STAR Dictionary Definition Language: Initial Specification†

Sydney R. Hall*

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

Anthony P. F. Cook

Synopsys Scientific Systems, 175 Woodhouse Lane, Leeds LS2 3AR, United Kingdom

Received January 18, 1995®

The self-defining text archival and retrieval (STAR) file is a universal approach to data exchange and archiving. Discipline-based applications of the STAR file use electronic dictionaries to define data items and to provide access and validation information. These dictionaries are constructed using special attribute data referred to as the STAR dictionary definition language (DDL).

INTRODUCTION

The STAR File[1,2] format provides a universal data exchange and archival approach based on a flat text file with a simple syntax. This approach has been used to develop two major discipline-specific interchange processes, the crystallographic information file (CIF)[3] for crystallographic structure and diffraction data and the molecular information file (MIF)[4] for more general structural chemistry applications. The CIF format is already is already in use by several international scientific databases (CCDC, PDB, ICDD) and is the preferred method[5] for submitting manuscripts to *Acta Crystallographica.*

Other universal exchange protocols, such ASN.1,[6] exist and have been used by the Chemical Abstracts Service in the development of CXF for chemical data. The STAR file approach is attractive as an interchange format because of its simplicity, generality, and extensibility. Extensibility is of increasing importance to disciplines utilizing new technologies because data types are continually changing. With the STAR file approach new items may be added easily because the file organization and syntax are independent of data definition. Data definitions reside in separate electronic STAR dictionary files. It is the language of these dictionaries which is the subject matter of this paper.

The data syntax and semantics of a STAR file are clearly separated. Data are stored as *tag-value* pairs, and loops to any depth of nesting are supported. There are no name, value, or record length restrictions. Various blocks or scopes are available within a file, two of the most important being global and saveframe blocks, the latter of which is a self-contained data block, which is de-referenced through a pointer. While not purporting to be a *database* standard, STAR encompasses properties of the relational database model. A single level loop is a *table*, the tuples of which can be in 1st Normal Form; the linking tags or *keys* between loops of data are provided in the DDL. The DDL defines conditions and constraints of data items, offering higher Normal Forms, and it can deal with *null* cases. The nested loop structures have the properties of an NF^2 or nested relational model.

Because data items are self-identifying, their arrangement in a file is arbitrary within a few syntax rules and independent of the data access processes. The data semantics are specified in the STAR dictionary file and may be expanded or updated independently of the data. This is essential. The most critical data requirement of technology-based disciplines is that newly defined data types must be incorporated, immediately and seamlessly, into the data interchange mechanism. The survival of an interchange protocol can hinge on this facility alone.

The basic STAR DDL concepts were proposed[7] in 1991. Since then the scope and facility of the language has expanded in response to the definition requirements of the two major interchange developments, CIF[3] and MIF.[4] The purpose of this paper is to formalize the DDL specifications and data items (usually referred to as DDL attributes) to ensure consistent application to existing, and to future, STAR dictionaries and parsing software.

DEFINITION STRUCTURE

A DDL dictionary is a STAR file composed of a sequence of discrete definitions: one for each defined data item. Each definition starts with a data block header and unique block name which matches that of the defined data name or a portion thereof. The header is followed by a sequence of DDL data items referred to as *DDL attributes*. These attributes contain the definition information, and only those attributes needed to uniquely define a data item are entered into the definition block.

The DDL attributes may be thought of as the vocabulary of the dictionary language, and they, individually and collectively, provide the semantic tools of the dictionary. The description and rationale for each attribute are detailed below. The formal DDL specification of each attribute is contained in the electronic file **ddl_core.dic**; which in itself a STAR DDL dictionary. Information on how to obtain a copy of this dictionary is given in the footnote labeled with the † symbol. The text version of the DDL dictionary, translated using the program *CIFtex*,[8] is listed in Chart 1.

To introduce the concept of DDL attributes and how they are used to form a definition, we will start with a simple definition from an existing dictionary. The definition of *chemical compound source* as defined in the CIF dictionary file **cif_core.dic** follows. In this example the data block

† **Dictionary Access:** The electronic version of the dictionary file *ddl_core.dic* containing the DDL definitions may be obtained by anonymous FTP from directory */star* at the Internet address 130.95.232.12 or on request from one of us (SRH) at the e-mail address *syd@crystal.uwa.edu.au.*

® Abstract published in *Advance ACS Abstracts,* August 15, 1995.

