J. Appl. Cryst. (1993). **26**, 469–473

CIF Applications. I. *QUASAR*: for Extracting Data from a CIF*

BY SYDNEY R. HALL

Crystallography Centre, University of Western Australia, Nedlands 6009, *Australia*

AND ROLF SIEVERS

Institut für Anorganische Chemie der Universität, Gerhard-Domagk-Strasse, Bonn, Germany

(*Received* 12 *December* 1992; *accepted* 15 *January* 1993)

Abstract

The program *QUASAR* performs two basic functions. It extracts specific data items from an input CIF and places them into an output CIF format and it tests the input CIF for logical integrity (*i.e.* that the file conforms to the STAR File syntax). *QUASAR* is written in Fortran77 and has been implemented on a wide range of computers. It is available as public-domain software.

Introduction

The Crystallographic Information File (CIF: Hall, Allen & Brown, 1991) has been recommended by the IUCr Commissions on Crystallographic Data and on Journals for electronic data exchange and archiving. From January 1992, the CIF is the preferred method for submitting machine-readable manuscripts to *Acta Crystallographica* (Allen, Bugg & Maslen, 1991). Considerable effort has gone into the preparation of computer software aimed at facilitating these objectives. Despite the relatively wide distribution of these programs, some users and developers are unaware of their availability.

This is the first of a series of papers describing computer programs written specifically for reading, writing, manipulating and validating CIFs. The description of *QUASAR* starts this series because it is the first program developed specifically to manipulate CIF data. Preliminary reports on *QUASAR* have appeared in the introductory STAR and CIF papers (Hall, 1991; Hall, Allen & Brown, 1991). The earliest version of *QUASAR* (July 1987) was used to test the feasibility of the STAR File process as a suitable universal archive and exchange format. It was the success of these trials that led to the decision by the IUCr Working Party on Crystallographic Information to develop a crystallographic application of the STAR File; namely the CIF. *QUASAR* was, therefore, the prototype CIF-processing software and today it remains the conformance standard for testing a CIF; *i.e.* if *QUASAR* cannot read data from a given file, then that file is not a properly constructed CIF! Currently, *QUASAR* is the principal interface to the *ciftex* publication system (McMahon, 1993), which is used to process manuscripts submitted as CIFs to *Acta Crystallographica* Section C.

Although the functions and facilities provided by *QUASAR* have evolved considerably since 1987, the version described in this paper is effectively unchanged since the publication of the CIF Core Dictionary (Hall *et al.*, 1991).

Other computer programs to be described in this series are *CIFIO*, an *Xtal* (Hall & Stewart, 1990) routine for generating and reading two different types of CIFs; *CYCLOPS*, which is for validating CIF data names in any ASCII file against a standard CIF dictionary file; *CIFtbx*, which is a tool box of routines for reading and writing a CIF; and *CIFER* (Allen & Edgington, 1993), which is for updating a CIF with missing standard items. Details of the program *DIFRAC* for converting diffractometer data to a CIF have already been reported (Flack, 1992). A number of other CIF-processing programs are under development and it is anticipated that these will also be published in this series.

Functions and features

The basic function of *QUASAR* is to extract specified data items from an existing CIF and place them into a new CIF. The order of data in the new CIF is determined by the order that data items are requested, not by the order of data in the original CIF. *QUASAR* may be used therefore to *reorder* as well as to *filter* data.

The basic features of *QUASAR* are as follows:

(*a*) Each data item is requested by data name. These names form a list which is input to *QUASAR* as a file referred to as the *request file*. Table 1(*a*) shows a simple example of a request file and Table 2 shows the CIF file to which it is applied.

(*b*) The list of requested data names in the request file *must* be preceded by a data-block statement. This specifies

* This paper is one of a series of papers on CIF applications. Offprints are available from The Technical Editor, 5 Abbey Square, Chester CH1 2HU, England. See text of paper for availability of program(s) by email.

