CIF - Changes to the specification

05 July 2010

This document specifies changes to the *syntax* of CIF. We refer to the current syntax specification of CIF as CIF1, and the new specification as CIF2. To date all archival CIFs are CIF1.

The changes to syntax are necessitated by the adoption of new dictionary functionalities that introduce several extensions, including new data types, and method definitions using dREL.

It is assumed the reader has a thorough understanding of the CIF1 specification.

TERMINOLOGY

Reference to **character(s)** means abstract characters assigned code points by **Unicode**. Specific characters are referenced according to Unicode convention, U+*xxxx*[*x*[*x*]], where *xxxx*[*x*[*x*]] is the four- to six-digit hexadecimal representation of the assigned code point. The designated character encoding for CIF2 is UTF-8.

Reference to **ASCII** characters means characters U+0000 through U+007F, or, equivalently the first 128 characters of the **ISO-8859-1** (**LATIN-1**) character set.

Reference to **newline** or **\n** means the sequence that conventionally terminates a line record (which is environment dependent). **See Change 3.**

Reference to **whitespace** means the characters ASCII space (U+0020), ASCII horizontal tab (U+0009) and the **newline** characters. Without regard to local convention, the various other characters that Unicode classifies as whitespace (character categories Zs and Zp) do not constitute **whitespace** for the purposes of CIF2.

PREAMBLE

CIF2 significantly extends CIF1 functionality, primarily through new dictionary features. CIF2 is not fully backwards-compatible with CIF1: many files compliant with CIF1 are also compliant with CIF2, but some are not (see especially change 5, below). The CIF1 standard will continue to operate for the foreseeable future in parallel with CIF2.

CHANGE 1 - NEW (MAGIC CODE)

A CIF2 file is uniquely identified by a required magic code at the beginning of its first line. The code is,

```
#\#CIF_2.0
```

followed immediately by **whitespace**.

CHANGE 2 – NEW (CHARACTER SET)

CIF2 files are standard variable length text files, which for compatibility with older processing systems will have a maximum line length of 2048 characters. As discussed

above and below, however, there are some restrictions on the character set for token delimiters, separators and data names.

In keeping with XML restrictions we allow the characters

```
U+0009 U+000A U+000D
U+0020 – U+007E
U+00A0 – U+D7FF
U+E000 – U+FDCF
U+FDF0 – U+FFFD
U+10000 – U+10FFFD
```

In addition, character U+FEFF and characters U+xFFFE or U+xFFFF where x is any hexadecimal digit are disallowed. Unicode reserves the code points E000 – F8FF for private use. The IUCr and only the IUCr may specify what characters are assigned to these code points in the context of CIF2.

Reasoning: There is growing demand for the wider character set afforded by Unicode to be made available in applications, especially those where internationalisation is an issue.

CHANGE 3 – RESTRICTION

Treatment of Newline

CIF2 processors are *required* to treat <U+000A>, <U+000D> and <U+000D><U+000A> as **newline** characters, by normalising them to <U+000A> on read. No other characters or character sequences may represent **newline**. In particular, CIF2 processors should not interpret the Unicode characters U+2028 (line separator) or U+2029 (paragraph separator) as **newline**.

CHANGE 4 - DEFINITION

Character set for *data names*.

In CIF2 the tags referred to as data names are composed of characters from the allowed set above, excluding a whitespace, since this terminates data name string. A data name begins with an **ASCII** _ and may be followed by any number of characters within the 2048 character restriction.

All data names in a **valid** CIF must be defined in a CIF dictionary referenced implicitly or explicitly by the CIF, and the DDLs in which such dictionaries are written may place additional restrictions on the data names that can be defined. For example, data names defined in a DDLm dictionary must match the regular expression `_[A-Za-z0-9_.]+` (the `.` is the explicit **ASCII** period character).

Note: In CIF2, as with CIF1, there is no explicit meaning to the sequence of characters, or the placement of _ or . in the data name. The separation of the category and attribute in a data name by a period (.) is purely a convention.

CHANGE 5 - RESTRICTION

Whitespace-delimited *data values*.

A data value in CIF2 may be a whitespace delimited string of allowed characters.

The first character of a whitespace delimited string cannot be any of the ASCII characters `" ' _ $ [ {`, and the terminal character cannot be `]` or `}`, since these have special meaning. STAR keywords may not appear as whitespace delimited strings: `loop_ global_ save_* stop_ data_*` (case insensitive), where * refers to zero or more characters.

Reasoning: The above exclusions are required for CIF2 syntax to be unambiguous.

CHANGE 6 – RESTRICTION

Delimited strings.

The delimited strings accepted in CIF2 are,

(1) A string delimited by ASCII single-quotes(`'`). The string is initiated by an ASCII single-quote, can consist of allowed characters excluding the newline, and is terminated by the first subsequent ASCII single-quote. **Clearly, the string within cannot contain ASCII single-quote characters.**

CIF2 does not specify any interpretation of the contents of the string. For example

```
loop_ _author.family _name 'Harris' 'Gr\"uber'
```

As far as CIF2 is concerned, the string values are `Harris` and `Gr\"uber`; handling of any elide characters is left to the calling application.

