```
################
## ATOM_SITE ##
################
data_atom_site_[]
    _name                      '_atom_site_[]'
    _category                    category_overview
    _type                        null
    loop_ _example
          _example_detail
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   Example 1 - based on data set TOZ of Willis,
    Beckwith & Tozer [Acta Cryst. (1991), C47,
    2276-2277].
;
;   loop_
    _atom_site_label
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
    _atom_site_U_iso_or_equiv
    _atom_site_adp_type
    _atom_site_calc_flag
    _atom_site_calc_attached_atom
O1   .4154(4)  .5699(1)  .3026(0)  .060(1)  Uani  ? ?
C2   .5630(5)  .5087(2)  .3246(1)  .060(2)  Uani  ? ?
C3   .5350(5)  .4920(2)  .3997(1)  .048(1)  Uani  ? ?
N4   .3570(3)  .5558(1)  .4167(0)  .039(1)  Uani  ? ?
# - - - - data truncated for brevity - - - -
H321C  .04(1)  .318(3)   .320(2)   .14000   Uiso  ? ?
H322A  .25(1)  .272(4)   .475(3)   .19000   Uiso  ? ?
H322B  .34976  .22118    .40954    .19000   Uiso
                                             calc C322
;
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   Example 2 - based on data set DPTD of Yamin,
    Suwandi, Fun, Sivakumar & bin Shawkataly
    [Acta Cryst. (1996), C52, 951-953].
;
;   loop_
    _atom_site_label
    _atom_site_chemical_conn_number
    _atom_site_fract_x
    _atom_site_fract_y
    _atom_site_fract_z
    _atom_site_U_iso_or_equiv
S1  1  0.74799(9)  -0.12482(11) 0.27574(9) 0.0742(3)
S2  2  1.08535(10)  0.16131(9)  0.34061(9) 0.0741(3)
N1  3  1.0650(2)   -0.1390(2)   0.2918(2)  0.0500(5)
C1  4  0.9619(3)   -0.0522(3)   0.3009(2)  0.0509(6)
# - - - - data truncated for brevity - - - -
;
    _definition
;    Data items in the ATOM_SITE category record
     details about the atom sites in a crystal
     structure, such as the positional
     coordinates, atomic displacement parameters,
     and magnetic moments and directions.
;
```

Fig. 2.5.5.5. DDL1 overview of a category of items.

2.5.5.9. Definition example 9: enumeration states

The last example, in Fig. 2.5.5.9, shows the definition of an item whose value is restricted to a predictable set of values known as enumeration states. The attributes `_enumeration` and `_enumeration_detail` are used to specify which enumeration states are permitted for the defined data item. Only one of these states may appear as the value for the defined item in a CIF. The attribute `_enumeration_default` specifies the state value that is used if an item is not instantiated.

```
data_atom_site_label
    _name                      '_atom_site_label'
    _category                    atom_site
    _type                        char
    _list                        yes
    _list_mandatory              yes
    loop_ _list_link_child
                  '_atom_site_aniso_label'
                  '_geom_angle_atom_site_label_1'
                  '_geom_angle_atom_site_label_2'
                  '_geom_angle_atom_site_label_3'
                  '_geom_bond_atom_site_label_1'
                  '_geom_bond_atom_site_label_2'

    loop_ _example    C12     Ca3g28     Fe3+17
                      H*251  boron2a  C_a_phe_83_a_0
                      Zn_Zn_301_A_0
    _definition
;    The _atom_site_label is a unique identifier for
     a particular site in the crystal.
;
```

Fig. 2.5.5.6. DDL1 definition of a 'mandatory' data item.

```
data_atom_site_aniso_label
    _name                     '_atom_site_aniso_label'
    _category                   atom_site
    _type                       char
    _list                       yes
    _list_link_parent         '_atom_site_label'
    _definition
;   Anisotropic atomic displacement parameters are
    usually looped in a separate list. If this is the
    case, this code must match the _atom_site_label
    of the associated atom in the atom coordinate
    list and conform with the same rules described
    in _atom_site_label.
;
```

