Dimension counts. `counts` is a 1-dimensional array containing one element per dimension. Each element of `counts` specifies the corresponding dimension count.

The only required preselected object/state is the current CDF.

`rVARs_DIMINTERVALS_` Selects the current dimension intervals for all rVariables in the current CDF. For 0-dimensional rVariables this operation is not applicable. Required arguments are as follows:

in: `long intervals[]`

Dimension intervals. `intervals` is a 1-dimensional array containing one element per dimension. Each element of `intervals` specifies the corresponding dimension interval.

The only required preselected object/state is the current CDF.

`rVAR_SEQPOS_` Selects the current sequential value for sequential access for the current rVariable (in the current CDF). Note that a current sequential value is maintained for each rVariable individually. Required arguments are as follows:

in: `long recNum`

Record number.

in: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per rVariable dimension. Each element of `indices` specifies the corresponding dimension index. For 0-dimensional rVariables this argument is ignored (but a place holder is necessary).

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_CACHESIZE_` Selects the number of cache buffers to be used for the current rVariable's file (of the current CDF). This operation is not applicable to a single-file CDF. Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long numBuffers`

The number of cache buffers to be used.

The required preselected objects/states are the current CDF and its current rVariable.

`rVARs_CACHESIZE_` Selects the number of cache buffers to be used for all of the rVariable files (of the current CDF). This operation is not applicable to a single-file CDF. Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long numBuffers`

The number of cache buffers to be used.

The only required preselected object/state is the current CDF.

`zVAR_` Explicitly selects the current zVariable (in the current CDF) by number. Required arguments are as follows:

in: `long varNum`

zVariable number.

The only required preselected object/state is the current CDF.

`zVAR_NAME_` Explicitly selects the current zVariable (in the current CDF) by name. Required arguments are as follows:

in: `char *varName`

zVariable name. This may be at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`zVARs_RECNUMBER_` Selects the current record number for each zVariable in the current CDF. This operation is provided to simplify the selection of the current record numbers for the zVariables involved in a multiple variable access operation (see Section 2.3.15). Required arguments are as follows:

in: `long recNum`

Record number.

The only required preselected object/state is the current CDF.

`zVAR_RECNUMBER_` Selects the current record number for the current zVariable in the current CDF. Required arguments are as follows:

in: `long recNum`

Record number.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_RECCOUNT_` Selects the current record count for the current zVariable in the current CDF. Required arguments are as follows:

in: `long recCount`

Record count.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_RECINTERVAL_` Selects the current record interval for the current zVariable in the current CDF. Required arguments are as follows:

in: `long recInterval`

Record interval.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMINDICES_` Selects the current dimension indices for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

in: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per dimension. Each element of `indices` specifies the corresponding dimension index.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMCOUNTS_` Selects the current dimension counts for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

in: `long counts[]`

Dimension counts. `counts` is a 1-dimensional array containing one element per dimension. Each element of `counts` specifies the corresponding dimension count.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMINTERVALS_` Selects the current dimension intervals for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

in: `long intervals[]`

Dimension intervals. `intervals` is a 1-dimensional array containing one element per dimension. Each element of `intervals` specifies the corresponding dimension interval.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_SEQPOS_` Selects the current sequential value for sequential access for the current zVariable (in the current CDF). Note that a current sequential value is maintained for each zVariable individually. Required arguments are as follows:

in: `long recNum`

Record number.

in: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per dimension. Each element of `indices` specifies the corresponding dimension index. For 0-dimensional zVariables this argument is ignored (but a place holder is necessary).

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_CACHESIZE_` Selects the number of cache buffers to be used for the current zVariable's file (of the current CDF). This operation is not applicable to a single-file CDF. Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long numBuffers`

The number of cache buffers to be used.

The required preselected objects/states are the current CDF and its current zVariable.

`zVARs_CACHESIZE_` Selects the number of cache buffers to be used for all of the zVariable files (of the current CDF). This operation is not applicable to a single-file CDF. Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long numBuffers`

The number of cache buffers to be used.

The only required preselected object/state is the current CDF.

