structure specified by *ddl*.

**RECEIVES**

| | |
|---|---|
| `dic` | pointer to the CifHandle data structure where the dictionary/cif file is loaded |
| `ddl` | pointer to the CifHandle data structure where the ddl/dictionary is loaded |
| `fp` | pointer to the file descriptor for the output file |
| `format` | pointer to the CifHandle data structure where the format is loaded |
| `implicit` | identifies does imlicit columns will be print or not |

## RETURN VALUE

Returns the number of data blocks written or 0 for failure.

## REMARKS

See also: *ndb_cif_read_dic*
*ndb_cif_read_dic_with_format*

### 2.0.9 ndb_cif_init

**NAME**

*ndb_cif_init*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_init(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_init* initializes CifHandle data structure specified by *cf*. This function must be called before any other CIFPARSE functions.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the initialized CifHandle data structure |

**RETURN VALUE**

Returns 1 for success or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_close*

### 2.0.10 ndb_cif_close

**NAME**

*ndb_cif_close*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_close(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_close* frees up all memory in the CifHanle data structure specified by *cf*. This function should be called at the end of the CIFPARSE application.

**RECEIVES**

| | |
|---|---|
| `cp` | pointer to the CifHandle data structure to be freed |

**RETURN VALUE**

Returns 1 for success or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_init*

### 2.0.11 ndb_cif_new_datablock

**NAME**

*ndb_cif_new_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_new_datablock(CifHandle *cf,
                          const char * datablockName);
```

**PURPOSE**

*ndb_cif_new_datablock* allocates space in the CifHande data structure, specified by *cf*, to store a new datablock with the name *datablockName*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | the name of the datablock to be added to the data structure |

**RETURN VALUE**

Returns the current datablock number.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_new_category*
*ndb_cif_new_row*

### 2.0.12 ndb_cif_put_datablock_name

**NAME**

*ndb_cif_put_datablock_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_put_datablock_name(CifHandle *cf,
                               const char * datablockName);
```

**PURPOSE**

*ndb_cif_put_datablock_name*, in the CifHandle data structure, specified by *cf*, changes the name of the current datablock to *datablockName*, or, if there are no datablocks, creates a new one with the name *datablockName*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | the new name of the current datablock |

**RETURN VALUE**

Returns a meaningless integer.

**REMARKS**

Function does provide new value to the cf.

### 2.0.13 ndb_cif_new_category

**NAME**

*ndb_cif_new_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_new_category(CifHandle *cf,
                         const char * categoryName);
```

**PURPOSE**

*ndb_cif_new_datablock* allocates space in the CifHandle data structure, specified by *cf*, to store a new category in the current datablock with the name *categoryName*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `categoryName` | the name of the category to be added to the current datablock |

**RETURN VALUE**

Returns the current category number.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_new_datablock* |
| | *ndb_cif_new_row* |

### 2.0.14 ndb_cif_new_row

**NAME**

*ndb_cif_new_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_new_row(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_new_row* allocates space in the CifHandle data structure, specified by *cf* to store a new row in the current category of the current datablock.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_new_datablock* |
| | *ndb_cif_new_category* |

### 2.0.15 ndb_cif_insert_new_row

**NAME**

*ndb_cif_insert_new_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_insert_new_row(CifHandle *cf,
                           const int rowNo);
```

**PURPOSE**

*ndb_cif_insert_new_row* allocates space in the CifHandle data structure, specified by *cf*, to store a new row in the current category of the current datablock and places it as row number *rowNo*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `rowNo` | where the new row will be placed |

**RETURN VALUE**

Returns the current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

### 2.0.16 ndb_cif_rewind_file

**NAME**

*ndb_cif_rewind_file*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_rewind_file(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_rewind_file* rewinds all datablocks, in the CifHandle data structure, specified by *cf* and sets current datablock to the first one.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current datablock or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_rewind_datablock*
*ndb_cif_rewind_category*
*ndb_cif_rewind_row*
*ndb_cif_rewind_column*

### 2.0.17 ndb_cif_rewind_datablock

**NAME**

*ndb_cif_rewind_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_rewind_datablock(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_rewind_datablock* rewinds all categories in the current datablock, in the CifHandle data structure, specified by *cf* and sets the current category to the first one.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current datablock or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_rewind_file*
*ndb_cif_rewind_category*
*ndb_cif_rewind_row*
*ndb_cif_rewind_column*

### 2.0.18 ndb_cif_rewind_category

**NAME**

*ndb_cif_rewind_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_rewind_category(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_rewind_category* rewinds all rows and all columns in the current category in the CifHandle data structure, specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current category or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_rewind_file* |
| | *ndb_cif_rewind_datablock* |
| | *ndb_cif_rewind_row* |
| | *ndb_cif_rewind_column* |

### 2.0.19 ndb_cif_rewind_row

**NAME**

*ndb_cif_rewind_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_rewind_row(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_rewind_row* in the CifHandle data structure, specified by *cf*, sets the current row for the current category to the first one.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_rewind_file*
*ndb_cif_rewind_datablock*
*ndb_cif_rewind_category*
*ndb_cif_rewind_column*

### 2.0.20 ndb_cif_rewind_column

#### NAME

*ndb_cif_rewind_column*

#### PROTOTYPE

```
#include "cifparse.h"

int ndb_cif_rewind_column(CifHandle *cf);
```

#### PURPOSE

*ndb_cif_rewind_column* sets the current column in the current category in the CifHandle data structure, specified by *cf*, to the first one.

#### RECEIVES

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

#### RETURN VALUE

Returns the current column or 0 for failure.

#### REMARKS

Function does provide new value to the cf.