0095-2338/95/1635-0819$09.00/0

Chart 1

Dictionary name: ddl_core.dic

`_category` (*char*)

Character string which identifies the natural grouping of data items to which the specified data item belongs. If the data item belongs in a looped list then it must be grouped only with items from the same category, but there may be more than one looped list of the same category provided that each loop has its own independent reference item (see `_list_reference`).

[category]

`_definition` (*char*)

The text description of the defined item.

[definition]

`_dictionary_history` (*char*)

A chronological record of the changes to the dictionary file containing the definition. Normally this item is stored in the separate data block labelled data_on_this_dictionary.

[dictionary]

`_dictionary_name` (*char*)

The name string which identifies the generic identity of dictionary. The standard construction for these names is

<application code>-<dictionary version>.dic

Normally this item is stored in the separate data block labelled data_on_this_dictionary.

Example(s): 'ddl_core.dic', 'cif_mm_core.dic' [dictionary]

`_dictionary_update` (*char*)

The date that the dictionary was last updated. Normally this item is stored in the separate data block labelled data_on_this_dictionary.

Permitted values may be constructed as the regular expression
`(_chronology_year)-(_chronology_month)-(_chronology_day)`

[dictionary]

`_dictionary_version` (*numb*)

The dictionary version number. Version numbers cannot decrease with updates. Normally this item is stored in the separate data block labelled data_on_this_dictionary.

[dictionary]

`_enumeration` (*char*)

Permitted value(s) for the defined item.

May appear in list as essential element of loop structure. [enumeration]

`_enumeration_default` (*char*)

The default value for the defined item if it is not specified explicitly. If a data value is not declared the default is assumed to be the "most-likely" or "natural" value.

[enumeration_default]

`_enumeration_detail` (*char*)

A description of a permitted value(s) for the defined item, as identified by `_enumeration`.

May appear in list containing `_enumeration`. [enumeration]

`_enumeration_range` (*char*)

The range of values permitted for a defined item. This can apply to 'numb' or 'char' items which have a preordained sequence (*e.g.* numbers or alphabetic characters). If 'max' is omitted then the item can have any permitted value greater than or equal to 'min'.

Permitted values may be constructed as the regular expression
`(_sequence_minimum):((_sequence_maximum)?)`

Example(s): '-4:10', 'a:z', 'B:R', '0:' [enumeration_range]

`_example` (*char*)

An example value of the defined item.

May appear in list as essential element of loop structure. [example]

`_example_detail` (*char*)

A description of an example value for the defined item.

May appear in list containing `_example`. [example]

`_list` (*char*)

Signals if the defined item is declared in a looped list.

`yes`	can only be declared in a looped list
`no`	cannot be declared in a looped list
`both`	declaration in a looped list optional

Where no value is given, the assumed value is '`no`'. [list]

`_list_level` (*numb*)

Specifies the level of the loop structure in which a defined item, with the attribute `_list` 'yes' or 'both', must be declared.

Where no value is given, the assumed value is '`1`'. The permitted range is $1 \rightarrow \infty$. [list]

`_list_link_child` (*char*)

Identifies data item(s) by name which must have a value which matches that of the defined item. These items are referred to as "child" references because they depend on the existence of the defined item.

May appear in list. [list_link_child]

`_list_link_parent` (*char*)

Identifies a data item by name which must have a value which matches that of the defined item, and which must be present in the same data block as the defined item. This provides for a reference to the "parent" data item.

May appear in list. [list_link_parent]

`_list_mandatory` (*char*)

Signals if the defined item must be present in the loop structure containing other items of the designated `_category`. This property is transferrable to another data item which is identified by `_related_item` and has `_related_function` set as 'alternate'.

`yes`	required item in this category of looped list
`no`	optional item in this category of looped list

Where no value is given, the assumed value is '`no`'. [list]

`_list_reference` (*char*)

Identifies the data item, or items, which must be present (collectively) in a looped list with the defined data item in order that the loop structure is valid. The data item(s) identified by `_list_reference` provides a unique access code to each loop packet. Note that this property may be transferred to another item with '`_related_function` alternate'.

