for that rVariable. When an rVariable is created (or for each rVariable in a CDF being opened), the current sequential value is set to the first physical value (even if no physical values exist yet). It may then be explicitly selected using the <SELECT_,rVAR_SEQPOS_> operation. Note that a current sequential value is maintained for each rVariable in a CDF.

record number, zVariable (state)

A zVariable read or write operation is always performed at (for single reads and writes) or starting at (for hyper reads and writes) the current record number for the current zVariable in the current CDF. A multiple variable read or write operation is performed at the current record number of each of the zVariables involved. (The record numbers do not have to be the same.) When a zVariable is created (or for each zVariable in a CDF being opened) the current record number for that zVariable is initialized to zero (0). It may then be explicitly selected using the <SELECT_,zVAR_RECNUMBER_> operation (which only affects the current zVariable in the current CDF). Note that a current record number is maintained for each zVariable in a CDF.

record count, zVariable (state)

A zVariable hyper read or write operation is always performed using the current record count for the current zVariable in the current CDF. When a zVariable is created (or for each zVariable in a CDF being opened), the current record count for that zVariable is initialized to one (1). It may then be explicitly selected using the <SELECT_,zVAR_RECCOUNT_> operation (which only affects the current zVariable in the current CDF). Note that a current record count is maintained for each zVariable in a CDF.

record interval, zVariable (state)

A zVariable hyper read or write operation is always performed using the current record interval for the current zVariable in the current CDF. When a zVariable is created (or for each zVariable in a CDF being opened) the current record interval for that zVariable is initialized to one (1). It may then be explicitly selected using the <SELECT_,zVAR_RECINTERVAL_> operation (which only affects the current zVariable in the current CDF). Note that a current record interval is maintained for each zVariable in a CDF.

dimension indices, zVariable (state)

A zVariable read or write operation is always performed at (for single reads and writes) or starting at (for hyper reads and writes) the current dimension indices for the current zVariable in the current CDF. When a zVariable is created (or for each zVariable in a CDF being opened), the current dimension indices for that zVariable are initialized to zeroes (0,0,...). They may then be explicitly selected using the <SELECT_,zVAR_DIMINDICES_> operation (which only affects the current zVariable in the current CDF). Note that current dimension indices are maintained for each zVariable in a CDF. For 0-dimensional zVariables the current dimension indices are not applicable.

dimension counts, zVariable (state)

A zVariable hyper read or write operation is always performed using the current dimension counts for the current zVariable in the current CDF. When a zVariable is created (or for each zVariable in a CDF being opened), the current dimension counts for that zVariable are initialized to the dimension sizes of that zVariable (which specifies the entire array). They may then be explicitly selected using the <SELECT_,zVAR_DIMCOUNTS_> operation (which only affects the current zVariable in the current CDF). Note that current dimension counts are maintained for each zVariable in a CDF. For 0-dimensional zVariables the current dimension counts are not applicable.

dimension intervals, zVariable (state)

A zVariable hyper read or write operation is always performed using the current dimension intervals

for the current zVariable in the current CDF. When a zVariable is created (or for each zVariable in a CDF being opened), the current dimension intervals for that zVariable are initialized to ones (1,1,...). They may then be explicitly selected using the <SELECT_,zVAR_DIMINTERVALS_> operation (which only affects the current zVariable in the current CDF). Note that current dimension intervals are maintained for each zVariable in a CDF. For 0-dimensional zVariables the current dimension intervals are not applicable.

sequential value, zVariable (state)

A zVariable sequential read or write operation is always performed at the current sequential value for that zVariable. When a zVariable is created (or for each zVariable in a CDF being opened) the current sequential value is set to the first physical value (even if no physical values exist yet). It may then be explicitly selected using the <SELECT_,zVAR_SEQPOS_> operation. Note that a current sequential value is maintained for each zVariable in a CDF.

status code (state)

When inquiring the explanation of a CDF status code, the text returned is always for the current status code. One current status code is maintained for the entire CDF library (regardless of the number of open CDFs). The current status code may be selected using the <SELECT_,CDF_STATUS_> operation. There is no default current status code. Note that the current status code is NOT the status code from the last operation performed.[9]

Returned Status

CDF_lib returns a status code of type INTEGER*4. Since more than one operation may be performed with a single call to CDF_lib, the following rules apply.

1. The first error detected aborts the call to CDF_lib, and the corresponding status code is returned.
2. In the absence of any errors, the status code for the last warning detected is returned.
3. In the absence of any errors or warnings, the status code for the last informational condition is returned.
4. In the absence of any errors, warnings, or informational conditions, CDF_OK is returned.

Section 5.9 explains how to interpret status codes. Appendix C lists the possible status codes and the type of each: error, warning, or informational.

Indentation/Style

Indentation should be used to make calls to CDF_lib readable. The following example shows a call to CDF_lib using proper indentation.

```
CALL CDF_lib (CREATE_, CDF_, CDF_path, num_dims, dim_sizes, id,
1             PUT_, CDF_FORMAT_, format,
2                   CDF_MAJORITY_, majority,
3             CREATE_, ATTR_, attr_name, scope, attr_num,
```

[9] The CDF library now maintains the current status code from one call to the next of CDF_lib.

```
4                       rVAR_, var_name, data_type, num_elements,
5                              rec_vary, dim_varys, var_num,
6             NULL_, status)
```

Note that the operations (CREATE_, PUT_, and NULL_) are indented the same and that the items (CDF_, CDF_FORMAT_, CDF_MAJORITY_, ATTR_, and rVAR_) are indented the same under their corresponding operations.

The following example shows the same call to CDF_lib without the proper indentation.

```
CALL CDF_lib (CREATE_, CDF_, CDF_path, num_dims, dim_sizes, id, PUT_,
1             CDF_FORMAT_, format, CDF_MAJORITY_, majority, CREATE_,
2             ATTR_, attr_name, scope, attr_num, rVAR_, var_name, data_type,
3             num_elements, rec_vary, dim_varys, var_num, NULL_, status)
```

The need for proper indentation to ensure the readability of your applications should be obvious.[10]

Syntax

CDF_lib takes a variable number of arguments. There must always be at least one argument (the first operation to perform). The maximum number of arguments is not limited by CDF but rather the Fortran compiler and operating system being used. Under normal circumstances that limit would never be reached (or even approached). Note that a call to CDF_lib with a large number of arguments can always be broken up into two or more calls to CDF_lib with fewer arguments.

The syntax for CDF_lib is as follows:

```
 CALL CDF_lib (op1, item1, arg1, arg2, ...argN,
+                   item2, arg1, arg2, ...argN,
                        .
                        .
+                   itemN, arg1, arg2, ...argN,
+              op2, item1, arg1, arg2, ...argN,
+                   item2, arg1, arg2, ...argN,
                        .
                        .
+                   itemN, arg1, arg2, ...argN,
                .
                .
+              opN, item1, arg1, arg2, ...argN,
+                   item2, arg1, arg2, ...argN,
                        .
                        .
+                   itemN, arg1, arg2, ...argN,
+              NULL_, status)
```

[10] If it isn't, call us — we can help you.

where opx is an operation to perform, itemx is the item on which to perform the operation, and argx is a required argument for the operation. The NULL_ operation must be used to end the call to CDF_lib. The completion status is placed into status.

Operations/Items/Arguments...

The supported operations are as follows:

CREATE_	Used to create an item.
OPEN_	Used to open an item.
DELETE_	Used to delete an item.
CLOSE_	Used to close an item.
SELECT_	Used to select the value of an item.
CONFIRM_	Used to confirm the value of an item.
PUT_	Used to put (write) something to an item.
GET_	Used to get (read) something from an item.

For each operation the supported items, required arguments, and required preselected objects/states are listed in the following sections. The required preselected objects/states are those objects/states that must be selected (typically with the SELECT_ operation) before a particular operation may be performed. Note that some of the required preselected objects/states have default values as described beginning on page 258.

CREATE_ The following items may be created:

CDF_ A new CDF will be created. It is illegal to create a CDF that already exists. The created CDF implicitly becomes the current CDF. Required arguments are as follows:

in: CHARACTER CDF_path*(*)

Pathname of the CDF to be created. (Do not append an extension.) This can be at most CDF_PATHNAME_LEN characters. A CDF pathname may contain disk and directory specifications that conform to the conventions of the operating system being used (including logical names on VMS systems and environment variables on UNIX systems). **NOTE:** On UNIX systems pathnames are case-sensitive. **NOTE:** On UNIX systems, for the proper operation of CDF_lib, CDF_path MUST be of type CHARACTER.

in: INTEGER*4 num_dims

Number of dimensions for the rVariables. This can be as few as zero (0) and at most CDF_MAX_DIMS. Note that this must be specified even if the CDF will contain only zVariables.

in: INTEGER*4 dim_sizes(*)

Dimension sizes for the rVariables. dim_sizes is a 1-dimensional array containing num_dims elements. Each element of dim_sizes specifies the corresponding dimension size. Each size must be

greater than zero (0). For 0-dimensional rVariables this argument is ignored (but a place holder is necessary). Note that this must be specified even if the CDF will contain only zVariables.

out: `INTEGER*4 id`

CDF identifier to be used in subsequent operations on the CDF.

A CDF is created with the default format, encoding, and variable majority as specified in the configuration file of your CDF distribution. Consult your system manager to determine these defaults. These defaults can be changed with the corresponding `PUT_` operations of `CDF_FORMAT_`, `CDF_ENCODING_`, and `CDF_MAJORITY_`.

`<CLOSE_,CDF_>` MUST be used to close the CDF before your application exits to ensure that the CDF will be correctly written to disk.

There are no required preselected objects/states.

`rVAR_` A new rVariable will be created in the current CDF. A variable (rVariable or zVariable) with the same name must not already exist in the CDF. The created rVariable implicitly becomes the current rVariable (in the current CDF). Required arguments are as follows:

in: `CHARACTER var_name*(*)`

Name of the rVariable to be created. This can be at most `CDF_VAR_NAME_LEN` characters. Variable names are case-sensitive. **NOTE:** On UNIX systems, for the proper operation of `CDF_lib`, `var_name` MUST be of type `CHARACTER`.

in: `INTEGER*4 data_type`

Data type of the new rVariable. 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 of the variable. For the non-character string data types this must be one (1) — multiple elements are not allowed for non-character string data types.

in: `INTEGER*4 rec_vary`

Record variance. Specify one of the variances described in Section 5.7.7.

in: `INTEGER*4 dim_varys(*)`

Dimension variances. `dim_varys` is a 1-dimensional array containing one element per rVariable 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. For 0-dimensional rVariables this argument is ignored (but a place holder is necessary).

out: INTEGER*4 var_num

rVariable. This number must be used in subsequent CDF function calls when referring to this rVariable. An existing rVariable's number may be determined with the <GET_,rVAR_NUMBER_> operation.

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

zVAR_ A new zVariable will be created in the current CDF. A variable (rVariable or zVariable) with the same name must not already exist in the CDF. The created zVariable implicitly becomes the current zVariable (in the current CDF). Required arguments are as follows:

in: CHARACTER var_name*(*)

Name of the zVariable to be created. This can be at most CDF_VAR_NAME_LEN characters. Variable names are case-sensitive. **NOTE:** On UNIX systems, for the proper operation of CDF_lib, var_name MUST be of type CHARACTER.

in: INTEGER*4 data_type

Data type of the new zVariable. 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 of the variable. For the non-character string data types this must be one (1) — multiple elements are not allowed for non-character string data types.

in: INTEGER*4 num_dims

Number of dimensions for the zVariable. This may be as few as zero and at most CDF_MAX_DIMS.

in: INTEGER*4 dim_sizes(*)

The dimension sizes. dim_sizes is a 1-dimensional array containing num_dims elements. Each element of dim_sizes specifies the corresponding dimension size. Each dimension size must be greater than zero (0). For a 0-dimensional zVariable this argument is ignored (but a place holder is necessary).

in: INTEGER*4 rec_vary

Record variance. Specify one of the variances described in Section 5.7.7.

in: INTEGER*4 dim_varys(*)

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. For a 0-dimensional zVariable this argument is ignored (but a place holder is necessary).

out: INTEGER*4 var_num

Number assigned to the new zVariable. This number must be used in subsequent CDF function calls when referring to this zVariable. An existing zVariable's number may be determined with the <GET_,zVAR_NUMBER_> operation.

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

ATTR_ A new attribute will be created in the current CDF. An attribute with the same name must not already exist in the CDF. The created attribute implicitly becomes the current attribute (in the current CDF). Required arguments are as follows:

in: CHARACTER attr_name*(*)

Name of the attribute to be created. This can be at most CDF_ATTR_NAME_LEN characters. Attribute names are case-sensitive. **NOTE:** On UNIX systems, for the proper operation of CDF_lib, attr_name MUST be of type CHARACTER.

in: INTEGER*4 scope

Scope of the new attribute. Specify one of the scopes described in Section 5.7.8.

out: INTEGER*4 attr_num

Number assigned to the new attribute. This number must be used in subsequent CDF function calls when referring to this attribute. An existing attribute's number may be determined with the <GET_,ATTR_NUMBER_> operation.

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

OPEN_ The following item(s) may be opened:

CDF_ Opens the named CDF. The opened CDF implicitly becomes the current CDF. Required arguments are as follows:

in: CHARACTER CDF_path*(*)

Pathname of the CDF to be opened. (Do not append an extension.) This can be at most CDF_PATHNAME_LEN characters. A CDF pathname may contain disk and directory specifications that conform to the conventions of the operating system being used (including logical names on VMS systems and environment variables on UNIX systems). **NOTE:** On UNIX systems pathnames are case-sensitive. **NOTE:** On UNIX systems, for the proper operation of CDF_lib, CDF_path MUST be of type CHARACTER.

out: INTEGER*4 id

CDF identifier to be used in subsequent operations of the CDF.

There are no required preselected objects/states.

`DELETE_` The following item(s) may be deleted:

`CDF_` Deletes the current CDF. A CDF must be opened before it can be deleted. When the CDF is deleted, there is no longer a current CDF.

There are no required arguments.

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

`CLOSE_` The following items may be closed:

`CDF_` Closes the current CDF. When the CDF is closed, there is no longer a current CDF. A CDF must be closed to ensure that it will be properly written to disk.

There are no required arguments.

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

`rVAR_` Closes the current rVariable (in the current CDF). This operation in only applicable to multi-file CDFs.

There are no required arguments.

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

`zVAR_` Closes the current zVariable (in the current CDF). This operation is only applicable to multi-file CDFs.

There are no required arguments.

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

`SELECT_` The following items may be selected:

`CDF_` Explicitly selects the current CDF. Required arguments are as follows:

in: `INTEGER*4 id`

Identifier of the CDF. This identifier must have been initialized by a successful `<CREATE_,CDF_>` or `<OPEN_,CDF_>` operation.

There are no required preselected objects/states.

`CDF_STATUS_` Selects the current status code. Required arguments are as follows:

in: `INTEGER*4 status`

CDF status code.

There are no required preselected objects/states.

`CDF_READONLY_MODE_` Selects a read-only mode (for the current CDF). Required arguments are as follows:

in: `INTEGER*4 mode`

The read-only mode. Specify one of the read-only modes described in Section 5.7.9.

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

`CDF_zMODE_` Selects a zMode (for the current CDF). Required arguments are as follows:

in: `INTEGER*4 mode`

The zMode. Specify one of the zModes described in Section 5.7.10.

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

`CDF_NEGtoPOSfp0_MODE_`

Selects a `-0.0` to `0.0` mode (for the current CDF). Required arguments are as follows:

in: `INTEGER*4 mode`

The `-0.0` to `0.0` mode. Specify one of the `-0.0` to `0.0` modes described in Section 5.7.11.

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

`CDF_DECODING_` Selects a decoding (for the current CDF). Required arguments are as follows:

in: `INTEGER*4 decoding`

The decoding. Specify one of the decodings described in Section 5.7.5.

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

`CDF_CACHESIZE_` Selects the number of cache buffers to be 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: `INTEGER*4 num_buffers`

The number of cache buffers to be used.

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

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

in: `INTEGER*4 var_num`

rVariable number.

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

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

in: `CHARACTER var_name*(*)`

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`.

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

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

in: `INTEGER*4 rec_num`

Record number.

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

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

in: `INTEGER*4 rec_count`

Record count.

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

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

in: `INTEGER*4 rec_interval`

Record interval.

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

`rVARs_DIMINDICES_` Selects 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:

in: `INTEGER*4 indices(*)`

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

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

`rVARs_DIMCOUNTS_` Selects 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:

in: `INTEGER*4 counts(*)`

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: `INTEGER*4 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 the current sequential value is maintained for each rVariable individually. Required arguments are as follows:

in: `INTEGER*4 rec_num`

Record number.

in: `INTEGER*4 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: `INTEGER*4 num_buffers`

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: `INTEGER*4 num_buffers`

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: `INTEGER*4 var_num`

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: `CHARACTER var_name*(*)`

zVariable 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`.

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: `INTEGER*4 rec_num`

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: `INTEGER*4 rec_num`

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: `INTEGER*4 rec_count`

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: `INTEGER*4 rec_interval`

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: `INTEGER*4 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: `INTEGER*4 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: `INTEGER*4 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 the current sequential value is maintained for each zVariable individually. Required arguments are as follows:

in: `INTEGER*4 rec_num`

Record number.

in: `INTEGER*4 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: `INTEGER*4 num_buffers`

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: `INTEGER*4 num_buffers`

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). Required arguments are as follows:

in: `INTEGER*4 attr_num`

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: `CHARACTER attr_name*(*)`

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 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: `INTEGER*4 entry_num`

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: `INTEGER*4 entry_num`

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: `CHARACTER var_name*(*)`

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`.

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: `INTEGER*4 entry_num`

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: `CHARACTER var_name*(*)`

zVariable 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`.

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: `CHARACTER CDF_path*(CDF_PATHNAME_LEN)`

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

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: INTEGER*4 mode

The read-only mode. The read-only modes are described in Section 5.7.9.

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: INTEGER*4 mode

The zMode. The zModes are described in Section 5.7.10.

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: INTEGER*4 mode

The -0.0 to 0.0 mode. The -0.0 to 0.0 modes are described in Section 5.7.11.

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: INTEGER*4 decoding

The decoding. The decodings are described in Section 5.7.5.

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: INTEGER*4 num_buffers

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: INTEGER*4 var_num

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: `CHARACTER var_name*(*)`

The rVariable name. This may be at most `CDF_VAR_NAME_LEN` characters.

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: `INTEGER*4 rec_num`

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: `INTEGER*4 rec_count`

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: `INTEGER*4 rec_interval`

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: `INTEGER*4 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: `INTEGER*4 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 applicable. Required arguments are as follows:

out: `INTEGER*4 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: `INTEGER*4 rec_num`

Record number.

out: `INTEGER*4 indices(*)`

Dimension indices. `indices` is a 1-dimensional array containing one element per rVariable 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: `INTEGER*4 num_buffers`

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: `INTEGER*4 var_num`

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: `CHARACTER var_name*(*)`

The zVariable name. This may be at most `CDF_VAR_NAME_LEN` characters.

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: `INTEGER*4 rec_num`

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: `INTEGER*4 rec_count`

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: `INTEGER*4 rec_interval`

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: `INTEGER*4 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: `INTEGER*4 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: `INTEGER*4 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: `INTEGER*4 rec_num`

Record number.

out: `INTEGER*4 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: `INTEGER*4 num_buffers`

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: `INTEGER*4 attr_num`

Attribute number.