See also: *ndb_cif_rewind_file*
*ndb_cif_rewind_datablock*
*ndb_cif_rewind_category*
*ndb_cif_rewind_row*

### 2.0.21 ndb_cif_reset_datablocks

**NAME**

*ndb_cif_reset_datablocks*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_reset_datablocks(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_reset_datablocks* deletes the data in CifHandle data structure, specified by *cf*, for all of the datablocks, but does not free up the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of datablocks reset or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_reset_datablock*
*ndb_cif_reset_datablock_by_id*
*ndb_cif_reset_category*
*ndb_cif_reset_category_by_id*

### 2.0.22 ndb_cif_reset_datablock

**NAME**

*ndb_cif_reset_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_reset_datablock(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_reset_datablock* deletes the data in the CifHandle data structures, specified by *cf*, for the current datablock, but does not free up the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to tne CifHandle data structure |

**RETURN VALUE**

Returns the number of categories in the datablock that were reset or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_reset_datablocks*
*ndb_cif_reset_datablock_by_id*
*ndb_cif_reset_category*
*ndb_cif_reset_category_by_id*

### 2.0.23 ndb_cif_reset_datablock_by_id

**NAME**

*ndb_cif_reset_datablock_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_reset_datablock_by_id(CifHandle *cf,
                                  const int datablockId);
```

**PURPOSE**

*ndb_cif_reset_datablock_by_id* deletes the data in the CifHandle data stucture, specified by *cf*, for the datablock specified by *datablockId*, but does not free up the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock to reset |

**RETURN VALUE**

Returns the number of categories in the datablock that were reset or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_reset_datablocks* |
| | *ndb_cif_reset_datablock* |
| | *ndb_cif_reset_category* |
| | *ndb_cif_reset_category_by_id* |

### 2.0.24 ndb_cif_reset_category

**NAME**

*ndb_cif_reset_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_reset_category(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_reset_category* deletes the data in the CifHandle data structure, specified by *cf*, for the current category of the current datablock, but does not free up the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns 1 for success and 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_reset_datablocks* |
| | *ndb_cif_reset_datablock* |
| | *ndb_cif_reset_datablock_by_id* |
| | *ndb_cif_reset_category_by_id* |

### 2.0.25 ndb_cif_reset_category_by_id

**NAME**

*ndb_cif_reset_category_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_reset_category_by_id(CifHandle *cf,
                                 const int datablockId,
                                 const int categoryId);
```

**PURPOSE**

*ndb_cif_reset_category_by_id* deletes the data in the category specified by *categoryId* in the datablock specified by *datablockId* in the CifHandle data structure specified by *cf*, but does not free up the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock |
| `categoryId` | identifies the category to reset |

**RETURN VALUE**

Returns 1 for success and 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_reset_datablocks* |
| | *ndb_cif_reset_datablock* |
| | *ndb_cif_reset_datablock_by_id* |
| | *ndb_cif_reset_category* |

### 2.0.26 ndb_cif_remove_datablock

**NAME**

*ndb_cif_remove_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_datablock(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_remove_datablock* deletes the data in the CifHandle data structure, specified by *cf*, for the current datablock and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of the current datablock.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_remove_datablock_by_id*
*ndb_cif_remove_datablock_by_name*
*ndb_cif_remove_category*
*ndb_cif_remove_category_by_id*
*ndb_cif_remove_category_by_name*
*ndb_cif_remove_row*
*ndb_cif_remove_row_by_id*
*ndb_cif_remove_item_keyword*

### 2.0.27 ndb_cif_remove_datablock_by_id

**NAME**

*ndb_cif_remove_datablock_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_datablock_by_id(CifHandle *cf,
                                    const int datablockId);
```

**PURPOSE**

*ndb_cif_remove_datablock_by_id* deletes the data in the datablock specified by *datablockId* in the CifHandle data structure specified by *cf* and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock to remove |

**RETURN VALUE**

Returns the number of the current datablock.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_remove_datablock*
*ndb_cif_remove_datablock_by_name*
*ndb_cif_remove_category*
*ndb_cif_remove_category_by_id*
*ndb_cif_remove_category_by_name*
*ndb_cif_remove_row*
*ndb_cif_remove_row_by_id*
*ndb_cif_remove_item_keyword*

### 2.0.28 ndb_cif_remove_datablock_by_name

**NAME**

*ndb_cif_remove_datablock_by_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_datablock_by_name(CifHandle *cf,
                                     const char * datablockName);
```

**PURPOSE**

*ndb_cif_remove_datablock_by_name* deletes the data in the CifHandle data structures specified by *cf* for the datablock specified by *datablockName* and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock to remove |

**RETURN VALUE**

Returns the number of the current datablock.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_remove_datablock* |
| | *ndb_cif_remove_datablock_by_id* |
| | *ndb_cif_remove_category* |
| | *ndb_cif_remove_category_by_id* |
| | *ndb_cif_remove_category_by_name* |
| | *ndb_cif_remove_row* |
| | *ndb_cif_remove_row_by_id* |
| | *ndb_cif_remove_item_keyword* |

### 2.0.29 ndb_cif_remove_category

**NAME**

*ndb_cif_remove_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_category(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_remove_category* deletes the data in the CifHandle data structure, specified by *cf*, for current category of the current datablock and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of the current category.

**REMARKS**

Function does provide new value to the cf.

See also:
*ndb_cif_remove_datablock*
*ndb_cif_remove_datablock_by_id*
*ndb_cif_remove_datablock_by_name*
*ndb_cif_remove_category_by_id*
*ndb_cif_remove_category_by_name*
*ndb_cif_remove_row*
*ndb_cif_remove_row_by_id*
*ndb_cif_remove_item_keyword*

### 2.0.30 ndb_cif_remove_category_by_id

**NAME**

*ndb_cif_remove_category_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_category_by_id(CifHandle *cf,
                                   const int datablockId,
                                   const int categoryId);
```

**PURPOSE**

*ndb_cif_remove_category_by_id* deletes the data in the CifHandle data structure, specified by *cf*, for the category specified by *categoryId* in the datablock specified by *datablockId* and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock |
| `categoryId` | identifies the category to remove |

**RETURN VALUE**

Returns the number of the current category.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_remove_datablock*
*ndb_cif_remove_datablock_by_id*
*ndb_cif_remove_datablock_by_name*
*ndb_cif_remove_category*
*ndb_cif_remove_category_by_name*
*ndb_cif_remove_row*
*ndb_cif_remove_row_by_id*
*ndb_cif_remove_item_keyword*

### 2.0.31 ndb_cif_remove_category_by_name

**NAME**

*ndb_cif_remove_category_by_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_category_by_name(CifHandle *cf,
                                    const char * datablockName,
                                    const char *categoryName);
```

