INTERNATIONAL TABLES FOR CRYSTALLOGRAPHY

Volume G

DEFINITION AND EXCHANGE
OF CRYSTALLOGRAPHIC DATA

Edited by

SYDNEY HALL AND BRIAN MCMAHON

Associate editors

H. M. Berman
F. C. Bernstein
H. J. Bernstein
P. E. Bourne
I. D. Brown
P. M. D. Fitzgerald
R. W. Grosse-Kunstleve
J. R. Hester
N. Spadaccini
B. H. Toby
J. D. Westbrook

Contributing authors

F. H. Allen, Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge, CB2 1EZ, England. [2.4, 4.1, 4.8]

J. M. Barnard, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England. [2.4, 4.8]

H. M. Berman, 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. [2.6, 3.6, 4.5, 5.5]

H. J. Bernstein, Department of Mathematics and Computer Science, Kramer Science Center, Dowling College, Idle Hour Blvd, Oakdale, NY 11769, USA. [2.2.7, 2.3, 3.7, 4.6, 5.1, 5.4, 5.6]

P. E. Bourne, Research Collaboratory for Structural Bioinformatics, San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0537, USA. [3.6, 4.5]

I. D. Brown, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1. [2.2.7, 3.5, 3.8, 4.1, 4.7]

A. P. F. Cook, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England. [2.4, 2.5, 4.8]

P. J. Ellis, Stanford Linear Accelerator Center, 2575 Sand Hill Road, Menlo Park, CA 94025, USA. [5.6]

Z. Feng, 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. [5.5]

P. M. D. Fitzgerald, Merck Research Laboratories, Rahway, New Jersey, USA. [3.2, 3.6, 4.5]

S. R. Hall, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia. [1.1, 2.1, 2.2, 2.4, 2.5, 2.6, 3.2, 4.1, 4.8, 4.9, 4.10, 5.2, 5.4]

A. P. Hammersley, ESRF/EMBL Grenoble, 6 rue Jules Horowitz, France. [2.3, 4.6]

K. Henrick, EMBL Outstation, The European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, England. [Appendix 3.6.2]

M. A. Hoyland, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England. [5.7]

B. McMahon, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England. [1.1, 2.2.7, 3.1, 3.2, 3.6, 4.5, 5.2, 5.3, 5.7]

G. Madariaga, Departamento de Física de la Materia Condensada, Facultad de Ciencia y Tecnología, Universidad del País Vasco, Apartado 644, 48080 Bilbao, Spain. [3.4, 4.3]

P. R. Mallinson, Department of Chemistry, University of Glasgow, Glasgow G12 8QQ, Scotland. [3.5, 4.4]

N. Spadaccini, School of Computer Science and Software Engineering, University of Western Australia, 35 Stirling Highway, Crawley, Perth, WA 6009, Australia. [2.1, 2.2.7, 5.2]

P. R. Strickland, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England. [5.7]

B. H. Toby, NIST Center for Neutron Research, National Institute of Standards and Technology, Gaithersburg, Maryland 20899-8562, USA. [3.3, 4.2]

K. D. Watenpaugh, retired; formerly Structural, Analytical and Medicinal Chemistry, Pharmacia Corporation, Kalamazoo, Michigan, USA. [3.6, 4.5]

J. 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. [2.2, 2.6, 3.6, 4.5, 4.6, 4.10, 5.5]

E. L. Ulrich, Department of Biochemistry, University of Wisconsin Madison, 433 Babcock Drive, Madison, WI 53706-1544, USA. [Appendix 3.6.2]

H. Yang, 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. [5.5]

Contents

Preface

By S. R. Hall and B. McMahon

Crystallography is a data-intensive discipline. This is true of the measurement and the analytical aspects of crystallographic studies, and so the orderly acquisition and retention of data is of great importance. There is also a need for computational tools that facilitate efficient data-handling processes. In this data-rich environment, *International Tables for Crystallography* provides supporting sources of reference data and guidelines for analysing, interpreting and modelling the data involved in a wide variety of studies.

This volume in the *International Tables* series, *Definition and exchange of crystallographic data*, deals with the precise definition of data items used in crystallography. It focuses on a particular data representation, the Crystallographic Information File (CIF), developed over the last 15 years for the computerized interchange of data important to structural crystallography, and for submissions to crystallographic publications and databases. The need for the volume stems from the IUCr's adoption in 1990 of CIF as the standard for exchanging crystallographic data across the entire community, and for the submission of research articles to *Acta Crystallographica* and other journals. The data items described in this volume appear as standard tags in CIF. Each data item has an associated definition, with attributes that characterize its allowed values and relationships with other defined items. The definitions provide a comprehensive machine-readable description of the information associated with a structural study, from area-detector counts through to the derived molecular geometry, and allow a full annotation suitable for publication as a primary research article.

The volume is arranged in five parts. Part 1 introduces the need and rationale for a standard interchange format, and the IUCr's commissioning and development of CIF in response to that need. Part 2 presents the formal specifications of the file formats involved in the CIF standard, and the related crystallographic binary file (CBF) and molecular information file (MIF) standards. Details are also given of the dictionary definition language (DDL) used to specify the names of data items and their attributes in a machine-readable way. The definitions and attributes of data items in different areas of crystallography are listed in separate dictionary files. Part 3 describes the structure and content of each of the public data dictionaries currently available. It provides commentaries on the dictionaries and guidance on best practice in their use for information interchange in several fields of crystallography. The dictionaries themselves are structured ASCII files that may be used directly by software parsers and validators. Part 4 presents the contents of the dictionaries in a text format that is easier for humans to read. Part 5 describes software applications and libraries that use CIF and related interchange standards.

The data definitions in this volume can be read and used from the printed page, but their principal use is in a computer-software environment. The volume therefore describes the large number of computer programs and libraries contributed by the crystallographic community for handling data files in the CIF format. Many of these programs use the dictionary files directly to validate and exchange data items. Software by its very nature is under continual development, and individual software implementations become obsolete with disturbing rapidity. Nevertheless, we have felt it important and useful to devote considerable space to the software aspects of CIF data exchange. The programs described in the volume illustrate various basic considerations and approaches to data exchange and provide a rich gallery of tools suitable for different applications. The volume also includes a CD-ROM containing many of these software packages, as well as machine-readable versions of the data dictionaries themselves, and many links to related web resources.

This volume therefore contains material that is relevant to both crystallographers and information-technology specialists. The data dictionaries that form the core of the volume provide the knowledge base for automated validation of data submitted for publication and for deposition in databases (tasks in which most structure-determination scientists are involved). Part 5 will be of particular interest to the non-specialist user of CIF, in that it contains practical advice on stand-alone application programs, the publication of crystal structure reports in primary research journals and the deposition of biological macromolecular structures in the Protein Data Bank. Software authors will benefit from the discussions on designing and implementing CIF-aware programs and from the detailed descriptions of a number of available libraries.

The data-representation and knowledge-management techniques used routinely in crystallography today were developed well in advance of recent approaches to information dissemination for the World Wide Web through open protocols. Nevertheless, CIF processes fit well with the enormous contemporary effort to build the so-called semantic web, in which information and knowledge are transported and interpreted by computer. The success of such knowledge management depends on well structured data vocabularies, data models and data structures appropriate for different subject areas. The term used to denote such semantically rich formulations is 'ontology'. We believe that the formal description of crystallographic data and information presented in this volume is an exemplary ontology in this sense, and will be a useful model for workers in related fields.

The volume has taken several years to complete. It covers subject matter that is relatively new and still changing, and indeed much of the material has been undergoing revision even as it was being compiled. Thanks to the determined effort by all involved in its production, this volume provides a comprehensive and up-to-date compendium of the current state of crystallographic data definition. This first edition also serves the role of providing the historical and scientific context for future developments in a rapidly evolving field. It is likely that new editions will be needed before long to stay abreast of such developments.

Since it began, the CIF project has involved the participation and collaboration of many colleagues across the breadth of crystallographic practice. In the production of this volume, we wish first to thank all the authors of the individual chapters for their major contributions. Each chapter was independently and anonymously peer-reviewed, and the reviewers' comments were forwarded to the authors for response. This has added to the time and effort involved on everyone's part, but we believe that the improved presentation and exposition of the detailed and often complex material will be of significant benefit to readers and users. We are grateful to the reviewers for their dedication and enthusiasm during this process. We particularly appreciate the help of many colleagues who have contributed to the CIF project generally and who have made valuable contributions to this volume. They are recognized as Associate Editors in the list of contributors. In addition to the individuals mentioned by name in individual chapters, we thank Doug du Boulay, Eldon Ulrich,

Frank Allen, Bill Clegg, Mark Spackman, Victor Streltsov, Bob Sweet, Howard Flack, Peter Murray-Rust and John Bollinger for thought-provoking discussions and contributions. We thank Philip Coppens for initially supporting the proposal for this volume, Theo Hahn and Hartmut Fuess, the previous and current Chairmen of the Commission on *International Tables for Crystallography*, and the other members of the Commission who welcomed this volume to the series. Thanks also go to Timo Vaalsta of the University of Western Australia for technical help with the CD-ROM. Finally, we are grateful to Nicola Ashcroft of the IUCr editorial office for her careful and insightful technical editing of this work.

SAMPLE PAGES

2.1.3.2. Data name

A data name (or tag) is the identifier of a data value (see Section 2.1.3.3) and is a sequence of non-white-space characters starting with an underscore character <_> (ASCII 95).

Example:

```
_publication_author_address
```

2.1.3.3. Data value

A data value is a text string preceded by its identifying data name. Privileged keywords, such as described in Sections 2.1.3.5 to 2.1.3.8, are excluded from this definition.

2.1.3.4. Data item

A data item is a data value and its associated data name. Each data item stored in a STAR File is specified with this combination.

2.1.3.5. Data loop list

A looped list consists of the keyword `loop_` followed by

(*a*) a sequence of data names (possibly with nested `loop_` constructs); and

(*b*) a sequence of loop packets, each containing data values which are identified in the same order as the data names in (*a*).

