The STAR File: Detailed Specifications

Sydney R. Hall*,† and Nick Spadaccini‡

Crystallography Centre and Department of Computer Science, University of Western Australia, Nedlands 6009, Australia

Received August 4, 1993®

The STAR File is a relatively simple but universal approach to electronic data exchange and archiving. This paper provides detailed specifications of the STAR File syntax and semantics. These include several extensions to the syntax.

INTRODUCTION

The STAR File was proposed in 1991 by Hall[1] as a universal approach to electronic data exchange and archiving. Hall detailed the concepts and general syntax specifications of the Self-defining Text Archive and Retrieval (STAR) File. These provide for a simple, easy to comprehend, flexible, and extensible data exchange format. The syntax permits most types of data items, data structures, and data cells.

The STAR File is currently being employed for a number of data-transfer applications. The principal of these is the Crystallographic Information File (CIF),[2] which is used widely for data entry into chemical structure databases (CCDC, ICDD, and PDB), and for electronic manuscript submission to the journal *Acta Crystallographica*. Other major applications include the Molecular Information File (MIF) data exchange mechanism[3] and the IUCr World Directory of Crystallographers (WDC) database.[4]

Data stored in a STAR File is self-descriptive because the underpinning data structure consists of *tag-value pairs* or *tuples*. These are also referred to as *data items*. A data item is a *data value* with its identifying unique tag, the *data name*. This self-descriptive approach is similar to some other data exchange formats, such as ASN.1,[5] but STAR is unique in a number of ways. For example, it provides special constructs for nested and repetitive data, which are particularly important for technical and scientific applications.

The paper of Hall[1] did not specify the syntax or semantics of STAR File constructs precisely. This was intended to promote application flexibility. Recent experience with applications such as CIF and MIF has, however, demonstrated that the lack of syntax precision can also inhibit data flexibility and reduce universality. Indeed, the recent *Star_Base*[6] development suggests that a more explicit specification of the STAR File syntax has distinct advantages for inherently complex applications.

This paper is in response to this. It provides a detailed specification of the STAR File syntax and semantics and includes several extensions to the STAR File syntax arising from the development of the *Star_Base* query language.[6]

DETAILED SPECIFICATIONS

A STAR File is a sequential file containing text lines of standard ASCII characters. A file may be divided into any number of discrete units, or cells, each containing unique sets of data items. A cell of data may be represented by a data block, a global block, or a save frame. The syntax rules which apply to a STAR file follow in descriptive form. The syntax in extended Backus–Naur form[7] is given in the Appendix. *The specifications which are an extension to those of Hall[1] are side lighted.*

The STAR File is a free form language and as such, blanks (ASCII 32), vertical (ASCII 11), and horizontal (ASCII 9) tabs, new lines (ASCII 10), and form feeds (ASCII 12), collectively referred to as *white space*, are ignored except as they separate tokens.

1. Text String or Token. A *text string* or *token* is defined as any of the following.

(a) a sequence of *non-white space* characters on a single line, e.g.

```
5.324
light_blue
```

(b) a sequence of characters on a single line containing a leading digraph <~><'> and the trailing digraph <'><~>. <'> is the single quotation character (ASCII 39) and <~> is *white space*, e.g.

```
'light blue'
'Patrick O'Connor'
```

(c) a sequence of characters on a single line containing a leading digraph <~><"> and the trailing digraph <"><~>. <"> is the double quotation (ASCII 34) character and <~> is *white space*, e.g.

```
"low melting point"
"classed as 'unknown'"
```

(d) a sequence of *lines* starting with, and finishing with, a semicolon character <;> (ASCII 59) as the first character of a line, e.g.

```
;
 Department of Computer Science
 University of Western Australia
 Nedlands WA 6009
 AUSTRALIA
;
```

2. Data Name. A *data name* is a sequence of *non-white space* characters starting with an underscore character <_> (ASCII 95), e.g.

```
_publication_author_address
```

3. Data Value. A *data value* is a text string *not* starting with an underscore character <_> *and* preceded by an identifying data name. Text strings representing privileged constructs (see section *10a* below) are excluded from this definition.

† Crystallography Centre.
‡ Department of Computer Science.
® Abstract published in *Advance ACS Abstracts*, March 1, 1994.

0095-2338/94/1634-0505$04.50/0

4. Data Item. A *data item* is a data value *and* a preceding data name. Each data item stored in a STAR File is specified with this combination.