0021-8898/93/030469-05$06.00

Table 1. *Example request files*

(*a*) An example request file to be used with `qtest.cif` (see Table 2)

```
star_arc_qtest.cif
star_out_qtest.out

data_                                          #<< wild card block name -- accepts first

_atom_site_fract_                              #<< this requests all fractional coord items
_atom_site_label
_atom_site_aniso_LABEL                         #<< capitals to test case insensitivity
_rubbish_to_see_what_happens                   #<< request something that isn't in the CIF
_atom_site_aniso_U_11

data_P6122
_                                              #<< this requests all data in this block
```

(*b*) An alternate request file to test `qtest.cif` (see Table 2) for logical integrity

```
star_arc_qtest.cif
star_log
```

Table 2. *Example input file named* `qtest.cif`

```
data_P6122

loop_
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_scat_dispersion_REAL                    #<< capitals to test case insensitivity
_atom_type_scat_dispersion_imag
_atom_type_scat_source
     S          0          6      .319       .557       'Int Tab Vol III p202 Tab. 3.3.1A'
     O          0          6      .047       .032       'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'
     C          0         20      .017       .009       'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'
     RU         0          1      -.105      3.296      'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
_atom_site_type_symbol
     s      .20200     .79800     .91667     .030(3)    Uij        ?          ?          s
     o      .49800     .49800     .66667     .02520     Uiso       ?          ?          o
     c1     .48800     .09600     .03800     .03170     Uiso       ?          ?          c

loop_
_atom_site_aniso_label
_atom_site_aniso_U_11
_atom_site_aniso_U_22
_atom_site_aniso_U_33
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_23
_atom_site_aniso_type_symbol
     s      .035(4)    .025(3)    .025(3)    .013(1)    .00000     .00000     s
```

the data block from which the items are extracted. More than one data block may be specified in the same request file. See the example in Table 1(*a*).

(*c*) The extraction process is insensitive to the character case (upper or lower) of the data names in the request file or the input CIF. The case of the output data names is consistent with the case of the requested data names. See Tables 1, 2 and 3.

(*d*) A wild-card option is available for requested data names and data-block names. A trailing underline character (_) signals a request for all names that match the preceding character string. For example, the request for

Table 3. *The CIF output from QUASAR after entering the request file in Table* 1(*a*)

```
data_P6122

loop_
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_label
    .20200      .79800      .91667      s
     .49800      .49800       .66667     o
     .48800      .09600       .03800     c1

loop_
_atom_site_aniso_label
_rubbish_to_see_what_happens                          # requested item not present
_atom_site_aniso_U_11
     s          ?        .035(4)

#                                        -----end-of-data-block-----

data_P6122

loop_
_atom_type_symbol
_atom_type_oxidation_number
_atom_type_number_in_cell
_atom_type_scat_dispersion_REAL
_atom_type_scat_dispersion_imag
_atom_type_scat_source
     S          0          6      .319       .557       'Int Tab Vol III p202 Tab. 3.3.1A'
     O          0          6      .047       .032       'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'
     C          0         20      .017       .009       'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'
     RU         0          1      -.105     3.296       'Cromer,D.T. & Mann,J.B. 1968 AC A24,321.'

loop_
_atom_site_label
_atom_site_fract_x
_atom_site_fract_y
_atom_site_fract_z
_atom_site_U_iso_or_equiv
_atom_site_thermal_displace_type
_atom_site_calc_flag
_atom_site_calc_attached_atom
_atom_site_type_symbol
     s      .20200      .79800      .91667      .030(3)     Uij         ?          ?          s
     o      .49800      .49800      .66667      .02520      Uiso        ?          ?          o
     c1     .48800      .09600      .03800      .03170      Uiso        ?          ?          c

loop_
_atom_site_aniso_label
_atom_site_aniso_U_11
_atom_site_aniso_U_22
_atom_site_aniso_U_33
_atom_site_aniso_U_12
_atom_site_aniso_U_13
_atom_site_aniso_U_23
_atom_site_aniso_type_symbol
     s      .035(4)     .025(3)      .025(3)     .013(1)     .00000      .00000      s

#                                        -----end-of-data-block-----
```

'_atom_site_' will extract all data items whose name starts with the string _atom_site_. A request for '_' will extract all data items in the designated data block. The entry 'data_' signals that data requests are from the next-encountered data block. See Table 1(*a*).

