CF & Strings

An exploration of possibilities

Issues & Questions

- Issues
  - The NC_STRING data type
  - Interpretation of NC_CHAR variables
  - Non-ASCII contents
- Questions
  - Does CF embrace NC_STRING type?
  - Does CF embrace non-ASCII encodings?
  - Are changes needed to disambiguate interpretation of NC_CHAR variables?

Issue: NC_STRING

- “Atomic” data type – each element is a character string.
- Only available in netCDF-4.
- Some existing applications (e.g. written in C or FORTRAN) can’t handle them without modifications.
- NC_STRING attributes
- NC_STRING variables

Issue: NC_STRING Attributes

- Functionally equivalent to NC_CHAR attributes when only one string is present.
- Allows an attribute to contain multiple strings, equivalent to attributes with numeric types containing multiple values.
- **NERC CF checker currently fails if it encounters an NC_STRING attribute.**
- Some language bindings (e.g. Python netCDF4 package) automatically force an attribute to NC_STRING type if the string is non-ASCII (more later).

Issue: NC_STRING Variables

- Dispenses with the need for “string length” dimensions.
- Makes interpretation of the variable as an array of strings unambiguous. (More about this later.)

Issue: Interpreting NC_CHAR Variables

- CF Conventions (Section 2.2) states that numeric data should be stored using the NC_BYTE type, that a 1-D NC_CHAR variable should be interpreted as a single string, and that a multi-string variable should have a fastest-moving dimension equal to the length of the longest string in the variable.
- Trac Ticket #158 (closed wontfix) raised the question of a variable that should be interpreted as an array of individual characters, instead of a single string.

Issue: Non-ASCII Contents

- A bit of background
  - ASCII is the “classic” character set containing 128 single-byte characters.
  - Unicode is a text representation standard that currently contains 136,755 characters. There are different ways to encode characters within the Unicode standard.
  - UTF-8 is a multi-byte Unicode encoding system. All ASCII characters are valid UTF-8 characters.
  - There are other old and new encodings that are not ASCII or Unicode.

Issue: Non-ASCII Contents

- CF Conventions don't address the question of non-ASCII characters in NC_CHAR or NC_STRING attributes or variables when the contents are not restricted to using a controlled vocabulary.
- Some existing applications may not handle non-ASCII contents correctly.
- NUG has explicitly allowed UTF-8 encoding for all netCDF-3 (version >= 3.6.3) & netCDF-4 strings (and names).
- NUG mentions a future attribute, _Encoding, that is reserved for declaring encodings other than UTF-8.
- Trac Ticket #159 (closed wontfix) raised the question of how to declare the encoding used in a NC_CHAR or NC_STRING variable.

Questions: NC_STRING

- Should CF embrace NC_STRING type?
  - For attributes? And do we also embrace multiple strings in an attribute?
  - For variables?

Questions: Non-ASCII Contents

- Should CF explicitly embrace non-ASCII encodings in attribute and variable contents?
  - UTF-8 only?
  - How do we specify other encodings?

Questions: Non-ASCII Contents

Trac Ticket #159 closed enhancement (wontfix)

In order to specify the character set of char and string variables, I propose that we append these two paragraphs to the end of CF section 2.2:

Each char array variable that is to be interpreted as an array of individual characters (not string(s)) must have a "charset" attribute which clarifies that the variable is to be interpreted as individual characters (not string(s)) and specifies the 8-bit character set used by the chars. Values for "charset" are case-insensitive. See http://www.iana.org/assignments/character-sets/character-sets.xhtml . Currently, the only values allowed for "charset" are "ISO-8859-1" and "ISO-8859-15". A scalar char variable may also use the "charset" attribute, which defaults to "ISO-8859-15" if it is not specified.

A string or string array variable (including a char array variable that is to be interpreted as a string or array of strings) may have an "_Encoding" attribute. Alternatively, a file may have a global "_Encoding" attribute which applies to all strings (scalar and array) in the file. Values for "_Encoding" are case-insensitive. See http://www.iana.org/assignments/character-sets/character-sets.xhtml . Currently, the only values allowed for "_Encoding" are "ISO-8859-1", "ISO-8859-15" and "UTF-8". A missing "_Encoding" attribute defaults to "UTF-8".

Questions: Interpreting NC_CHAR Variables

- Are there any changes to CF needed to disambiguate the interpretation of NC_CHAR variables?

Questions: Interpreting NC_CHAR Variables

Trac Ticket #158 closed enhancement (wontfix)

In NetCDF-3 files, in order to make it clear whether an array of chars should be interpreted as an array of chars or an array of Strings, I propose that we replace this sentence in CF section 2.2:

NetCDF does not support a character string type, so these must be represented as character arrays. In this document, a one dimensional array of character data is simply referred to as a "string".

with:

Since NetCDF-3 files do not have a built-in string data type, strings in NetCDF-3 files must be represented as character arrays. To clarify how a char array should be interpreted, char arrays must have a "data_type" attribute with a value of "char" (for individual chars) or "string". (In older files with char variables that lack a data_type attribute, it remains ambiguous whether a char array should be interpreted as an array of individual characters or an array of Strings.)

Questions: Interpreting NC_CHAR Variables

CF Section 2.2

The netCDF data types **char**, **byte**, **short**, **int**, **float** or **real**, and **double** are all acceptable. The **char** type is not intended for numeric data. One byte numeric data should be stored using the **byte** data type. All integer types are treated by the netCDF interface as signed. It is possible to treat the **byte** type as unsigned by using the NUG convention of indicating the unsigned range using the **valid_min**, **valid_max**, or **valid_range** attributes.

NetCDF does not support a character string type, so these must be represented as character arrays. In this document, a one dimensional array of character data is simply referred to as a "string". An n-dimensional array of strings must be implemented as a character array of dimension (n,max_string_length), with the last (most rapidly varying) dimension declared large enough to contain the longest string in the array. All the strings in a given array are therefore defined to be equal in length. For example, an array of strings containing the names of the months would be dimensioned (12,9) in order to accommodate "September", the month with the longest name.