May appear in list. [list_reference]

`_list_uniqueness` (*char*)

Identifies data items which, collectively, must have a unique value for the loop structure of the designated `_category` items to be deemed valid. This attribute is specified in the definition of a data item with `_list_mandatory` set to 'yes'.

May appear in list. [list_uniqueness]

`_name` (*char*)

The data name(s) of the defined item(s). If data items are closely related, or represent an irreducible set, their names may be declared as a looped sequence in the same definition.

May appear in list.

Example(s):
'_atom_site_label',
'_atom_attach_all _atom_attach_ring',
'_index_h _index_k _index_l',
'_matrix_11 _matrix_12 _matrix_21 _matrix_22' [name]

`_related_item` (*char*)

Identifies data item(s) which have a classified relationship to the defined data item. The nature of this relationship is specified by `_related_function`.

May appear in list as essential element of loop structure. [related]

`_related_function` (*char*)

Specifies the relationship between the defined item and the item specified by `_related_item`. The following classifications are recognised.

'alternate' signals that the item referred to in `_related_item` has attributes that permit it to be used alternately to the defined item for validation purposes.

'convention' signals that the item referred to in `_related_item` is equivalent to the defined item except for a predefined convention which requires a different `_enumeration` set.

'conversion' signals that the item referred to in `_related_item` is equivalent to the defined item except that different scaling or conversion factors are applied.

'replace' signals that the item referred to in `_related_item` may be used identically to replace the defined item.

`alternate`	used alternatively for validation tests
`convention`	equivalent except for defined convention
`conversion`	equivalent except for conversion factor
`replace`	new definition replaces the current one

Appears in list containing `_related_item`. [related]

`_type` (*char*)

The type specification of the defined item.

Type 'numb' identifies items which must have values that are identifiable numbers. The acceptable syntax for these numbers is application dependent, but the formats illustrated by the following identical numbers are considered to be interchangeable.
42 42.000 0.42E2 .42E+2 4.2E1 420000D-4 0.0000042D+07

Type 'char' identifies items which need not be interpretable numbers. The specification of these items must comply with the STAR syntax specification of either a 'contiguous single line string' bounded by blanks or blank-quotes, or a 'text string' bounded by semi-colons as first character of a line.

Type 'null' identifies items which appear in the dictionary for data definition and descriptive purposes. These items serve no function outside of the dictionary files.

`numb`	numerically-interpretable string
`char`	character or text string
`null`	for dictionary purposes only

[type]

`_type_conditions` (*char*)

Codes defining conditions on the `_type` specification.

'esd' permits a number string to contain an appended standard deviation number enclosed within parentheses. E.g. 4.37(5)

'seq' permits data to be declared as a sequence of values separated by a comma <,> or a colon <:>.

* The sequence v1,v2,v3 signals that v1, v2, v3 *etc.* are alternative values.
* The sequence v1:v2 signals that v1 and v2 are the boundary values of a continuous range of values satisfying the requirements of `_enumeration` for the defined item.

Combinations of alternate and range sequences are permitted.

`none`	no extra conditions apply to the defined `_type`
`esd`	numbers *may* have esd's appended within ()
`seq`	data may be declared as a permitted sequence

May appear in list. [type_conditions]

`_type_construct` (*char*)

String of characters specifying the construction of the data value for the defined data item. The construction is composed of two entities:
(1) data names
(2) construction characters
The rules of construction conform to the the regular expression (REGEX) specifications detailed in the IEEE document P1003.2 Draft 11.2 Sept 1991 (ftp file '/doc/POSIX/1003.2/p121-140').

Example(s): '(_year)-(_month)-(_day)'(a typical construction for `_date`) [type_construct]

`_units` (*char*)

A unique code which identifies the units of the defined data item. A description of the units is provided in `_units_detail`.

Example(s): 'K'(degrees Kelvin), 'C'(degrees Celsius), 'rad' (radians of angle), 'e'(electrons), 'V'(volts), 'Dal'(Daltons), 'kg'(kilogrammes), 's'(seconds) [units]

`_units_detail` (*char*)

A description of the numerical units applicable to the defined item and identified by the code `_units`.

[units]

header is shown in italics, and the attribute names are shown in bold type.

```
data_chemical_compound_source
    _name                '_chemical_compound_source'
    _category             chemical
    _type                 char
    loop_ _example        'From Norilsk (Russia)'
                          'Extracted from the bark of Cinchona Naturalis'
    _definition
;               Description of the origins of the compound under study, or
                of the parent molecule if a simple derivative is studied.
                This includes the place of discovery for minerals or the
                actual source of a natural chemical product.
;
```

Here is the same definition translated using *CIFtex*.[7]

`_chemical_compound_source` *(char)*
Description of the origins of the compound under study, or of the parent molecule if a simple derivative is studied. This includes the place of discovery for minerals or the actual source of a natural chemical product. Examples: 'From Norilsk (Russia)' 'Extracted from the bark of Cinchona Naturalis' [chemical]

Note that this definition contains only those attributes needed to uniquely specify the quantity "chemical compound source" and to enable its validation. A detailed description of the complete set of DDL attributes is given in later sections and in Chart 1.

As stated above, the organization of attributes within each definition, and of the definitions within a dictionary file, conform to the requirements of a STAR file. The sequence order of both the attributes and the definition blocks is arbitrary. STAR compliance enables DDL dictionaries to be accessed using standard STAR parsing tools.[9,10,11]

ATTRIBUTE DESCRIPTION

The description of data attributes is simplified by grouping them according to functionality. The five basic functional groups are as follows. Those that

1. **name** a data item,
2. **describe** a data item,
3. **type** a data item,
4. **relate** data items in lists,
5. **identify** a dictionary.

1. Data Name. Identifying data items is the primary function of the dictionary. This is achieved by a unique name, referred to in the STAR syntax as a *data name*. This quantity provides the simplest form of data validation. The DDL attribute used to specify a data name is called **_name** and is invoked as the tuple (note that imbedding of the defined data name in quotes is essential)

_name *'<data name>'*

Data names of items having a common function, or forming an irreducible set, may be entered into the same definition block as a *name list* (i.e. they follow the header "**loop_ _name**").

The specification of a data name represents the first level of validation for a STAR file. It provides a check that an item is recognizable for a particular application and also a spelling list for validation software such as *Cyclops*.[8] It is important to emphasize here that a STAR File will not be corrupted by the presence of "stranger" data (i.e., unrecognised by a particular application dictionary), but the file validation process can fail if certain data items are not present. This is because undefined data is ignored by a validation process, whereas the presence of certain items in a file are required because of data dependencies (the attributes that specify data relationships and links are considered below).

2. Data Description. The descriptive DDL attributes are as follows.

_definition

_example

_example_detail

This group of attributes provides *human-readable* information about the defined data item. They are used for dictionary browser software and for the production of text information (after all, humans need to understand definitions too!). This group of attributes is not intended to be machine interpretable.

The **_definition** attribute specifies the text description in a definition and is the primary semantic information about defined data item. It may also be used provide supplementary descriptive (semantic) information about machine-parsable attributes of the item. Here is a definition in which this attribute is used to describe atomic charge.

```
data_atom_charge
    _name                       '_atom_charge'
    _type                        numb
    _list                        yes
    _definition
;             Specifies the formal electronic charge on the atom for the
              different atomic representation conventions. The convention
              for charge is specified by _define_bonding_convention.
;
```

The attributes **_example** and **_example_detail** specify typical applications of the defined item. An example of these two attributes is shown in the following definition.

```
data_atom_site_attached_hydrogens
    _name                       '_atom_site_attached_hydrogens'
    _type                        numb
    loop_ _example
          _example_detail        2    'water oxygen'
                                 1    'hydroxyl oxygen'
                                 4    'ammonium nitrogen'
    _definition
;             The number of hydrogen atoms attached to the atom at this site
              excluding any H atoms for which coordinates (measured or
              calculated) are given.
;
```

3. Type Attributes. The attributes which specify the data type are as follows.

_enumeration

_enumeration_default

_enumeration_detail

_enumeration_range

_list

_list_level

_type

_type_conditions

_type_construct

_units

_units_detail

These are machine parsable attributes of particular importance in the validation of a STAR file. A complete description of these attributes is given in Chart 1.

