International Tables for Crystallography (2006). Vol. G, Section 2.2.2, pp. 20–21.

2.2. Specification of the Crystallographic Information File (CIF)

BY S. R. HALL AND J. D. WESTBROOK
WITH SECTION 2.2.7 BY S. R. HALL, N. SPADACCINI, I. D. BROWN, H. J. BERNSTEIN, J. D. WESTBROOK AND B. MCMAHON

2.2.1. Introduction

The term 'Crystallographic Information File' (CIF) refers to data and dictionary files conforming to the conventions adopted by the IUCr in 1990 and revised by the IUCr Committee for the Maintenance of the CIF Standard (COMCIFS). The CIF format is intended to meet the needs of a wide range of scientific applications within, and without, the discipline of crystallography. Parts 2 and 3 of this volume provide the full specification of the contents of CIF across the different crystallographic applications. The files used in these applications must conform to the same rules of syntax, and share certain properties and conventions in the way that information is presented. It is these common features that are discussed in this chapter.

The CIF family of applications uses a proper subset of the STAR File syntax described in Chapter 2.1. The STAR File grammar provides a very general approach to storing and accessing data values through the use of an associated data name, or tag. A CIF search tool, such as *Star_Base* (Chapter 5.2), can readily access a single data value, or set of values, using this tag without prior knowledge of the order of the file contents. It can also provide details of the context of the data within the file structure. Context, in this sense, is a fully annotated indication of the file structure in which the retrieved value was located. That is, whether it was located in a global declaration, a named save frame, or a looped list. In every case the data 'value' is simply a character string and the STAR File protocol itself imposes absolutely no meaning on that string. This leaves the interpretation of the value string (*e.g.* whether it is numerical or text) to the conventions of the applications used to read and write the STAR File.

The CIF approach to the permissive STAR File syntax is restrictive. In the first place, the earliest version of the CIF syntax (Hall *et al.*, 1991) did not adopt some of the grammatical (or syntactical) constructs available to the STAR File in order to facilitate existing crystallographic software approaches. This was in anticipation of likely short-term developments, and to encourage a rapid take-up of the CIF approach. For these reasons it was considered appropriate to adopt only data-block partitioning and a single, rather than multiple, level of looped lists. Save frames were adopted into the CIF later but only in CIF dictionary files written using the DDL2 dictionary definition language (see Chapter 2.6). It is relevant to point out, however, that the full STAR File syntax has been adopted for the storage of NMR experimental and structure data (Ulrich *et al.*, 1998).

In 2002, a COMCIFS review of the design and implementation of CIF led to a revised syntax specification, which was published in February 2003. This revised specification is reproduced in full in Section 2.2.7. It is important to note that after more than a decade of CIF usage, this revision contains few substantial changes to the design choices of the original version (Hall *et al.*, 1991). There have been some modest extensions to the lengths of data names and text lines, and a number of clarifications are introduced. In addition, various privileged labels used for STAR File constructs (*e.g.* global blocks, save frames and nested loops, as described in Chapter 2.1) have now been explicitly reserved (*i.e.* excluded from appearing in unquoted form in an existing CIF). This will allow the clean upward migration of future CIF syntax versions to the more complex data structures permitted in a STAR File, when and if these are later required by the community.

The remainder of this chapter is structured as follows. First, there is a brief description of CIF terminology (Section 2.2.2). This is followed by the syntax rules, corresponding to a subset of the STAR syntax, used by CIF data files (Section 2.2.3). The portability and archival issues that programmers must be aware of in applying CIF data in different computing environments are described in Section 2.2.4. They are also detailed in the formal specifications given at the end of the chapter. Section 2.2.5 describes the conventions regarding data typing and embedded semantics that are common to all CIF applications, and Section 2.2.6 outlines future possible ways of introducing metadata which would enable files to be linked to each other, and which would establish the nature of CIF contents within a more general framework of information storage systems. The final section of the chapter, Section 2.2.7, reproduces in full the formal specification documents approved by COMCIFS.

2.2.2. Terminology

A summary of the basic terminology used throughout this chapter and the volume follows. A more extensive description of this terminology is given as formal specifications in Section 2.2.7.1.2.

(i) A **CIF** is a file conforming to the specification presented in this chapter. The term includes both **data files** containing information on a structural experiment or its results (or similar scientific content) and the **dictionary files** that provide descriptions of the data identifiers used in such data files.

(ii) A **data name** or **tag** is an identifier (a string of characters beginning with an underscore character) of the content of an associated data value.

(iii) A **data value** is a string of characters representing a particular item of information. It may represent a single numerical value; a letter, word or phrase; extended discursive text; or in principle any coherent unit of data such as an image, audio clip or virtual-reality object.

(iv) A **data item** is a specific piece of information defined by a data name and its associated data value.

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; JOHN D. WESTBROOK, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA; NICK SPADACCINI, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia; I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1; HERBERT J. BERNSTEIN, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England.

(v) A **data block** is the highest-level component of a CIF, containing data items or (in the case of dictionary files only) save frames. A data block is identified by a **data-block header**, which is an isolated character string (that is, bounded by white space and not forming part of a data value) beginning with the case-insensitive reserved characters `data_`. A **block code** is the variable part of a data-block header, *e.g.* the string *foo* in the header `data_foo`.

(vi) A **looped list** of data is a set of data items represented as a table or matrix of values. The data names are assembled immediately following the word `loop_`, each separated by white space, and the associated data values are then listed in strict rotation. The table of values is assembled in row-major order; that is, the first occurrence of each of the data items is assembled in sequence, then the second occurrence of each item, and so forth. In a CIF, looped lists may not be nested.