`ATTR_` Explicitly selects the current attribute (in the current CDF) by number. Required arguments are as follows:

in: `long attrNum`

Attribute number.

The only required preselected object/state is the current CDF.

`ATTR_NAME_` Explicitly selects the current attribute (in the current CDF) by name. Required arguments are as follows:

in: `char *attrName`

Attribute name. This may be at most `CDF_ATTR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`gENTRY_` Selects the current gEntry number for all gAttributes in the current CDF. **NOTE:** Only use this operation when dealing with gAttributes. Required arguments are as follows:

in: `long entryNum`

gEntry number.

The only required preselected object/state is the current CDF.

`rENTRY_` Selects the current rEntry number for all vAttributes in the current CDF. **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

in: `long entryNum`

rEntry number.

The only required preselected object/state is the current CDF.

`rENTRY_NAME_` Selects the current rEntry number for all vAttributes (in the current CDF) by rVariable name. The number of the named rVariable becomes the current rEntry number. (The current rVariable is not changed.) **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

in: `char *varName`

rVariable name. This may be at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`zENTRY_` Selects the current zEntry number for all vAttributes in the current CDF. **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

in: `long entryNum`

zEntry number.

The only required preselected object/state is the current CDF.

`zENTRY_NAME_` Selects the current zEntry number for all vAttributes (in the current CDF) by zVariable name. The number of the named zVariable becomes the current zEntry number. (The current zVariable is not changed.) **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

in: `char *varName`

zVariable name. This may be at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`CONFIRM_` The following items may be confirmed:

`CDF_NAME_` Confirms the pathname of the current CDF. Required arguments are as follows:

out: `char *CDFpath`

Pathname of the CDF. This character string must be large enough to hold `CDF_PATHNAME_LEN + 1` characters (including the `NUL` terminator).

The only required preselected object/state is the current CDF.

`CDF_READONLY_MODE_` Confirms the read-only mode for the current CDF. Required arguments are as follows:

out: `long *mode`

The read-only mode. The read-only modes are described in Section 4.6.10.

The only required preselected object/state is the current CDF.

`CDF_zMODE_` Confirms the zMode for the current CDF. Required arguments are as follows:

in: `long *mode`

The zMode. The zModes are described in Section 4.6.11.

The only required preselected object/state is the current CDF.

`CDF_NEGtoPOSfp0_MODE_`

Confirms the `-0.0` to `0.0` mode for the current CDF. Required arguments are as follows:

in: `long *mode`

The `-0.0` to `0.0` mode. The `-0.0` to `0.0` modes are described in Section 4.6.12.

The only required preselected object/state is the current CDF.

`CDF_DECODING_` Confirms the decoding for the current CDF. Required arguments are as follows:

in: `long *decoding`

The decoding. The decodings are described in Section 4.6.6.

The only required preselected object/state is the current CDF.

`CDF_CACHESIZE_` Confirms the number of cache buffers being used for the "`.cdf`" file (for the current CDF). Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long *numBuffers`

The number of cache buffers being used.

The only required preselected object/state is the current CDF.

`rVAR_` Confirms the current rVariable (in the current CDF). Required arguments are as follows:

out: `long *varNum`

rVariable number.

The only required preselected object/state is the current CDF.

`rVAR_EXISTENCE_` Confirms the existence of the named rVariable (in the current CDF). If the rVariable does not exist, an error code will be returned. In any case the current rVariable is not affected. Required arguments are as follows:

out: `char *varName`

The rVariable name. This may be at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`rVARs_RECNUMBER_` Confirms the current record number for all rVariables in the current CDF. Required arguments are as follows:

out: `long *recNum`

Record number.

The only required preselected object/state is the current CDF.

`rVARs_RECCOUNT_` Confirms the current record count for all rVariables in the current CDF. Required arguments are as follows:

out: `long *recCount`

Record count.

The only required preselected object/state is the current CDF.

`rVARs_RECINTERVAL_` Confirms the current record interval for all rVariables in the current CDF. Required arguments are as follows:

out: `long *recInterval`

Record interval.

The only required preselected object/state is the current CDF.