5. Data Loop Structure. A *data loop structure* consists of a loop_ statement followed by (a) a list of data names and (b) a list of *data packets*, each of which contains data values which match the data names in (a). Data loops may be nested to any level. Each loop level is opened with a loop_ statement 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 (referred to as loop packets). Each level of looped data must be terminated with a stop_, except the outermost (level 1) which is terminated by a new data item, a save_*framecode* statement, a data_*blockcode* statement, a global_ statement, or an end of file.

An example of a simple one level loop structure is

```
loop_
      _atom_identity_node
      _atom_identity_symbol   1 C
                              2 C
                              3 O
```

Nested (multilevel) loop structures contain matching data packets as per b and an additional stop_ to terminate each level of data. Here is an example of nested loop data.

```
loop_
      _atom_identity_node
      _atom_identity_symbol
            loop_
                  _atom_bond_node_1
                  _atom_bond_node_2
                  _atom_bond_order

      A1 B1       1 2 single                    stop_
      A2 B2       1 6 double  30 40 triple      stop_
      A3 B3       1 7 single                    stop_
```

The matching of data name packets 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_
       _scheme
       _atomic_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_
```

6. Save Frame. A *save frame* is a data cell wholly contained within a data block. It starts with save_*framecode* statement where *framecode* is a unique identifying code within a data block. A save frame is closed with a save_ statement [*this changes the earlier specification*[1] *where stop_ was used to close a save frame.*], e.g. see Chart 1.

Chart 1

```
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.

(c) Data specifications in a save frame are insulated from the parent data block specifications.

(d) A *framecode* must be unique within a data block.

(e) A save frame may be referenced within the data block in which it is specified using a data value with the construct $*framecode*, e.g.

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

where *arg*, *tyr* and *leu* are *framecodes* identifying three save frames of data.

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

7. Data Block. A *data block* is a data cell containing a sequence of data items, data loops, and save frames. It starts with a data_*blockcode* statement where *blockcode* is a unique identifying code within a file. A data block is closed by another data_*blockcode* statement, a global_ statement, or an end of file, e.g.

```
data_rhinovirus
```

all information relevant to rhinovirus included here

```
data_influenza
```

all information relevant to influenza included here

A data block has the following attributes:

(a) A *blockcode* must be unique within a file.

(b) Data blocks may *not* be referenced from within a file. [*Unlike save frames there is no internal referencing mechanism for data blocks.*]

Chart 2

<star_file>	::=	{<data_block> | <global_block> }*
<data_block>	::=	<data_heading> <data_block_body>+
<global_block>	::=	<global_heading> <global_block_body>+
<global_heading>	::=	**global_**
<data_block_body>	::=	{<data> | <save_frame> }+
<global_block_body>	::=	<data>+
<data>	::=	<data_name> <data_value> | <data_loop>
<save_frame>	::=	<save_heading> <data>+ **save_**
<data_loop>	::=	**loop_** <data_loop_definition> <data_loop_values>
<data_loop_definition>	::=	<data_loop_field>+
<data_loop_field>	::=	<data_name> | <nested_loop>
<nested_loop>	::=	**loop_** <data_loop_definition> [**stop_**]
<data_loop_values>	::=	<data_loop_item>+
<data_loop_item>	::=	<data_value> | **stop_**
<data_value>	::=	<non_quoted_text_string> | <double_quoted_text_string> | <single_quoted_test_string> | <semi_colon_bounded_text_string> | <frame_code>
<data_heading>	::=	**data_**<non_blank_char>+
<data_name>	::=	_<non_blank_char>+
<save_heading>	::=	**save_**<non_blank_char>+
<non_quoted_text_string>	::=	<not_underscore> <non_blank_char>*
<double_quoted_text_string>	::=	<D_quote> <D_quote_string> <D_quote>
<D_quote_string>	::=	{ <D_quote> <no_blank_char> | <not_a_D_quote> }*
<single_quoted_text_string>	::=	<S_quote> <S_quote_string> <S_quote>
<S_quote_string>	::=	{ <S_quote> <non_blank_char> | <not_a_S_quote> }*
<semi_colon_bounded_text_string>	::=	<line_begin> <semi_colon> <line_of_text>+ <line_begin>
<frame_code>	::=	**$**<non_blank_char>+
<line_begin>	::=	*beginning of a new line of input*
<blank>	::=	*space character (ASCII 32)* | *tab character (ASCII 9)*
<non_blank_char>	::=	*! shriek character -> ~ tilde character (ASCII 33 – 126)*
<char>	::=	<blank> | <not_a_blank>
<line_of_text>	::=	<char>+ <newline>
<newline>	::=	*nl character (ASCII 10)*
<D_quote>	::=	" *character (ASCII 34)*
<S_quote>	::=	' *character (ASCII 39)*
<not_a_D_quote>	::=	*! shriek character (ASCII 33)* | *# sharp character -> ~ tilde character (ASCII 35-126)*
<not_a_S_quote>	::=	*! shriek character -> & ampersand character (ASCII 33-38)* | *(left bracket character -> ~ tilde character (ASCII 40-126)*
<semi_colon>	::=	*; semicolon character (ASCII 59)*

(c) The scope of data specified in a data block is the data block. The value of a data item is always associated with the data block in which it is specified.

