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: `CHARACTER attr_name*(*)`

The attribute name. This may be at most `CDF_ATTR_NAME_LEN` characters.

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: `INTEGER*4 entry_num`

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: `INTEGER*4 entry_num`

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: `INTEGER*4 entry_num`

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: `INTEGER*4 entry_num`

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: `INTEGER*4 entry_num`

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: `INTEGER*4 entry_num`

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: `INTEGER*4 encoding`

New data encoding. Specify one of the encodings described in Section 5.7.4.

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: `INTEGER*4 majority`

New variable majority. Specify one of the majorities described in Section 5.7.6.

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: `INTEGER*4 format`

New CDF format. Specify one of the formats described in Section 5.7.2.

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: `CHARACTER var_name*(*)`

New name of the rVariable. This may consist of at most `CDF_VAR_NAME_LEN` characters. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `var_name` MUST be of type `CHARACTER`.

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: `INTEGER*4 data_type`

New data type. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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. (A string exists at each value.) For non-character string data types this must be one (1) — multiple elements 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: `INTEGER*4 rec_vary`

New record variance. Specify one of the variances described in Section 5.7.7.

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: `INTEGER*4 dim_varys(*)`

New dimension variances. `dim_varys` is a 1-dimensional array containing one element per dimension. Each element of `dim_varys` specifies the corresponding dimension variance. For each dimension specify one of the variances described in Section 5.7.7.

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: `<type> value`

Pad value. `<type>` depends on the data type of the rVariable. The pad value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 n_initial_recs`

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: `INTEGER*4 n_extend_recs`

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: `<type> value`

Value. `<type>` depends on the data type of the rVariable. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `<type> buffer`

Values. `<type>` depends on the data type of the rVariable. The values starting at memory address `buffer` are written to the CDF. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `<type> value`

Value. `<type>` depends on the data type of the rVariable. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 num_vars`

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

in: `INTEGER*4 var_nums(*)`

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

in: `<type> buffer`

The buffer of full-physical rVariable records to be written. `<type>` must be a Fortran variable that will be passed by reference and cannot be of type `CHARACTER`. (The CDF library is expecting an address at which to get the full-physical records being 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 Fortran STRUCTUREs to store multiple full-physical rVariable records. Fortran compilers on some operating systems will pad between the elements of a STRUCTURE in order to prevent memory alignment errors (i.e., the elements of a STRUCTURE 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: `CHARACTER var_name*(*)`

New name of the zVariable. This may consist of at most `CDF_VAR_NAME_LEN` characters. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `var_name` MUST be of type `CHARACTER`.

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 spec-

ification 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: `INTEGER*4 data_type`

New data type. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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. (A string exists at each value.) For non-character string data types this must be one (1) — multiple elements 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: `INTEGER*4 rec_vary`

New record variance. Specify one of the variances described in Section 5.7.7.

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: `INTEGER*4 dim_varys(*)`

New dimension variances. `dim_varys` is a 1-dimensional array containing one element per dimension. Each element of `dim_varys` specifies the corresponding dimension variance. For each dimension specify one of the variances described in Section 5.7.7.

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: `<type> value`

Pad value. `<type>` depends on the data type of the zVariable. The pad value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the zVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 n_initial_recs`

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: `INTEGER*4 n_extend_recs`

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: `<type> value`

Value. `<type>` depends on the data type of the zVariable. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the zVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `<type> buffer`

Values. `<type>` depends on the data type of the zVariable. The values starting at memory address `buffer` are written to the CDF. **NOTE:** On UNIX systems, if the zVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `<type> value`

Value. `<type>` depends on the data type of the zVariable. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the zVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 num_vars`

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

in: `INTEGER*4 var_nums(*)`

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

determined by the value of num_vars, contains zVariable numbers. The zVariable numbers can be listed in any order.

in: <type> buffer

The buffer of full-physical zVariable records to be written. <type> must be a Fortran variable that will be passed by reference and cannot be of type CHARACTER. (The CDF library is expecting an address at which to get the full-physical records being 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 Fortran STRUCTUREs to store multiple full-physical zVariable records. Fortran compilers on some operating systems will pad between the elements of a STRUCTURE in order to prevent memory alignment errors (i.e., the elements of a STRUCTURE 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: INTEGER*4 scope

New attribute scope. Specify one of the scopes described in Section 5.7.8.

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: CHARACTER attr_name*(*)

New attribute name. This may be at most CDF_ATTR_NAME_LEN characters. **NOTE:** On UNIX systems, for the proper operation of CDF_lib, attr_name MUST be of type CHARACTER.

