in: `long dataType`

New data type of the zEntry. 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 zEntry number.

`gENTRY_DATA_` Writes a gEntry to the current attribute at the current gEntry number (in the current CDF). An existing gEntry may be overwritten with a new gEntry having the same data specification (data type and number of elements) or a different data specification. **NOTE:** This operation should only be performed on gAttributes. Required arguments are as follows:

in: `long dataType`

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. This may be greater than one (1) for any of the supported data types. For character string data types (`CDF_CHAR` and `CDF_UCHAR`) this is the number of characters in the string (an array of characters). For all other data types this is the number of elements in an array of that data type.

in: `void *value`

Value(s). The entry value is written to the CDF from memory address `value`.

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

`rENTRY_DATA_` Writes an rEntry to the current attribute at the current rEntry number (in the current CDF). An existing rEntry may be overwritten with a new rEntry having the same data specification (data type and number of elements) or a different data specification. **NOTE:** This operation should only be performed on vAttributes. Required arguments are as follows:

in: `long dataType`

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. This may be greater than one (1) for any of the supported data types. For character string data types (`CDF_CHAR` and `CDF_UCHAR`) this is the number of characters in the string (an array of characters). For all other data types this is the number of elements in an

array of that data type.

in: `void *value`

Value(s). The entry value is written to the CDF from memory address `value`.

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

`zENTRY_DATA_` Writes a zEntry to the current attribute at the current zEntry number (in the current CDF). An existing zEntry may be overwritten with a new zEntry having the same data specification (data type and number of elements) or a different data specification. **NOTE:** This operation should only be performed on vAttributes. Required arguments are as follows:

in: `long dataType`

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

in: `long numElements`

Number of elements of the data type. This may be greater than one (1) for any of the supported data types. For character string data types (`CDF_CHAR` and `CDF_UCHAR`) this is the number of characters in the string (an array of characters). For all other data types this is the number of elements in an array of that data type.

in: `void *value`

Value(s). The entry value is written to the CDF from memory address `value`.

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

`GET_` The following items may be read (gotten):

`rVARs_NUMDIMS_` Inquires the number of dimensions for the rVariables in the current CDF. Required arguments are as follows:

out: `long *numDims`

Number of dimensions.

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

`rVARs_DIMSIZES_` Inquires the size of each dimension for the rVariables in the current CDF. For 0-dimensional rVariables this operation is not applicable. Required arguments are as follows:

out: `long dimSizes[]`

Dimension sizes. `dimSizes` is a 1-dimensional array containing one element per dimension. Each element of `dimSizes` receives the corresponding dimension size.

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

`zVAR_NUMDIMS_` Inquires the number of dimensions for the current zVariable in the current CDF. Required arguments are as follows:

out: `long *numDims`

Number of dimensions.

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

`zVAR_DIMSIZES_` Inquires the size of each dimension for the current zVariable in the current CDF. For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

out: `long dimSizes[]`

Dimension sizes. `dimSizes` is a 1-dimensional array containing one element per dimension. Each element of `dimSizes` receives the corresponding dimension size.

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

`CDF_ENCODING_` Inquires the data encoding of the current CDF. Required arguments are as follows:

out: `long *encoding`

Data encoding. The encodings are described in Section 4.6.5.

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

`CDF_MAJORITY_` Inquires the variable majority of the current CDF. Required arguments are as follows:

out: `long *majority`

Variable majority. The majorities are described in Section 4.6.7.

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

`CDF_FORMAT_` Inquires the format of the current CDF. Required arguments are as follows:

out: `long *format`

CDF format. The formats are described in Section 4.6.3.

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

`CDF_COPYRIGHT_` Reads the copyright notice for the CDF library that created the current CDF. Required arguments are as follows:

out: `char *copyRight`

CDF copyright text. This character string must be large enough to hold `CDF_COPYRIGHT_LEN + 1` characters (including the `NUL` terminator).

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

LIB_COPYRIGHT_ Reads the copyright notice of the CDF library being used. Required arguments are as follows:

out: `char *copyRight`

CDF library copyright text. This character string must be large enough hold `CDF_COPYRIGHT_LEN + 1` characters (including the `NUL` terminator).

There are no required preselected objects/states.

CDF_NUMrVARS_ Inquires the number of rVariables in the current CDF. Required arguments are as follows:

out: `long *numVars`

Number of rVariables.