**PURPOSE**

*ndb_cif_remove_category_by_name* deletes the data in the CifHandle data structure, specified by *cf*, for category specified by *categoryName* in the datablock specified by *datablockName* and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock to remove |

**RETURN VALUE**

Returns the number of the current category.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_remove_datablock* |
| | *ndb_cif_remove_datablock_by_id* |
| | *ndb_cif_remove_datablock_by_name* |
| | *ndb_cif_remove_category* |
| | *ndb_cif_remove_category_by_id* |
| | *ndb_cif_remove_row* |
| | *ndb_cif_remove_row_by_id* |
| | *ndb_cif_remove_item_keyword* |

### 2.0.32 ndb_cif_remove_row

**NAME**

*ndb_cif_remove_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_row(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_remove_row* deletes the data in the CifHandle data structure, specified by *cf*, for the current row in the current category and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of the current row.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_remove_datablock*
*ndb_cif_remove_datablock_by_id*
*ndb_cif_remove_datablock_by_name*
*ndb_cif_remove_category*
*ndb_cif_remove_category_by_id*
*ndb_cif_remove_category_by_name*
*ndb_cif_remove_row_by_id*
*ndb_cif_remove_item_keyword*

### 2.0.33 ndb_cif_remove_row_by_id

**NAME**

*ndb_cif_remove_row_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_row_by_id(CifHandle *cf,
                             const int datablockId,
                             const int categoryId,
                             const int rowId);
```

**PURPOSE**

*ndb_cif_remove_row_by_id* deletes the data in the row specified by *rowId* in the category specified by *categoryId* in the datablock specified by *datablockId* in the CifHandle dat structures specified by *cf* and frees the memory.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock |
| `categoryId` | identifies the category |
| `rowId` | identifies the row to remove |

**RETURN VALUE**

Returns the number of the current row.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_remove_datablock*
*ndb_cif_remove_datablock_by_id*
*ndb_cif_remove_datablock_by_name*
*ndb_cif_remove_category*
*ndb_cif_remove_category_by_id*
*ndb_cif_remove_category_by_name*
*ndb_cif_remove_row*
*ndb_cif_remove_item_keyword*

### 2.0.34 ndb_cif_next_datablock

**NAME**

*ndb_cif_next_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_next_datablock(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_next_datablock* changes the current datablock to the next one in the CifHandle dat structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of the new current datablock or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_next_category*
*ndb_cif_next_row*
*ndb_cif_next_column*

### 2.0.35 ndb_cif_next_category

#### NAME

*ndb_cif_next_category*

#### PROTOTYPE

```
#include "cifparse.h"

int ndb_cif_next_category(CifHandle *cf);
```

#### PURPOSE

*ndb_cif_next_category* changes the current category, in the CifHandle data stucture specified by *cf*, to the next one.

#### RECEIVES

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

#### RETURN VALUE

Returns the number of the new current category or 0 for failure.

#### REMARKS

Function does provide new value to the cf.

See also: *ndb_cif_next_datablock*
*ndb_cif_next_row*
*ndb_cif_next_column*

### 2.0.36 ndb_cif_next_row

**NAME**

*ndb_cif_next_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_next_row(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_next_row* changes the current row, in the CifHandle data structure specified by *cf*, to the next one.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of the new current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_next_datablock*
*ndb_cif_next_category*
*ndb_cif_next_column*

### 2.0.37 ndb_cif_next_column

**NAME**

*ndb_cif_next_column*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_next_column(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_next_column* changes the current colum , in the CifHandle data structure specified by *cf*, to the next one.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of the new current column or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_next_datablock* |
| | *ndb_cif_next_category* |
| | *ndb_cif_next_row* |

### 2.0.38 ndb_cif_move_datablock

**NAME**

*ndb_cif_move_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_datablock(CifHandle *cf,
                           const char * datablockName);
```

**PURPOSE**

*ndb_cif_move_datablock* makes the current datablock refer to the datablock specified by *datablockName* in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock to become the current one |

**RETURN VALUE**

Returns the number of the current datablock or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_move_datablock_by_id*
*ndb_cif_move_category*
*ndb_cif_move_category_by_id*
*ndb_cif_move_category_by_name*
*ndb_cif_move_row*
*ndb_cif_move_row_by_id*
*ndb_cif_move_row_by_name*

### 2.0.39 ndb_cif_move_datablock_by_id

**NAME**

*ndb_cif_move_datablock_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_datablock_by_id(CifHandle *cf,
                                 const int datablockId);
```

**PURPOSE**

*ndb_cif_move_datablock_by_id* makes the current datablock refer to the datablock specified by *datablockId* in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock to become the current datablock |

**RETURN VALUE**

Returns the number of the current datablock or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_move_datablock* |
| | *ndb_cif_move_category* |
| | *ndb_cif_move_category_by_id* |
| | *ndb_cif_move_category_by_name* |
| | *ndb_cif_move_row* |
| | *ndb_cif_move_row_by_id* |
| | *ndb_cif_move_row_by_name* |

### 2.0.40 ndb_cif_move_category

**NAME**

*ndb_cif_move_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_category(CifHandle *cf,
                          const int categoryId);
```

**PURPOSE**

*ndb_cif_move_category* makes the current category refer to the category specified by *categoryId* in the current datablock in the CifHandle data structures specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `categoryId` | identifies the category to become the current category |

**RETURN VALUE**

Returns the number of the current category or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_move_datablock* |
| | *ndb_cif_move_datablock_by_id* |
| | *ndb_cif_move_category_by_id* |
| | *ndb_cif_move_category_by_name* |
| | *ndb_cif_move_row* |
| | *ndb_cif_move_row_by_id* |
| | *ndb_cif_move_row_by_name* |

### 2.0.41 ndb_cif_move_category_by_id

#### NAME

*ndb_cif_move_category_by_id*

#### PROTOTYPE

```
#include "cifparse.h"

int ndb_cif_move_category_by_id(CifHandle *cf,
                                const int datablockId,
                                const int categoryId);
```

#### PURPOSE

*ndb_cif_move_category_by_id* makes the current category refer to the category specified by *categoryId* in the datablock specified by *datablockId* in the CifHandle data structure specified by *cf*.