`rVARs_DIMINDICES_` Confirms the current dimension indices for all rVariables in the current CDF. For 0-dimensional rVariables this operation is not applicable. Required arguments are as follows:

out: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per dimension. Each element of `indices` receives the corresponding dimension index.

The only required preselected object/state is the current CDF.

`rVARs_DIMCOUNTS_` Confirms the current dimension counts for all rVariables in the current CDF. For 0-dimensional rVariables this operation is not applicable. Required arguments are as follows:

out: `long counts[]`

Dimension counts. `counts` is a 1-dimensional array containing one element per dimension. Each element of `counts` receives the corresponding dimension count.

The only required preselected object/state is the current CDF.

`rVARs_DIMINTERVALS_` Confirms the current dimension intervals for all rVariables in the current CDF. For 0-dimensional rVariables this operation is not applica-

ble. Required arguments are as follows:

out: `long intervals[]`

Dimension intervals. `intervals` is a 1-dimensional array containing one element per dimension. Each element of `intervals` receives the corresponding dimension interval.

The only required preselected object/state is the current CDF.

`rVAR_SEQPOS_` Confirms the current sequential value for sequential access for the current rVariable (in the current CDF). Note that a current sequential value is maintained for each rVariable individually. Required arguments are as follows:

out: `long *recNum`

Record number.

out: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per dimension. Each element of `indices` receives the corresponding dimension index. For 0-dimensional rVariables this argument is ignored (but a place holder is necessary).

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_CACHESIZE_` Confirms the number of cache buffers being used for the current rVariable's file (of the current CDF). This operation is not applicable to a single-file CDF. Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long *numBuffers`

The number of cache buffers being used.

The required preselected objects/states are the current CDF and its current rVariable.

`zVAR_` Confirms the current zVariable (in the current CDF). Required arguments are as follows:

out: `long *varNum`

zVariable number.

The only required preselected object/state is the current CDF.

`zVAR_EXISTENCE_` Confirms the existence of the named zVariable (in the current CDF). If the zVariable does not exist, an error code will be returned. In any case the current zVariable is not affected. Required arguments are as follows:

out: `char *varName`

The zVariable name. This may be at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`zVAR_RECNUMBER_` Confirms the current record number for the current zVariable in the current CDF. Required arguments are as follows:

out: `long *recNum`

Record number.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_RECCOUNT_` Confirms the current record count for the current zVariable in the current CDF. Required arguments are as follows:

out: `long *recCount`

Record count.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_RECINTERVAL_` Confirms the current record interval for the current zVariable in the current CDF. Required arguments are as follows:

out: `long *recInterval`

Record interval.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMINDICES_` Confirms the current dimension indices for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

out: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per dimension. Each element of `indices` receives the corresponding dimension index.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMCOUNTS_` Confirms the current dimension counts for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

out: `long counts[]`

Dimension counts. `counts` is a 1-dimensional array containing one element per dimension. Each element of `counts` receives the corresponding dimension count.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMINTERVALS_` Confirms the current dimension intervals for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

out: `long intervals[]`

Dimension intervals. `intervals` is a 1-dimensional array containing one element per dimension. Each element of `intervals` receives the corresponding dimension interval.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_SEQPOS_` Confirms the current sequential value for sequential access for the current zVariable (in the current CDF). Note that a current sequential value is maintained for each zVariable individually. Required arguments are as follows:

out: `long *recNum`

Record number.

out: `long indices[]`

Dimension indices. `indices` is a 1-dimensional array containing one element per dimension. Each element of `indices` receives the corresponding dimension index. For 0-dimensional zVariables this argument is ignored (but a place holder is necessary).

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_CACHESIZE_` Confirms the number of cache buffers being used for the current zVariable's file (of the current CDF). This operation is not applicable to a single-file CDF. Section 2.1.4 describes the caching scheme used by the CDF library in greater detail. Required arguments are as follows:

in: `long *numBuffers`

The number of cache buffers being used.

The required preselected objects/states are the current CDF and its current zVariable.

`ATTR_` Confirms the current attribute (in the current CDF). Required arguments are as follows:

out: `long *attrNum`

Attribute number.