A looped list specifies a table of data in which the data names represent the 'header descriptors' for columns of data and the packets represent the rows in the table. Looped lists may be nested to any level. Each loop level is initialized with the `loop_` keyword and is followed by the names of data items in this level. Data values that follow the nested data declarations must be in exact multiples of the number of data names. Each loop level must be terminated with a `stop_`, except the outermost (level 1) which is terminated by either a new data item or the privileged strings indicating a save frame (Section 2.1.3.6), a data block (Section 2.1.3.7), a global block (Section 2.1.3.8) or an end of file.

An example of a simple one-level loop structure is:

```
loop_
    _atom_identity_number
    _atom_type_symbol       1 C   2 C    3 O
```

Nested (multi-level) looped lists contain matching data packets [as per (*b*) above] and an additional `stop_` to terminate each level of data. Here is a simple example of a two-level nested list.

```
loop_
    _atom_id_number
    _atom_type_symbol
    loop_
        _atom_bond_id_1
        _atom_bond_id_2
        _atom_bond_order
             1 C     1 2 single  1 3 double stop_
             2 C     2 1 single             stop_
             3 O     3 1 double             stop_
```

The matching of data names to value packets is applied at each loop level. Initially the data values are matched to the data names listed in the outermost level loop. This process is iterated to successively inner levels. At the innermost loop level, data matching is maintained until a `stop_` is encountered. This returns the matching process to the next outer level. The matching process is recursive until the loop structure is depleted. Here is an example of a three-level loop structure.

```
loop_
    _atomic_name
     loop_
         _level_scheme
         _level_energy
          loop_
              _function_exponent
              _function_coefficient
 hydrogen
        (2)->[2]    -0.485813
        1.3324838E+01    1.0
        2.0152720E-01    1.0 stop_
        (2)->[2]    -0.485813
        1.3326990E+01    1.0
        2.0154600E-01    1.0 stop_
        (2)->[1]    -0.485813
        1.3324800E-01    2.7440850E-01
        2.0152870E-01    8.2122540E-01 stop_
        (3)->[2]    -0.496979
        4.5018000E+00    1.5628500E-01
        6.8144400E-01    9.0469100E-01
        1.5139800E-01    1.0000000E+01 stop_ stop_
```

2.1.3.6. Save frame

A save frame is a set of unique data items wholly contained within a data block. The frame starts with a `save_`*framecode* statement, where the *framecode* is a unique identifying code within the data block. Each frame is closed with a `save_` statement.

Example:

```
data_example
save_phenyl
    _object_class    molecular_fragment
     loop_
         _atom_identity_node
         _atom_identity_symbol 1 C 2 C 3 C 4 C 5 C 6 C
save_
loop_ _molecular_fragments   $ethyl  $phenyl $methyl
```

A save frame has the following attributes:

(*a*) A save frame may contain data items and loop structures but not other save frames [see (*f*)].

(*b*) The scope of the data specified in a save frame is the save frame in which it is specified.

(*c*) Data values in a save frame are distinct from any identical items in the parent data block.

(*d*) A save frame may be referenced within the data block in which it is specified using a data item with a value of `$`*framecode*.

Example:

```
loop_ _amino_acid_seq
      _amino_acid_data   1 $tyr 2 $arg 3 $arg 4 $leu
```

where 'arg', 'tyr' and 'leu' are frame codes identifying three save frames of data.

(*e*) A frame code must be unique within a data block.

(*f*) A save frame may not contain another save frame, but it may contain references to other save frames in the same data block using frame codes.

2.1.3.7. Data block

A data block is a set of data containing any number of unique items and save frames. A data block begins with a `data_`*blockcode* statement, where *blockcode* is a unique identifying name within a file. A data block is closed by another `data_`*blockcode* statement, a `global_` statement or an end of file.

Example:

```
data_rhinovirus
all information relevant to rhinovirus included here

data_influenza
all information relevant to influenza virus
included here
```

(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).

```
data_cyclohexane

_molecule_name_common            cyclohexane

    loop_
        _atom_id
        _atom_type
        _atom_attach_h          1  C  2   2  C  2   3
                  C  2   4  C  2   5  C  2   6  C  2
loop_
    _bond_id_1
    _bond_id_2
    _bond_type_mif
           1 2 S  2 3 S  3 4 S  4 5 S  5 6 S  6 1 S

loop_
    _reference_conformation
                      $chair   $boat   $twisted_boat

save_chair
    loop_
        _atom_id
        _atom_coord_x
        _atom_coord_y
        _atom_coord_z     1   1.579  0.159  0.263
                          2   0.756  0.507 -0.986
                          3   0.825  0.493  1.541
                          4  -0.549 -0.131  1.590
                          5  -1.377  0.222  0.347
                          6  -0.626 -0.158 -0.937
save_

save_boat
    loop_
        _atom_id
        _atom_coord_x
        _atom_coord_y
        _atom_coord_z     1   1.657 -0.426  0.356
                          2   1.031  0.133 -0.927
                          3   0.960  0.133  1.602
                          4  -0.568 -0.040  1.558
                          5  -1.051 -0.738  0.279
                          6  -0.499 -0.028 -0.964
save_

save_twisted_boat
    loop_
        _atom_id
        _atom_coord_x
        _atom_coord_y
        _atom_coord_z     1   0.933  0.922  0.971
                          2   1.186  0.220 -0.368
                          3  -0.119  0.161  1.796
                          4  -1.135 -0.581  0.911
                          5  -1.371  0.181 -0.397
                          6  -0.083  0.236 -1.238
save_
```

Fig. 2.4.4.3. Atom and bond properties for cyclohexane, together with 3D coordinate representations of three alternative conformations: chair, boat and twisted boat.

2.4.4.5. Global blocks

A global block is similar to a data block except that it is opened with a **global_** statement and contains data that are common or 'default' to all subsequent data blocks in a file. Global data items remain active until re-specified in a subsequent data block or global block.

In some applications it may be efficient to place data that are common to all data blocks within a global block. In particular, save frames may be defined within global blocks and then referenced in subsequent data blocks [this statement corrects an error in Hall & Spadaccini (1994)]. Examples of global data are shown in Figs. 2.4.7.1 and 2.4.7.2, in which a variety of frequently referenced structural units are encapsulated within save frames specified in global blocks.

2.4.5. Atoms, bonds and molecular representations

The MIF dictionary (see Chapter 4.8) contains definitions of the principal data items needed to specify molecular connectivity and spatial representations. These definitions are grouped according to purpose or, as referred to in the DDL dictionary language (Hall & Cook, 1995), by category. Categories are formally specified in the MIF dictionary using the data attribute **_category** but they may also be identified from the data-name construction '***_<category>_<subcategory>_<descriptor>***'. Note that data items appearing in the same looped list must belong to the same category.

The values of some data items are restricted, by definition in the MIF dictionary, to standard codes or states. For example, the item **_bond_type_mif** can only have values S, D, T or O as in its dictionary definitions:

S: single (two-electron) bond;
D: double (four-electron) bond;
T: triple (six-electron) bond;
O: other (*e.g.* coordination) bond.

The MIF dictionary plays the important additional role of validating and standardizing data values. This is illustrated with the data item **_display_colour**, which identifies the colours of 'atom' and 'bond' graphical objects. The colour codes or states for this item are specified in its dictionary definitions as a set of permitted red/green/blue (RGB) ratios, and no other colours may be used in a MIF. This has the technical advantage of making colour states searchable for chemical applications.

Fig. 2.4.4.2 shows MIF data for the molecule (+)-3-bromocamphor. The 'atom' list contains the items **_atom_id**, **_atom_type** and **_atom_attach_h**, which identify the chemical properties of the atoms, plus the items **_atom_coord_x**, ***_y** and ***_z**, which specify the 3D molecular structure in Cartesian coordinates [these are taken from diffraction results (Allen & Rogers, 1970)]. The item **_atom_label** is also used with any graphical depiction of the 3D model. The 'bond' loop in this example uses the simple **_bond_type_mif** conventions described above. The data names needed to depict stereochemistry are discussed with examples (Figs. 2.4.8.1, 2.4.8.2 and 2.4.8.3) in Section 2.4.8.

The MIF approach to representing 2D chemical structure separates the specification of chemical atom and bond properties. This provides additional flexibility in the description of the graphical objects, such as atomic nodes and bonded connections. The MIF data required to generate a 2D chemical diagram are shown in Fig. 2.4.4.2. The diagram generated from this data will be in a display area of 500 × 500 coordinate units at a scale of 50 units per cm (the 2D chemical diagram shown in Fig. 2.4.4.2 is not to this scale). The default origin (the bottom left corner of the display area) can be specified with the item **_display_define_origin**. The data used to depict a 2D structure form a two-level loop with the 'atomic' graphical objects at level 1 and the 'bond' graphical objects at level 2. The item **_display_object** has the values '.' (null or no object), 'text' (an element or number string) or 'icon'. The size and colour of the atom site are specified with **_display_size** and **_display_colour**. The bonds connected to each atom site are specified as a sequence of **_display_conn_id** numbers (in loop level 2). These numbers must match one of the

2.5. Specification of the core CIF dictionary definition language (DDL1)

BY S. R. HALL AND A. P. F. COOK

2.5.1. Introduction

The CIF approach to data representation described in Chapter 2.2 is based on the STAR File universal data language (Hall, 1991; Hall & Spadaccini, 1994) detailed in Chapter 2.1. An important advantage of the CIF approach is the self-identification of data items through the use of tag–value tuples. This syntax removes the need for preordained data ordering in a file or stream of data and enables appropriate parsing tools to automate access independently of the data source. In this chapter, we will show that the CIF syntax also provides a higher level of abstraction for managing data storage and exchange – that of defining the meaning of data items (*i.e.* their properties and characteristics) as attribute descriptors linked to the identifying data tag.

Each attribute descriptor specifies a particular characteristic of a data item, and a collection of these attributes can be used to provide a unique definition of the data item. Moreover, placing the definitions of a selected set of data items into a CIF-like file provides an electronic dictionary for a particular subject area. In the modern parlance of knowledge management and the semantic web, such a data dictionary represents a *domain ontology*.

In most respects, data dictionaries serve a role similar to spoken-word dictionaries and as such are an important adjunct to the CIF data-management approach by providing semantic information that is necessary for automatic validation and compliance procedures. That is, prior lexical knowledge of the nature of individual items ensures that each item in a CIF can be read and interpreted correctly *via* the unique tag that is the link to descriptions in a data dictionary file. Because the descriptions in the dictionary are machine-parsable, the semantic information they contain forms an integral part of a data-handling process. In other words, machine-interpretable semantic knowledge embedded in data dictionaries leads directly to the automatic validation and manipulation of the relevant items stored in any CIF.

2.5.1.1. The concept of a dictionary definition language (DDL)

The structure or arrangement of data in a CIF is well understood and predictable because the CIF syntax may be specified succinctly (see Chapter 2.2 for CIF syntax expressed using extended Backus–Naur form). In contrast, the meaning of individual data values in a file is only known if the nature of these items is understood. For CIF data this critical link between the value and the meaning of an item is achieved using an electronic 'data dictionary' in which the definitions of relevant data items are catalogued according to their data name and expressed as attribute values, one set of attributes per item.

The dictionary definitions describe the characteristics of each item, such as data type, enumeration states and relationships between data. The more precise the definitions, the higher the level of semantic knowledge of defined items and the better the efficiency achievable in their exchange. To a large degree, the precision achievable hinges on the attributes selected for use in dictionary definitions, these being the semantic vocabulary of a dictionary. Within this context, attribute descriptors constitute a dictionary definition language (DDL). Definitions of the attributes described in this chapter are provided in Chapter 4.9.

The main purpose of this chapter is to describe the DDL attributes used to construct the core, powder, modulated structures and electron density CIF dictionaries detailed in Chapters 3.2–3.5 and 4.1–4.4. We shall see that each item definition in these dictionaries is constructed separately by appropriate choice from the attribute descriptors available, and that a sequence of definition blocks (one block per item) constitutes a CIF dictionary file. The organization of attributes and definition blocks in a dictionary file need not be related to the syntax of a data CIF, but in practice there are significant advantages if they are. Firstly, a common syntax for data and dictionary files enables the same software to be used to parse both. Secondly, and of equal importance, data descriptions and dictionaries need continual updating and additions, and the CIF syntax provides a high level of extensibility and flexibility, whereas most other formats do not. Finally, the use of a consistent syntax permits the dictionary attribute descriptors themselves to be described in their own DDL dictionary file.

While the basic functionality and flexibility of a dictionary is governed by the CIF syntax, the precision of the data definitions contained within it is determined entirely by the scope and number of the attribute descriptors representing the DDL. Indeed, the ability of a DDL to permit the simple and seamless evolution of data definitions and the scope of the DDL to precisely define data items both play absolutely pivotal roles *via* the supporting dictionaries in determining the power of the CIF data-exchange process.

2.5.2. The organization of a CIF dictionary

The precision and efficiency of a data definition language are directly related to the scope of the attribute vocabulary. In other words, the lexical richness of the DDL depends on the number and the specificity of the available language attributes. The breadth of these attributes, in terms of the number of separate data characteristics that can be specified, largely controls the precision of data definitions. However, it is the functionality of attributes that determines the information richness and enables higher-level definition complexity. For example, the attributes that define child–parent relationships between data and key pointers to items in list packets are essential to understanding the data hierarchy and to its validation. Attributes provide the semantic tools of a dictionary.

The choice and scope of attributes in the DDL are governed by both semantic and technical considerations. Attributes need to have a clear purpose to facilitate easy definition and comprehension, and their routine application in automatic validation processes. A CIF dictionary is much more than a list of unrelated data definitions. Each definition conforms to the CIF syntax, which requires each data block in the dictionary to be unique. However, the functionality of a dictionary involves elements of both relational and object-oriented processes. For example, attributes in one definition may refer to another definition *via* `_list_link_parent` or `_list_link_child` attributes, so as to indicate the dependency

Affiliations: SYDNEY R. HALL, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; ANTHONY P. F. COOK, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England.

```
_chemical_optical_rotation
_chemical_properties_biological
_chemical_properties_physical
_chemical_temperature_decomposition
_chemical_temperature_decomposition_gt
_chemical_temperature_decomposition_lt
_chemical_temperature_sublimation
_chemical_temperature_sublimation_gt
_chemical_temperature_sublimation_lt
```

(*b*) CHEMICAL_FORMULA

```
_chemical_formula_analytical
_chemical_formula_iupac
_chemical_formula_moiety
_chemical_formula_structural
_chemical_formula_sum
_chemical_formula_weight
_chemical_formula_weight_meas
```

The CHEMICAL category itself deals with the large-scale chemical properties of the compound from which the crystal under study was formed: its various formal and common names, its source, melting point, decomposition and sublimation temperatures (as experimentally determined values, or as upper or lower possible values if not measured directly), its biological or physical properties, and where applicable the absolute configuration and optical rotation.

The optical rotation in solution may be reported using the data name **_chemical_optical_rotation** by an expression of the form

$$[\alpha]_W^T = \pm\frac{100\alpha}{lc} \qquad (c = CONC, \mathrm{SOLV}),$$

where $[\alpha]_W^T$ is the signed optical rotation in degrees at temperature T and wavelength labelled by code W, l is the length of the optical cell, *CONC* is the concentration of the solution (given as the mass of the substance in g in a standard 100 ml of solution), and SOLV is the chemical formula of the solvent. This can be marked up within the constraints of the ASCII character set to which CIF is restricted as `[\a]^25^~D~ = +108 (c = 3.42, CHCl~3~)`, where the measurement is taken using the *D* line of the atomic spectrum of sodium.

Data items in the CHEMICAL_FORMULA category describe the chemical formula and formula mass of the compound under study. The quoted formula must reflect the overall stoichiometry of the crystal under study, and must, when multiplied by the Z value **_cell_formula_units_Z**, account for the total contents of the unit cell.

A number of data names are provided to account for different conventions in the presentation of chemical formulae. **_chemical_formula_analytical** is appropriate for a gross formula determined by standard chemical analysis, including all trace elements identified in the sample. Standard uncertainties on the proportions of elements present are acceptable, *e.g.*

```
_chemical_formula_analytical 'Fe2.45(2) Ni1.60(3) S4'
```

_chemical_formula_sum is another aggregate formula, in which all discrete bonded residues and ions are summed over the constituent elements. Where appropriate, the formulae of separate residues of a complex may be described by **_chemical_formula_moiety**, in which the formula for each moiety is supplied as a sum of the individual elements within the moiety, or by **_chemical_formula_structural**, in which sub-components within individual moieties are further identified, so that the overall expression permits the identification of particular bonded groups. Within these formula expressions, certain rules must be observed to allow parsing by software. The final data item relating to the chemical formula, **_chemical_formula_iupac**, is for formulae that are constructed according to the rules of the International Union for Pure and Applied Chemistry.

The ordering and notation rules are explained n detail in the dictionary, but are repeated here for convenience. Within each group of atoms for which a formula is present:

(i) only recognized element symbols may be used;

(ii) each element symbol is followed by a 'count' number ('1' is implicit and may be omitted);

(iii) a space or parenthesis must separate each cluster of (element symbol + count);

(iv) where a group of elements is enclosed in parentheses, the multiplier for the group must follow the closing parentheses. That is, all element and group multipliers are assumed to be printed as subscripted numbers. (An exception to this rule exists for **_chemical_formula_moiety**, where pre- and post-multipliers are permitted for molecular units.)

(v) Unless the elements are ordered in a manner that corresponds to their chemical structure, as in **_chemical_formula_structural**, the order of the elements within any group or moiety depends on whether or not carbon is present. If carbon is present, the order should be: C, then H, then the other elements in alphabetical order of their symbol. If carbon is not present, the elements are listed purely in alphabetic order of their symbol. This is the 'Hill' system used by *Chemical Abstracts*. This ordering is used in **_chemical_formula_moiety** and **_chemical_formula_sum**.

For **_chemical_formula_moiety** some additional rules apply:

(i) Moieties are separated by commas, ','.

(ii) The order of elements within a moiety follows the general rules outlined above as the 'Hill' system.

(iii) Parentheses are *not* used within moieties but may surround a moiety. Parentheses may not be nested.

(iv) Charges should be placed at the end of the moiety. The charge '+' or '-' may be preceded by a numerical multiplier and should be separated from the last (element symbol + count) by a space. Pre- or post-multipliers may be used for individual moieties.

Example 3.2.4.6 illustrates the differences between some of these data items.

3.2.4.2.2. *Chemical connectivity*

The data items in these categories are as follows:

(*a*) CHEMICAL_CONN_ATOM

```
• _chemical_conn_atom_number
  _chemical_conn_atom_charge
  _chemical_conn_atom_display_x
  _chemical_conn_atom_display_y
  _chemical_conn_atom_NCA
  _chemical_conn_atom_NH
  _chemical_conn_atom_type_symbol
```

(*b*) CHEMICAL_CONN_BOND

```
• _chemical_conn_bond_atom_1
        → _chemical_conn_atom_number
• _chemical_conn_bond_atom_2
        → _chemical_conn_atom_number
  _chemical_conn_bond_type
```

The bullet (•) indicates a category key. Where multiple items within a category are marked with a bullet, they must be taken together to form a compound key. The arrow (→) is a reference to a parent data item.

The CHEMICAL_CONN_ATOM category labels the chemical atoms in a connected representation of the molecular species and can also give the coordinates for the atoms in a two-dimensional chemical diagram (Example 3.2.4.7). Each atom may also carry an indication of the number of connected non-hydrogen atoms (**_NCA**) and the number of hydrogen atoms (**_NH**) to which it

Data items revised in these categories are as follows:

(*a*) AUDIT
```
_audit_block_code
```
(*b*) AUDIT_LINK
```
_audit_link_block_code
```

The core dictionary definitions of these items are revised in order to formalize the relationships between multiple data blocks representing reference and modulated structures. Guidance is provided in the msCIF dictionary on how to label data blocks in a way that makes their mutual relationships clear.

3.4.4. Use of the msCIF dictionary

In this section, some of the capabilities of the dictionary will be demonstrated using simple examples. More detailed examples can be found at http://www.iucr.org/iucr-top/cif/ms and on the CD-ROM accompanying this volume.

3.4.4.1. Description of reciprocal space

Modulated and composite structures need more than three reciprocal vectors in order to index the whole set of reflections with integer numbers. Hence a diffraction vector is written as

$$\mathbf{H} = h\mathbf{a}^* + k\mathbf{b}^* + l\mathbf{c}^* + m_1\mathbf{q}_1 + \ldots + m_d\mathbf{q}_d, \qquad (3.4.4.1)$$

where the notation has been chosen according to the core CIF dictionary. In the case of a modulated structure, $\mathbf{a}^*$, $\mathbf{b}^*$ and $\mathbf{c}^*$ are the reciprocal vectors of the reference structure (and therefore h, k and l index the main reflections). $\mathbf{q}_1, \ldots, \mathbf{q}_d$ are the modulation wave vectors. They are three-dimensional vectors with some irrational component (if the modulated structure is incommensurate) in the lattice spanned by $\mathbf{a}^*$, $\mathbf{b}^*$ and $\mathbf{c}^*$. d is the dimension of the modulation. In the case of composite structures, the diffraction pattern can be indexed using $3 + d$ (arbitrarily selected) vectors $\mathbf{a}_k^*$ ($k = 1, \ldots, 3+d$). $\mathbf{a}_1^*$ ($\equiv \mathbf{a}^*$), $\mathbf{a}_2^*$ ($\equiv \mathbf{b}^*$) and $\mathbf{a}_3^*$ ($\equiv \mathbf{c}^*$) normally span the reciprocal lattice of the main reflections of one of the substructures (notice that this is only one particular, but highly intuitive, choice). The remaining d vectors with $k = 4, \ldots, d$ are the wave vectors of the modulation [$\mathbf{q}_1, \ldots, \mathbf{q}_d$ in equation (3.4.4.1)].

In a composite structure, the $(3 + d)$-dimensional reciprocal basis of the subsystem ν is determined by a $(3 + d) \times (3 + d)$ matrix W^ν [see van Smaalen (1995) and references therein]:

$$\mathbf{a}_i^{*\nu} = \sum_{k=1}^{3+d} W_{ik}^\nu \mathbf{a}_k^*, \quad i = 1, \ldots, 3 + d, \qquad (3.4.4.2)$$

where the subscripts $i = 1$, 2 and 3 label the reciprocal vectors $\mathbf{a}^{*\nu}$, $\mathbf{b}^{*\nu}$ and $\mathbf{c}^{*\nu}$, and $i = 4, \ldots, d$ label the wave vectors of the modulation expressed as linear combinations of $\mathbf{a}^{*\nu}$, $\mathbf{b}^{*\nu}$ and $\mathbf{c}^{*\nu}$.

The simplest case corresponds to a one-dimensional ($d = 1$) modulated structure. Consider for example the incommensurate phase of K_2SeO_4. The wave vector of the modulation can be chosen to be $\mathbf{q}_1 = \alpha\mathbf{a}^*$. Relevant information about the diffraction pattern of this compound is expressed using both the core CIF and msCIF dictionaries as shown in Example 3.4.4.1.

A more complicated example is the composite structure $(LaS)_{1.14}NbS_2$. The two mutually incommensurate subsystems (along the a axis) are (van Smaalen, 1991) NbS_2 ($\nu = 1$) and LaS ($\nu = 2$). The reciprocal basis can be chosen to be $\mathbf{a}_1^* = \mathbf{a}^{*1}$, $\mathbf{a}_2^* = \mathbf{b}^{*1}$, $\mathbf{a}_3^* = \mathbf{c}^{*1}$ and $\mathbf{a}_4^* = \mathbf{a}^{*2}$. For this particular choice, the two W matrices [see equation (3.4.4.2)] are

Example 3.4.4.1. *msCIF description of the diffraction pattern of a one-dimensional modulated structure.*

```
_exptl_crystal_type_of_structure         'mod'
_cell_reciprocal_basis_description
; a*,b*,c* (reciprocal basis spanning the lattice of
  main reflections), q modulation wave vector.
;

_diffrn_symmetry_description
; The whole diffraction pattern shows orthorhombic
  symmetry. The following extinction rules were
  detected:
     0kl0   k+l=odd
     h0lm   h+m=odd
     hk0m   m=odd
     h00m   h,m=odd
  Superspace group: P:Pnam:-1ss
;

_diffrn_reflns_satellite_order_max           1

_diffrn_reflns_theta_max                  40.14
_diffrn_reflns_theta_min                   3.32
_diffrn_reflns_limit_h_max                    8
_diffrn_reflns_limit_k_max                   18
_diffrn_reflns_limit_l_max                   10
_diffrn_reflns_limit_index_m_1_max            1
_diffrn_reflns_limit_h_min                    0
_diffrn_reflns_limit_k_min                    0
_diffrn_reflns_limit_l_min                    0
_diffrn_reflns_limit_index_m_1_min           -1

# Modulation wave vector
  loop_
     _cell_wave_vector_seq_id
     _cell_wave_vector_x
         1        0.318(5)
```

$$W^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad W^2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

This information is transcribed to CIF format as shown in Example 3.4.4.2. (Note that the default values for the wave vector components and the elements of W are 0.)

3.4.4.2. Description of symmetry

The symmetry of a modulated or composite structure is described by a superspace group which leaves the $(3 + d)$-dimensional embedding of the structure invariant. Superspace is built of two orthogonal subspaces and both of them are kept invariant separately by the superspace symmetry operations. (In reciprocal space this means that, for these structures, main reflections and satellite reflections are never transformed into one another by superspace symmetry operations.) Consequently, superspace groups are not general $(3 + d)$-dimensional space groups. The standard notation for superspace groups only covers the one-dimensional superspace groups, which are listed in Janssen *et al.* (2004). As a consequence, msCIFs must include a list of all the symmetry operations in an (x, y, z) format (using as symbols $x_1 \ldots x_{3+d}$) similar to that used in the core CIF dictionary. Superspace-group names for one-dimensional structures can be expressed either according to Janssen *et al.* (2004) or according to the original notation of de Wolff *et al.* (1981). Alternative names or higher-dimensional superspace groups can also be included, but not parsed.

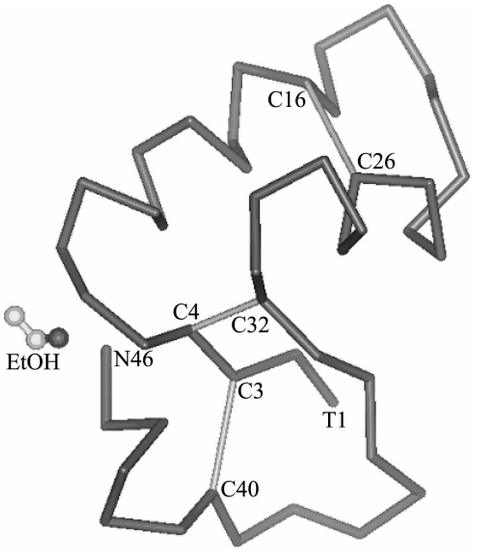

Fig. 3.6.3.1. A representation of crambin (PDB 3CNR) with a co-crystallized ethanol molecule.

(vii) Data items should be present for describing the characteristics and geometry of canonical and non-canonical amino acids, nucleotides, sugars and ligand groups.

(viii) Data items should be provided that permit a detailed description of the chemistry of the component parts of the macromolecule to be given.

(ix) Data items should be present that provide specific pointers from elements of the structure (*e.g.* the sequence, bound inhibitors) to appropriate entries in publicly available databases.

(x) Data items should be present that provide meaningful three-dimensional views of the structure so as to highlight functional and structural aspects of the macromolecule.

(xi) Data items specific to an NMR experiment or modelling study would not in general be included in version 1. However, data items that summarize the features of an ensemble of structures and permit a description of each member of the ensemble to be given should be available.

(xii) A comprehensive set of data items for providing a higher-order structure description (for example, to cover supersecondary structure and functional classification) was considered to be beyond the scope of version 1.

Based on the above, the first version of the mmCIF dictionary with approximately 1700 data items (including those data items taken from the core CIF dictionary) was developed and officially approved in October 1997. Subsequent revisions have increased the number of data items to over 2000. It is not expected that all the data items will be present in every mmCIF data file. Instead, the goal was to provide a wide range of data items from which users can select those that best suit the structure they wish to describe.

3.6.3. Overview of the mmCIF data model

The solution and refinement of a macromolecular structure is complex and often difficult, as there are a large number of atoms in a typical macromolecule, the molecular conformation can be complex and it can be difficult to model included solvent molecules. However, even when a satisfactory structural model has been derived, describing the structure can be a considerable challenge. Using diagrams can help, but two-dimensional projections are often inadequate for illustrating important features and a complete understanding of the three-dimensional structure of a macromolecule can often only be reached by using interactive molecular graphics software.

Example 3.6.3.1. *Specification of the three distinct components of the crambin structure.*

```
loop_
  _struct_asym.id
  _struct_asym.entity_id
  _struct_asym.details
   chain_a A       'single polypeptide chain'
   ethanol ethanol 'cocrystallized ethanol molecule'
   water   HOH     .
```

The mmCIF dictionary provides several ways for describing the structure. The PUBL categories can be used to record text describing the structure. The complete list of atomic coordinates may be used as input for visualization programs that allow a range of wire-frame, stick, space-filling, ribbon or cartoon representations to be generated based upon inbuilt heuristics and user interaction. However, most importantly, the mmCIF approach also offers a large collection of categories which are designed to provide descriptions of the structure at different levels of detail, and the relationships between data items in different categories permit the function of an individual atom site at any particular level of detail to be traced.

Before beginning the detailed description of the full mmCIF dictionary, it is helpful to demonstrate how it is used to describe the structure of a biological macromolecule. Fig. 3.6.3.1 shows the small protein crambin, which is a single polypeptide chain of 48 residues. The molecule co-crystallizes with a molecule of ethanol, although this is not thought to have any biological effect. Almost a quarter of the residues have side chains that adopt alternative conformations, and there is sequence heterogeneity at positions 22 (Pro/Ser) and 25 (Leu/Ile). Three disulfide links stabilize the structure.

The highest level of the description of the structure uses data items from the STRUCT category group. The crystallographic asymmetric unit contains one protein molecule, one co-crystallization ethanol molecule and a water solvent molecule. These are described with data items from the STRUCT_ASYM category (Example 3.6.3.1).

Each entry in this list assigns a label to a discrete component of the asymmetric unit and associates it with an entry in the entity list that defines each distinct chemical species in the crystal (Example 3.6.3.2).

The biological functions of the components of the crystal structure are described using data items in the STRUCT_BIOL and related categories. For crambin, the biological function is still unknown (see Example 3.6.3.3). This example also shows how the biological unit is generated from specific discrete objects in the asymmetric unit. In this case the relationship is trivial, but it will often be much more complex.

The secondary structure of the protein is described using data items in the STRUCT_CONF category (and in the STRUCT_SHEET category where relevant). The beginning and end labels for each

Example 3.6.3.2. *Specification of the distinct chemical entities in the crambin structure.*

```
loop_
  _entity.id
  _entity.type
  _entity.formula_weight
  _entity.src_method
    A        polymer       4716     natural
    ethanol  non-polymer     52     synthetic
    HOH      water           18     .
```

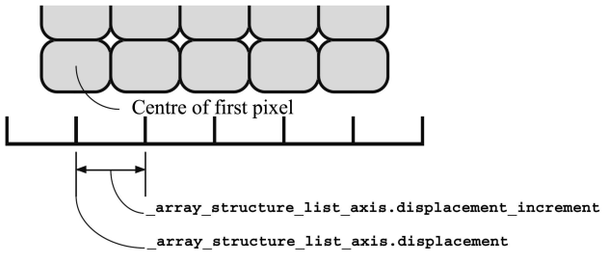

Fig. 3.7.2.1. ARRAY_STRUCTURE_LIST specification of linearly organized image elements.

Note that a spiral scan uses two coupled axes, one for the angular direction and one for the radial direction. This differs from a cylindrical scan for which the two axes are not coupled into one set.

Multiple related axes are gathered together into sets. Each set is identified by the value of the axis set identifier, **_array_structure_list_axis.axis_set_id**, and each axis within a set is identified by the value of **_array_structure_list_axis.axis_id**. Each set given by a value of ***.axis_set_id** is linked to a corresponding value for **_array_structure_list.axis_set_id** to relate settings of the axes in the axis set to particular image elements in ARRAY_STRUCTURE_LIST.

If axes are all independent, no value need be given for **_array_structure_list_axis.axis_set_id**, which is then implicitly given the corresponding value of **_array_structure_list_axis.axis_id**. Each axis given by a value of **_array_structure_list_axis.axis_id** is linked to a corresponding value for **_axis.id** to provide a physical description of the axis. **_array_structure_list_axis.axis_id** and **_array_structure_list_axis.axis_set_id** together uniquely identify a row of data in an ARRAY_STRUCTURE_LIST_AXIS table.

For the remaining data items, there are two important cases to consider: axes that step by Euclidean distance and axes that step by angle. Fig. 3.7.2.1 shows a portion of an array of image elements laid out on a rectangular grid. The starting point of an axis is specified in millimetres by the value of **_array_structure_list_axis.displacement** and the centre-to-centre distance between pixels is specified in millimetres by the value of **_array_structure_list_axis.displacement_increment**.

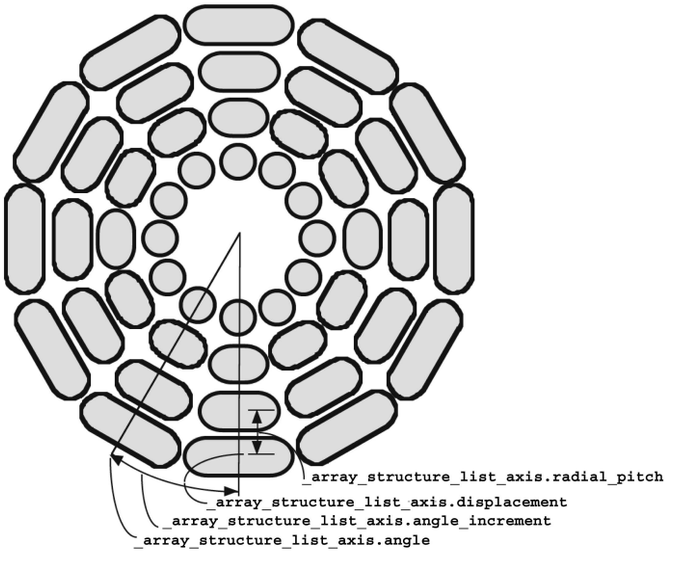

Fig. 3.7.2.2. ARRAY_STRUCTURE_LIST specification of 'constant-angle' image elements in a cylindrical scan. The angular and radial axes are independent. Note that outer-zone image elements are further apart, centre-to-centre, than inner-zone image elements.

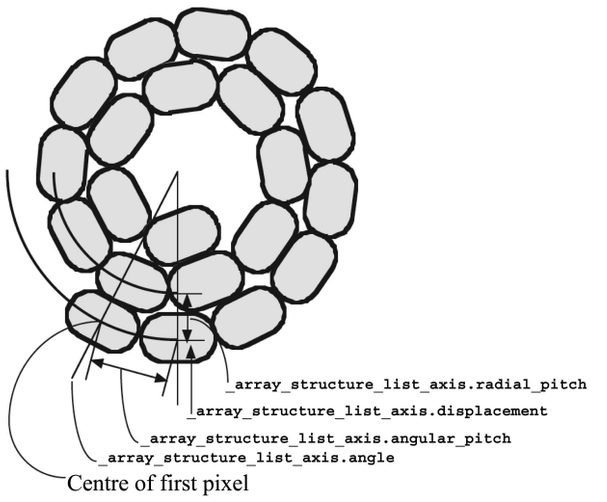

Fig. 3.7.2.3. ARRAY_STRUCTURE_LIST specification of 'constant-velocity' image elements in a cylindrical scan. The angular and radial axes are coupled. Note that outer-zone image elements are the same linear distance apart, centre-to-centre, as the inner-zone image elements.

Fig. 3.7.2.2 shows a portion of an array of image elements laid out in concentric cylinders. The starting point of the angular axis is specified in degrees by the value of **_array_structure_list_axis.angle** and the centre-to-centre angular distance between pixels is specified in degrees by the value of **_array_structure_list_axis.angle_increment**. The starting point of the radial axis is specified by the value of **_array_structure_list_axis.displacement** and the radial distance between cylinders of pixels is specified in millimetres by the value of **_array_structure_list_axis.radial_pitch**. Note that the image elements further from the centre are larger than the image elements closer to the centre.

Fig. 3.7.2.3 shows a portion of a spiral scan array in which the angular and radial axes are coupled. This example represents a 'constant-velocity' scan, in which the size of the image elements does not depend on the distance from the centre. The starting point of the angular axis is again specified in degrees by the value of **_array_structure_list_axis.angle**, but the centre-to-centre distance between pixels is specified in millimetres by the value of **_array_structure_list_axis.angular_pitch**. The coupled radial axis is handled in much the same way as for the uncoupled radial axis in the cylindrical array.

These examples show some of the more common two-dimensional data structures. By coupling an additional axis not in the plane of the first two, regular three-dimensional arrays of data can be represented without additional tags. The categories in the DIFFRN group allow arrays of data to be associated with frames and thereby with time and/or wavelength. More general data structures, for example ones based on dope vectors or hash tables, would require the definition of additional tags, but any data structure (see Aho *et al.*, 1987) that can be handled by a modern computer should be manageable within this framework.

3.7.3. Axes

The category describing the axes required to specify the data collection is as follows:

AXIS group

AXIS

4.2. Powder dictionary (pdCIF)

BY B. H. TOBY

This is version 1.0.1 of the powder CIF dictionary (pdCIF). The data names defined in this dictionary complement those in the core dictionary (Chapter 4.1) and should be used to describe the results of powder diffraction studies. The organization of powder data sets, especially for studies of multiphase samples and for studies referring to external calibration standards, is discussed in Chapter 3.3.

The pdCIF data structure departs from the rigorous relational nature of the core dictionary in that it does not adhere fully to the close coupling of data names and associated category names. The dictionary is therefore presented here strictly alphabetically by data name. Not all pdCIF categories are described in this dictionary. Further, the pdCIF dictionary defines items that belong to categories in the core CIF dictionary. Care must therefore be taken in checking the category membership of each data name; this is particularly important to ensure that items in the same category are presented together in the same looped lists. See Chapters 3.1 and 3.3 for a more complete discussion.

PD_BLOCK

`_pd_block_id` is used to assign a unique ID code to a data block. This code is then used for references between different blocks (see `_pd_block_diffractogram_id`, `_pd_calib_std_external_block_id` and `_pd_phase_block_id`). Note that a data block may contain only a single diffraction data set or information about a single crystalline phase. However, a single diffraction measurement may yield structural information on more than one phase, or a single structure determination may use more than one data set. Alternatively, results from a single data set, such as calibration parameters from measurements of a standard, may be used for many subsequent analyses. Through use of the ID code, a reference made between data sets may be preserved when the file is exported from the laboratory from which the CIF originated. The ID code assigned to each data block should be unique with respect to an ID code assigned for any other data block in the world. The naming scheme chosen for the block-ID format is designed to ensure uniqueness. It is the responsibility of a data archive site or local laboratory to create a catalogue of block IDs if that site wishes to resolve these references.

`_pd_block_diffractogram_id` (*char*)

A block ID code (see `_pd_block_id`) that identifies diffraction data contained in a data block other than the current block. This will occur most frequently when more than one set of diffraction data is used for a structure determination. The data block containing the diffraction data will contain a `_pd_block_id` code matching the code in `_pd_block_diffractogram_id`.

Appears in list. **[pd_proc]**

`_pd_block_id` (*char*)

Used to assign a unique character string to a block. Note that this code is not intended to be parsed; the concatenation of several strings is used in order to generate a string that can reasonably be expected to be unique.

This code is assigned by the originator of the data set and is used for references between different CIF blocks. The ID will normally be created when the block is first created. It is possible to loop more than one ID for a block: if changes or additions are made to the block later, a new ID may be assigned, but the original name should be retained.

The format for the ID code is

$$\langle \textit{date-time} \rangle | \langle \textit{block\_name} \rangle | \langle \textit{creator\_name} \rangle | \langle \textit{instr\_name} \rangle .$$

$\langle$*date-time*$\rangle$ is the date and time the CIF was created or modified. $\langle$*block_name*$\rangle$ is an arbitrary name assigned by the originator of the data set. It will usually match the name of the phase and possibly the name of the current CIF data block (*i.e.* the string *xxxx* in a `data_`*xxxx* identifier). It may be a sample name. $\langle$*creator_name*$\rangle$ is the name of the person who measured the diffractogram, or prepared or modified the CIF. $\langle$*instr_name*$\rangle$ is a unique name (as far as possible) for the data-collection instrument, preferably containing the instrument serial number for commercial instruments. It is also possible to use the Internet name or address for the instrument computer as a unique name.

As blocks are created in a CIF, the original sample identifier (*i.e.* $\langle$*block_name*$\rangle$) should be retained, but the $\langle$*creator_name*$\rangle$ may be changed and the $\langle$*date-time*$\rangle$ will always change. The $\langle$*date-time*$\rangle$ will usually match either the `_pd_meas_datetime_initiated` or the `_pd_proc_info_datetime` entry.

Within each section of the code, the following characters may be used:

`A-Z a-z 0-9 # & * . : , - _ + / ( ) \ [ ]`

The sections are separated with vertical rules '|', which are not allowed within the sections. Blank spaces may also not be used. Capitalization may be used within the ID code but should not be considered significant – searches for data-set ID names should be case-insensitive.

Date–time entries are in the standard CIF format '*yyyy-mm-dd*T*hh*:*mm*:*ss*+*zz*'. Use of seconds and a time zone is optional, but use of hours and minutes is strongly encouraged as this will help to ensure that the ID code is unique.

An archive site that wishes to make CIFs available *via* the web may substitute the URL for the file containing the appropriate block for the final two sections of the ID ($\langle$*creator_name*$\rangle$ and $\langle$*instr_name*$\rangle$). Note that this should not be done unless the archive site is prepared to keep the file available online indefinitely.

May appear in list.

Examples: '`1991-15-09T16:54|Si-std|B.Toby|D500#1234-987`', '`1991-15-09T16:54|SEPD7234|B.Toby|SEPD.IPNS.ANL.GOV`'. **[pd_block]**

PD_CALC

This section is used for storing a computed diffractogram trace. This may be a simulated powder pattern for a material from a program such as *LAZY/PULVERIX* or the computed intensities from a Rietveld refinement.

Affiliation: BRIAN H. TOBY, NIST Center for Neutron Research, National Institute of Standards and Technology, Gaithersburg, Maryland 20899-8562, USA.

4.5. Macromolecular dictionary (mmCIF)

BY P. M. D. FITZGERALD, J. D. WESTBROOK, P. E. BOURNE, B. MCMAHON, K. D. WATENPAUGH AND H. M. BERMAN

This is version 2.0.09 of the macromolecular CIF dictionary (mmCIF). The philosophy behind this dictionary and the history of its development are described in Chapter 1.1. A detailed commentary on the use of the dictionary is given in Chapter 3.6.

Category groups

atom_group	Categories that describe the properties of atoms.
audit_group	Categories that describe dictionary maintenance and identification.
cell_group	Categories that describe the unit cell.
chemical_group	Categories that describe chemical properties and nomenclature.
chem_comp_group	Categories that describe components of chemical structure.
chem_link_group	Categories that describe links between components of chemical structure.
citation_group	Categories that provide bibliographic references.
computing_group	Categories that describe the computational details of the experiment.
compliance_group	Categories that are included in this dictionary specifically to comply with previous dictionaries.
database_group	Categories that hold references to entries in databases that contain related information.
diffrn_group	Categories that describe details of the diffraction experiment.
entity_group	Categories that describe chemical entities.
entry_group	Categories that pertain to the entire data block.
exptl_group	Categories that hold details of the experimental conditions.
geom_group	Categories that hold details of molecular and crystal geometry.
iucr_group	Categories that are used for manuscript submission and internal processing by the staff of the International Union of Crystallography.
pdb_group	Categories that pertain to the file-format or data-processing codes used by the Protein Data Bank.
phasing_group	Categories that describe phasing.
refine_group	Categories that describe refinement.
refln_group	Categories that describe the details of reflection measurements.
struct_group	Categories that contain details about the crystallographic structure.
symmetry_group	Categories that describe symmetry information.

Affiliations: PAULA M. D. FITZGERALD, Merck Research Laboratories, Rahway, New Jersey, USA; 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, USA; PHILLIP E. BOURNE, Research Collaboratory for Structural Bioinformatics, San Diego Supercomputer Center, University of California, San Diego, 9500 Gilman Drive, La Jolla, CA 92093-0537, USA; BRIAN MCMAHON, International Union of Crystallography, 5 Abbey Square, Chester CH1 2HU, England; KEITH D. WATENPAUGH, retired; formerly Structural, Analytical and Medicinal Chemistry, Pharmacia Corporation, Kalamazoo, Michigan, USA; HELEN M. BERMAN, 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, USA.

ATOM_SITE

Data items in the ATOM_SITE category record details about the atom sites in a macromolecular crystal structure, such as the positional coordinates, atomic displacement parameters, magnetic moments and directions. The data items for describing anisotropic atomic displacement factors are only used if the corresponding items are not given in the ATOM_SITE_ANISOTROP category.

Category group(s): **inclusive_group**
atom_group
Category key(s): **_atom_site.id**

Example 1 – based on PDB entry 5HVP and laboratory records for the structure corresponding to PDB entry 5HVP.

```
loop_
_atom_site.group_PDB
_atom_site.type_symbol
_atom_site.label_atom_id
_atom_site.label_comp_id
_atom_site.label_asym_id
_atom_site.label_seq_id
_atom_site.label_alt_id
_atom_site.Cartn_x
_atom_site.Cartn_y
_atom_site.Cartn_z
_atom_site.occupancy
_atom_site.B_iso_or_equiv
_atom_site.footnote_id
_atom_site.auth_seq_id
_atom_site.id
ATOM N  N   VAL  A  11  .  25.369  30.691  11.795  1.00
     17.93  .  11   1
ATOM C  CA  VAL  A  11  .  25.970  31.965  12.332  1.00
     17.75  .  11   2
ATOM C  C   VAL  A  11  .  25.569  32.010  13.808  1.00
     17.83  .  11   3
ATOM O  O   VAL  A  11  .  24.735  31.190  14.167  1.00
     17.53  .  11   4
ATOM C  CB  VAL  A  11  .  25.379  33.146  11.540  1.00
     17.66  .  11   5
ATOM C  CG1 VAL  A  11  .  25.584  33.034  10.030  1.00
     18.86  .  11   6
ATOM C  CG2 VAL  A  11  .  23.933  33.309  11.872  1.00
     17.12  .  11   7
ATOM N  N   THR  A  12  .  26.095  32.930  14.590  1.00
     18.97  4  12   8
ATOM C  CA  THR  A  12  .  25.734  32.995  16.032  1.00
     19.80  4  12   9
ATOM C  C   THR  A  12  .  24.695  34.106  16.113  1.00
     20.92  4  12  10
ATOM O  O   THR  A  12  .  24.869  35.118  15.421  1.00
     21.84  4  12  11
ATOM C  CB  THR  A  12  .  26.911  33.346  17.018  1.00
     20.51  4  12  12
ATOM O  OG1 THR  A  12  3  27.946  33.921  16.183  0.50
     20.29  4  12  13
```

4.7. Symmetry dictionary (symCIF)

BY I. D. BROWN

This is version 1.0.1 of the symmetry CIF dictionary (symCIF), which gives a more complete description of symmetry than was included in the original core CIF dictionary. A detailed commentary on the philosophy behind the dictionary and its use may be found in Chapter 3.8.

SPACE_GROUP

Contains all the data items that refer to the space group as a whole, such as its name, Laue group *etc.* It may be looped, for example in a list of space groups and their properties. Space-group types are identified by their number as listed in *International Tables for Crystallography* Volume A, or by their Schoenflies symbol. Specific settings of the space groups can be identified by their Hall symbol, by specifying their symmetry operations or generators, or by giving the transformation that relates the specific setting to the reference setting based on *International Tables* Volume A and stored in this dictionary. The commonly used Hermann–Mauguin symbol determines the space-group type uniquely but several different Hermann–Mauguin symbols may refer to the same space-group type. A Hermann–Mauguin symbol contains information on the choice of the basis, but not on the choice of origin.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

Mandatory category.

Category key(s): **`_space_group.id`**

Example 1 – description of the C2/c space group, No. 15 in International Tables for Crystallography Volume A.

```
_space_group.id                    1
_space_group.name_H-M_ref          'C 2/c'
_space_group.name_Schoenflies      C2h.6
_space_group.IT_number             15
_space_group.name_Hall             '-C 2yc'
_space_group.Bravais_type          mS
_space_group.Laue_class            2/m
_space_group.crystal_system        monoclinic
_space_group.centring_type         C
_space_group.Patterson_name_H-M    'C 2/m'
```

`_space_group.Bravais_type` (*char*)

The symbol denoting the lattice type (Bravais type) to which the translational subgroup (vector lattice) of the space group belongs. It consists of a lower-case letter indicating the crystal system followed by an upper-case letter indicating the lattice centring. The setting-independent symbol *mS* replaces the setting-dependent symbols *mB* and *mC*, and the setting-independent symbol *oS* replaces the setting-dependent symbols *oA*, *oB* and *oC*.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed., p. 15. Dordrecht: Kluwer Academic Publishers.

The data value must be one of the following:

`aP`
`mP mS`
`oP oS oI oF`
`tP tI`
`hP hR`
`cP cI cF`

Example: '`aP`' (triclinic (anorthic) primitive lattice). **`[space_group]`**

`_space_group.IT_coordinate_system_code` (*char*)

A qualifier taken from the enumeration list identifying which setting in *International Tables for Crystallography* Volume A (2002) (*IT*) is used. See *IT* Table 4.3.2.1, Section 2.2.16, Table 2.2.16.1, Section 2.2.16.1 and Fig. 2.2.6.4. This item is not computer-interpretable and cannot be used to define the coordinate system. Use **`_space_group.transform_`*** instead.

Reference: *International Tables for Crystallography* (2002). Volume A, *Space-group symmetry*, edited by Th. Hahn, 5th ed. Dordrecht: Kluwer Academic Publishers.

The data value must be one of the following:

`b1`	monoclinic unique axis *b*, cell choice 1, **abc**
`b2`	monoclinic unique axis *b*, cell choice 2, **abc**
`b3`	monoclinic unique axis *b*, cell choice 3, **abc**
`-b1`	monoclinic unique axis *b*, cell choice 1, **c$\bar{b}$a**
`-b2`	monoclinic unique axis *b*, cell choice 2, **c$\bar{b}$a**
`-b3`	monoclinic unique axis *b*, cell choice 3, **c$\bar{b}$a**
`c1`	monoclinic unique axis *c*, cell choice 1, **abc**
`c2`	monoclinic unique axis *c*, cell choice 2, **abc**
`c3`	monoclinic unique axis *c*, cell choice 3, **abc**
`-c1`	monoclinic unique axis *c*, cell choice 1, **ba$\bar{c}$**
`-c2`	monoclinic unique axis *c*, cell choice 2, **ba$\bar{c}$**
`-c3`	monoclinic unique axis *c*, cell choice 3, **ba$\bar{c}$**
`a1`	monoclinic unique axis *a*, cell choice 1, **abc**
`a2`	monoclinic unique axis *a*, cell choice 2, **abc**
`a3`	monoclinic unique axis *a*, cell choice 3, **abc**
`-a1`	monoclinic unique axis *a*, cell choice 1, **$\bar{a}$cb**
`-a2`	monoclinic unique axis *a*, cell choice 2, **$\bar{a}$cb**
`-a3`	monoclinic unique axis *a*, cell choice 3, **$\bar{a}$cb**
`abc`	orthorhombic
`ba-c`	orthorhombic
`cab`	orthorhombic
`-cba`	orthorhombic
`bca`	orthorhombic
`a-cb`	orthorhombic
`1abc`	orthorhombic origin choice 1
`1ba-c`	orthorhombic origin choice 1
`1cab`	orthorhombic origin choice 1
`1-cba`	orthorhombic origin choice 1
`1bca`	orthorhombic origin choice 1
`1a-cb`	orthorhombic origin choice 1
`2abc`	orthorhombic origin choice 2
`2ba-c`	orthorhombic origin choice 2
`2cab`	orthorhombic origin choice 2
`2-cba`	orthorhombic origin choice 2
`2bca`	orthorhombic origin choice 2
`2a-cb`	orthorhombic origin choice 2
`1`	tetragonal or cubic origin choice 1
`2`	tetragonal or cubic origin choice 2
`h`	trigonal using hexagonal axes
`r`	trigonal using rhombohedral axes

`[space_group]`

Affiliation: I. DAVID BROWN, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada L8S 4M1.

and a mechanism for representing metadata (*e.g.* as dictionaries or schemas). Four are of particular importance in crystallography: CIF, ASN.1, HDF and XML.

As noted in Chapter 1.1, CIF was created to rationalize the publication process for small molecules. It combines a very simple tag–value data representation with a dictionary definition language (DDL) and well populated dictionaries. CIF is table-oriented, naturally row-based, has case-insensitive tags and allows two levels of nesting. CIF is order-independent and uses its dictionaries both to define the meanings of its tags and to parameterize its tags. It is interesting to note that, even though CIF is defined as order-independent, it effectively fills the role of an order-dependent markup language in the publication process. We will discuss this issue later in this chapter.

Abstract Syntax Notation One (ASN.1) (Dubuisson, 2000; ISO, 2002) was developed to provide a data framework for data communications, where great precision in the bit-by-bit layout of data to be seen by very different systems is needed. Although targeted for communications software, ASN.1 is suitable for any application requiring precise control of data structures and, as such, primarily supports the metadata of an application, rather than the data. ASN.1 can be compiled directly to C code. The resulting C code then supports the data of the application. ASN.1 notation found application in NCBI's macromolecular modelling database (Ohkawa *et al.*, 1995). ASN.1 has case-sensitive tags and allows case-insensitive variants. It manages order-dependent data structures in a mixed order-dependent/order-independent environment.

HDF (NCSA, 1993) is 'a machine-independent, self-describing, extendible file format for sharing scientific data in a heterogeneous computing environment, accompanied by a convenient, standardized, public domain I/O library and a comprehensive collection of high quality data manipulation and analysis interfaces and tools' (http://ssdoo.gsfc.nasa.gov/nost/formats/hdf.html). HDF was adopted by the Neutron and X-ray Data Format (NeXus) effort (Klosowski *et al.*, 1997). HDF allows the building of a complete data framework, representing both data and metadata. Two parallel threads of software development, focused on the management and exchange of raw data from area detectors, began in the mid-1990s: the Crystallographic Binary File (CBF) (Hammersley, 1997) and NeXus. The volumes of data involved were daunting and efficiency of storage was important. Therefore both proposed formats assumed a binary format. CBF was based on a combination of CIF-like ASCII headers with compressed binary images. NeXus was based on HDF. The first API for CBF was produced by Paul Ellis in 1998. CBF rapidly evolved into CBF/imgCIF with a complete DDL2 dictionary and a fully CIF-compliant API (Chapter 5.6). As of mid-2004, NeXus was still evolving (see http://www.nexus.anl.gov/).

XML is a simplified form of SGML, drawing on years of development of tools for SGML and HTML. XML is tree-oriented with case-sensitive entity names. It allows unlimited nesting and is order-dependent. Metadata are managed as a 'document type definition' (DTD), which provides minimal syntactic information, or as schemas, which allow for more detail and are more consistent with database conventions. In fields close to crystallography, the first effort at adopting XML was the chemical markup language (CML) (Murray-Rust & Rzepa, 1999). CML is intentionally imprecise in its ontology to allow for flexibility in development. The CSD and PDB have released their own XML representations (http://www.ccdc.cam.ac.uk/support/prods_doc/relibase/Sysdocn1119.html; http://www.rcsb.org/pdb/newsletter/2003q2/xml_beta.html).

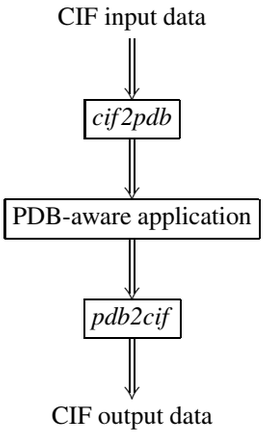

Fig. 5.1.3.1. Example of using filters to make a PDB-aware application CIF-aware.

It may seem from this discussion that the application designer faces an unmanageable variety of data frameworks in an unstable, evolving environment. To some extent this is true. Fortunately, however, there are signs of convergence on CIF dictionary-based ontologies and the use of transliterated CIFs. This means that an application adapted to CIF should be relatively easy to adapt to other data frameworks.

5.1.3. Strategies in designing a CIF-aware application

There are multiple strategies to consider when designing a CIF-aware application. One can use external filters. One can use existing CIF-aware libraries. One can write CIF-aware code from scratch.

5.1.3.1. Working with filter utilities

One solution to making an existing application aware of a new data format is to leave the application unchanged and change the data instead. For almost all crystallographic formats other than CIF, the Swiss-army knife of conversion utilities is *Babel* (Walters & Stahl, 1994). *Babel* includes conversions to and from PDB format. Therefore, by the use of *cif2pdb* (Bernstein & Bernstein, 1996) and *pdb2cif* (Bernstein *et al.*, 1998) combined with *Babel*, many macromolecular applications can be made CIF-aware without changing their code (see Figs. 5.1.3.1 and 5.1.3.2). If the need is to extract mmCIF data from the output of a major application, the PDB provides *PDB_EXTRACT* (http://sw-tools.pdb.org/apps/PDB_EXTRACT/).

Creating a filter program to go from almost any small-molecule format to core CIF is easy. In many cases one need only insert the appropriate '`loop_`' headers. Creating a filter to go from CIF to a particular small-molecule format can be more challenging, because a CIF may have its data in any order. This can be resolved by use of *QUASAR* (Hall & Sievers, 1993) or *cif2cif* (Bernstein, 1997), which accept request lists specifying the order in which data are to be presented (see Fig. 5.1.3.3).

There are a significant and growing number of filter programs available. Several of them [*QUASAR*, *cif2cif*, *ciftex* (ftp://ftp.iucr.org/pub/ciftex.tar.Z) (to convert from CIF to TEX) and ZINC (Stampf, 1994) (to unroll CIFs for use by Unix utilities)] are discussed in Chapter 5.3. In addition there are *CIF2SX* by Louis J. Farrugia (http://www.chem.gla.ac.uk/~louis/software/utils/), to convert from CIF to *SHELXL* format, and *DIFRAC* (Flack *et al.*, 1992) to translate many diffractometer output formats to CIF. The program *cif2xml* (Bernstein & Bernstein, 2002) translates from CIF to XML and CML. The PDB

5.3.2.1.3. *Limitations of vcif*

Because the program is testing certain properties of character strings within logical lines of a file, it stores a line at a time for further internal processing. If a line contains a null character (an ASCII character with integer value zero), this will be taken as the termination of the string currently being processed, according to the normal conventions in the C programming language for marking the end of a text string. In this case, subsequent error messages may not reflect the real problem. The null character, of course, is not allowed in a CIF.

vcif also interprets syntax rules literally, so a misplaced semicolon might mean that a large section of the file is regarded as a text field and too many or too few error messages are generated. This can make a correct interpretation of the causative errors difficult for a novice user.

5.3.3. Editors with graphical user interfaces

A useful class of editing tool is the graphical editor, where different types of access can be provided through icons, windows or frames, menus and other graphical representations. The availability of standard instructions through drop-down menus makes such tools particularly suitable for users who are not expert on the fine details of the file format. The ability within the program to restrict access to particular regions of the file makes it easier to modify the contents of a CIF without breaking the syntax rules. A small but growing number of such editors are becoming available, such as those described here.

5.3.3.1. *enCIFer*

The program *enCIFer* (Allen *et al.*, 2004) has been developed as a graphical utility designed to indicate clearly to a novice user where errors are present in a CIF, to permit interactive editing and revalidation of the file, and to allow visualization of three-dimensional structures described in the file. In its early releases, it was targeted at the community of small-molecule crystallographers interested in publishing structures or depositing them directly in a structure database. Version 1.0 depended on a compiled version of the CIF core dictionary, but subsequent versions allow external CIF dictionaries to be imported. At the time of publication (2005), development is concentrating on support for DDL1 dictionaries.

Given its target user base, the purpose of the program is to permit the following operations within single- or multi-block CIFs:

(i) Location and reporting of syntax and/or format violations using the current CIF dictionary.

(ii) Correction of these syntax and/or format violations.

(iii) Editing of existing individual data items or looped data items.

(iv) Addition of new individual data items or looped data items.

(v) Addition of some standard additional information *via* two data-entry utilities prompting the user for required input ('wizards'): the *publication wizard*, for entering the basic bibliographic information required by most journals and databases that accept CIFs for publication or deposition; and the *chemical and crystal data wizard*, for entering chemical and physical property information in a CIF for publication in a journal or deposition in a database.

(vi) Visualization of the structure(s) in the CIF.

In all cases where data are edited or added, *enCIFer* can be used to check the format integrity of the amended file.

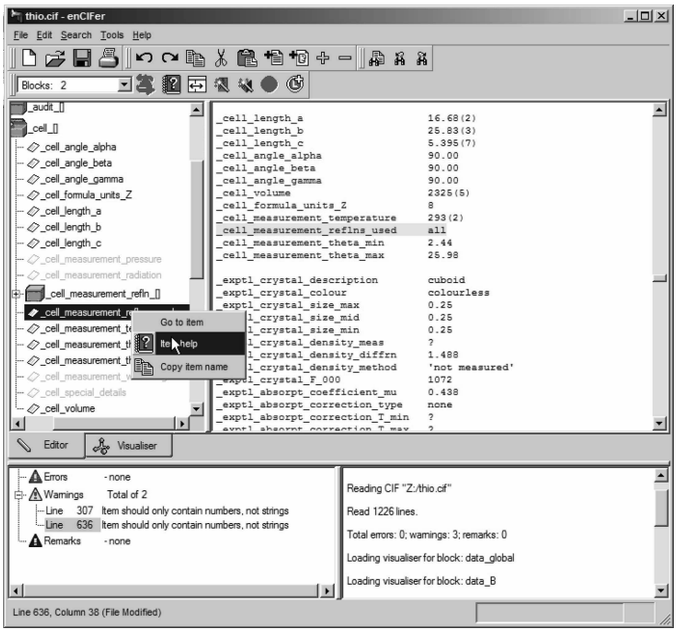

Fig. 5.3.3.1. The *enCIFer* graphical user interface.

5.3.3.1.1. *The main graphical window*

Fig. 5.3.3.1 is an example of the use of *enCIFer* to read and modify a CIF. The figure shows the components of the main window after a file has been opened. Beneath the standard toolbar that provides access to operating-system utilities and to the main functions of the program itself is a task bar (here split over two lines) providing rapid access to a subset of the program's features. Under this are two large panes. The pane on the right is the editing window, where the content of the CIF is displayed and may be modified. The left-hand pane is a user-selectable view by category of the data names stored in the CIF dictionary against which the file is to be validated. At the bottom are two smaller panes. The one on the right logs the session activities and displays informational messages. The left-hand pane lists errors and warning notices generated by the validation system. Errors are labelled by line number, and selection of a specific message (by a mouse double-click) scrolls the content of the main text-editing window to that line number.

Tabs in the middle of the display allow the user to switch rapidly between the editing mode and a visualization of the three-dimensional structures described in the CIF.

These components are described more fully below, followed by a description of the other windows that may be created by a user: the help viewer, the loop editor and the data-entry wizards.

5.3.3.1.2. *The interface toolbar*

This toolbar provides menus labelled 'File', 'Edit', 'Search', 'Tools' and 'Help' that provide the expected functionality of graphical interfaces: the ability to open, close and save files, store a list of recently accessed files, spawn help and other windows, allow searching for strings within the document, allow the user to modify aspects of the behaviour or the look and feel of the program, and provide entry points for specific modes of operation. The most useful of these utilities can also be accessed from icons on the task bar. They are discussed in more detail in the following section.

This main menu is structured in a way familar to users of popular applications designed for the Microsoft Windows operating

5.5.2.3. Supporting other data formats and data delivery methods

One of the greatest benefits of a dictionary-based informatics infrastructure is the flexibility that it provides in supporting alternative data formats and delivery methods. Because the data and all of their defining attributes are electronically encoded, translation between data and dictionary formats can be achieved using lightweight software filters without loss of any information.

XML provides a particularly good example of the ease with which data can be converted to and from the mmCIF format. XML translations of mmCIF data files are currently provided on the RCSB PDB beta ftp site (ftp://beta.rcsb.org/pub/pdb/uniformity/data/XML/). These XML files use mmCIF dictionary data-item names as XML tags. These files were created by a translation tool (http://sw-tools.pdb.org/apps/MMCIF-XML-UTIL/) that translates mmCIF data files to XML in compliance with an XML schema. The XML schema is similarly software-translated from the PDB exchange data dictionary.

Other delivery methods such as Corba (http://www.omg.org/cgi-bin/doc?lifesci/00-02-02) do not require a data format, as data are exchanged using an application program interface (API). A Corba API for macromolecular structure (Greer *et al.*, 2002) based on the content of the mmCIF data dictionary has been approved by the Object Management Group (OMG). Software tools supporting this Corba API (*OpenMMS*, http://openmms.sdsc.edu, and *FILM*, http://sw-tools.pdb.org/apps/FILM) take full advantage of the data dictionary in building the interface definitions and supporting server on which the API is based (see also Section 5.3.8.2).

5.5.3. Integrated data-processing system: overview

The RCSB PDB data-processing system has been designed to take full advantage of the features of the mmCIF metadata framework. The AutoDep Input Tool (*ADIT*) is an integrated data-processing system developed to support deposition, data processing and annotation of three-dimensional macromolecular structure data.

This system, which is outlined in Fig. 5.5.3.1, accepts experimental and structural data from a user for deposition. Data are input in the form of data files or through a web-based form interface. The input data can be validated in a very basic sense for syntax compliance and internal consistency. Other computational validation can also be applied, including checking the input structure data against a variety of community standard geometrical criteria and comparing the input experimental data with the derived structure model. The suite of validation software used within *ADIT* is distributed separately (http://sw-tools.pdb.org/apps/VAL/). All of this validation information is returned to the user as a collection of HTML reports.

In addition to providing data-validation reports, *ADIT* also encodes data in archival data files and loads data into a relational database. The loading of data into the relational database is aided by an expert annotator. The *ADIT* system customizes its behaviour according to the user's requirements. One important distinction is between the behaviour of the interface provided for depositing data and that of the interface used for annotating the data. The depositor is focused only on data collection and provides the simplest possible presentation of the information to be input. The annotator sees the detail of all possible data items as well as the full functionality of the supporting data-processing software and database system.

Although the *ADIT* system was originally developed to support the centralized data deposition and annotation of macromolecular structure data, it is not limited to these particular applications. Because the architecture of the *ADIT* system derives the full scope of information to be processed from a data dictionary, the system can transparently provide data input and processing functionality for any content domain. This feature has been exploited in building a data-input tool for the BioSync project (Kuller *et al.*, 2002). The *ADIT* system can also be configured in workstation mode to provide single-user data collection and processing functionality. This version of the *ADIT* system as well as the supporting mmCIF parsing and data-management tools are currently distributed by the RCSB PDB under an open-source licence (http://sw-tools.pdb.org/apps/ADIT).

5.5.3.1. *ADIT*: functional description

The basic functions of the *ADIT* deposition system are shown in Fig. 5.5.3.2. Users interact with the *ADIT* system through a web server. The CGI components of the *ADIT* system (that is, functional software components interacting with web input data through the Common Gateway Interface protocol) dynamically build the HTML that provides the system user interface. These CGI components are currently implemented as compiled binaries from C++ source code.

User data can be provided in the form of data files or as keyboard input. Input files can be accepted in a variety of formats. *ADIT* uses a collection of format filters to convert input data to the data specification defined in a persistent data dictionary. Data in the form of data files are typically loaded first. Any input data that are not included in uploaded files can be keyed in by the user. *ADIT* builds a set of HTML forms for each category of data to be input. At any point during an input session, a user may choose to view or deposit the input data. Users who are depositing data may also use the data-validation services through the *ADIT* interface.

Comprehensive data ontologies like the PDB exchange dictionary contain vast numbers of data definitions. A data-input application may only need to access a small fraction of these definitions at any point. To address the problem of selecting only the relevant set of input data items from a data dictionary *ADIT* uses a view database. In addition to defining the scope of the data items to be edited by the *ADIT* application, an *ADIT* data view also stores

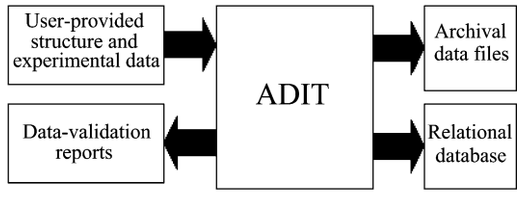

Fig. 5.5.3.1. Functional diagram of the *ADIT* system.

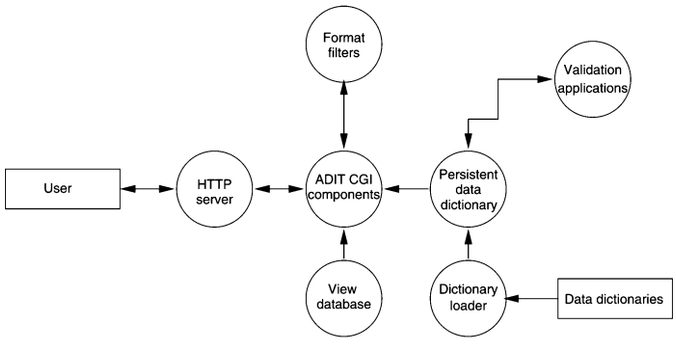

Fig. 5.5.3.2. Schematic diagram of *ADIT* editing, format translation and validation functions.