#### RECEIVES

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock |
| `categoryId` | identifies the category to become the current category |

#### RETURN VALUE

Returns the number of the current category or 0 for failure.

#### REMARKS

Function does provide new value to the cf.

See also: *ndb_cif_move_datablock*
*ndb_cif_move_datablock_by_id*
*ndb_cif_move_category*
*ndb_cif_move_category_by_name*
*ndb_cif_move_row*
*ndb_cif_move_row_by_id*
*ndb_cif_move_row_by_name*

### 2.0.42 ndb_cif_move_category_by_name

**NAME**

*ndb_cif_move_category_by_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_category_by_name(CifHandle *cf,
                                   const char * datablockName,
                                   const char * categoryName);
```

**PURPOSE**

*ndb_cif_move_category_by_name* makes the current category refer to the category specified by *categoryName* in the datablock specified by *datablockName* in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock |
| `categoryName` | identifies the category to become the current category |

**RETURN VALUE**

Returns the number of the current category or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_move_datablock* |
| | *ndb_cif_move_datablock_by_id* |
| | *ndb_cif_move_category* |
| | *ndb_cif_move_category_by_name* |
| | *ndb_cif_move_row* |
| | *ndb_cif_move_row_by_id* |
| | *ndb_cif_move_row_by_name* |

### 2.0.43 ndb_cif_move_row

**NAME**

*ndb_cif_move_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_row(CifHandle *cf,
                     const int rowId);
```

**PURPOSE**

*ndb_cif_move_row* makes the current row refer to the row specified by *rowId* in the current category and datablock, in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `rowId` | identifies the row to become the current row |

**RETURN VALUE**

Returns the number of the current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_move_datablock* |
| | *ndb_cif_move_datablock_by_id* |
| | *ndb_cif_move_category* |
| | *ndb_cif_move_category_by_id* |
| | *ndb_cif_move_category_by_name* |
| | *ndb_cif_move_row_by_id* |
| | *ndb_cif_move_row_by_name* |

### 2.0.44 ndb_cif_move_row_by_id

**NAME**

*ndb_cif_move_row_by_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_row_by_id(CifHandle *cf,
                           const int datablockId,
                           const int categoryId,
                           const int rowId);
```

**PURPOSE**

*ndb_cif_move_row_by_id* makes the current row refer to the row specified by *rowId* in the category specified by *categoryId*, in the datablock specified by *datablockId*, in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockId` | identifies the datablock |
| `categoryId` | identifies the category |
| `rowId` | identifies the row to become the current row |

**RETURN VALUE**

Returns the number of the current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_move_datablock*
*ndb_cif_move_datablock_by_id*
*ndb_cif_move_category*
*ndb_cif_move_category_by_id*
*ndb_cif_move_category_by_name*
*ndb_cif_move_row*
*ndb_cif_move_row_by_name*

### 2.0.45 ndb_cif_move_row_by_name

**NAME**

*ndb_cif_move_row_by_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_move_row_by_name(CifHandle *cf,
                             const char * datablockName,
                             const char * categoryName,
                             const int rowId);
```

**PURPOSE**

*ndb_cif_move_row_by_name* makes the current row refer to the row specified by *rowId* in the category specified by *categoryName*, in the datablock specified by *datablockName*, in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock |
| `categoryName` | identifies the category |
| `rowId` | identifies the row to become the current row |

**RETURN VALUE**

Returns the number of the current row or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_move_datablock*
*ndb_cif_move_datablock_by_id*
*ndb_cif_move_category*
*ndb_cif_move_category_by_id*
*ndb_cif_move_category_by_name*
*ndb_cif_move_row*
*ndb_cif_move_row_by_id*

### 2.0.46 ndb_cif_count_datablock

**NAME**

*ndb_cif_count_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_count_datablock(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_count_datablock* counts the number of datablocks represented in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of datablocks.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_count_category* |
| | *ndb_cif_count_row* |
| | *ndb_cif_count_column* |

### 2.0.47 ndb_cif_count_category

**NAME**

*ndb_cif_count_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_count_category(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_count_category* counts the number of categories in the current datablock, in the CifHande data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of categories or 0 for failure.

**REMARKS**

See also: *ndb_cif_count_datablock*
*ndb_cif_count_row*
*ndb_cif_count_column*

### 2.0.48 ndb_cif_count_row

**NAME**

*ndb_cif_count_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_count_row(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_count_row* counts the number of rows in the current category, in the current datablock, in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of rows or 0 for failure.

**REMARKS**

See also: *ndb_cif_count_datablock*
*ndb_cif_count_category*
*ndb_cif_count_column*

### 2.0.49 ndb_cif_count_column

**NAME**

*ndb_cif_count_column*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_count_column(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_count_column* counts the number of columns in the current category, in the current datablock, in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the number of columns or 0 for failure.

**REMARKS**

See also: *ndb_cif_count_datablock*
*ndb_cif_count_category*
*ndb_cif_count_row*

### 2.0.50 ndb_cif_current_datablock

**NAME**

*ndb_cif_current_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_current_datablock(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_current_datablock* returns the current datablock number of the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current datablock number.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_current_datablock_name* |
| | *ndb_cif_current_category* |
| | *ndb_cif_current_category_name* |
| | *ndb_cif_current_row* |
| | *ndb_cif_current_col* |

### 2.0.51 ndb_cif_current_datablock_name

#### NAME

*ndb_cif_current_datablock_name*

#### PROTOTYPE

```
#include "cifparse.h"

int ndb_cif_current_datablock_name(CifHandle *cf,
                                    char datablockName[MxNameLen]);
```

#### PURPOSE

*ndb_cif_current_datablock_name* stores the name of the current datablock in *datablockName*, which must be of the fixed length *MxNameLen*.

#### RECEIVES

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | a fixed length string to hold the name of the current datablock |

#### RETURN VALUE

Returns the current datablock number.

#### REMARKS

See also: *ndb_cif_current_datablock*
*ndb_cif_current_category*
*ndb_cif_current_category_name*
*ndb_cif_current_row*
*ndb_cif_current_col*

### 2.0.52 ndb_cif_current_category

**NAME**

*ndb_cif_current_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_current_category(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_current_category* returns the current category number of the CifHandle data structure specified by it cf.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current category number or 0 for failure.