The only required preselected object/state is the current CDF.

`ATTR_EXISTENCE_` Confirms the existence of the named attribute (in the current CDF). If the attribute does not exist, an error code will be returned. In any case the current attribute is not affected. Required arguments are as follows:

out: `char *attrName`

The attribute name. This may be at most `CDF_ATTR_NAME_LEN` characters (excluding the `NUL` terminator).

The only required preselected object/state is the current CDF.

`gENTRY_` Confirms the current gEntry number for all attributes in the current CDF. **NOTE:** Only use this operation when dealing with gAttributes. Required arguments are as follows:

out: `long *entryNum`

The gEntry number.

The only required preselected object/state is the current CDF.

`rENTRY_` Confirms the current rEntry number for all attributes in the current CDF. **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

out: `long *entryNum`

The rEntry number.

The only required preselected object/state is the current CDF.

`zENTRY_` Confirms the current zEntry number for all attributes in the current CDF. **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

out: `long *entryNum`

The zEntry number.

The only required preselected object/state is the current CDF.

`gENTRY_EXISTENCE_` Confirms the existence of the specified gEntry (for the current attribute in the current CDF). If the gEntry does not exist, an error code will be returned. In any case the current gEntry number is not affected. **NOTE:** Only use this operation when dealing with gAttributes. Required arguments are as follows:

in: `long entryNum`

The gEntry number.

The required preselected objects/states are the current CDF and its current attribute.

`rENTRY_EXISTENCE_` Confirms the existence of the specified rEntry (for the current attribute in the current CDF). If the rEntry does not exist, an error code will be returned. In any case the current rEntry number is not

affected. **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

in: `long entryNum`

The rEntry number.

The required preselected objects/states are the current CDF and its current attribute.

`zENTRY_EXISTENCE_` Confirms the existence of the specified zEntry (for the current attribute in the current CDF). If the zEntry does not exist, an error code will be returned. In any case the current zEntry number is not affected. **NOTE:** Only use this operation when dealing with vAttributes. Required arguments are as follows:

in: `long entryNum`

The zEntry number.

The required preselected objects/states are the current CDF and its current attribute.

`PUT_` The following items may be written (put):

`CDF_ENCODING_` Respecifies the data encoding of the current CDF. A CDF's data encoding may not be changed after any variable values (including the pad value) or attribute entries have been written. Required arguments are as follows:

in: `long encoding`

New data encoding. Specify one of the encodings described in Section 4.6.5.

The only required preselected object/state is the current CDF.

`CDF_MAJORITY_` Respecifies the variable majority of the current CDF. A CDF's variable majority may not be changed after any variable values have been written. Required arguments are as follows:

in: `long majority`

New variable majority. Specify one of the majorities described in Section 4.6.7.

The only required preselected object/state is the current CDF.

`CDF_FORMAT_` Respecifies the format of the current CDF. A CDF's format may not be changed after any variables have been created. Required arguments are as follows:

in: `long format`

New CDF format. Specify one of the formats described in Section 4.6.3.

The only required preselected object/state is the current CDF.

`rVAR_NAME_` Renames the current rVariable (in the current CDF). A variable (rVariable or zVariable) with the same name must not already exist in the CDF. Required arguments are as follows:

in: `char *varName`

New name of the rVariable. This may consist of at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_DATASPEC_` Respecifies the data specification (data type and number of elements) of the current rVariable (in the current CDF). An rVariable's data specification may not be changed if the new data specification is not equivalent to the old data specification and any values (including the pad value) have been written. Data specifications are considered equivalent if the data types are equivalent (see Section 2.2.9) and the number of elements are the same. Required arguments are as follows:

in: `long dataType`

New data type. Specify one of the data types described in Section 4.6.4.

in: `long numElements`

Number of elements of the data type at each value. For character string data types (`CDF_CHAR` and `CDF_UCHAR`), this is the number of characters in each string (an array of characters). A string exists at each value. For the non-character string data types this must be one (1) — arrays of values are not allowed for non-character string data types.