The ***enumeration*** attributes stipulate which values are permitted for a data item. If no enumeration attributes are specified, a data value is assumed to be unrestricted. The attributes **_enumeration** and **_enumeration_detail** are used to list allowed values and descriptions, respectively. For instance, in a definition of *atomic element symbols* these attributes would probably list the IUPAC "element symbols" and "element names". A definition from the MIF dictionary involving both these attributes follows.

```
data_bond_type_mif
    _name                       '_bond_type_mif'
    _category                    bond
    _type                        char
    _list                        yes
    loop_ _list_reference       '_bond_id_1'    '_bond_id_2'
    loop_ _related_item
          _related_function     '_bond_type_casreg3' convention
                                '_bond_type_ccdc'    convention
    loop_ _enumeration
          _enumeration_detail     S  'Single (2-electron) bond'
                                  D  'Double (4-electron) bond'
                                  T  'Triple (6-electron) bond'
                                  O  'Other (e.g. coordination) bond'
    _definition
;             Code indicating the nature of the bond according to the
              MIF bonding convention. Aromatic and normalised tautomeric
              bonds cannot be shown. The convention is specified with
              _define_bonding_convention.
;
```

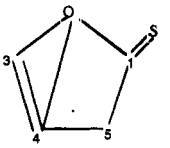
When a data item is restricted to an *ordinal set* of values, the attribute **_enumeration_range** is used to define the minimum and maximum values of the logical sequence. The **_enumeration_default** attribute specifies the value assumed if a value is not set in the application. Here is a definition which uses these two attributes.

```
data_atom_site_occupancy
    _name                       '_atom_site_occupancy'
    _type                        numb
    _enumeration_range           0.0:1.0
    _enumeration_default         1.0
    _definition
;             The fraction of the atom type present at this site.
              The sum of the occupancies of all the atom types at this site
              may not significantly exceed 1.0 unless it is a dummy site.
;
```

List attributes specify how a data item is used in a repetitive (looped) list. The attribute **_list** has a value of *yes* if the defined item must be in a list, and *no* if it must not. A value of *both* allows for both occurrences. The attribute **_list_level** specifies the nested level of the loop which the defined item is used. The default level is 1. Additional _list attributes for specifying relational dependencies between data items are described in the next attribute group (see 4).

The use of nested list data can be illustrated with the display data for a 2D chemical diagram of a hypothetical

molecule. The four items in the first name list (category display) are in level 1, and the two items in the second name list (category display_conn) are in level 2. Observe how, according to the STAR syntax rules, the nest level is automatically incremented after the first four values and decremented when a stop_ signal is encountered.




```
loop_
    _display_id
    _display_symbol
    _display_coord_x
    _display_coord_y
    loop_
        _display_conn_id
        _display_conn_symbol
            1  1    180  110       2 1b       stop_
            2  O    140  140       3 1b  4 1b stop_
            3  3    100  110       4 2b       stop_
            4  4    120   60       5 1b       stop_
            5  5    160   60       1 1b       stop_
            6  O    205  140       1 2b       stop_
```

The list levels of data items used in this example are specified in the dictionary definitions which follow.

```
data_display_coord_
    loop_ _name                 '_display_coord_x'
                                '_display_coord_y'
    _type                        numb
    _list                        yes
    _list_level                  1
    _definition
;               The projected coordinates of the display object identified in
                _display_object and _display_symbol.;

data_display_conn_symbol
    _name                       '_display_conn_symbol'
    _type                        char
    _list                        yes
    _list_level                  2
    _definition
;               The symbol code for the bond connection to the displayed
                objects identified by _display_id and _display_conn_id.
                The symbol codes and descriptions are stored in the
                validation file 'mif_core_bonds.val'.
;
```

Normally "_list_level 1" is specified as a **global_** parameter in a dictionary.

Type attributes fix the form of a data item. The attribute **_type** identifies a *number*, a *text string*, or a dictionary descriptor as the coded values *numb*, *char*, or *null*, respectively. What is classified as a number, or a character string, depends on the application, not on the STAR File syntax. Examples of _type attribute specification are shown in earlier definitions.