**REMARKS**

See also: *ndb_cif_current_datablock*
*ndb_cif_current_datablock_name*
*ndb_cif_current_category_name*
*ndb_cif_current_row*
*ndb_cif_current_col*

### 2.0.53 ndb_cif_current_category_name

**NAME**

*ndb_cif_current_category_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_current_category_name(CifHandle *cf,
                                  char categoryName[MxNameLen]);
```

**PURPOSE**

*ndb_cif_current_category_name* stores the current category's name in *categoryName*, which must be of the fixed length *MxNameLen*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `categoryName` | a fixed size string to hold the name of the current category |

**RETURN VALUE**

Returns the current category number or 0 for failure.

**REMARKS**

See also: *ndb_cif_current_datablock*
*ndb_cif_current_datablock_name*
*ndb_cif_current_category*
*ndb_cif_current_row*
*ndb_cif_current_col*

### 2.0.54 ndb_cif_current_row

**NAME**

*ndb_cif_current_row*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_current_row(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_current_row* returns the current row number of the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current row number or 0 for failure.

**REMARKS**

See also: *ndb_cif_current_datablock*
*ndb_cif_current_datablock_name*
*ndb_cif_current_category*
*ndb_cif_current_category_name*
*ndb_cif_current_col*

### 2.0.55 ndb_cif_current_col

**NAME**

*ndb_cif_current_col*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_current_col(CifHandle *cf);
```

**PURPOSE**

*ndb_cif_current_col* returns the current column number of the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |

**RETURN VALUE**

Returns the current column number or 0 for failure.

**REMARKS**

See also: *ndb_cif_current_datablock*
*ndb_cif_current_datablock_name*
*ndb_cif_current_category*
*ndb_cif_current_category_name*
*ndb_cif_current_row*

### 2.0.56 ndb_cif_put_item_keyword

**NAME**

*ndb_cif_put_item_keyword*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_put_item_keyword(CifHandle *cf,
                             const char * itemKeyword);
```

**PURPOSE**

*ndb_cif_put_item_keyword* adds a new column with the name *itemKeyword* to the current category in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `itemKeyword` | the name of the new column |

**RETURN VALUE**

Returns the current column number or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_remove_item_keyword*

### 2.0.57 ndb_cif_remove_item_keyword

**NAME**

*ndb_cif_remove_item_keyword*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_remove_item_keyword(CifHandle *cf,
                               const char * itemKeyword);
```

**PURPOSE**

*ndb_cif_remove_item_keyword* removes the item keyword specified by the name *itemKeyword*, and all values from from the specified column in the current category in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `itemKeyword` | the name of the column to remove |

**RETURN VALUE**

Returns the current column number or 0 for failure.

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_put_item_keyword*

### 2.0.58 ndb_cif_get_item_name

**NAME**

*ndb_cif_get_item_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_item_name(CifHandle *cf,
                          const int colId,
                          char itemName[MxNameLen]);
```

**PURPOSE**

*ndb_cif_get_item_name* concatenates the current category name with the item keyword of the column identified by *colId* of CifHandle data structure specified by *cf* to return a CIF compliant item name stored in *itemName*, which must be of the fixed length *MxNameLen*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |
| `itemName` | a string of fixed length to store the item name |

**RETURN VALUE**

Returns 1 upon success and 0 upon failure, with success leading to the item name being stored in the third argument.

**REMARKS**

See also: *ndb_cif_get_item_shname*

### 2.0.59 ndb_cif_get_item_shname

**NAME**

*ndb_cif_get_item_shname*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_item_shname(CifHandle *cf,
                            const int colId,
                            char itemName[MxNameLen]);
```

**PURPOSE**

*ndb_cif_get_item_shname* extracts the item name (short name) from the complete item name identified by *colId* of CifHandle data structure specified by *cf* to return a CIF compliant item name stored in *itemName*, which must be of the fixed length *MxNameLen*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |
| `itemName` | a string of fixed length to store the item name |

**RETURN VALUE**

Returns 1 upon success and 0 upon failure, with success leading to the item name being stored in the third argument.

**REMARKS**

See also: *ndb_cif_get_item_name*

### 2.0.60 ndb_cif_get_item_value

**NAME**

*ndb_cif_get_item_value*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_item_value(CifHandle *cf,
                           const int colId,
                           char fieldValue[maxFieldLen],
                           const int maxFieldLen);
```

**PURPOSE**

*ndb_cif_get_item_value* copies the item value in the current row and column identified by *colId* in the CifHandle data structure specified by *cf* into the buffer *fieldValue*, which must be of the fixed length *maxFieldLen*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |
| `fieldValue` | a fixed length buffer to hold the item value |
| `maxFieldLen` | the length of the buffer *fieldValue* |

**RETURN VALUE**

Returns 1 upon success and 0 upon failure, with success leading to the item value being stored in the third argument.

**REMARKS**

See also: *ndb_cif_output_item*
*ndb_cif_copy_item_value*
*ndb_cif_get_item_value_length*

### 2.0.61 ndb_cif_put_item_value

**NAME**

*ndb_cif_put_item_value*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_put_item_value(CifHandle *cf,
                           const int colId,
                           const char * fieldValue);
```

**PURPOSE**

*ndb_cif_put_item_value* copies the item value stored in fieldValue into the current row and the column identified by *colId* in the CifHandle data structures specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |
| `fieldValue` | holds the item value to be copied |

**RETURN VALUE**

Returns 1 upon success and 0 upon failure.

**REMARKS**

Function does provide new value to the cf.