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_RECVARY_` Respecifies the record variance of the current rVariable (in the current CDF). An rVariable's record variance may not be changed if any values have been written (except for an explicit pad value — it may have been written). Required arguments are as follows:

in: `long recVary`

New record variance. Specify one of the variances described in Section 4.6.8.

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_DIMVARYS_` Respecifies the dimension variances of the current rVariable (in the current CDF). An rVariable's dimension variances may not be changed if any values have been written (except for an explicit pad value — it may have been written). For 0-dimensional rVariables this operation is not applicable. Required arguments are as follows:

in: `long dimVarys[]`

New dimension variances. `dimVarys` is a 1-dimensional array containing one element per dimension. Each element of `dimVarys` specifies the corresponding dimension variance. For each dimension specify one of the variances described in Section 4.6.8.

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_PADVALUE_` Specifies the pad value for the current rVariable (in the current CDF). An rVariable's pad value may be specified (or respecified) at any time without affecting already written values (including where pad values were used). Section 2.3.16 describes variable pad values. Required arguments are as follows:

in: `void *value`

Pad value. The pad value is written to the CDF from memory address `value`.

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_INITIALRECS_` Specifies the number of records to initially write to the current rVariable (in the current CDF). This may be specified only once per rVariable and before any other records have been written to that rVariable. If a pad value has not yet been specified, the default is used (see Section 2.3.16). If a pad value has been explicitly specified, that value is written to the records. Section 2.3.7 describes initial records. Required arguments are as follows:

in: `long NinitialRecs`

Number of records to write.

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_EXTENDRECS_` Specifies the number of physical records to extend the current rVariable (in the current CDF) whenever that rVariable needs to be extended. Section 2.3.7 describes extend records. **NOTE:** The number of extend records has no effect for NRV rVariables or multi-file CDFs. Required arguments are as follows:

in: `long NextendRecs`

Number of records to extend. A value of zero (0) indicates that the default number of records should be used.

The required preselected objects/states are the current CDF and its current rVariable.

`rVAR_DATA_` Writes one value to the current rVariable (in the current CDF). The value is written at the current record number and current dimension indices for the rVariables (in the current CDF). Required arguments

are as follows:

in: `void *value`

Value. The value is written to the CDF from memory address `value`.

The required preselected objects/states are the current CDF, its current rVariable, its current record number for rVariables, and its current dimension indices for rVariables.

`rVAR_HYPERDATA_` Writes one or more values to the current rVariable (in the current CDF). The values are written based on the current record number, current record count, current record interval, current dimension indices, current dimension counts, and current dimension intervals for the rVariables (in the current CDF). Required arguments are as follows:

in: `void *buffer`

Values. The values starting at memory address `buffer` are written to the CDF.

The required preselected objects/states are the current CDF, its current rVariable, its current record number, record count, and record interval for rVariables, and its current dimension indices, dimension counts, and dimension intervals for rVariables.

`rVAR_SEQDATA_` Writes one value to the current rVariable (in the current CDF) at the current sequential value for that rVariable. After the write the current sequential value is automatically incremented to the next value (crossing a record boundary if necessary). If the current sequential value is past the last record for the rVariable, the rVariable is extended as necessary. Required arguments are as follows:

in: `void *value`

Value. The value is written to the CDF from memory address `value`.

The required preselected objects/states are the current CDF, its current rVariable, and the current sequential value for the rVariable. Note that the current sequential value for an rVariable increments automatically as values are written.

`rVARs_RECDATA_` Writes full-physical records to one or more rVariables (in the current CDF). The full-physical records are written at the current record number for rVariables. This operation does not affect the current rVariable (in the current CDF). Required arguments are as follows:

in: `long numVars`

The number of rVariables to which to write. This must be at least one (1).

in: `long varNums[]`

The rVariables to which to write. This array, whose size is

determined by the value of `numVars`, contains rVariable numbers. The rVariable numbers can be listed in any order.

in: `void *buffer`

The buffer of full-physical rVariable records to be written. The order of the full-physical rVariable records in this buffer must agree with the rVariable numbers listed in `varNums`, and this buffer must be contiguous — there can be no spacing between full-physical rVariable records. Be careful if using C `struct` objects to store multiple full-physical rVariable records. C compilers on some operating systems will pad between the elements of a `struct` in order to prevent memory alignment errors (i.e., the elements of a `sturct` may not be contiguous). See Section 2.3.15 for more details on how to create this buffer.

