

Extensions to the STAR File Syntax

Nick Spadaccini* and Sydney R. Hall*

School of Chemistry and Biochemistry, The University of Western Australia, Nedlands 6009, Australia

S Supporting Information

ABSTRACT: The STAR File [Hall *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 326–333; Hall and Spadaccini *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 505–508] format represents a universal language adopted for electronic data and metadata exchange in the molecular-structure sciences [Hall et al. *Acta Crystallogr. Sect. A* **1991**, *47*, 655–685; *International Tables for Crystallography. Vol. G: Definition and exchange of crystallographic data*; Springer: Dordrecht, The Netherlands, 2005; Allen et al. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 412–427] and used by the International Union for Crystallography for publication submissions and database depositions. This paper describes an extended STAR syntax that facilitates richer and more specific data definition and typing and a commensurate improvement in precise data description.

```
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_
```

INTRODUCTION

A 2009 editorial⁶ in *Nature* entitled *Data's Shameful Neglect*, along with several accompanying articles, drew attention to the massive gap in data sharing between researchers in the sciences. In particular, these highlighted the lack of technical, institutional, and cultural frameworks for supporting open data access and archiving. In the chemical and biological sciences, the STAR File format^{1,2} is one of the more successful approaches to the coordination of data and metadata management. This approach was developed in the late 1980s and predates the creation of another universal data language, XML. STAR syntax underpins the development of the *Crystallographic Information Framework* (CIF)³ that is used widely for electronic data exchange, publication submission, and database deposition in the molecular-structure sciences.^{4,7} Most importantly, STAR is the syntax of the data definition language used to build electronic data dictionaries for these disciplines, and these provide the primary reference for numerous editing and validation tools.^{4,5,8}

STAR FILE DEVELOPMENT

The STAR data language is predicated on a simple free-format, text-based syntax suitable for expressing data as *tag–value* pairs. The *tag* serves to uniquely identify the data item, and is followed immediately by an instance value. The file is portable, machine-parsable, and easily read by humans. This approach has general applicability but is particularly suited to well-defined numerical and descriptive data. CIF^{3,4,7,8} is a discipline-specific instance of the STAR File and has been adopted by the International Union for Crystallography (IUCr) for its publication and data archiving activities.⁴

In the twenty years since the publication of the initial STAR file syntax,^{1,2} the electronic data needs of science has grown enormously, and today's research activities require much richer metadata descriptors and more flexible approaches to data internationalization. Clearly, internet access to widely disparate and rapidly expanding information continues to be a strong

driver for these requirements. To facilitate the precise identification and description of data, and their inter-relationships, a STAR-based dictionary definition language (DDL) has been developed.^{4,9–14}

The extended STAR file syntax described in this paper provides for a higher level of data specificity, validation, and automation. The changes from the original syntax are summarized separately in Appendix A in the Supporting Information. Companion papers^{13,14} describe the application of the extended STAR file syntax within a semantically rich dictionary definition language (DDLm) and the purpose-built language dREL for DDLm methods scripts. Using methods expressions, DDLm can define machine-parsable and executable relationships between data items, as well as facilities for user defined types and functions.

STAR DATA MODEL

The description of the STAR file extensions that follows assume that the reader is familiar with the rationale and motivation for this approach described in the 1991 and subsequent publications.^{1–5}

There were many data models under consideration during the initial development of STAR in 1988; a development that preceded HTML and the logical infrastructure that became the World Wide Web. At its simplest level, the STAR file data model consists of whitespace-separated *tag–value* pairs constructed from printable characters. Within certain scoping rules the *tag* is unique and therefore is a unique key to the *value*. The *tag* is generally referred to as a *data name*. The combination of a data name and its value is referred to as a *data item*. The syntax of a *data name* and the *data value* as a *number*, *nondelimited* and *delimited string*, *list*, *table* (associative array), and *ref-table* is given below. The latter two are recursive compound data types.

Received: February 7, 2012

Published: June 22, 2012

A single tag with multiple values can be concisely expressed using a *looped list* (detailed in part 4 of section STAR SYNTAX below), hereafter referred to simply as a *loop*. This construction is analogous to a relational table consisting of column headers (composed of *tags*), followed by a sequence of rows (composed of *values* as an integer multiple of the number of tags). That is, each value is implicitly matched to its tag by the order. More complex data relationships can be expressed as *nested loops*. The structure of any loop is identical and independent of the nesting depth. That is, a sequence of tags followed by the sequence of values, where values are matched to tags by their order at each level of nesting (examples shown in part 4 of section STAR SYNTAX).