(*e*) The user need not know whether a data item is in a repeated list or not. *QUASAR* automatically extracts the data in the construction of the input CIF. However, if the output data items are to be retained in the same loop_ structure as the input CIF, these data items must be requested consecutively (*i.e.* they may be reordered but not separated). See Tables 1, 2 and 3.

(*f*) The file name of the input CIF is specified as the first line of the request file. Its format is _star_arc_<*filename*>. For example, if the input filename is TOZ.cif, the first line will read _star_arc_TOZ.cif. See Table 1(*a*).

(*g*) The file name of the output CIF, if required, is specified as the second line of the request file. Its format is _star_out_<*filename*>. For example, if the output filename is to be TOZ.new, the second line would read _star_out_TOZ.new.

(*h*) If the input CIF is to be tested only for logical integrity, an output file name is not specified in the request

Table 4. *Description of major storage variables in the QUASAR common block*

```
C >>>>>>> QUASAR common: description of the array variables. <<<<<<<<
C
C
C     For i=1,2500 in the order of data items in the archive file.
C
C          REQU(n,i)        Contains request sequence number for each (n) request.
C          RCNT(i)          Contains the number of requests for this item.
C          TYPE(i)          Data type code 'numb', 'char' or 'text' for all items.
C
C
C     For j=1,200 in the order of items on the request file.
C
C          NAME(j)          Data name of the requested item.
C          ITEM(j)          Sequence number of this item in the archive file.
C          LOOP(j)          The loop number of looped items, otherwise 0.
C          CLEN(j)          Nchr of non-LOOP items; max char length of LOOP items.
C          SCR1(j)          Scratch record number for non-LOOP items;
C                           first scratch record number for LOOP items.
C          SCR2(j)          First char position in scratch record for non-LOOP items;
C                           order of items in scratch file LOOP items.
C
C
C     For k=1,50 in order of the current LOOP items in scratch file.
C
C          LREC(k)          Record number on the scratch file for non-text item.
C                           first record number on the scratch file for text item.
C          LPOS(k)          Char pointer to the scratch record for a non-text item.
C                           last record number on the scratch file for text item.
C          LLEN(k)          Length of string on the scratch record for non-text item.
C                           -1 for text item.
C
C     For l=1,10 in order of wild-card data names in the request file.
C
C          WNAM(l,n)        For up to 100 data names per wild-card request.
C
C     NTYP = 1 for a data name; = 3 for number data;
C            = 4 for char data;    = 5 for text data
C
```

file. Instead, the line `_star_log` is entered. All lines following this line in the request file will be ignored. See Table 1(*b*).

Program algorithm

The program algorithm and modules are discussed to provide insight into a typical CIF parsing process. It is possible that some of the *QUASAR* modules are directly adaptable to other CIF software. Programmers are encouraged to adapt *QUASAR* for this purpose. It is anticipated, however, that the *CIFtbx* routines (Hall, 1993) will be more convenient for local adaptations.

The basic algorithmic steps used in *QUASAR* are as follows:

1. Read the request file and store the list of data names for the next specified data block. See the variables NAME(), ITEM() *etc.* in Table 4.

2. Expand the stored request list if 'wild-card' names are specified by reading the input CIF and extracting all data names that match the 'wild-card' entries.

3. Read the input CIF again and store all data items that are requested on a direct-access file. The direct-access record numbers and the line positions are saved in the request-list variables shown in Table 4. Only one pass of the input sequential CIF is necessary.

4. The items are extracted from the direct-access file in the order of the request list. The stored pointers enable data items to be transferred directly from the scratch file to the CIF output buffer.