The required preselected objects/states are the current CDF and its current record number for rVariables.

`zVAR_NAME_` Renames the current zVariable (in the current CDF). A variable (rVariable or zVariable) with the same name must not already exist in the CDF. Required arguments are as follows:

in: `char *varName`

New name of the zVariable. This may consist of at most `CDF_VAR_NAME_LEN` characters (excluding the `NUL` terminator).

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DATASPEC_` Respecifies the data specification (data type and number of elements) of the current zVariable (in the current CDF). A zVariable's data specification may not be changed if the new data specification is not equivalent to the old data specification and any values (including the pad value) have been written. Data specifications are considered equivalent if the data types are equivalent (see Section 2.2.9) and the number of elements are the same. Required arguments are as follows:

in: `long dataType`

New data type. Specify one of the data types described in Section 4.6.4.

in: `long numElements`

Number of elements of the data type at each value. For character string data types (`CDF_CHAR` and `CDF_UCHAR`), this is the number of characters in each string (an array of characters). A string exists at each value. For the non-character string data types this must be one (1) — arrays of values are not allowed for non-character string data types.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_RECVARY_` Respecifies the record variance of the current zVariable (in the current CDF). A zVariable's record variance may not be changed if any values have been written (except for an explicit pad value — it may have been written). Required arguments are as follows:

in: `long recVary`

New record variance. Specify one of the variances described in Section 4.6.8.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DIMVARYS_` Respecifies the dimension variances of the current zVariable (in the current CDF). A zVariable's dimension variances may not be changed if any values have been written (except for an explicit pad value — it may have been written). For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

in: `long dimVarys[]`

New dimension variances. `dimVarys` is a 1-dimensional array containing one element per dimension. Each element of `dimVarys` specifies the corresponding dimension variance. For each dimension specify one of the variances described in Section 4.6.8.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_PADVALUE_` Specifies the pad value for the current zVariable (in the current CDF). A zVariable's pad value may be specified (or respecified) at any time without affecting already written values (including where pad values were used). Section 2.3.16 describes variable pad values. Required arguments are as follows:

in: `void *value`

Pad value. The pad value is written to the CDF from memory address `value`.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_INITIALRECS_` Specifies the number of records to initially write to the current zVariable (in the current CDF). This may be specified only once per zVariable and before any other records have been written to that zVariable. If a pad value has not yet been specified, the default is used (see Section 2.3.16). If a pad value has been explicitly specified, that value is written to the records. Section 2.3.7 describes initial records. Required arguments are as follows:

in: `long NinitialRecs`

Number of records to write.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_EXTENDRECS_` Specifies the number of physical records to extend the current zVariable (in the current CDF) whenever that zVariable needs to be extended. Section 2.3.7 describes extend records. **NOTE:** The number of extend records has no effect for NRV zVariables or multi-file CDFs. Required arguments are as follows:

in: `long NextendRecs`

Number of records to extend. A value of zero (0) indicates that the default number of records should be used.

The required preselected objects/states are the current CDF and its current zVariable.

`zVAR_DATA_` Writes one value to the current zVariable (in the current CDF). The value is written at the current record number and current dimension indices for that zVariable (in the current CDF). Required arguments are as follows:

in: `void *value`

Value. The value is written to the CDF from memory address `value`.

The required preselected objects/states are the current CDF, its current zVariable, the current record number for the zVariable, and the current dimension indices for the zVariable.

`zVAR_HYPERDATA_` Writes one or more values to the current zVariable (in the current CDF). The values are written based on the current record number, current record count, current record interval, current dimension indices, current dimension counts, and current dimension intervals for that zVariable (in the current CDF). Required arguments are as follows:

in: `void *buffer`

Values. The values starting at memory address `buffer` are written to the CDF.

The required preselected objects/states are the current CDF, its current zVariable, the current record number, record count, and record interval for the zVariable, and the current dimension indices, dimension counts, and dimension intervals for the zVariable.