A STAR file is partitioned into discrete *cells*. The cells are of three types: *global block*, *data block*, and *save frame*. At least one data block must exist in a STAR file; global blocks and save frames are optional. If a save frame is used, it must reside within a data block. Save frames may be nested. The different cell types serve particular functions in the organization and the management of data in the file. Cells are identified by the *case insensitive* keywords *global_*, *data_*, or *save_*, and the latter two must include a concatenated identifier code. A detailed description of the syntax for these cells is described below.

In summary the data model of the STAR File is as follows:

- a STAR file may be partitioned into *global* and *data blocks*;
- a *global block* contains default data common to all subsequent *data blocks*;
- a *data block* may be further partitioned into *save frames*;
- a *data block* identifier code must be unique within a file;
- a *data block* is closed by a subsequent data or global block;
- a *save frame* must reside within a *data block*;
- a *save frame* may reside in another *save frame*;
- a *save frame* identifier code must be unique within its containing cell;
- a *data item* must reside in a cell, and be unique within that cell.

A detailed description of the syntax, keywords, and scoping rules that facilitate the organization and access of data items within this data model is given below.

■ EXTENDED CHARACTER SET

The character set of the STAR file has been extended from printable ASCII characters (i.e., UNICODE Code Page 0 in UTF-8 encoding or the first 127 characters of the LATIN-1 set) to include the complete UNICODE character set, excluding a small number of nonprintable or conflicting characters.

For the purposes of encoding data, the extended STAR character set is the UNICODE subset U+0009, U+000A, U+000D, U+0020 to U+D7FF, U+E000 to U+FFFD, and U+10000 to U+10FFF. In addition the **BEL** character U+0007 is used as an escape character within delimited strings (this is detailed in part 2 of section STAR SYNTAX below). No restriction is placed on the preferred encoding of this character set.

■ STAR SYNTAX

A STAR file is a sequential text file containing lines of allowed UNICODE characters. As detailed in the data model above a file is divided into any number of discrete *data cells* each

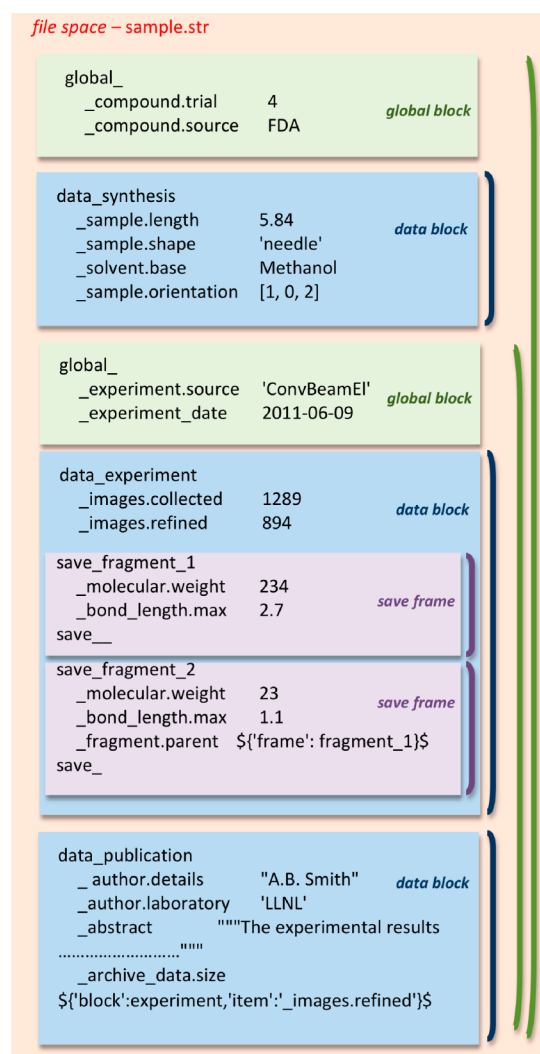


Figure 1. Star file Data Model is shown in terms of example data items in a file. The data cells *global blocks*, *data blocks*, and *save frames* are shown as colored boxes representing the boundaries of the cells. The colored bands at the right-hand side show the scope of the respective cells.

containing any number of data items that have *data names* unique to that data cell. The syntax rules for these cells follow in descriptive form. A more rigorous Extended Backus-Naur Form description of the STAR file syntax is provided as Appendix B in the Supporting Information.