Fig. 2.5.5.7. DDL1 definition of a 'parent' data item.

```
data_atom_site_aniso_U_
    loop_ _name                '_atom_site_aniso_U_11'
                               '_atom_site_aniso_U_12'
                               '_atom_site_aniso_U_13'
                               '_atom_site_aniso_U_22'
                               '_atom_site_aniso_U_23'
                               '_atom_site_aniso_U_33'
    _category                   atom_site
    _type                       numb
    _type_conditions            su
    _list                       yes
    _list_reference           '_atom_site_aniso_label'
    _related_item             '_atom_site_aniso_B_'
    _related_function           conversion
    _units                     A^2^
    _units_detail             'angstroms squared'
    _definition
;   These are the standard anisotropic atomic
    displacement components in angstroms squared.
;
```

Fig. 2.5.5.8. DDL1 definition showing 'related' data items.

2.5.6. DDL1 attribute descriptions

This section provides an overview of the different attributes that make up the core data dictionary language DDL1. A more detailed description of attributes is given in the DDL1 dictionary in Chapter 4.9. In this dictionary the attributes are used to define themselves!

```
data_atom_site_adp_type
    _name                       '_atom_site_adp_type'
    _category                     atom_site
    _type                         char
    _list                         yes
    _list_reference             '_atom_site_label'
    loop_ _enumeration
          _enumeration_detail
                        Uani    'anisotropic Uij'
                        Uiso    'isotropic U'
                        Uovl    'overall U'
                        Umpe    'multipole expansion U'
                        Bani    'anisotropic Bij'
                        Biso    'isotropic B'
                        Bovl    'overall B'
    _definition
;   A standard code used to describe the type of
    atomic displacement parameters used for the site.
;
```

Fig. 2.5.5.9. DDL1 definition showing enumeration states.

DDL1 attributes are most easily understood when considered in groups with common descriptive functions. There are five basic functional groups of attributes, which perform the definition tasks of identifying, describing, typing, relating and registering data items.

2.5.6.1. Identification attributes

Establishing the identity of a data item is a primary function of a dictionary. In a data instantiation each item is recognized by a unique code, known as its data name or tag. This tag provides the most fundamental level of data validation.

The 'identification' attribute in DDL1 dictionaries is **_name**, and appears in a definition as

_name '<*dataname*>'

The tag of the defined item is the value of **_name** and, because it starts with an underscore, it must always be bounded by quotes to prevent it from being interpreted as the start of another tag–value pair. If there is more than one data item in a definition, as in the case of an irreducible set of items, the data names are entered as a list starting with the statement

loop_ _name

(see the example in Fig. 2.5.5.8).

The presence of the identification attribute **_name** in a definition is mandatory. Its value, the name of the defined items, provides for spelling validation. Note that if a data item in a CIF is not defined in a dictionary the CIF is still valid, although CIF parsers that employ dictionaries as part of the scanning process usually flag undefined items. The accepted practice to date is that undefined data are largely ignored by a dictionary validation process. In contrast, the need for certain defined items to be present in CIF data can be crucial because of list dependencies. The attributes that specify data relationships and links are considered in Section 2.5.6.4.

2.5.6.2. Descriptive attributes

Three DDL1 attributes are used to provide text descriptions of a defined data item. These are present in a dictionary for human readability, browser software or for the production of text dictionaries such as those in Part 4. This group of attributes is not machine interpretable.

_definition
_example
_example_detail

The **_definition** attribute provides a text description and as such is the primary semantic content in a defined data item. It may also be used to provide supplementary information about other machine-parsable attributes in the definition (see the definition in Fig. 2.5.5.3). The attributes **_example** and **_example_detail** are used to show typical instantiations of the defined item, as also shown in Fig. 2.5.5.6.

2.5.6.3. Typing attributes

This class of attributes is used to specify the fundamental characteristics of data items and how they may be instantiated. These attributes are machine-parsable and of particular importance in the validation of CIF data. They are

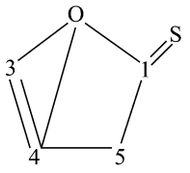
_enumeration
_enumeration_default
_enumeration_detail
_enumeration_range
_list
_list_level
_type
_type_conditions
_type_construct
_units
_units_detail

Enumeration attributes are used to specify restricted values, or 'states', of a data item (see the example in Fig. 2.5.5.9). They are not applicable in a definition of an item with unrestricted values. The attributes **_enumeration** and **_enumeration_detail** are used in definitions to specify permitted states and their descriptions. For instance, in a definition of atomic element symbols these attributes would be used to list the IUPAC 'element symbols' and 'element names'. 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 in the format <*min*>:<*max*> The attribute **_enumeration_default** may be used to specify the default value if an item is not present in a data instantiation.

