Table 5.6.2.4. *Values for encodings in* `cbf_write_file`

Values may be combined bit-wise.

`ENC_BASE64`	Use BASE64 encoding (default)
`ENC_QP`	Use quoted-printable encoding
`ENC_BASE8`	Use BASE8 (octal) encoding
`ENC_BASE10`	Use BASE10 (decimal) encoding
`ENC_BASE16`	Use BASE16 (hexadecimal) encoding
`ENC_FORWARD`	For BASE8, BASE10 or BASE16 encoding, map bytes to words forward (1234) (default on little-endian machines)
`ENC_BACKWARD`	For BASE8, BASE10 or BASE16 encoding, map bytes to words backward (4321) (default on big-endian machines)
`ENC_CRTERM`	Terminate lines with CR
`ENC_LFTERM`	Terminate lines with LF (default)

block and `cbf_next_datablock` selects the data block following the current data block.

All of these functions return `CBF_NOTFOUND` if the requested object does not exist.

The '`count`' functions evaluate the number of data blocks in the data set, the number of categories in the current data block and the number of columns or rows in the current category:

```
int cbf_count_datablocks (cbf_handle handle,
     unsigned int *datablocks);
int cbf_count_categories (cbf_handle handle,
     unsigned int *categories);
int cbf_count_columns (cbf_handle handle,
     unsigned int *columns);
int cbf_count_rows (cbf_handle handle,
     unsigned int *rows);
```

The '`name`' functions retrieve the current data block, category or column names:

```
int cbf_datablock_name (cbf_handle handle,
     const char **datablockname);
int cbf_set_datablockname (cbf_handle handle,
     const char *datablockname);
int cbf_category_name (cbf_handle handle,
     const char **categoryname);
```

As rows do not have names, the corresponding function is:

```
int cbf_row_number (cbf_handle handle,
     unsigned int *row);
```

To create new entities within the tree, *CBFlib* provides the functions:

```
int cbf_new_datablock (cbf_handle handle,
     const char *datablockname);
int cbf_new_category (cbf_handle handle,
     const char *categoryname);
int cbf_new_column (cbf_handle handle,
     const char *columnname);
int cbf_new_row (cbf_handle handle);
int cbf_insert_row (cbf_handle handle,
     unsigned int row);
int cbf_force_new_datablock (cbf_handle handle,
     const char *datablockname);
int cbf_force_new_category (cbf_handle handle,
     const char *categoryname);
```

The '`new`' functions add a new data block within the data set, a new category in the current data block, or a new column or row within the current category, and make it the current data block, category, column or row, respectively. If the data block, category or column already exists, then the function simply makes it the current data block, category or column. The function `cbf_new_row` adds the row to the end of the current category. `cbf_insert_row` provides the ability to insert a row before ordinal *row*, starting from 0. The newly inserted row gets the row ordinal *row* and the row that originally had that ordinal and all rows with higher ordinals are pushed downwards.

In general, CIF does not permit duplication of the names of data blocks or categories. In practice, however, duplications do occur. *CBFlib* provides '`force`' variants to allow creation of duplicate data-block and category names. Because, in this case, the program analysing the resulting file can only distinguish the duplicates by ordinal, these variants are not recommended for general use.

The following functions are used to remove entities from the tree:

```
int cbf_remove_datablock (cbf_handle handle);
int cbf_remove_category (cbf_handle handle);
int cbf_remove_column (cbf_handle handle);
int cbf_remove_row (cbf_handle handle);
int cbf_delete_row (cbf_handle handle,
     unsigned int row);
int cbf_reset_datablocks (cbf_handle handle);
int cbf_reset_datablock (cbf_handle handle);
int cbf_reset_category (cbf_handle handle);
```

The basic '`remove`' functions delete the current data block, category, column or row. Note that removing a data block makes the current data block, category, column and row undefined; removing a category makes the current category, column and row undefined. Removing a column makes the current column undefined, but leaves the current row intact, and removing a row leaves the current column intact. The function `cbf_delete_row` is similar to `cbf_remove_row` except that it removes the specified row in the current category. If the current row is not the deleted row, then it will remain valid.

All the categories in all data blocks, all the categories in the current data block or all the entries in the current category may be removed using the '`reset`' functions.

When a column and row within a category have been selected, the entry value may be examined or modified:

```
int cbf_get_value (cbf_handle handle,
     const char **value);
int cbf_set_value (cbf_handle handle,
     const char *value);
int cbf_get_integervalue (cbf_handle handle,
     int *number);
int cbf_set_integervalue (cbf_handle handle,
     int number);
int cbf_get_doublevalue (cbf_handle handle,
     double *number);
int cbf_set_doublevalue (cbf_handle handle,
     const char *format;
int cbf_get_integerarrayparameters (cbf_handle handle,
     unsigned int *compression, size_t *elsize,
     size_t *elements, int *maxelement);
int cbf_get_integerarray (cbf_handle handle,
     int *binary_id, int elsigned,
     size_t *elements_read);
int cbf_set_integerarray (cbf_handle handle,
     unsigned int compression, void *array,
     size_t elements);
```

A value within a CBF/imgCIF data set may be a simple character string, an integer or real number, or an array of integers. The functions `cbf_get_value` and `cbf_set_value` provide the basic functionality for normal CIF values, retrieving and modifying the