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: `INTEGER*4 data_type`

New data type of the gEntry. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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: `INTEGER*4 data_type`

New data type of the rEntry. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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:

in: `INTEGER*4 data_type`

New data type of the zEntry. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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: `INTEGER*4 data_type`

Data type of the gEntry. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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. For all other data types this is the number of elements in an array of that data type.

in: `<type> value`

Value(s). `<type>` depends on the data type of the gEntry. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the gEntry has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 data_type`

Data type of the rEntry. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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. For all other data types this is the number of elements in an array of that data type.

in: `<type> value`

Value(s). `<type>` depends on the data type of the rEntry. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the rEntry has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 data_type`

Data type of the zEntry. Specify one of the data types described in Section 5.7.3.

in: `INTEGER*4 num_elements`

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. For all other data types this is the number of elements in an array of that data type.

in: `<type> value`

Value(s). `<type>` depends on the data type of the zEntry. The value is written to the CDF from memory address `value`. **NOTE:** On UNIX systems, if the zEntry has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 num_dims`

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: `INTEGER*4 dim_sizes(*)`

Dimension sizes. `dim_sizes` is a 1-dimensional array containing one element per rVariable dimension. Each element of `dim_sizes` 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: `INTEGER*4 num_dims`

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: `INTEGER*4 dim_sizes(*)`

Dimension sizes. `dim_sizes` is a 1-dimensional array containing one element per dimension. Each element of `dim_sizes` 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: `INTEGER*4 encoding`

Data encoding. The encodings are described in Section 5.7.4.

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: `INTEGER*4 majority`

Variable majority. The majorities are described in Section 5.7.6.

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: `INTEGER*4 format`

CDF format. The formats are described in Section 5.7.2.

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: `CHARACTER copy_right*(CDF_COPYRIGHT_LEN)`

CDF copyright text. This character string must be large enough hold `CDF_COPYRIGHT_LEN` characters and will be blank padded if necessary. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `copy_right` MUST be of type `CHARACTER`.

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

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

out: `CHARACTER copy_right*(CDF_COPYRIGHT_LEN)`

CDF library copyright text. This character string must be large enough hold `CDF_COPYRIGHT_LEN` characters and will be

blank padded if necessary. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `copy_right` MUST be of type `CHARACTER`.

There are no required preselected objects/states.

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

out: `INTEGER*4 num_vars`

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: `INTEGER*4 num_vars`

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: `INTEGER*4 num_attrs`

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: `INTEGER*4 num_attrs`

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: `INTEGER*4 num_attrs`

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. A value of zero (0) indicates that no records have been written to the rVariables in the CDF. The maximum record number for an individual rVariable may be inquired using `<GET_,rVAR_MAXREC_>`. Required arguments are as follows:

out: `INTEGER*4 max_rec`

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. A value of zero (0) indicates that no records have been written to the zVariables in the CDF. The maximum record number for an individual zVariable may be inquired using <GET_,zVAR_MAXREC_>. Required arguments are as follows:

out: INTEGER*4 max_rec

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: INTEGER*4 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: INTEGER*4 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: INTEGER*4 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: INTEGER*4 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: INTEGER*4 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: `INTEGER*4 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: `CHARACTER*1 subincrement`

Subincremental character. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `subincrement` MUST be of type `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: `CHARACTER var_name*(CDF_VAR_NAME_LEN)`

Name of the rVariable. This character string must be large enough to hold `CDF_VAR_NAME_LEN` characters and will be blank padded if necessary. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `var_name` MUST be of type `CHARACTER`.

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: `INTEGER*4 data_type`

Data type. The data types are described in Section 5.7.3.

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: `INTEGER*4 num_elements`

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: `INTEGER*4 rec_vary`

Record variance. The variances are described in Section 5.7.7.

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: `INTEGER*4 dim_varys(*)`

Dimension variances. `dim_varys` is a 1-dimensional array containing one element per rVariable dimension. Each element of `dim_varys` receives the corresponding dimension variance. The variances are described in Section 5.7.7.

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: `CHARACTER var_name*(*)`

The rVariable name. This may be at most `CDF_VAR_NAME_LEN` characters. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `var_name` MUST be of type `CHARACTER`.

out: `INTEGER*4 var_num`

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: `<type> value`

Pad value. `<type>` depends on the data type of the rVariable. The pad value is read from the CDF and placed into memory at address `value`. This buffer must be large enough to hold the pad value. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `INTEGER*4 n_extend_recs`

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 one (1). A value of zero (0) indicates that no records have been written. Required arguments are as follows:

out: `INTEGER*4 var_max_rec`

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: `INTEGER*4 var_max_rec_alloc`

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 *num_records`

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 *num_entries`

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: `<type> value`

Value. `<type>` depends on the data type of the rVariable. This buffer must be large enough to hold the value. The value is read from the CDF and placed into memory at address `value`. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `<type> buffer`

Value(s). `<type>` depends on the data type of the rVariable. The values are read from the CDF and placed into memory starting at address `buffer`. This buffer must be large enough to hold the values. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

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: `<type> value`

Value. `<type>` depends on the data type of the rVariable. The value is read from the CDF and placed into memory at address `value`. This buffer must be large enough to hold the value. **NOTE:** On UNIX systems, if the rVariable has a character string data type, then `<type>` MUST be `CHARACTER` to ensure the proper operation of `CDF_lib`.

The required preselected objects/states are the current CDF, its current rVariable, and the current sequential value for the rVariable. Note