(2) A string delimited by ASCII double quotes ("). The string is initiated by an ASCII double-quote, can consist of allowed characters excluding the newline, and is terminated by the first subsequent ASCII double-quote. **Clearly, the string within cannot contain ASCII double-quote characters.**

CIF2 does not specify any interpretation of the contents of the string. For example

```
_quote.literal "He said, 'We're going in circles'"
```

The string value is `He said, 'We're going in circles'`, including the embedded single-quotes.

(3) A string delimited by ASCII newline semi-colon(`\n;`). The string is initiated by an ASCII newline semi-colon sequence, consists of any of the allowed characters, and is terminated by the first subsequent ASCII newline semi-colon sequence. **Clearly, the strings within cannot contain an ASCII newline semi-colon sequence.**

CIF2 does not specify any interpretation of the contents of the string. For example

```
_recipe.ingredients
;Sugar
Flour
```

```
Butter
;
```

The string value is `Sugar\nFlour\nButter`, where `\n` is the literal newline sequence.

With respect to changes 7, 8, and 9, the newline in the opening delimiter is considered to separate a delimited string of this kind from the preceding syntax element.

CHANGE 7 – NEW

Triple-quote delimited strings.

The ASCII """ sequence (alternatively ASCII ''') delimits the beginning of a string that may contain any printable character and whitespace and is terminated by the first subsequent """ sequence (alternatively '''). CIF2 does not specify any interpretation of the contents of the string. The string can contain separable " and "" characters, (alternatively ' and ''). **Clearly, the string within cannot contain an ASCII """ (or alternatively ASCII ''')**.

For example

```
"""He said "His name is O'Hearly"."""

'''In {\bf \TeX} the accents are \' and \".'''
```

The string values are, `He said "His name is O'Hearly".` and `In {\bf \TeX} the accents are \' and \".`. No interpretation of any elides is undertaken; this is the responsibility of the calling application. The triple quote string supports embedded newlines, which are considered part of the string.

CHANGE 8 – NEW

List data type.

The ASCII square bracket (`[]`) is accepted in STAR for delimiting the List compound data type. A List is an ordered sequence of values. A data value of type List is initiated by an ASCII left square bracket(`[`) and terminated by the pair-matching ASCII right square bracket (`]`). The List elements are separated from each other by whitespace. For example

```
loop_
     _colour_name   _colour_value_rgb
               red    [1 0 0]
             green    [0 1 0]
```

The elements of a List can be any CIF2 data values, and hence it is a recursive data type. For example

```
_refln.hklFoFc [[1 3 -4] 23.32(9) 22.97(11)]
```

Since List elements are whitespace separated (and may include `\n;`-delimited strings), a List can span more than one physical line. For example

```
_refln.hklFoFc [[1 3 -4]
                    23.32(9) 22.97(11)]
```

is identical to the previous example list.

Whitespace is allowed, but not required, between the `[]` delimiters and the values within, and between the opening and closing delimiters of an empty List.

CHANGE 9 – NEW

Table data type.

The ASCII curly brace (`{}`) is accepted in STAR for delimiting the Table (*Associative Array*) compound data type. A Table is an unordered mapping from string labels ("indices") to CIF2 data values. Labels must be unique within each Table (excluding Tables nested within it).

A data value of type Table is initiated by an ASCII left curly brace(`{`) and terminated by the pair-matching ASCII right curly brace (`}`).The Table elements are separated from each other by **whitespace**, and consist of: an index label as a single- or double-quote delimited string, or as a triple-quoted string; an ASCII colon(`:`); and a CIF2 data value. For example

```
{"symm":"P 4n 2 3 -1n" 'avec':[10.3 0.0 0.0]
 'bvec':[0.0 10.3 0.0] 'cvec':[0.0 0.0 10.3]
     "description":"""Cubic space group
        and metric cell vectors"""}
```

A Table is a recursive data type. Since Table elements are whitespace separated (and also since values may be `\n;`-delimited strings), a Table may span more than one physical line.

Whitespace is allowed, but not required, between the `{}` delimiters and the elements within, and between the opening and closing delimiters of an empty Table.

CHANGE 10 – REFINEMENT to CIF1

Separating syntax elements.

CIF2 keywords, data block headers, save frame headers, data names, and data values must all be separated from each other by **whitespace**. Whitespace not otherwise part of a CIF2 syntax element is significant only for this purpose.

Reasoning: The CIF1 specification relies implicitly on the syntactic structure of the language to require whitespace separators between syntax elements. The CIF2 syntax no longer implicitly provides whitespace separators in some cases (notably, after most types of data values), therefore the requirement is now made explicit.