(d) Data specifications in a data block are unique, except if contained with a save frame. Data specifications in a save frame are insulated from the parent data block specifications.

(e) If a data item is not specified in a given data block, the global value is assumed. If a global value is not specified, the value is unknown.

8. Global Block. A *global block* is a data cell containing a sequence of data items. It starts with a global_ statement

and is closed by a data_*blockcode* statement or an end of file. [*The global data concept is an extension to earlier specifications.*[1] *It is particularly important for abbreviating some types of data exchange applications.*[3]] E.g.

```
global_

all information global to subsequent data blocks included here

data_influenza
```

A global block has the following attributes:

(a) The scope of global data is from the point of declaration to the end of the file.

(b) A global block may contain data items and loop structures but not save frames.

(c) Multiple global blocks are permitted, but all global data exist as a single data list in which the latest specification has precedence.

(d) A data item specified within a data block has precedence over a globally specified data item.

9. Data Cells and Scopes. A *data cell* is the generic term for a unique set of data. A STAR file may contain three types of data cells: global blocks, data blocks, and save frames. The attributes of data cells are as follows.

(a) A file may contain any number of data cells.

(b) The *scope* of a data cell is the region of a file to which a data item is attached and may be accessed.

(c) The scope of data cell types is hierarchical.

(d) The scope of a save frame is the boundaries of the save frame.

(e) The scope of a data block is the boundaries of the data block, including any contained save frames. The same data item may be defined within a save frame and within the parent data block. All specifications of this item will be recognized when accessing the data block.

(f) The scope of a global block is the file, from the point of invocation to the end of the file. It encompasses all contained global blocks, data blocks, and save frames. Globally specified data are active if identical items are not specified in subsequent data blocks or save frames.

10. Privileged Constructs. The following constructs are privileged syntax or key words.

a. Text strings starting with the character sequences: data_, loop_, global_, save_, or stop_ are key words and may not be used for data item specification.

b. An isolated (i.e. preceded by *white space* <~>) sharp character <#> (ASCII 35) is an explicit *end of line* signal *provided* it is not contained within a text string of type *1b, 1c,* or *1d.* Characters on a line following a sharp character are treated as inaccessible data and represent comment text only.

APPENDIX: EXTENDED BACKUSNAUR FORM

The extended Backus–Naur form[7] of the STAR File syntax is given in Chart 2. The following conventions have been employed:

(a) Terminal symbols are denoted in **bold**.

(b) Nonterminal symbols (syntactic categories) are enclosed in < >.

(c) Alternation is represented by the | symbol.

(d) < > followed by the Kleene cross + represents "one or more of" that syntactic category.

(e) < > followed by the Kleene star * represents "zero or more of" that syntactic category.

(f) Groups of syntactic categories are enclosed in { }.

(g) Syntactic categories which are optional are enclosed in [ ].

A comment is started with an isolated hash (i.e. <~><#>, where <~> is white space) except if it contained within the text strings of type *1a, 1b,* or *1c.* A comment is closed by a new line (ASCII 10). Otherwise the hash character (ASCII 35) is treated identically to other characters (see Chart 2).

REFERENCES AND NOTES

(1) Hall, S. R. The STAR File: A New Format for Electronic Data Transfer and Archiving. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 326–333.
(2) Hall, S. R.; Allen, F. H.; Brown, I. D. The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography. *Acta Crystallogr.* **1991**, *A47*, 655–685.
(3) Allen, F. H.; Barnard, J. M.; Cook, A. F. P.; Hall, S. R. The Molecular Information File (MIF): Initial Specifications. *J. Chem. Inf. Comput. Sci.*, to be submitted.
(4) Epelboin, Y. Keywords for the Database of Crystallographers and the World Directory. *Acta Crystallogr.* **1992**, *A48*, 949–954.
(5) Abstract Syntax Notation 1. ISO Standards 1990, ISO/IEC 8824 and ISO/IEC 8825.
(6) Spadaccini, N.; Hall, S. R. *Star_Base*: Accessing STAR File Data. *J. Chem. Inf. Comput. Sci.*, following paper in this issue.
(7) MacLennan, B. J. Principles of Programming Languages: Design, Evaluation and Implementation, Holt , Rinehart and Winston: New York, 1983.