The STAR file is a free-form language in which *whitespace* (defined below) separates lexical tokens. It is a line-oriented text file, where operating system dependent mechanisms indicate the occurrence of new lines of text. For commonly used operating systems, the newline character U+000A terminates a line record for *Mac OS X* and *Unix*, and the digraph U+000DU+000A terminates a line record for *Windows*.

1. Syntax of Tag–Value Pairs. The basic element for expressing data in a STAR File is a *tag* (*data name*) followed by a *data value* i.e. a *tag–value* pair.

(a). *Data Name.* A data name is a data identifier composed of a sequence of *nonwhitespace* characters starting with an underscore character `_` (U+005F).

Examples:

```
_publication.author_name
_publication.author_address
```

(b). *Data Value*. A data value is a text string, preceded by its identifying data name. Details of the syntax of data values follow in the next section.

(c). *Data Item*. A data item is the tag–value pair, i.e., a data name and its data value. A data item may be specified as a single tag–value pair or within a loop of tags, each with multiple values and this is referred to as a *loop* (see part 4)

2. Syntax of Single Data Values. A *single* data value in a STAR file is expressed as a text string delimited by whitespace, quotes, apostrophes, triple quotes, triple apostrophes, or semicolons.

(a). *Whitespace Delimited Value*. A whitespace delimited value is a sequence of characters **on a single line, excluding**:

- a space (U+0020), a left curly bracket { (U+007B), a right curly bracket } (U+007D), a left square bracket [(U+005B), a right square bracket] (U+005D) or a comma , (U+002C);
- a leading
 - underscore _ (U+005F) or,
 - apostrophe ' (U+0027) or,
 - quote " (U+0022) or,
 - semicolon ; (U+003B)
- a string whose leading characters exactly match one of the reserved keywords `loop_`, `global_`, `save_`, `stop_`, or `data_`. The keywords in STAR are *case insensitive*.

Examples of whitespace-delimited values are

```
5.3 6.083(1)e+23 light-blue O'Connor
```

(b). *Quote or Apostrophe Delimited Value*. A quote-delimited value is a sequence of characters **on a single line**, starting with a quote " (U+0022) and terminating with the first subsequent quote. The leading and trailing quote characters serve only to delimit the string and *do not form part of the data*.

If a quote is preceded by the *control* character BEL (U+0007), its action is to *escape* the quote from interpretation as the terminating delimiter. The lexeme returned is the original string with the digraph ␣ replaced by the single quote. In all other cases, a value enclosed between single quotes is treated as a raw string.

An *apostrophe* ' (U+0027) may be used equivalently to a quote for delimiting values.

Examples of delimited values are

```
"low melting point" "Patrick O'Connor"
"classed as ␣"unknown␣"

'light blue' 'classed as "unknown"'
'Patrick O␣'Connor'
```

Consider a UTF8 encoded STAR file containing the data item `PatientDiagnosis.CommonName` with the value of `Hashimoto's disease` (橋本病). Figure 2 shows

5f50	6174	6965	6e74	_	P	a	t	i	e	n	t
5f44	6961	676e	6f73	_	D	i	a	g	n	o	s
6973	2e43	6f6d	6d6f	i	s	.	C	o	m	m	o
6e4e	616d	6520	2748	n	N	a	m	e	'	H	
6173	6869	6d6f	746f	a	s	h	i	m	o	t	o
0727	7320	6469	7365	␣	'	s	D	i	s	e	
6173	6520	28e6	a98b	a	s	e	(U	U	U	
e69c	ace7	9785	2927	U	U	U	U	U)	'	

Figure 2. Portion of a UTF8 encoded STAR data file in binary (left) and the text representation (right).

the hexadecimal representation of the binary contents of the file and its textual representation on the right-hand side. Both show the BEL character prior to the apostrophe in the disease name, so as to protect it from interpretation as a delimiter. The nine bytes representing the three Kanji characters are shown as U in the text representation for simplicity.

(c). *Triple Quote or Triple Apostrophe Delimited Value*. A triple-quote-delimited value is a sequence of characters on one or more lines, started by a sequence of three quotes """" (U+0022U+0022U+0022) and terminated by the first subsequent sequence of three quotes. The leading and trailing sequence of quote characters serve to delimit the string and do not form part of the data.

If a quote is preceded by the *control* character BEL (U+0007), its action is to *escape* the quote from interpretation as part of the terminating delimiter. The lexeme returned is the original string with each of the digraph ␣ replaced by the single quote. In all other cases, a value enclosed between three double quotes is treated as a raw string and this includes all *newlines*.

A sequence of *three apostrophes* "" may be used equivalently to a triple quote """" for delimiting values.

Examples of delimited values are

```
""""A triple quote ␣""␣""␣"" string that
includes newlines is supported in STAR""""

'''<expression> :=
  <term> | <term> op <term>'''
```

(d). *Semicolon Delimited Value*. A semicolon-delimited value is a sequence of characters **on multiple lines**, started by a semicolon character ; (U+003B) as the first character on a line and terminated by the first subsequent semicolon as the first character on a line. The leading and trailing semicolon characters serve to delimit the string and do not form part of the data. In all other cases the value enclosed between the semicolons is treated as a raw string and including all *newlines*.

An example of semicolon-delimited value is

```
;
School of Chemistry and Biochemistry
The University of Western Australia
;
```

3. Syntax of a Compound Data Value. A *compound data value* is composed of multiple elements and may be expressed either as a *List*, delimited by right and left square brackets, or a *Table*, delimited by left and right curly brackets.

(a). *List Value*. A list data item is an ordered set of data elements. The List starts with an left square bracket character [(U+005B) and ends with its *pair-matched* right square bracket character] (U+005D). The List elements are comma character, (U+002C) separated (whitespace between values has no meaning), and the lists may be nested. The List is a recursive data type.

Three examples:

```
[1, 0, 1] [119,136,153,"slate gray"]
[[119,136,153], "slate gray"]
```

A data value of type List may occupy more than one physical line. For example, the value

```
[[119, 136, 153],
 "slate gray"]
```

is identical to the last of the previous examples.

(b). *Table Value*. A table (associative array) data item is an *unordered* set of data elements, each indexed by a string label. The Table starts with a left curly bracket character { (U+007B)

and ends with its *pair-matched* right curly bracket character } (U+007D). The Table elements are comma separated (whitespace between values has no meaning). Each element consists of a quote- or apostrophe-delimited index label, a separating colon character : (U+003A), followed by a STAR data value. The Table is a recursive data type.

An example:

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

4. Syntax of Looped Data Items. A single data name can have multiple values when expressed within a *looped list*, hereafter referred to simply as a *loop*.

A loop is constructed

- starting with the *case insensitive* reserved keyword `loop_`;
- with `loop_` followed by either a sequence of *data names*, or other nested loop constructs (referred to collectively as the *loop header*);
- with the loop header followed by a sequence of *data values* in multiples of the number of data names in the header—each set of data values matching the set of data names, both in number and order, is referred to as a *loop packet*.

A single level loop is equivalent to a relational table in which the data names are the headers for the table *columns*, and the packets are the *rows* in the table. Loops may be nested to any level. Each loop level is initialized with the `loop_` keyword and is followed by the data names of data items recorded at this level. Data values that follow the nested data declarations are in exact multiples of the number of data names. Each loop level must be terminated with the *case insensitive* keyword `stop_`, except the outermost (level 1), which may also be terminated in context by a new data item or by the *case insensitive* privileged strings `save_`, `data_`, `global_` (definitions follow).

An example of a simple **one-level** loop is

```
loop_
  atom_identity_number  1 C  2 C  3 O
  atom_type_symbol
```

Nested (multilevel) loops contain matching data packets and an additional `stop_` to terminate each level of data. Here is an example of a **two-level** nested loop.

```
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_
  stop_
```

The matching of data names to values is applied at each loop level. Initially the data values are matched to the data names in the outermost level loop. This process is repeated 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** nested loop.

```
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_
  stop_
```

5. Syntax of Data Cells. All data items in a STAR file are contained within one or more data cells of the type *data block*, *global block*, or *save frame*.

The uniqueness of a data item is relative to the cell partition in which it is defined so that this does not preclude the existence of an identically named data item in a separate cell partition.

The value of a data item is associated with the cell partition in which it is specified.

(a). **Data Block.** A data block is a cell of unique data items and save frames. The block begins with the statement `data_blockcode`, where *blockcode* is a unique identifying name within a file. A data block is closed by any subsequent `data_blockcode` statement, or `global_` keyword.

For example:

```
data_rhinovirus
  data_relevant_to_rhinovirus
data_influenza_virus
  data_relevant_to_influenza_virus
```

A data block has the following attributes:

- a STAR file must contain at least one data block;
- each *blockcode* name must be unique within the containing file;
- a data block may contain data items and save frames.

(b). **Save Frame.** A save frame is a cell partition of unique data items wholly contained within a data block, or another save frame. The frame starts with the statement `save_framecode`, where the *framecode* is a unique identifying code within the parent cell. Each level of save frame is closed with a `save_` statement.

An example:

```
data_molecule_A
save_phenyl
  object_class  molecular_template
  loop_
    atom_identity_node
    atom_identity_symbol
    1 C  2 C  3 C  4 C  5 C  6 C
  save_
  loop_
    molecular_fragments
    ${"block":"molecule_B","frame":"ethyl"}$
    ${"frame":"phenyl"}$
    ${"block":"molecule_C","frame":"methyl"}$
```

A save frame has the following attributes:

- a *framecode* name must be unique within the parent cell;
- a save frame may contain data items and other save frames.

(c). *Global Block*. A global block is a cell partition of data items that specify default values for cell partitions that follow its declaration. The default value is ignored if the data item is explicitly specified in a cell. A global block starts with the keyword `global_` and is closed by a `data_` declaration.

An example:

```
global_
  data specifications that are defaults
data_rhinovirus
  data specification for rhinovirus
data_influenza_virus
  data specification for influenza virus
```

A global block has the following attributes:

- the scope of global data is from the point of declaration to the *end-of-file*;
- the global block may contain data items but not save frames;
- data items specified in successive global blocks accumulate to form a single global block, in which the latest specification of an item has precedence;
- a data item specified within a data block has precedence over the same data item specified in a prior global block.

(d). *Referencing Data Across Cells*. The scoping rules in the 1991 STAR file^{1,2} restricted a direct reference to a data item to the data block in which it is specified. In an era when the sharing of data resources across the internet is essential, this limitation is now considered too restrictive. The extended STAR file model permits data files, blocks, frames, and items to be referenced from anywhere in a data resource.

The STAR file data reference-value encoding is a special *Table*, referred to as a *ref-table*

```
{
  "source": "location",
  "block": "blockcode",
  "frame": "framecode",    or
  "frame": ["framecode", ["framecode", ...]],
  "item": "dataname",      or
  "item": ["dataname", "dataname", ...]
  "key": "value"           }$
```

in which:

- the delimiter digraph `${` (U+0024U+007B) initiates a *reference-value* terminated by a subsequent digraph `}$` (U+007DU+0024). A *reference-value* is a special instance of a *Table*;
- the only allowed labels in this table are "**source**", "**block**", "**frame**", "**item**" and "**key**";
- location* is the file being referenced—how locations are represented is left to the application domain, though clearly the URL syntax would be a common choice [*default*: parent file];
- blockcode* is the name of the data block being referenced [*default*: parent block];
- framecode* is the name for the save frame being referenced [*default*: parent frame, if relevant];
- dataname* is the data name whose value is being referenced [*default*: all contained data items];
- value* is the key to the loop packet for a referenced data name that resides in a loop.

The *ref-table* permits

- labels to be order independent and nonobligatory (defaults then apply);
- interpretation and application of a reference-value to be domain dependent.

Typical invocations of reference-values, using data structure in Figure 1 are as follows:

- `${'block':synthesis,'item':"_sample.shape"}` within a STAR file is a reference to the value associated with the data name `_sample.shape` that is resident in the data block named `synthesis`, within the same file. Defined within the data block named `synthesis`, `${"item": "_sample.shape"}` refers to the same data item (i.e., defaults to the parent block and file). The referenced value is **needle**;
- `${"block": "experiment", "frame": "fragment_1", "item": "_molecular.weight"}` is a reference to the data item `_molecular.weight` within a save frame named `fragment_1` inside the data block named `experiment`. The default for absent block and frame labels is the same as the previous example. The referenced value is 234;
- in the parent file `${"block": "publication"}` refers to all data in the block named `publication`;
- defined within the parent block `experiment`, `${"frame": "fragment_2"}` refers to all data in the frame `fragment_2`;
- defined outside the block `experiment`, `${"block": "experiment", "frame": "fragment_2"}` is required to refer to all of the data in the frame `fragment_2`;
- within a local resource `${"source": "blat.cif"}` or `${"source": "archive/data/blat.cif"}` refers to all data in some file named `blat.cif`;
- for remote files, using a URL such as `${'source': "file:///iucr.org/CIF/archive/data/blat.cif"}` may be needed to reference all data in the file named `blat.cif`.