`zVAR_SEQDATA_` Writes one value to the current zVariable (in the current CDF) at the current sequential value for that zVariable. After the write the current sequential value is automatically incremented to the next value (crossing a record boundary if necessary). If the current sequential value is past the last record for the zVariable, the zVariable is extended as necessary. Required arguments are as follows:

in: `void *value`

Value. The value is written to the CDF from memory address `value`.

The required preselected objects/states are the current CDF, its current zVariable, and the current sequential value for the zVariable. Note that the current sequential value for a zVariable increments automatically as values are written.

`zVARs_RECDATA_` Writes full-physical records to one or more zVariables (in the current CDF). The full-physical record for a particular zVariable is written at the current record number for that zVariable. (The record numbers do not have to be the same but in most cases probably will be.) This operation does not affect the current zVariable (in the current CDF). Required arguments are as follows:

in: `long numVars`

The number of zVariables to which to write. This must be at least one (1).

in: `long varNums[]`

The zVariables to which to write. This array, whose size is determined by the value of `numVars`, contains zVariable numbers. The zVariable numbers can be listed in any order.

in: `void *buffer`

The buffer of full-physical zVariable records to be written. The order of the full-physical zVariable records in this buffer must agree with the zVariable numbers listed in `varNums`, and this buffer must be contiguous — there can be no spacing between full-physical zVariable records. Be careful if using C `struct` objects to store multiple full-physical zVariable records. C compilers on some operating systems will pad between the elements of a `struct` in order to prevent memory alignment errors (i.e., the elements of a `struct` may not be contiguous). See Section 2.3.15 for more details on how to create this buffer.

The required preselected objects/states are the current CDF and the current record number for each of the zVariables specified. A convenience operation exists, `<SELECT_,zVARs_RECNUMBER_>`, that allows the current record number for each zVariable to be selected at one time (as opposed to selecting the current record numbers one at a time using `<SELECT_,zVAR_RECNUMBER_>`).

`ATTR_SCOPE_` Respecifies the scope for the current attribute (in the current CDF). Required arguments are as follows:

in: `long scope`

New attribute scope. Specify one of the scopes described in Section 4.6.9.

The required preselected objects/states are the current CDF and its current attribute.

`ATTR_NAME_` Renames the current attribute (in the current CDF). An attribute with the same name must not already exist in the CDF. Required arguments are as follows:

in: `char *attrName`

New attribute name. This may be at most `CDF_ATTR_NAME_LEN` characters (excluding the `NUL` terminator).

The required preselected objects/states are the current CDF and its current attribute.

`gENTRY_DATASPEC_` Modifies the data specification (data type and number of elements) of the gEntry at the current gEntry number of the current attribute (in the current CDF). The new and old data types must be equivalent, and the number of elements must not be changed. Equivalent data types are described in Section 2.2.9. **NOTE:** This operation should only be performed on gAttributes. Required arguments are as follows:

in: `long dataType`

New data type of the gEntry. Specify one of the data types described in Section 4.6.4.

in: `long numElements`

Number of elements of the data type.

The required preselected objects/states are the current CDF, its current attribute, and its current gEntry number.

`rENTRY_DATASPEC_` Modifies the data specification (data type and number of elements) of the rEntry at the current rEntry number of the current attribute (in the current CDF). The new and old data types must be equivalent, and the number of elements must not be changed. Equivalent data types are described in Section 2.2.9. **NOTE:** This operation should only be performed on vAttributes. Required arguments are as follows:

in: `long dataType`

New data type of the rEntry. Specify one of the data types described in Section 4.6.4.

in: `long numElements`

Number of elements of the data type.

The required preselected objects/states are the current CDF, its current attribute, and its current rEntry number.

`zENTRY_DATASPEC_` Modifies the data specification (data type and number of elements) of the zEntry at the current zEntry number of the current attribute (in the current CDF). The new and old data types must be equivalent, and the number of elements must not be changed. Equivalent data types are described in Section 2.2.9. **NOTE:** This operation should only be performed on vAttributes. Required arguments are as follows: