

```

netcdf_file = header data
header      = magic numrecs dim_list gatt_list var_list
magic        = 'C' 'D' 'F' VERSION
VERSION      = \x01 |                                // classic format
                \x02 |                                // 64-bit offset format
                \x05 |                                // 64-bit data format
numrecs     = NON_NEG | STREAMING                 // length of record dimension
dim_list    = ABSENT | NC_DIMENSION nelems [dim ...]
gatt_list   = att_list                          // global attributes
att_list    = ABSENT | NC_ATTRIBUTE nelems [attr ...]
var_list    = ABSENT | NC_VARIABLE nelems [var ...]
ABSENT      = ZERO ZERO |                      // list is not present for CDF-1 and 2
                ZERO ZERO64 |                  // for CDF-5
ZERO        = \x00 \x00 \x00 \x00                   // 32-bit zero
ZERO64     = \x00 \x00 \x00 \x00 \x00 \x00 \x00 \x00 // 64-bit zero
NC_DIMENSION = \x00 \x00 \x00 \x0A             // tag for list of dimensions
NC_VARIABLE = \x00 \x00 \x00 \x0B             // tag for list of variables
NC_ATTRIBUTE = \x00 \x00 \x00 \x0C             // tag for list of attributes
nelems      = NON_NEG           // number of elements in following sequence
dim         = name dim_length
name        = nelems namestring
                // Names a dimension, variable, or attribute.
                // Names should match the regular expression
                // ([a-zA-Z0-9_]|{MUTF8})({^|\x00-\x1F|\x7F-\xFF}|{MUTF8})*
                // For other constraints, see "Note on names", below.
namestring = ID1 [IDN ...] padding
ID1         = alphanumeric | '_'
IDN         = alphanumeric | special1 | special2
alphanumeric = lowercase | uppercase | numeric | MUTF8
lowercase   = 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' |
                'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'
uppercase   = 'A' | 'B' | 'C' | 'D' | 'E' | 'F' | 'G' | 'H' | 'I' | 'J' | 'K' | 'L' | 'M' |
                'N' | 'O' | 'P' | 'Q' | 'R' | 'S' | 'T' | 'U' | 'V' | 'W' | 'X' | 'Y' | 'Z'
numeric    = '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
                // special1 chars have traditionally been
                // permitted in netCDF names.
special1   = '_' | '.' | '@' | '+' | '-'
                // special2 chars are recently permitted in
                // names (and require escaping in CDL).
                // Note: '/' is not permitted.
special2   = ' ' | '!' | '"' | '#' | '$' | '%' | '&'amp; | '\'' |
                '(' | ')' | '*' | ',' | ':' | ';' | '<' | '=' | |
                '>' | '?' | '[' | '\\\' | ']' | '^' | ` | '{' | |
                '|' | '}' | '~'
MUTF8       = <multibyte UTF-8 encoded, NFC-normalized Unicode character>
dim_length = NON_NEG           // If zero, this is the record dimension.
                // There can be at most one record dimension.
attr        = name nc_type nelems [values ...]
nc_type     = NC_BYTE | 
                NC_CHAR | 
                NC_SHORT | 
                NC_INT | 
                NC_FLOAT | 
                NC_DOUBLE | 
                NC_UBYTE | 
                NC USHORT | 
                NC_UINT | 
                NC_INT64 | 
                NC_UINT64 | 
                NC_STRING
var         = name nelems [dimid ...] vatt_list nc_type vszie begin