### 2.0.62 ndb_cif_output_item

**NAME**

*ndb_cif_output_item*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_output_item(CifHandle *cf,
                        FILE * fp,
                        const int colId);
```

**PURPOSE**

*ndb_cif_output_item* outputs the value of the column specified by *colId* in the current row in CifHandle data strucutre specified by *cf* into the file pointed to by it fp.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `fp` | pointer to the file descriptor of the output file |
| `colId` | identifies the column |

**RETURN VALUE**

Returns 1 upon success and 0 upon failure..

**REMARKS**

See also: *ndb_cif_get_item_value*
*ndb_cif_copy_item_value*

### 2.0.63 ndb_cif_copy_item_value

**NAME**

*ndb_cif_copy_item_value*

**PROTOTYPE**

```
#include "cifparse.h"

char *ndb_cif_copy_value(CifHandle *cf,
                         const int colId);
```

**PURPOSE**

*ndb_cif_copy_item_value* copies the value of the column specified by *colId* in the current row in CifHandle data strucutre specified by *cf* into the newly allocated string.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |

**RETURN VALUE**

Returns the the newly allocated string that contains a copy value of the column or NULL if failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_get_item_value* |
| | *ndb_cif_output_item* |

### 2.0.64 ndb_cif_get_item_value_length

**NAME**

*ndb_cif_get_item_value_length*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_item_value_length(CifHandle *cf,
                           const int colId);
```

**PURPOSE**

*ndb_cif_get_item_value_length* returns the length of the item value in the current row and column identified by *colId* in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |

**RETURN VALUE**

Returns the length of the item value.

**REMARKS**

See also: *ndb_cif_get_item_value*

### 2.0.65 ndb_cif_item_value_strncmp

**NAME**

*ndb_cif_item_value_strncmp*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_item_value_strncmp(CifHandle *cf,
                                const int catId,
                                const int colId,
                                const int rowId,
                                const int startPos,
                                char fieldValue[maxFieldLen],
                                const int maxFieldLen);
```

**PURPOSE**

*ndb_cif_item_value_strncmp* compares the item value in the CifHandle data stucture specified by *cf* in the category *catId*, in the column *colId* in the row *rowId* with the string specified by *fieldValue* ,which must be of the fixed length *maxFieldLen*. Starts at the position *startPos* in the row, and compares *maxFiledLen* charasters.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `catId` | identifies the category |
| `colId` | identifies the column |
| `rowId` | identifies the row |
| `startPos` | number of the start position |
| `fieldValue` | pointer to the string to be compared |
| `maxFieldLen` | the length of the string *fieldValue* |

**RETURN VALUE**

Returns 0 if the srings are equal, or nonzero value if aren't.

**REMARKS**

See also: *ndb_cif_item_values_strncmp*

### 2.0.66 ndb_cif_item_values_strncmp

**NAME**

*ndb_cif_item_values_strncmp*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_item_values_strncmp(CifHandle *cf,
                                const int catId1,
                                const int colId1,
                                const int rowId1,
                                const int startPos1,
                                const int catId2,
                                const int colId2,
                                const int rowId2,
                                const int startPos2,
                                const int maxFieldLen);
```

**PURPOSE**

*ndb_cif_item_values_strncmp* compares the values in the two rows of the CifHandle data stucture specified by *cf*. Rows are specifieded by the category *catId1*, the column *colId1*, the row *rowId1* and category *catId2*, the column *colId2*, the row *rowId1*, respectively. Comparing starts at the position *startPos1* in the first row and at the position *startPos2* in the second row, and compares *maxFiledLen* charasters.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `catId1` | identifies the category |
| `colId1` | identifies the column |
| `rowId1` | identifies the row |
| `startPos1` | number of the start position |
| `catId2` | identifies the category |
| `colId2` | identifies the column |
| `rowId2` | identifies the row |
| `startPos2` | number of the start position |
| `maxFieldLen` | the length of the string *fieldValue* |

## RETURN VALUE

Returns 0 if the srings are equal, or nonzero value if aren't.

## REMARKS

See also: *ndb_cif_item_value_strncmp*

### 2.0.67 ndb_cif_item_row_1_key

**NAME**

*ndb_cif_item_row_1_key*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_item_row_1_key(CifHandle *cf,
                           const int colId,
                           char *fieldValue);
```

**PURPOSE**

*ndb_cif_item_row_1_key* in the CifHandle data stucture specified by *cf* examines all the rows and return the row number if the *fieldValue* is found in the column specified by *colId*, and also set the current row to the row where the value was found.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId` | identifies the column |
| `fieldValue` | pointer to the value which is being looked for |

**RETURN VALUE**

Returns the row number where the value were found or 0 for failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_item_row_2_key*<br>*ndb_cif_item_row_3_key*<br>*ndb_cif_item_row_4_key* |

### 2.0.68 ndb_cif_item_row_2_key

**NAME**

*ndb_cif_item_row_2_key*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_item_row_2_key(CifHandle *cf,
                           const int colId1,
                           char *fieldValue1,
                           const int colId2,
                           char *fieldValue2);
```

**PURPOSE**

*ndb_cif_item_row_2_key* in the CifHandle data stucture specified by *cf* examines all the rows and return the row number if the *fieldValue1* and *fieldValue2* are found in the columns *colId1* and *colId2* respectively, and also set the current row to the row where the values were found.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId1` | identifies the first column |
| `fieldValue1` | pointer to the value which is being looked for in the first column |
| `colId2` | identifies the second column |
| `fieldValue2` | pointer to the value which is being looked for in the second column |

**RETURN VALUE**

Returns the row number where the values was found or 0 for failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_item_row_1_key* |
| | *ndb_cif_item_row_3_key* |
| | *ndb_cif_item_row_4_key* |

### 2.0.69 ndb_cif_item_row_3_key

**NAME**

*ndb_cif_item_row_3_key*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_item_row_3_key(CifHandle *cf,
                           const int colId1,
                           char *fieldValue1,
                           const int colId2,
                           char *fieldValue2,
                           const int colId3,
                           char *fieldValue3);
```

**PURPOSE**

*ndb_cif_item_row_3_key* in the CifHandle data stucture specified by *cf* examines all the rows and return the row number if the *fieldValue1*, *fieldValue2* and *fieldValue3* are found in the columns *colId1*, *colId2* and *colId3* respectively, and also set the current row to the row where the values were found.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId1` | identifies the first column |
| `fieldValue1` | pointer to the value which is being looked for in the first column |
| `colId2` | identifies the second column |
| `fieldValue2` | pointer to the value which is being looked for in the second column |
| `colId3` | identifies the third column |
| `fieldValue3` | pointer to the value which is being looked for in the third column |

**RETURN VALUE**

Returns the row number where the values was found or 0 for failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_item_row_1_key*<br>*ndb_cif_item_row_2_key*<br>*ndb_cif_item_row_4_key* |

### 2.0.70 ndb_cif_item_row_4_key

**NAME**

*ndb_cif_item_row_4_key*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_item_row_4_key(CifHandle *cf,
                           const int colId1, char *fieldValue1,
                           const int colId2, char *fieldValue2,
                           const int colId3, char *fieldValue3,
                           const int colId4, char *fieldValue4);
```