2.2.3. The syntax of a CIF

The essential syntax rules for a CIF data file are discussed alongside an example (Fig. 2.2.3.1), which is an extract from a file used to exemplify the reporting of a small-molecule crystal structure to *Acta Crystallographica Section C*. The following discussion is tutorial in nature and is intended to give an overview of the syntactic features of CIF to the general reader. The special use of save frames in dictionary files is not discussed in this summary. Software developers will find the full specification at the end of the chapter. If there are any real or apparent discrepancies between the two treatments, the full specification is to be taken as definitive.

A CIF contains only ASCII characters, organized as lines of text.

Tokens (the discrete components of the file) are separated by white space; layout is not significant. Thus, in the list of atom-site coordinates in Fig. 2.2.3.1, the hydrogen-atom entries are cosmetically aligned in columns, but the non-aligned entries for the other atoms are equally valid. Indeed, there is no requirement that each cluster of looped data values be confined to a separate row; contrast the cosmetic ordering of the atom-sites loop with the loop of symmetry-equivalent positions, where entries run on the same or following lines indiscriminately.

A comment is a token introduced by a hash character # and extending to the end of the line. Comments are considered to have no portable information content and may freely be discarded by a parser. However, revision 1.1 of the CIF specification introduces a *recommendation* that a CIF begin with a comment taking the form

```
#\#CIF_1.1
```

where the 1.1 is a version identifier of the reference CIF specification. This is primarily for the benefit of general file-handling software on current operating systems (*e.g.* graphical file managers that associate software applications with files of specific type), and its presence or absence does not guarantee the integrity of the file with respect to any particular revision of the CIF specification.

The first non-comment token of a CIF must be a data-block header, which is a character string that does not include white space and begins with the case-insensitive characters `data_`.

The file may be partitioned into multiple data blocks by the insertion of further data-block headers. Data-block headers are case-insensitive (that is, two headers differing only in whether corresponding letter characters are upper or lower case are considered identical). Within a single data file identical data-block headers are not permitted.

```
data_99107abs

# Chemical data
_chemical_name_systematic
; 3-Benzo[b]thien-2-yl-5,6-dihydro-1,4,2-oxathiazine
  4-oxide
;
_chemical_formula_moiety         "C11 H9 N O2 S2"
_chemical_formula_weight          251.31

# Crystal data
_symmetry_cell_setting            orthorhombic
_symmetry_space_group_name_H-M    'P 21 21 21'

loop_
   _symmetry_equiv_pos_as_xyz
 'x, y, z' 'x+1/2, -y+1/2, -z' '-x, y+1/2, -z+1/2'
 '-x+1/2, -y, z+1/2'

_cell_length_a                    7.4730(11)
_cell_length_b                    8.2860(11)
_cell_length_c                    17.527(2)
_cell_angle_alpha                 90.00
_cell_angle_beta                  90.00
_cell_angle_gamma                 90.00

# Atomic coordinates and displacement parameters
loop_
  _atom_site_label
  _atom_site_type_symbol
  _atom_site_fract_x
  _atom_site_fract_y
  _atom_site_fract_z
  _atom_site_U_iso_or_equiv
S4 S 0.32163(7) 0.45232(6) 0.52011(3) 0.04532(13)
S11 S 0.39642(7) 0.67998(6) 0.29598(2) 0.04215(12)
O1 O -0.00302(17) 0.67538(16) 0.47124(8) 0.0470(3)
O4 O 0.2601(2) 0.28588(16) 0.50279(10) 0.0700(5)
N2 N 0.14371(19) 0.66863(19) 0.42309(9) 0.0402(3)
C3 C 0.2776(2) 0.57587(19) 0.43683(9) 0.0332(3)
C5 C 0.1497(3) 0.5457(3) 0.57608(11) 0.0498(5)
C6 C -0.0171(3) 0.5529(2) 0.52899(12) 0.0460(4)
C12 C 0.4215(2) 0.57488(19) 0.38139(9) 0.0344(3)
C13 C 0.5830(2) 0.4995(2) 0.38737(10) 0.0386(4)
C13A C 0.6925(2) 0.5229(2) 0.32123(10) 0.0399(4)
C14 C 0.8631(3) 0.4608(3) 0.30561(13) 0.0532(5)
C15 C 0.9423(3) 0.4948(3) 0.23709(15) 0.0644(7)
C16 C 0.8563(3) 0.5917(3) 0.18349(14) 0.0667(7)
C17 C 0.6901(3) 0.6568(3) 0.19729(12) 0.0546(5)
C17A C 0.6090(3) 0.6204(2) 0.26670(10) 0.0396(4)
H5A   H    0.1284   0.4834   0.6221   0.060
H5B   H    0.1861   0.6537   0.5908   0.060
H6A   H   -0.0374   0.4490   0.5050   0.055
H6B   H   -0.1186   0.5762   0.5617   0.055
H13   H    0.6182   0.4397   0.4297   0.046
H14   H    0.9218   0.3972   0.3414   0.064
H15   H    1.0548   0.4527   0.2262   0.077
H16   H    0.9127   0.6130   0.1373   0.080
H17   H    0.6340   0.7227   0.1616   0.066
```

Fig. 2.2.3.1. Typical small-molecule CIF.

Data names are character strings that begin with an underscore character _ and do not contain white-space characters. Data names serve to index data values and are case-insensitive.

Where a data name indexes a single data value, that value follows the data name separated by white space.

Where a data name indexes a set of data values (conceptually a vector or table column), the relevant data items are preceded by the case-insensitive string `loop_` separated by white space.

The examples of Fig. 2.2.3.1 show the use of `loop_` to specify a vector or one-dimensional list of values (the symmetry-equivalent positions) and a tabular or matrix list (the atom-site positions).