Table 5.6.2.5. *Values for the parameter compression in cbf_get_integerarrayparameters and cbf_set_integerarray*

CBF_CANONICAL	Canonical-code compression (Section 5.6.3.1)
CBF_PACKED	*CCP4*-style packing (Section 5.6.3.2)
CBF_NONE	No compression

current entry as a string. The functions cbf_get_integervalue and cbf_get_doublevalue interpret the retrieved string as an integer or real value and the functions cbf_set_integer and cbf_set_doublevalue convert the *number* argument into a string before setting the entry.

The functions for working with binary sections are more complicated as they must take into account compression, array size and the variety of different integer types available on different systems: signed/unsigned and various sizes.

The function cbf_get_integerarrayparameters retrieves the parameters of the current, binary, entry. The *compression* argument is set to the compression type used (Table 5.6.2.5). At present, this may take one of three values: CBF_CANONICAL, for canonical-code compression (see Section 5.6.3.1 below); CBF_PACKED, for *CCP4*-style packing (see Section 5.6.3.2 below); or CBF_NONE, for no compression. [*Note*: CBF_NONE is by far the slowest scheme of the three and uses much more disk space. It is intended for routine use with small arrays only. With large arrays (like images) it should be used only for debugging.] The *binary_id* value is a unique integer identifier for each binary section, *elsize* is the size in bytes of the array entries, *elsigned* and *elunsigned* are nonzero if the array can be read as unsigned or signed, respectively, *elements* is the number of entries in the array, and *minelement* and *maxelement* are the lowest and highest elements. If a destination argument is too small to hold a value, it will be set to the nearest value and the function will return CBF_OVERFLOW. If the current entry is not binary, cbf_get_integerarrayparameters will return CBF_ASCII.

cbf_get_integerarray reads the current binary entry into an integer array. The parameter *array* points to an array of elements interpreted as integers. Each element in the array is signed if *elsigned* is nonzero and unsigned otherwise, and each element occupies *elsize* bytes. The argument *elements_read* is set to the number of elements actually obtained. If the binary section does not contain sufficient entries to fill the array, the function returns CBF_ENDOFDATA. As before, the function will return CBF_OVERFLOW on overflow and CBF_ASCII if the entry is not binary.

cbf_set_integerarray sets the current binary or ASCII entry to the binary value of an integer array. As before, the acceptable values for compression are CBF_PACKED, CBF_CANONICAL and CBF_NONE. Each binary section should be given a unique integer identifier *binary_id*.

Two macros are provided to facilitate processing and propagation of error returns: one to return from the current function immediately and one to execute a given command first:

```
#define cbf_failnez(f) \
     {int err; err = (f); if (err) return err; }
#define cbf_onfailnez(f,c) \
     {int err; err = (f); if (err) {{c; }return err; }}
```

If the symbol CBFDEBUG is defined, alternative definitions that print out the error number as given in Table 5.6.1.1 are used:

```
#define cbf_failnez(x) \
{int err; err = (x); \
  if (err) { fprintf (stderr, \
     "\nCBFlib error %d in \"x\"\n", \
     err); return err; }}

#define cbf_onfailnez(x,c) \
{int err; err = (x); \
  if (err) { fprintf (stderr, \
    "\nCBFlib error %d in \"x\"\n", \
     err); \
     { c; } return err; }}
```

5.6.2.2. High-level *CBFlib* functions

The high-level *CBFlib* functions provide a level of abstraction above the CIF file structure and their prototypes are defined in the header file cbf_simple.h. Most of these functions simply use the low-level routines to navigate the CBF/imgCIF structure and read and modify data entries, and consequently expect a cbf_handle argument. There are also, however, additional sets of functions used to analyse the geometry of the goniometer and detector. These functions use additional handles of type cbf_goniometer and cbf_detector, respectively. All functions return the same error codes as the low-level functions do. The function return values are given in Table 5.6.1.1. The formal parameters for the high-level *CBFlib* functions are given in Table 5.6.2.6.

5.6.2.3. General high-level functions

The general high-level functions use the low-level routines to accomplish common tasks with a single call.

The first of these is used to facilitate the preparation of the complex CBF/imgCIF header structure:

```
int cbf_read_template (cbf_handle handle, FILE *file);
```

cbf_read_template simply reads the CBF/imgCIF file *file* into the data structure associated with the given handle and selects the first data block. It is typically used to read a template – an imgCIF file populated with data entries, but without any binary sections, into which experimental information can then be inserted. Template files are discussed further in Section 5.6.4 below.

The value of **_diffrn_radiation_wavelength.wavelength** can be retrieved or set. The functions

```
int cbf_get_wavelength (cbf_handle handle,
     double *wavelength);
int cbf_set_wavelength (cbf_handle handle,
     double wavelength);
```

operate on the categories DIFFRN_RADIATION and DIFFRN_RADIATION_WAVELENGTH. The wavelength is found indirectly. The value of **_diffrn_radiation.wavelength_id** is retrieved and used to find a matching row in the DIFFRN_RADIATION_WAVELENGTH category, from which the value of **_diffrn_radiation_wavelength.wavelength** is obtained.

The value of the ratio of the intensities of the polarization components **_diffrn_radiation.polarizn_source_ratio** and the value of the angle **_diffrn_radiation.polarizn_source_norm** between the normal to the polarization plane and the laboratory *Y* axis can be retrieved or set. The functions

```
int cbf_get_polarization (cbf_handle handle,
     double *polarizn_source_ratio,
     double *polarizn_source_norm);
int cbf_set_polarization (cbf_handle handle,
     double polarizn_source_ratio,
     double polarizn_source_norm);
```

operate on the DIFFRN_RADIATION category.