The attribute **_type_conditions** is used to identify application-specific conditions which apply globally in the separation of *numb* and *char* data items. This attribute is used to alert general (i.e., nonapplication-specific) parsing software of such conditions. The attribute values are codes. For example, CIF dictionaries set _type_conditions to *esd* so that an error value may be appended, within parentheses, to any CIF number. Other _type_conditions codes are described in Chart 1.

The attribute **_type_construct** stipulates how particular data items should be encoded. The language used for this construction is *REGEX* (version *POSIX*[12]). The specification of _type_construct can include the data names of other defined items. Validation software interprets this attribute as format specifications and constraints and expands the imbedded data items according to their separate definition. For example, the chronological date is made up of day, month, and year. These may be ordered and represented in a variety of ways. The _type_construct attribute is used to specify the date representation. The two definitions below illustrate how a date value of the construction "1995/03/25" may be imbedded into the dictionary definitions. The month and day definitions have been omitted for brevity.

```
data_publ_date
    _name                       '_publ_date'
    _type                        char
    _type_construct             (_publ_year)/(_publ_month)/(_publ_day)
    _definition
;       Specifies the syntax for declaring the publication date.
;

data_publ_year
    _name                       '_publ_year'
    _type                        char
    _type_construct              (19|20)[0-9][0-9]
    _definition
;               Specifies the how the publication year will
                appear. Syntax is valid only for publication
                in the 20th and 21st Centuries!
;
```

Units attributes specify the measurement units permitted for a numerical data item. The attribute **_units** specifies a code which uniquely identifies the measurement units. A description of the units identified bt the code is given by the attribute **_units_detail.**

4. List Relationships. This class of attributes is used to specify the relationships between data items in a repetitive list and between those in different lists. In particular, they are responsible for identifying data dependencies across lists. The attributes are machine parsable and are formally defined in Chart 1.

_category

_list_link_child

_list_link_parent

_list_mandatory

_list_reference

_list_uniqueness

_related_item

_related_function

The **_category** attribute is used to specify the group, or basis set, to which a data item belongs. This attribute is specified as a character string which matches the portion of the data name, following the leading underscore. Data items in a list must be of the same _category. Data items of a given category may exist in different lists provided each list contains an appropriate "reference" data item (see _list_reference below). Items belonging to the different categories should not appear in the same list.

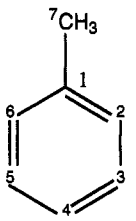
List link attributes specify relationships between different lists. The **_list_link_parent** attribute identifies an item in another list which must exist in order that the defined data item is interpretable. The "parent" data item (or items, in the case of irreducible sets of data) must be a unique value within its own list. The **_list_link_child** attribute is declared in the definition of a "parent" item and identifies items in other lists which depend implicitly on the defined data item being present in the same file. The functionality of these two attribute mirror each other. The parent item must be present if any of the child items exit in a file but not the converse.

List dependent items are present in the chemical connectivity data shown in Chart 2a. Each atom site in this data is identified by a unique identification number _atom_id. The bonds between sites are represented by pairs of id

Chart 2

CHART 2a

Molecular connectivity data for toluene in STAR format.




```
data_toluene

loop_
    _atom_id
    _atom_type
    _atom_attach_h
                    1  C  0    2  C  1    3  C  1    4  C  1
                    5  C  1    6  C  1    7  C  3

loop_
    _bond_id_1
    _bond_id_2
    _bond_type_mif
                    1  2  D    2  3  S    3  4  D    4  5  S    5  6  D
                    1  6  S    1  7  S
```

CHART 2b

The definitions of three data items used in toluene data.

```
data_atom_id
    _name                       '_atom_id'
    _category                    atom
    _type                        char
    _list                        yes
    _list_mandatory              yes
    _list_uniqueness            '_atom_id'
    loop_ _list_link_child      '_bond_id_1'
                                '_bond_id_2'
                                '_stereo_vertex_id'
    _definition
;               This specifies a unique code for an 'atom site' in a molecule
                or fragment. A designated atom site may be occupied by a 'dummy'
                atom (see _atom_type).
;

data_bond_id_
    loop_ _name                 '_bond_id_1'
                                '_bond_id_2'
    _category                    bond
    _type                        char
    _list                        yes
    _list_mandatory              yes
    _list_link_parent           '_atom_id'
    loop_ _list_uniqueness      '_bond_id_1'
                                '_bond_id_2'
    _definition
;               Specify the atom site codes of chemically "connected sites".
                The atom id codes may appear in either order except for
                asymmetric bonds. Each "bond" pair may be represented only once.
                The special syntax for representing atom sites within molecular
                templates is described in the _atom_id definition.
```

numbers _bond_id_1, and _bond_id_2. The _atom_id, _bond_id_1 and _bond_id_2 items refer to identical quantities, namely the identity of the atom sites. Because the interpretation of the bond id's is dependent on the existence of the atom id's they are child dependent data. The definitions of these items in Chart 2b indicates this dependence.