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

CDF_NUMzVARS_ Inquires the number of zVariables in the current CDF. Required arguments are as follows:

out: `long *numVars`

Number of zVariables.

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

CDF_NUMATTRS_ Inquires the number of attributes in the current CDF. Required arguments are as follows:

out: `long *numAttrs`

Number of attributes.

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

CDF_NUMgATTRS_ Inquires the number of gAttributes in the current CDF. Required arguments are as follows:

out: `long *numAttrs`

Number of gAttributes.

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

CDF_NUMvATTRS_ Inquires the number of vAttributes in the current CDF. Required arguments are as follows:

out: `long *numAttrs`

Number of vAttributes.

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

rVARs_MAXREC_ Inquires the maximum record number of the rVariables in the current CDF. Note that this is not the number of records but rather the maximum record number (which is one less than the number of records).

A value of negative one (-1) indicates that the rVariables contain no records. The maximum record number for an individual rVariable may be inquired using <GET_, rVAR_MAXREC_>. Required arguments are as follows:

out: long *maxRec

Maximum record number.

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

zVARs_MAXREC_ Inquires the maximum record number of the zVariables in the current CDF. Note that this is not the number of records but rather the maximum record number (which is one less than the number of records). A value of negative one (-1) indicates that the zVariables contain no records. The maximum record number for an individual zVariable may be inquired using <GET_, zVAR_MAXREC_>. Required arguments are as follows:

out: long *maxRec

Maximum record number.

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

CDF_VERSION_ Inquires the version number of the CDF library that created the current CDF. Required arguments are as follows:

out: long *version

Version number.

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

CDF_RELEASE_ Inquires the release number of the CDF library that created the current CDF. Required arguments are as follows:

out: long *release

Release number.

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

CDF_INCREMENT_ Inquires the incremental number of the CDF library that created the current CDF. Required arguments are as follows:

out: long *increment

Incremental number.

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

LIB_VERSION_ Inquires the version number of the CDF library being used. Required arguments are as follows:

out: long *version

Version number.

There are no required preselected objects/states.

`LIB_RELEASE_` Inquires the release number of the CDF library being used. Required arguments are as follows:

out: `long *release`

Release number.

There are no required preselected objects/states.

`LIB_INCREMENT_` Inquires the incremental number of the CDF library being used. Required arguments are as follows:

out: `long *increment`

Incremental number.

There are no required preselected objects/states.

`LIB_subINCREMENT_` Inquires the subincremental character of the CDF library being used. Required arguments are as follows:

out: `char *subincrement`

Subincremental character.

There are no required preselected objects/states.

`rVAR_NAME_` Inquires the name of the current rVariable (in the current CDF). Required arguments are as follows:

out: `char *varName`

Name of the rVariable. This character string must be large enough to hold `CDF_VAR_NAME_LEN + 1` characters (including the `NUL` terminator).

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

`rVAR_DATATYPE_` Inquires the data type of the current rVariable (in the current CDF). Required arguments are as follows:

out: `long *dataType`

Data type. The data types are described in Section 4.6.4.

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

`rVAR_NUMELEMS_` Inquires the number of elements (of the data type) for the current rVariable (in the current CDF). Required arguments are as follows:

out: `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 the string. (Each value consists of the entire string.) For all other data types this will always be

one (1) — multiple elements at each value are not allowed for non-character string data types.

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

`rVAR_RECVARY_` Inquires the record variance of the current rVariable (in the current CDF). Required arguments are as follows:

out: `long *recVary`

Record variance. The variances are described in Section 4.6.8.

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

`rVAR_DIMVARYS_` Inquires the dimension variances of the current rVariable (in the current CDF). For 0-dimensional rVariables this operation is not applicable. Required arguments are as follows:

out: `long dimVarys[]`

Dimension variances. `dimVarys` is a 1-dimensional array containing one element per dimension. Each element of `dimVarys` receives the corresponding dimension variance. The variances are described in Section 4.6.8.

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

`rVAR_NUMBER_` Gets the number of the named rVariable (in the current CDF). Note that this operation does not select the current rVariable. Required arguments are as follows:

in: `char *varName`

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

out: `long *varNum`

The rVariable number.