5. Go back to 1 and process the next requested data block.

The principal source modules which perform these functions are as follows:

QUASAR (main program which cycles steps 1 to 5)
  Calls *REQIN*, *ADDREQ*, *GETDAT*, *REQOUT*
  REQIN (performs step 1)
    Calls *GETSTR*, *ERR*
  ADDREQ (performs step 2)
    Calls *GETSTR*, *ERR*
  GETDAT (performs step 3)
    Calls *GETSTR*, *PUTSCR*, *ERR*
  REQOUT (performs step 4)
    Calls *GETSCR*, *PUTOUT*, *ERR*
    GETSTR (get a character string from the request file or input CIF)

PUTSCR (put a character string in the direct-access file)
GETSCR (get a character string from the direct-access scratch file)
PUTOUT (put a character string into the output CIF)
ERR (print an error message and either exit if fatal or return).

CIF error detection

An important function of *QUASAR* is to test the logical integrity of the input CIF. An indication of the types of checks is evident from the error messages output by *QUASAR*. Here is a summary of these messages, with the likely causes of error.

`No data requests? Check request file`
The request file is empty, not available or misconstructed.

`Request dataname > 32 chars` ***<data name>***
The requested data name is too long

`Request count > 200`
The current limit to requested items in *QUASAR* is 200 per data block (see Table 4). This can easily be increased by the user.

`Wild card count > 10`
The current limit to requested items with wild-card options in *QUASAR* is 10 per data block (see Table 4). This can easily be increased by the user in *ADDREQ*.

`Wild card name expansion > 100`
The number of data names that can be matched to the wild-card request is 100 per data block. This can be increased by modifying *ADDREQ*. See also Table 4.

`No items in data block` ***<data block name>***
There are no requests for data in this block *or* there is no data-block request.

`Archive data mis-count in loop_` ***<loop number>***
The count of data items in a list of repeated data is not a multiple of the number of data names in the designated `loop_` structure. The input CIF is corrupted.

`Archive data name > 32 chars` ***<data name>***
The specified data name in the input CIF exceeds the 32-character limit.

`Data structure error before` ***<data name>***
There is a logical error in the construction of the input CIF prior to this name.

`Data item count > 2500`
The number of data items in the input CIF exceeds 2500. See Table 4.

`Requests for same item > 5`
The same data items may only be requested five times in a data block. See Table 4.

`Data structure error at data item` ***<data name>***
A logical data structure error is detected when transferring data from the scratch file.

`Terminating ; missing from text near` ***<line string>***
Text data has not been terminated correctly. The semicolon character is missing from the first character of the last text line.

`Warning: string starts with ; on line` ***<line string>***
A character string starts with a semicolon. This warns of a possible text error.

`Quoted string in archive not closed` ***<line string>***
Unmatched quotes have been detected on this line.

Distribution

The *QUASAR* software is distributed as the file `quasar` containing the Fortran source, the common file and some small test files. These test files are shown as Tables 1, 2 and 3. The file `quasar` may be obtained free of charge in several different ways. The simplest and fastest approach is to use anonymous FTP to *get* the file from the directory cif on the host 130.95.232.12. Alternatively, send an email containing the line `send quasar` to sendcif@crystal.uwa.edu.au or containing the line `send quasar.src` to sendcif@iucr.ac.uk. As a last resort, airmail a floppy disk to the first-named author stating the mode of copy required.

References

ALLEN, F. H., BUGG, C. E. & MASLEN, E. N. (1991). *Acta Cryst.* B**47**, 821–823.
ALLEN, F. H. & EDGINGTON, P. R. (1991). *CIFER: a Program for CIF Generation.* Crystallographic Data Centre, Cambridge, England.
FLACK, H. D. (1992). *J. Appl. Cryst.* **25**, 455–459.
HALL, S. R. (1991). *J. Chem. Inf. Comput. Sci.* **31**, 326–333.
HALL, S. R. (1993). *J. Appl. Cryst.* **26**, 482–494.
HALL, S. R., ALLEN, F. H. & BROWN, I. D. (1991). *Acta Cryst.* A**47**, 655–685.
HALL, S. R. & STEWART, J. M. (1990). Editors. *Xtal3.0 Users Manual.* Univs. of Western Australia, Australia, and Maryland, USA.
MCMAHON, B. (1993). *Acta Cryst.* C**49**, 418–423.