The attributes **_list_mandatory** and **_list_reference** are closely related. The former signals (with *yes* or *no*) if a data item is essential to a list of items. The attribute _list_reference specifies which items distinguish list packet from each other. A "packet" is composed of a value from each item in the name list. In other words, _list_reference identifies items which are used as reference pointers to a specific packet in the list. In the connectivity example (Chart 2), _atom_id is the reference item for both the _atom_type and the _atom_attach_h values since these are properties dependent on the identity of the atom site. It follows that each value of _atom_id in the list *must be unique*. In the bond list, the items _bond_id_1 and _bond_id_2 are the *joint* reference pointers for _bond_type_mif.

The attribute **_list_uniqueness** identifies items which must be unique for a list to be valid and accessible. This attribute is similar to that of _list_reference except that it appears only in the definition in which _list_mandatory is set to *yes*. This

Chart 3. Recommended Construction of a Dictionary Using the DDL

```
##############################################################################
#                                                                            #
#                          XYZ DATA DICTIONARY                               #
#                                                                            #
##############################################################################


data_on_this_dictionary

    _dictionary_name            xyz_core.dic
    _dictionary_version         0.1
    _dictionary_update          1994-02-22
    _dictionary_history
;
  1994-02-22  Created as a typical dictionary construction.
;


data_include_related_dictionaries

#\include "ftp://anonymous//tobago.crystal.uwa.edu.au/star/star_core.dic"


global_
    _list                        no
    _list_mandatory              no
    _list_level                  1
    _type_conditions             seq


data_atom_attach_
    loop_ _name                 '_atom_attach_all'
                                '_atom_attach_ring'
                                '_atom_attach_nh'
                                '_atom_attach_h'
    _category                    atom
    _type                        numb
    _list                        yes
    _list_reference             '_atom_id'
    _enumeration_range           0:
    _definition
;               The number of atom sites considered to be attached to this site.
                    all    all sites
                    ring   all sites forming rings
                    nh     all sites excl. hydrogens and unshared electron pairs
                    h      hydrogen sites
;

data_atom_charge
    _name                       '_atom_charge'
    _category                    atom
    _type                        numb
    _list                        yes
    _list_reference             '_atom_id'
    _enumeration_range           -99:99
    _definition
;               Specifies the formal electronic charge on the atom for the
                different atomic representation conventions. The convention
                for charge is specified by _define_bonding_convention.
;

data_atom_cip
    _name                       '_atom_cip'
    _category                    atom
    _type                        char
    _list                        yes
    _list_reference             '_atom_id'
    _definition
;               Specifies the Cahn-Ingold-Prelog designation for the atom.
                The designators are by Prelog and Helmchen (Angew. Chem.
                Int. Ed. Engl. 1982, 21, 567-583).
;

data_atom_coord_
    loop_ _name                 '_atom_coord_x'
                                '_atom_coord_y'
                                '_atom_coord_z'
    _category                    atom
    _type                        numb
    _list                        yes
    _list_reference             '_atom_id'
    _units                       Ang
    _units_detail               'angstroems'
    _definition
;               Specifies the Cartesian coordinates for the atom at an arbitary
                origin and arbitary orthogonal axes.
;

data_atom_id
    _name                       '_atom_id'
    _category                    atom
    _type                        char
    _list                        yes
    _list_mandatory              yes
    _list_uniqueness            '_atom_id'
    loop_ _list_link_child      '_bond_id_1'
                                '_bond_id_2'
                                '_stereo_vertex_id'
    _definition
;               This specifies a unique numeric identifier for an 'atom site'
                in a molecule or fragment. A designated atom site may be
                occupied by a 'dummy' atom (see _atom_type).
;
```

attribute is intended to simplify validation because it collects in one place all items which jointly identify the uniqueness of a list packet. This is in contrast to the attribute **_list_reference** which appears in the definition of each item dependent on this item.