**PURPOSE**

*ndb_cif_item_row_4_key* in the CifHandle data stucture specified by *cf* examines all the rows and return the row number if the *fieldValue1*, *fieldValue2*, *fieldValue3* and *fieldValue4* are found in the columns *colId1*, *colId2*, *colId3* and *colId4* respectively, and also set the current row to the row where the values were found.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `colId1` | identifies the first column |
| `fieldValue1` | pointer to the value which is being looked for in the first column |
| `colId2` | identifies the second column |
| `fieldValue2` | pointer to the value which is being looked for in the second column |
| `colId3` | identifies the third column |
| `fieldValue3` | pointer to the value which is being looked for in the third column |
| `colId4` | identifies the fourth column |
| `fieldValue4` | pointer to the value which is being looked for in the fourth column |

## RETURN VALUE

Returns the row number where the values was found or 0 for failure.

## REMARKS

See also: *ndb_cif_item_row_1_key*
*ndb_cif_item_row_2_key*
*ndb_cif_item_row_3_key*

### 2.0.71 ndb_cif_get_category_name_from_item_name

**NAME**

*ndb_cif_get_category_name_from_item_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_category_name_from_item_name(CifHandle *cf,
                                             char categoryName[MxNameLen],
                                             const char * itemName);
```

**PURPOSE**

*ndb_cif_get_category_name_from_item_name* extracts the category name from the complete item name *itemName* in the CifHandle data structure specified by *cf* and stores the result in *categoryName*, which must be of the fixed length *MxNameLen*. Checks are made to ensure that the extraction leads to a valid category name.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `categoryName` | a fixed length string to hold the category name |
| `itemName` | the name of the item from which the category name is to be extracted |

**RETURN VALUE**

Returns the category's number or 0 upon failure.

**REMARKS**

See also: *ndb_cif_get_item_keyword_from_item_name*

### 2.0.72 ndb_cif_get_item_keyword_from_item_name

**NAME**

*ndb_cif_get_item_keyword_from_item_name*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_item_keyword_from_item_name(CifHandle *cf,
                                            char itemKeyword[MxNameLen],
                                            const char * itemName);
```

**PURPOSE**

*ndb_cif_get_item_keyword_from_item_name* extracts the item keyword (column name) from the complete item name *itemName* in the CifHandle data structure specified by *cf* and stores the result in *itemKeyword*, which must be of the fixed length *MxNameLen*. Checks are made to ensure that the extraction leads to a valid item keyword.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `itemKeyword` | a fixed length string to hold the item keyword |
| `itemName` | the name of the item from which the item keyword is to be extracted |

**RETURN VALUE**

Returns the column's number or 0 upon failure.

**REMARKS**

See also: *ndb_cif_get_category_name_from_item_name*

### 2.0.73 ndb_cif_get_datablock_id

**NAME**

*ndb_cif_get_datablock_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_datablock_id(CifHandle *cf,
                             const char * datablockName);
```

**PURPOSE**

*ndb_cif_get_datablock_id* returns the datablock number of the datablock specified by *datablockName* in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock |

**RETURN VALUE**

Returns the number of the datablock or 0 upon failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_get_category_id* |
| | *ndb_cif_get_column_id* |

### 2.0.74 ndb_cif_get_category_id

**NAME**

*ndb_cif_get_category_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_category_id(CifHandle *cf,
                            const char * datablockName,
                            const char * categoryName);
```

**PURPOSE**

*ndb_cif_get_category_id* returns the category number of the category specified by *categoryName* in the datablock *datablockName* in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock |
| `categoryName` | identifies the category |

**RETURN VALUE**

Returns the number of the category or 0 upon failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_get_datablock_id* |
| | *ndb_cif_get_column_id* |

### 2.0.75 ndb_cif_get_column_id

**NAME**

*ndb_cif_get_column_id*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_get_column_id(CifHandle *cf,
                          const char * datablockName,
                          const char * categoryName,
                          const char * itemKeyword);
```

**PURPOSE**

*ndb_cif_get_column_id* returns the column number of the column specified by *itemKeyword* in the category *categoryName* in datablock *datablockName* and CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `datablockName` | identifies the datablock |
| `categoryName` | identifies the category |
| `itemKeyword` | identifies the column |

**RETURN VALUE**

Returns the number of the column or 0 upon failure.

**REMARKS**

See also: *ndb_cif_get_datablock_id*
*ndb_cif_get_category_id*

### 2.0.76 ndb_cif_print_datablock

#### NAME

*ndb_cif_print_datablock*

#### PROTOTYPE

```
#include "cifparse.h"

void ndb_cif_print_datablock(CifHandle *cf,
                             FILE *fp);
```

#### PURPOSE

*ndb_cif_print_datablock* dumps the CifHandle data structure representation, specified by *cf*, of the current datablock into the file pointed to by *fp*.

#### RECEIVES

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `fp` | a pointer to the file descriptor of the output file |

#### RETURN VALUE

None

#### REMARKS

See also: *ndb_cif_print_datablocks*

### 2.0.77 ndb_cif_pretty_print_datablock

**NAME**

*ndb_cif_pretty_print_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

void ndb_cif_pretty_print_datablock(CifHandle *cf,
                                    FILE *fp);
```

**PURPOSE**

*ndb_cif_pretty_print_datablock* dumps the CifHandle data structure representation, specified by *cf*, of the current datablock into the file pointed to by *fp*. File will be column formated.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `fp` | a pointer to the file descriptor of the output file |

**RETURN VALUE**

None

**REMARKS**

See also: *ndb_cif_print_datablocks*

### 2.0.78 ndb_cif_print_datablocks

**NAME**

*ndb_cif_print_datablocks*

**PROTOTYPE**

```
#include "cifparse.h"

void ndb_cif_print_datablocks(CifHandle *cf,
                              FILE *fp);
```