```

```

// nelems is the dimensionality (rank) of the
// variable: 0 for scalar, 1 for vector, 2
// for matrix, ...
dimid      = NON_NEG           // Dimension ID (index into dim_list) for
                                // variable shape. We say this is a "record
                                // variable" if and only if the first
                                // dimension is the record dimension.

vatt_list   = att_list          // Variable-specific attributes
vsize       = NON_NEG           // Variable size. If not a record variable,
                                // the amount of space in bytes allocated to
                                // the variable's data. If a record variable,
                                // the amount of space per record. See "Note
                                // on vsize", below.

begin       = OFFSET             // Variable start location. The offset in
                                // bytes (seek index) in the file of the
                                // beginning of data for this variable.

data        = non_recs  recs
non_recs    = [vardata ...]     // The data for all non-record variables,
                                // stored contiguously for each variable, in
                                // the same order the variables occur in the
                                // header.

vardata     = [values ...]      // All data for a non-record variable, as a
                                // block of values of the same type as the
                                // variable, in row-major order (last
                                // dimension varying fastest).

recs        = [record ...]       // The data for all record variables are
                                // stored interleaved at the end of the
                                // file.

record      = [varslab ...]      // Each record consists of the n-th slab
                                // from each record variable, for example
                                // x[n,...], y[n,...], z[n,...] where the
                                // first index is the record number, which
                                // is the unlimited dimension index.

varslab     = [values ...]      // One record of data for a variable, a
                                // block of values all of the same type as
                                // the variable in row-major order (last
                                // index varying fastest).

values      = bytes | chars | shorts | ints | floats | doubles |
              ubytes | ushorts | uints | int64s | uint64s
string      = nelems [chars]
bytes       = [BYTE ...] padding
chars       = [CHAR ...] padding
shorts      = [SHORT ...] padding
ints        = [INT ...]
floats      = [FLOAT ...]
doubles     = [DOUBLE ...]
ubytes      = [UBYTE ...] padding
ushorts     = [USHORT ...] padding
uints       = [UINT ...]
int64s      = [INT64 ...]
uint64s     = [UINT64 ...]
padding     = <0, 1, 2, or 3 bytes to next 4-byte boundary>
              // Header padding uses null (\x00) bytes. In
              // data, padding uses variable's fill value.
              // See "Note on padding", below, for a special
              // case.

NON_NEG     = <non-negative INT> |
              <non-negative INT64> // for 64-bit data format
STREAMING   = \xFF \xFF \xFF \xFF // Indicates indeterminate record
                                // count, allows streaming data
OFFSET      = <non-negative INT> // For classic format or

```

```

        <non-negative INT64> // for 64-bit offset format
BYTE      = <8-bit byte>           // See "Note on byte data", below.
CHAR      = <8-bit byte>           // See "Note on char data", below.
SHORT     = <16-bit signed integer, BigEndian, two's complement>
INT       = <32-bit signed integer, BigEndian, two's complement>
FLOAT    = <32-bit IEEE single-precision float, BigEndian>
DOUBLE   = <64-bit IEEE double-precision float, BigEndian>
UBYTE    = <8-bit unsigned byte>
USHORT   = <16-bit unsigned integer, BigEndian, two's complement>
UINT     = <32-bit unsigned integer, BigEndian, two's complement>
INT64    = <64-bit signed integer, BigEndian, two's complement>
UINT64   = <64-bit unsigned integer, BigEndian, two's complement>

// following type tags are 32-bit integers
// \x is a way to indicate that the next two characters represent
// hexadecimal digits. Two hexadecimal digits (each of them 4 bits)
// make a byte.
NC_BYTE   = \x00 \x00 \x00 \x01      // 8-bit signed integers
NC_CHAR   = \x00 \x00 \x00 \x02      // text characters
NC_SHORT  = \x00 \x00 \x00 \x03      // 16-bit signed integers
NC_INT    = \x00 \x00 \x00 \x04      // 32-bit signed integers
NC_FLOAT  = \x00 \x00 \x00 \x05      // IEEE single precision floats
NC_DOUBLE = \x00 \x00 \x00 \x06      // IEEE double precision floats
NC_UBYTE  = \x00 \x00 \x00 \x07      // unsigned 1 byte int
NC USHORT = \x00 \x00 \x00 \x08      // unsigned 2-byte int
NC_UINT   = \x00 \x00 \x00 \x09      // unsigned 4-byte int
NC_INT64  = \x00 \x00 \x00 \x0A      // signed 8-byte int
NC_UINT64 = \x00 \x00 \x00 \x0B      // unsigned 8-byte int
NC_STRING = \x00 \x00 \x00 \x0C      // string

                                // Default fill values for each type, may be
                                // overridden by variable attribute named
                                // '_FillValue'. See "Note on fill values",
                                // below.
FILL_CHAR = \x00                  // ((char)0) null byte
FILL_BYTE  = \x81                  // (signed char) -127
FILL_SHORT = \x80 \x01              // (short) -32767
FILL_INT   = \x80 \x00 \x00 \x01    // (int) -2147483647
FILL_FLOAT = \x7C \xF0 \x00 \x00    // (float) 9.9692099683868690e+36
FILL_DOUBLE= \x47 \x9E \x00 \x00 \x00 \x00 \x00 \x00 // (double) 9.9692099683868690e+36
FILL_UBYTE = \xFF                  // (255)
FILL USHORT= \xFF \xFF             // (65535)
FILL_UINT  = \xFF \xFF \xFF \xFF   // (4294967295U)
FILL_INT64 = \x80 \x00 \x00 \x00 \x00 \x00 \x00 \x01 // ((long long)-9223372036854775807LL)
FILL_UINT64= \xFF \xFF \xFF \xFF \xFF \xFF \xFF \xFF // ((unsigned long long)18446744073709551615ULL)
FILL_STRING= \x00                  // "" null string

```