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

`rVAR_PADVALUE_` Inquires the pad value of the current rVariable (in the current CDF). Required arguments are as follows:

out: `void *value`

Pad value. This buffer must be large enough to hold the pad value. The pad value is read from the CDF and placed in memory at address `value`.

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

`rVAR_EXTENDRECS_` Inquires the number of extend records for the current rVariable (in the current CDF). Extend records are described in Section 2.3.7. Required

arguments are as follows:

out: `long *NextendRecs`

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

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

`rVAR_MAXREC_` Inquires the maximum record number for the current rVariable (in the current CDF). For rVariables with a record variance of `NOVARY`, this will be at most zero (0). A value of negative one (-1) indicates that no records have been written. Required arguments are as follows:

out: `long *varMaxRec`

Maximum record number.

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

`rVAR_MAXallocREC_` Inquires the maximum record number allocated for the current rVariable (in the current CDF). This only has significance for rVariables that are not NRV and are in single-file CDFs. Required arguments are as follows:

out: `long *varMaxRecAlloc`

Maximum record number allocated.

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

`rVAR_nINDEXRECORDS_` Inquires the number of index records for the current rVariable (in the current CDF). This only has significance for rVariables that are in single-file CDFs. Section 2.2.6 describes the indexing scheme used for variable records in a single-file CDF. Required arguments are as follows:

out: `long *numRecords`

Number of index records.

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

`rVAR_nINDEXENTRIES_` Inquires the number of index entries for the current rVariable (in the current CDF). This only has significance for rVariables that are in single-file CDFs. Section 2.2.6 describes the indexing scheme used for variable records in a single-file CDF. Required arguments are as follows:

out: `long *numEntries`

Number of index entries.

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

`rVAR_DATA_`

Reads a value from the current rVariable (in the current CDF). The value is read at the current record number and current dimension indices for the rVariables (in the current CDF). Required arguments are as follows:

out: `void *value`

Value. This buffer must be large enough to hold the value. The value is read from the CDF and placed into memory at 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_`

Reads one or more values from the current rVariable (in the current CDF). The values are read 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:

out: `void *buffer`

Values. This buffer must be large enough to hold the values. The values are read from the CDF and placed into memory starting at address `buffer`.

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_`

Reads one value from the current rVariable (in the current CDF) at the current sequential value for that rVariable. After the read the current sequential value is automatically incremented to the next value (crossing a record boundary if necessary). An error is returned if the current sequential value is past the last record for the rVariable. Required arguments are as follows:

out: `void *value`

Value. This buffer must be large enough to hold the value. The value is read from the CDF and placed into memory at 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 read.

`rVARs_RECDATA_`

Reads full-physical records from one or more rVariables (in the current CDF). The full-physical records are read 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 from which to read. This must be at least one (1).

in: `long varNums[]`

The rVariables from which to read. 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 into which the full-physical rVariable records being read are to be placed. This buffer must be large enough to hold the full-physical records. The order of the full-physical rVariable records in this buffer will correspond to the rVariable numbers listed in `varNums`, and this buffer will be contiguous — there will be no spacing between full-physical rVariable records. Be careful if using C `struct` objects to receive 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 `struct` may not be contiguous). See Section 2.3.15 for more details on how to allocate this buffer.

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

`zVAR_NAME_` Inquires the name of the current zVariable (in the current CDF). Required arguments are as follows:

out: `char *varName`

Name of the zVariable. This character string must be large enough to hold `CDF_VAR_NAME_LEN + 1` characters (including the `NUL` terminator).

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

`zVAR_DATATYPE_` Inquires the data type of the current zVariable (in the current CDF). Required arguments are as follows:

out: `long *dataType`

Data type. The data types are described in Section 4.6.4.

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

`zVAR_NUMELEMS_` Inquires the number of elements (of the data type) for the current zVariable (in the current CDF). Required arguments are as follows:

out: `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 the string. (Each value consists of the entire string.) For all other data types this will always be one (1) — multiple elements at each value are not allowed for non-character string data types.

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

`zVAR_RECVARY_` Inquires the record variance of the current zVariable (in the current CDF). Required arguments are as follows:

out: `long *recVary`

Record variance. The variances are described in Section 4.6.8.