6. Syntax of Reserved Keywords. The following keywords are reserved:

- text strings starting with the *case insensitive* character sequences `data_`, `loop_`, `global_`, `save_`, or `stop_` are reserved keywords and may not be used as values in nondelimited text strings.
- a *sharp* (hash) character `#` (U+0023) outside of any STAR token initiates a comment that is terminated by a first subsequent *newline*. Such a sharp character is an explicit *end-of-data* signal. The characters that follow and are contained on one line are *comment* text. A comment is considered *whitespace*.

SUMMARY

The STAR file extensions described in this paper facilitate richer and more precise data definition by supporting the full UNICODE character set, a wider range of data structure containers (e.g., lists, matrices and tables), nested save frames, and by enabling encapsulated and distributed data systems through the universal addressing of data cells. In the two companion papers,^{13,14} application examples of the extended STAR file syntax are presented.

ASSOCIATED CONTENT

Supporting Information

The salient differences between this specification of STAR and the previously published version are listed in Appendix A. The formal specification of the STAR syntax and grammar as an

annotated EBNF is provided as Appendix B. This material is available free of charge via the Internet at <http://pubs.acs.org>.

AUTHOR INFORMATION

Corresponding Author

*E-mail: Nick.Spadaccini@uwa.edu.au (N.S.); Sydney.Hall@uwa.edu.au (S.R.H.).

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

We would like to thank Brian McMahon and James Hester for their contributions and advice on the STAR file extensions. We gratefully acknowledge the support of this work by the Australian Research Council Discovery Grant DP0344560.

REFERENCES

- (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.; Spadaccini, N. The STAR File: Detailed Specifications. *J. Chem. Inf. Comput. Sci.* **1994**, *34*, 505–508.
- (3) Hall, S. R.; Allen, F. H.; Brown, I. D. The Crystallographic Information File (CIF): A New Standard Archive File for Crystallography. *Acta Crystallogr., Sect. A* **1991**, *A47*, 655–685.
- (4) *International Tables for Crystallography. Vol. G: Definition and exchange of crystallographic data*; Hall, S. R., McMahon, B., Eds; Springer: Dordrecht, The Netherlands, 2005.
- (5) Allen, F. H.; Barnard, J. M.; Cook, A. P. F.; Hall, S. R. The molecular Information File (MIF): core specifications of a new standard format for chemical data. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 412–427.
- (6) Data's Shameful Neglect *Nature* **2009**, *461*, 145.
- (7) Bourne, P.; Berman, H.; McMahon, B.; Watenpaugh, K.; Westbrook, J.; Fitzgerald, P. M. D. Macromolecular Crystallographic Information File. *Methods Enzymol.* **1997**, *277*, 571–590.
- (8) Berman, H.; Bernstein, H. J.; Ellis, P. J.; Feng, Z.; Hall, S. R.; Hoyland, M. A.; McMahon, B.; Spadaccini, N.; Strickland, P. R.; Westbrook, J. D.; Yang, H. Applications. In *International Tables for Crystallography. Vol. G: Definition and exchange of crystallographic data*; Hall, S. R., McMahon, B., Eds; Springer: Dordrecht, The Netherlands, 2005; Chapter 5; pp 479–569.
- (9) Hall, S. R.; Cook, A. P. F. STAR Data Definition Language: Initial Specification. *J. Chem. Inf. Comput. Sci.* **1995**, *35*, 819–825.
- (10) Fitzgerald, P. M. D.; Berman, H.; Bourne, P.; McMahon, B.; Watenpaugh, K.; Westbrook, J. The mmCIF dictionary: community review and final approval. *Acta Crystallogr., Sect. A* **1996**, *A52*, C575.
- (11) Westbrook, J. D.; Hall, S. R. *A Dictionary Language for Macromolecular Structure. DDL 2.1.11*, mmcif.rcsb.org (accessed May 25, 2011).
- (12) Spadaccini, N.; Hall, S. R.; Castleden, I. R. Relational Expressions in STAR File Dictionaries. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 1289–1301.
- (13) Spadaccini, N.; Hall, S. R. DDLm: A New Dictionary Definition Language. *J. Chem. Inf. Model.* **2012**, DOI: 10.1021/ci300075z.
- (14) Spadaccini, N.; Castleden, I. R.; du Boulay, D.; Hall, S. R. dREL Relational Expression Language for Dictionary Methods. *J. Chem. Inf. Model.* **2012**, DOI: 10.1021/ci300076w.