Related attributes specify substitution information. The **_related_item** attribute identifies items related to the defined data item. The nature of this relationship is specified by the **_related_function** attribute which has value codes of *alternate*, *convention*, *conversion*, and *replace*. The definition of these codes is detailed in Chart 1. These attributes

are used to substitute data items (for validation), replace definitions (when definitions are superseded), or to change access pathways. This is particularly important for archiving because it enables old data definitions to remain in a dictionary, even when superseded by new definitions. The old and new definitions are linked by these attributes so that both data items can be validated and accessed.

5. Dictionary Controls. The final group of attributes is used exclusively for DDL dictionary purposes. They are

_dictionary_history

_dictionary_name

_dictionary_update

_dictionary_version

Dictionary_ attributes are used to audit and identify dictionary information. The attribute **_dictionary_history** specifies the entry and update information of the dictionary, and **_dictionary_name** specifies the *generic* name of the electronic file containing the dictionary (the *actual* name of the file can vary from site to site). The attributes **_dictionary_version** and **_dictionary_update** specify the version number and the date of the last change in the dictionary. Both items represent important external reference information.

DICTIONARY ORGANIZATION

The STAR File syntax permits a DDL dictionary to be organized in a wide variety of ways, all of which will be accessible to validation software. The recommended layout for DDL dictionaries is illustrated in Chart 3.

ACKNOWLEDGMENT

The DDL attributes described in this paper provide the basic vocabulary for defining data used in STAR file applications. These have been developed largely in response to the data needs of the various CIF and MIF dictionaries, and a number of colleagues have contributed significantly to expanding the DDL for this purpose. The attributes described in this paper are intended to provide the basis for existing data validation tools. The contributions to the DDL evolution must be appropriately acknowledged. Particular thanks are due to Phil Bourne at Colombia University, NY, and Paula Fitzgerald of Merck, NJ, for the suggested changes to the DDL arising from the development of the macromolecular CIF dictionary and also for their organization of *CIFmm* workshops which contributed to DDL development; to Brian Toby, NIST, MD, who coordinated the development of the Powder CIF dictionary and made numerous suggestions for improving the DDL specification; to Brian McMahon, IUCr Office, UK, who, as Secretary of COMCIFS and as a developer of CIF validation software, contributed significantly to the scope and the precision of the DDL definition; to Nick Spadaccini, U. Western Australia, and Peter Murray-Rust, Glaxo, UK, for ensuring that some of the relational aspects of the DDL were met; to Frank Allen, CCDC, UK, and John Barnard, BCI, UK, for their input from the MIF dictionary perspective. We also thank the informatics and database people involved in the Star file applications for their patience. The requirement that the DDL attributes be comprehensive enough to cope with the diversity of existing applications and yet be simple enough to be understandable by the average user, provided a significant challenge that has taken some long time to meet.

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.; 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.* **1991**, *A47*, 655–685.
(4) 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.* (accepted for publication).
(5) Notes for Authors *Acta Crystallogr.* **1994**, *C50*, 145–159.
(6) Cook, A. F. P. Dictionary Definition Language in STAR File format; *ORAC report*, 1991.
(7) Abstract Standard Notation 1. ISO Standards, ISO/IEC 8824 and ISO/IEC 8825, 1990.
(8) McMahon, B. CIFtex; *IUCr report*, Chester, UK, 1992.
(9) Hall, S. R. CIF Applications III. CYCLOPS: for validating CIF Data Names. *J. Appl. Crystallogr.* **1993**, *26*, 480–481.
(10) Hall, S. R.; Sievers, R. CIF Applications. I. *QUASAR*: for extracting CIF data. *J. Appl. Crystallogr.* **1993**, *26*, 469–473.
(11) Hall, S. R. CIF Applications. IV. *CIFtbx* a toolbox for Manipulating CIF's. *J. Appl. Crystallogr.* **1993**, *26*, 482–494.
(12) IEEE P1003.2 Draft 11.2 September 1991 IEEE Office, New York, U.S.A.

CI950010X