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

`zVAR_DIMVARYS_` Inquires the dimension variances of the current zVariable (in the current CDF). For 0-dimensional zVariables this operation is not applicable. Required arguments are as follows:

out: `long dimVarys[]`

Dimension variances. `dimVarys` is a 1-dimensional array containing one element per dimension. Each element of `dimVarys` receives the corresponding dimension variance. The variances are described in Section 4.6.8.

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

`zVAR_NUMBER_` Gets the number of the named zVariable (in the current CDF). Note that this operation does not select the current zVariable. Required arguments are as follows:

in: `char *varName`

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

out: `long *varNum`

The zVariable number.

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

`zVAR_PADVALUE_` Inquires the pad value of the current zVariable (in the current CDF). Required arguments are as follows:

out: `void *value`

Pad value. This buffer must be large enough to hold the pad value. The pad value is read from the CDF and placed in memory at address `value`.

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

`zVAR_EXTENDRECS_` Inquires the number of extend records for the current zVariable (in the current CDF). Extend records are described in Section 2.3.7. Required arguments are as follows:

out: `long *NextendRecs`

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

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

`zVAR_MAXREC_` Inquires the maximum record number for the current zVariable (in the current CDF). For zVariables with a record variance of `NOVARY`, this will be at most zero (0). A value of negative one (-1) indicates that no records have been written. Required arguments are as follows:

out: `long *varMaxRec`

Maximum record number.

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

`zVAR_MAXallocREC_` Inquires the maximum record number allocated for the current zVariable (in the current CDF). This only has significance for zVariables that are not NRV and are in single-file CDFs. Required arguments are as follows:

out: `long *varMaxRecAlloc`

Maximum record number allocated.

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

`zVAR_nINDEXRECORDS_` Inquires the number of index records for the current zVariable (in the current CDF). This only has significance for zVariables that are in single-file CDFs. Section 2.2.6 describes the indexing scheme used for variable records in a single-file CDF. Required arguments are as follows:

out: `long *numRecords`

Number of index records.

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

`zVAR_nINDEXENTRIES_` Inquires the number of index entries for the current zVariable (in the current CDF). This only has significance for zVariables that are in single-file CDFs. Section 2.2.6 describes the indexing scheme used for variable records in a single-file CDF. Required arguments are as follows:

out: `long *numEntries`

Number of index entries.

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

`zVAR_DATA_` Reads a value from the current zVariable (in the current CDF). The value is read at the current record number and current dimension indices for that zVariable (in the current CDF). Required arguments are as follows:

out: `void *value`

Value. This buffer must be large enough to hold the value. The value is read from the CDF and placed into memory at 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_` Reads one or more values from the current zVariable (in the current CDF). The values are read 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:

out: `void *buffer`

Values. This buffer must be large enough to hold the values. The values are read from the CDF and placed into memory starting at address `buffer`.

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_` Reads one value from the current zVariable (in the current CDF) at the current sequential value for that zVariable. After the read the current sequential value is automatically incremented to the next value (crossing a record boundary if necessary). An error is returned if the current sequential value is past the last record for the zVariable. Required arguments are as follows:

out: `void *value`

Value. This buffer must be large enough to hold the value. The value is read from the CDF and placed into memory at 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 read.

`zVARs_RECDATA_` Reads full-physical records from one or more zVariables (in the current CDF). The full-physical record for a particular zVariable is read 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 from which to read. This must be at least one (1).

in: `long varNums[]`

The zVariables from which to read. 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 into which the full-physical zVariable records being read are to be placed. This buffer must be large enough to hold the full-physical records. The order of the full-physical zVariable records in this buffer will correspond to the zVariable numbers listed in `varNums`, and this buffer will be contiguous — there will be no spacing between full-physical zVariable records. Be careful if using C `struct` objects to receive 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 allocate 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_` Inquires the scope of the current attribute (in the current CDF). Required arguments are as follows:

out: `long *scope`

Attribute scope. The scopes are described in Section 4.6.9.

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

`ATTR_NAME_` Inquires the name of the current attribute (in the current CDF). Required arguments are as follows:

out: `char *attrName`

Attribute name. This character string must be large enough to hold `CDF_ATTR_NAME_LEN + 1` characters (including the `NUL` terminator).

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

`ATTR_NUMBER_` Gets the number of the named attribute (in the current CDF). Note that this operation does not select the current attribute. Required arguments are as follows:

in: `char *attrName`

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

out: `long *attrNum`

The attribute number.