**PURPOSE**

*ndb_cif_print_datablocks* dumps the CifHandle data structure representation, specified by *cf*, of the all the datablocks into the file pointed to by *fp*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `fp` | a pointer to the file descriptor of the output file |

**RETURN VALUE**

None

**REMARKS**

See also: *ndb_cif_print_datablock*

### 2.0.79 ndb_cif_write_category

**NAME**

*ndb_cif_write_category*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_write_category(CifHandle *cf,
                           FILE *fp);
```

**PURPOSE**

*ndb_cif_write_category* writes all of the data of current category loaded into the CifHande data structure specified by *cf* to the file pointed to by *fp*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to CifHandle structure where the data blocks loaded |
| `fp` | pointer to the file descriptor for the mm-CIF output file |

**RETURN VALUE**

Returns the number of data blocks written or 0 for failure.

**REMARKS**

None

### 2.0.80 ndb_cif_write_file_pr

**NAME**

*ndb_cif_write_file_pr*

**PROTOTYPE**

```
#include "cifparse.h"

void ndb_cif_write_file_pr(CifHandle *cf,
                           FILE *fp);
```

**PURPOSE**

*ndb_cif_write_file* writes all of the data blocks loaded into the CifHande data structure specified by *cf* to the file pointed to by the *fp*. File will be column formated.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure where the data blocks loaded |
| `fp` | pointer to the file descriptor for the mmCIF output file |

**RETURN VALUE**

Returns the number of data blocks written or 0 for failure.

**REMARKS**

| | |
|---|---|
| See also: | *ndb_cif_write_file* |

### 2.0.81 ndb_cif_compare

**NAME**

*ndb_cif_compare*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_compare(CifHandle *cf1,
                    char *name1,
                    CifHandle *cf2,
                    char *name2,
                    FILE *msg);
```

**PURPOSE**

*ndb_cif_compare* compares two CifHandle structures specified by *cif1* and *cif2*, with the names *name1* and *name2* recpectively. Result of this comparison is reported into the file pointed to by the input file descriptor *msg*.

**RECEIVES**

| | |
|---|---|
| `cf1` | pointer to the first CifHandle data structure |
| `name1` | name of first cif structure (file) |
| `cf2` | pointer to the second CifHandle data structure |
| `name2` | to the name of second cif structure (file) |
| `fp` | pointer to the file descriptor where the results would be reported |

**RETURN VALUE**

Returns the 1 if the structures are the same 0 if not.

**REMARKS**

None

### 2.0.82 ndb_cif_compress_file

**NAME**

*ndb_cif_compress_file*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_compress_file(CifHandle *cf,
                          CifHandle *ddl);
```

**PURPOSE**

*ndb_cif_compress_file* compresses all datablocks in the CifHandle data structure, specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `ddl` | pointer to the CifHandle data structure where is the ddl/dictionary saved |

**RETURN VALUE**

None

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_compress_datablock* |
| | *ndb_cif_compress_category* |

### 2.0.83 ndb_cif_compress_datablock

**NAME**

*ndb_cif_compress_datablock*

**PROTOTYPE**

```
#include "cifparse.h"

int ndb_cif_compress_datablock(CifHandle *cf,
                               CifHandle *ddl);
```

**PURPOSE**

*ndb_cif_compress_datablock* compresses all categories in the current datablock in the CifHandle data structure specified by *cf*.

**RECEIVES**

| | |
|---|---|
| `cf` | pointer to the CifHandle data structure |
| `ddl` | pointer to the CifHandle data structure where is the ddl/dictionary saved |

**RETURN VALUE**

None

**REMARKS**

Function does provide new value to the cf.

See also: *ndb_cif_compress_file*
*ndb_cif_compress_category*

### 2.0.84 ndb_cif_compress_category

**NAME**

*ndb_cif_compress_category*

**PROTOTYPE**

```
#include "cifparse.h"

void ndb_cif_compress_category(CifHandle *cf,
                               CifHandle *ddl);
```

**PURPOSE**

*ndb_cif_compress_category* delete redundant rows from CifHandle structure specified by *cf* in the curent category. Knowledge about what rows are redundant is found in the *ddl* for dictionaries, i.e. in the dictionary for data files.

**RECEIVES**

| | |
|---|---|
| `cf1` | pointer to the CifHandle data structure wich are compressing |
| `ddl` | pointer to the CifHandle data structure where the knowledge is saved (ddl/dictionary) |

**RETURN VALUE**

None

**REMARKS**

Function does provide new value to the cf.

| | |
|---|---|
| See also: | *ndb_cif_compress_file* |
| | *ndb_cif_compress_datablock* |

### 2.0.85 ndb_cif_encapsulate

**NAME**

*ndb_cif_encapsulate*

**PROTOTYPE**

```
#include "cifparse.h"

void ndb_cif_encapsulate(char ** a);
```

**PURPOSE**

*ndb_cif_encapsulate* puts the encapsulation characters into string. At the begining of string puts 5 dollar signs ($$$$$), and one dollar sign ($) at the begining of every new line in the string. This function is used for embedded cif file.

**RECEIVES**

| | |
|---|---|
| `a` | pointer to string |

**RETURN VALUE**

None

**REMARKS**

Function does provide new value to the string.

See also: *ndb_cif_de_encapsulate*

### 2.0.86 ndb_cif_de_encapsulate

#### NAME

*ndb_cif_de_encapsulate*

#### PROTOTYPE

```
#include "cifparse.h"

void ndb_cif_de_encapsulate(char ** a);
```

#### PURPOSE

*ndb_cif_de_encapsulate* takes off the encapsulation characters from an ecapsulated string.

#### RECEIVES

| | |
|---|---|
| `a` | pointer to string |

#### RETURN VALUE

None

#### REMARKS

Function does provide new value to the string.

See also: *ndb_cif_encapsulate*

# References


[1] P. M. D. Fitzgerald, H. M. Berman, P. E. Bourne, and K. Watenpaugh. *Macromolecular CIF Working Group.* International Union of Crystallography, 1992.

[2] P. E. Bourne, H. M. Berman, K. Watenpaugh, J. D. Westbrook, and P. M. D. Fitzgerald. The Macromolecular Crystallographic Information File (mmCIF). *Methods in Enzymology*, 1996. submitted.

[3] 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.

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

[5] 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.

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