List attributes specify how, and when, data items are used in a (looped) list. The attribute **_list** has a value of `yes` if a defined item must be in a list, and `no` if it must not. A value of `both` allows for both uses. The attribute **_list_level** specifies the nesting level of a list defined item. Only level 1 data are allowed in a CIF (see Chapter 2.2). Higher-nested list levels are permitted for MIF data (Allen *et al.*, 1995; see Chapter 2.4 for a description of the MIF format). Fig. 2.5.6.1 shows MIF data specifying a 2D chemical molecular diagram. The first four items in the category DISPLAY are in a level 1 list and the next two items in the category DISPLAY_CONN are in a level 2 list. Note that, according to the STAR File syntax, the nest level is automatically incremented after the first four values, and only decremented when a **stop_** signal is encountered.

Additional **_list** attributes for specifying relational dependencies between data items are described in the next attribute group.

Type attributes are used to specify the form of data. The attribute **_type** is restricted to the states `numb`, `char` and `null` that code for a number, a text string and a dictionary descriptor, respectively. The implications of this typing, in terms of how numbers or character strings are interpreted, is not specified by the CIF syntax. That is, a 'number' is simply a string of characters having one of several possible representations (*e.g.* as an integer, a floating point or in scientific notation) and it is up to the parsing software to interpret these strings appropriately. The same applies in the treatment of text strings defined as 'character'. Such strings may be bounded




```
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  S    205  140    1 2b    stop_
```

(*a*)

```
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'.
;
```

(*b*)

Fig. 2.5.6.1. (*a*) Hypothetical example and (*b*) the typical definition of nested items.

by white space, quotation characters or, in the case of multi-line text, by semicolon characters in column 1 of a line.

The attribute **_type_conditions** is used to identify application-specific conditions that apply to the typing of data items. This attribute is used to alert parsing software of string constructions that may disrupt the normal identification and validation of data types. The permitted attribute states are `none`, `esd` (and its preferred synonym `su`) and `seq`. In the definition example in Fig. 2.5.5.3 the code `esd` signals that cell-length values are measurements and may have a standard uncertainty value appended in parentheses. Fig 2.5.5.8 indicates the use of the code `su` for the same purpose. A full description of **_type_conditions** states is given in the DDL1 dictionary in Chapter 4.9.

If a data item consists of a concatenation of values, the attribute **_type_construct** may be used to specify the encoding algorithm for the component values. The language used for this algorithm is POSIX regex (IEEE, 1991). 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 embedded data items according to their

```
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 how the publication year will
    appear. Syntax is valid only for publication
    in the 20th and 21st centuries!
;
```

Fig. 2.5.6.2. Definition example: type constructions.

separate definition. For example, the chronological date is a composite of day, month and year. These may be represented in many different ways and the **_type_construct** attribute enables a specific date representation to be defined. The two definitions shown in Fig. 2.5.6.2 illustrate how a date value of the construction '1995/03/25' may be embedded into the dictionary definitions (the month and day definitions have been omitted here for brevity).

Units attributes specify the measurement units permitted for a numerical data item. The attribute **_units** specifies a code that uniquely identifies the measurement units. A description of the units identified by the code is given by the attribute **_units_detail**. Typical uses of this attribute are shown in Fig. 2.5.5.8.

2.5.6.4. Relational attributes

This class of attributes is used to describe special relationships and dependencies between data items. These attributes are machine parsable and are formally defined in the DDL1 dictionary in Chapter 4.9. They are

```
_category
_list_link_child
_list_link_parent
_list_mandatory
_list_reference
_list_uniqueness
_related_item
_related_function
```

The attribute **_category** is used to specify to which group, or basis set, a data item belongs. The value of this attribute is a name string that identifies the group and it is usually the leading part of the tags for all items in this group. Data items in a list must have the same category value. Data items in the same category may, however, be divided into different lists, provided each list contains an appropriate key data item (see **_list_reference** below).