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

`ATTR_MAXgENTRY_` Inquires the maximum gEntry number used for the current attribute (in the current CDF). This does not necessarily correspond with the number of gEntries for the attribute. **NOTE:** Use this operation with gAttributes only. Required arguments are as follows:

out: `long *maxEntry`

The maximum gEntry number for the attribute.

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

`ATTR_NUMgENTRIES_` Inquires the number of gEntries for the current attribute (in the current CDF). This does not necessarily correspond with the maximum gEntry number used. **NOTE:** Use this operation with gAttributes only. Required arguments are as follows:

out: `long *numEntries`

The number of gEntries for the attribute.

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

`ATTR_MAXrENTRY_` Inquires the maximum rEntry number used for the current attribute (in the current CDF). This does not necessarily correspond with the number of rEntries for the attribute. **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *maxEntry`

The maximum rEntry number for the attribute.

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

`ATTR_NUMrENTRIES_` Inquires the number of rEntries for the current attribute (in the current CDF). This does not necessarily correspond with the maximum rEntry number used. **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *numEntries`

The number of rEntries for the attribute.

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

`ATTR_MAXzENTRY_` Inquires the maximum zEntry number used for the current attribute (in the current CDF). This does not necessarily correspond with the number of zEntries for the attribute. **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *maxEntry`

The maximum zEntry number for the attribute.

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

`ATTR_NUMzENTRIES_` Inquires the number of zEntries for the current attribute (in the current CDF). This does not necessarily correspond with the maximum zEntry number used. **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *numEntries`

The number of zEntries for the attribute.

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

`gENTRY_DATATYPE_` Inquires the data type of the gEntry at the current gEntry number for the current attribute (in the current CDF). **NOTE:** Use this operation with gAttributes only. Required arguments are as follows:

out: `long *dataType`

Data type. The data types are described in Section 4.6.4.

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

`gENTRY_NUMELEMS_` Inquires the number of elements (of the data type) of the gEntry at the current gEntry number for the current attribute (in the current CDF). **NOTE:** Use this operation with gAttributes only. Required arguments are as follows:

out: `long *numElements`

Number of elements of the data type. For character string data types (`CDF_CHAR` and `CDF_UCHAR`) this is the number of characters in the string (an array of characters). For all other data types this is the number of elements in an array of that data type.

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

`gENTRY_DATA_` Reads the gEntry data value from the current attribute at the current gEntry number (in the current CDF). **NOTE:** Use this operation with gAttributes only. Required arguments are as follows:

out: `void *value`

Value. This buffer must be large to hold the value. The value is read from the CDF and placed into memory at address `value`.

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

`rENTRY_DATATYPE_` Inquires the data type of the rEntry at the current rEntry number for the current attribute (in the current CDF). **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *dataType`

Data type. The data types are described in Section 4.6.4.

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

`rENTRY_NUMELEMS_` Inquires the number of elements (of the data type) of the rEntry at the current rEntry number for the current attribute (in the current CDF). **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *numElements`

Number of elements of the data type. For character string data types (`CDF_CHAR` and `CDF_UCHAR`) this is the number of characters in the string (an array of characters). For all other data types this is the number of elements in an array of that data type.

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

`rENTRY_DATA_` Reads the rEntry data value from the current attribute at the current rEntry number (in the current CDF). **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `void *value`

Value. This buffer must be large to hold the value. The value is read from the CDF and placed into memory at address `value`.

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

`zENTRY_DATATYPE_` Inquires the data type of the zEntry at the current zEntry number for the current attribute (in the current CDF). **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *dataType`

Data type. The data types are described in Section 4.6.4.

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

`zENTRY_NUMELEMS_` Inquires the number of elements (of the data type) of the zEntry at the current zEntry number for the current attribute (in the current CDF). **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `long *numElements`

Number of elements of the data type. For character string data types (`CDF_CHAR` and `CDF_UCHAR`) this is the number of characters in the string (an array of characters). For all other data types this is the number of elements in an array of that data type.

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

`zENTRY_DATA_` Reads the zEntry data value from the current attribute at the current zEntry number (in the current CDF). **NOTE:** Use this operation with vAttributes only. Required arguments are as follows:

out: `void *value`

Value. This buffer must be large to hold the value. The value is read from the CDF and placed into memory at address `value`.

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

`STATUS_TEXT_` Inquires the explanation text for the current status code. Note that the current status code is NOT the status from the last operation performed. Required arguments are as follows:

out: `char *text`

Text explaining the status code. This string must be large enough to hold `CDF_STATUSTEXT_LEN + 1` characters.

The only required preselected object/state is the current status code.

More Examples

Several more examples of the use of `CDFlib` follow. In each example it is assumed that the current CDF has already been selected (either implicitly by creating/opening the CDF or explicitly with `<SELECT_,CDF_>`).

rVariable Creation. In this example an rVariable will be created with a pad value being specified; initial records will be written; and the number of records to extend the rVariable when necessary will be specified.

Note that the pad value was specified before the initial records. This results in the specified pad value being written. Had the pad value not been specified first, the initial records would have been written with the default pad value. It is assumed that the current CDF has already been selected.

```
.
.
#include "cdf.h"
.
.
CDFstatus   status;                     /* Status returned from CDF
                                           library. */
long        dimVarys[2];                /* Dimension variances. */
long        varNum;                     /* rVariable number. */
float       padValue = -999.9;          /* Pad value. */
.
.
dimVarys[0] = VARY;
dimVarys[1] = VARY;
status = CDFlib (CREATE_, rVAR_, "HUMIDITY", CDF_REAL4, 1, VARY,
                                 dimVarys, &varNum,
                 PUT_, rVAR_PADVALUE_, &padValue,
                       rVAR_INITIALRECS_, (long) 500,
                       rVAR_EXTENDRECS_, (long) 50,
                 NULL_);
if (status != CDF_OK) UserStatusHandler (status);
.
.
```

zVariable Creation (Character String Data Type). In this example a zVariable with a character string data type will be created with a pad value being specified. It is assumed that the current CDF has already been selected.

```
.
.
#include "cdf.h"
.
.
CDFstatus   status;                     /* Status returned from CDF
                                           library. */
long        dimVarys[1];                /* Dimension variances. */
long        varNum;                     /* zVariable number. */
long        numDims = 1;                /* Number of dimensions. */
static long dimSizes[1] = { 20 };       /* Dimension sizes. */
long        numElems = 10;              /* Number of elements (characters
                                           in this case). */
static char padValue = "**********";    /* Pad value. */
.
.
dimVarys[0] = VARY;
```

```
status = CDFlib (CREATE_, zVAR_, "Station", CDF_CHAR, numElems, numDims,
                                 dimSizes, NOVARY, dimVarys, &varNum,
                 PUT_, zVAR_PADVALUE_, padValue,
                 NULL_);
if (status != CDF_OK) UserStatusHandler (status);
.
.
```

Hyper Read with Subsampling. In this example an rVariable will be subsampled in a CDF whose rVariables are 2-dimensional and have dimension sizes `[100,200]`. The CDF is row major, and the data type of the rVariable is `CDF_UINT2`. It is assumed that the current CDF has already been selected.

```
.
.
#include "cdf.h"
.
.
CDFstatus      status;               /* Status returned from CDF library. */
unsigned short values[50][100];      /* Buffer to receive values. */
long           recCount = 1;         /* Record count, one record per hyper
                                        get. */
long           recInterval = 1;      /* Record interval, set to one to
                                        indicate contiguous records (really
                                        meaningless since record count is
                                        one). */
static long    indices[2] = {0,0};   /* Dimension indices, start each read
                                        at 0,0 of the array. */
static long    counts[2] = {50,100}; /* Dimension counts, half of the
                                        values along each dimension will
                                        be read. */
static long    intervals[2] = {2,2}; /* Dimension intervals, every other
                                        value along each dimension will be
                                        read. */
long           recNum;               /* Record number. */
long           maxRec;               /* Maximum rVariable record
                                        number in the CDF - this was
                                        determined with a call to
                                        CDFinquire. */
.
.
status = CDFlib (SELECT_, rVAR_NAME_, "BRIGHTNESS",
                          rVARs_RECCOUNT_, recCount,
                          rVARs_RECINTERVAL_, recInterval,
                          rVARs_DIMINDICES_, indices,
                          rVARs_DIMCOUNTS_, counts,
                          rVARs_DIMINTERVALS_, intervals,
                 NULL_);
if (status != CDF_OK) UserStatusHandler (status);
```