List-link attributes are used to specify dependencies between data items in different lists. The attribute **_list_link_parent** identifies an item in another list from which the defined item was derived. The parent data item, or items in the case of irreducible sets, must have 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 that depend implicitly on the defined data item being present in the same data instantiation. The functionalities of these two attributes mirror each other. The parent item must be present in any data instantiation containing the child items, but not the converse. The definition examples in Figs 2.5.5.6 and 2.5.5.7 illustrate the use of these attributes.

The attributes **_list_mandatory** and **_list_reference** are also connected to each other. The former signals (with values of yes or no) whether the presence of a data item is essential to preserving the validity of a list of items. The latter attribute identifies the data item in the list that provides the unique key value to each packet or row of items in a list. A packet is made up of listed values, one for each item in the name list (*i.e.* the packet size matches the number of data names at the head of the list). The **_list_reference** attribute identifies items that are the keys to specific packets in the list. In Example 2 of Fig. 2.5.5.5 the key item is **_atom_site_label** and the listed labels S1, S2, N1 and C1 must be unique.

The attribute **_list_uniqueness** is used to identify items that must be unique for a list to be valid and accessible. This attribute is similar to **_list_reference** except that it appears only in a definition in which **_list_mandatory** is set to yes. This simplifies validation because it may be used as the placeholder for all items that jointly identify the uniqueness of a list packet. This is in contrast to the attribute **_list_reference**, which appears in the definition of every item dependent on this item.

Relational attributes are used to link equivalent data. The **_related_item** attribute identifies items related to the defined data item. The nature of this relationship is specified with the **_related_function** attribute according to the restricted value states of alternate, convention, conversion and replace. The definition of these states is detailed in Chapter 4.9. Relational attributes are used to provide equivalent data items, to replace definitions when definitions are superseded or to change access pathways. These facilities are for archives, because they enable old data to be accessed and the associated definitions to remain in a dictionary even when superseded by new definitions. The old and new definitions are linked by these attributes so that all related data items can be validated and accessed.

2.5.6.5. Dictionary registration attributes

This class of attributes is used to register the dictionary version and audit information. They are

_dictionary_history
_dictionary_name
_dictionary_update
_dictionary_version

Dictionary attributes are used to record the creation and update history of a dictionary. 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. An example application of these attributes is shown in Fig. 2.5.5.2.

References

Allen, F. H., Barnard, J. M., Cook, A. F. P. & Hall, S. R. (1995). *The Molecular Information File (MIF): core specifications of a new standard format for chemical data. J. Chem. Inf. Comput. Sci.* **35**, 412–427.

Bourne, P. E., Berman, H. M., McMahon, B., Watenpaugh, K. D., Westbrook, J. D. & Fitzgerald, P. M. D. (1997). *Macromolecular Crystallographic Information File. Methods Enzymol.* **277**, 571–590.

Gray, P. M. D., Kulkarni, K. G. & Paton, N. W. (1992). *Object oriented databases*. New York: Prentice Hall.

Hall, S. R. (1991). *The STAR File: a new format for electronic data transfer and archiving. J. Chem. Inf. Comput. Sci.* **31**, 326–333.

Hall, S. R., Allen, F. H. & Brown, I. D. (1991). *The Crystallographic Information File (CIF): a new standard archive file for crystallography. Acta Cryst.* A**47**, 655–685.

Hall, S. R. & Cook, A. P. F. (1995). *STAR dictionary definition language: initial specification. J. Chem. Inf. Comput. Sci.* **35**, 819–825.

Hall, S. R. & Spadaccini, N. (1994). *The STAR File: detailed specifications. J. Chem. Inf. Comput. Sci.* **34**, 505–508.

IEEE (1991). *IEEE Standard for Information Technology – Portable Operating System Interface (POSIX) – Part 2: Shell and utilities*, Vol. 1, IEEE Standard 1003.2-1992. New York: The Institute of Electrical and Electronic Engineers, Inc.

Kim, W. (1990). *Introduction to object oriented databases*. Boston: MIT Press.

Westbrook, J. D. & Hall, S. R. (1995). *A dictionary description language for macromolecular structure.* http://ndbserver.rutgers.edu/mmcif/ddl/.