International Tables for Crystallography (2006). Vol. G, Section 3.1.5, pp. 76–79.

several tables, or the implied key may not have a unique value unless concatenated with other fields in the table row. For many applications this is only of academic interest; but in some subdisciplines it is important that the data model is constrained strictly to a relational one, and for those applications dictionaries built on the DDL2 formalism are more appropriate.

Of the dictionaries presented in this volume, the core, powder, modulated structures and electron density dictionaries use the DDL1 formalism and the symmetry, macromolecular and image dictionaries use the DDL2 formalism. The core dictionary uses DDL1 so that it can be used alongside other less rigorous dictionaries. The powder dictionary is one case of this, where the need to tabulate and merge extensive lists of raw or processed data is not well served by a relational model. Modulated structures are also best served by a data model that is not rigorously relational. The macromolecular dictionary uses DDL2 because many of the major database applications in macromolecular crystallography are relational in nature, but in consequence it contains a copy of the core data items re-expressed in DDL2 formalism. The image dictionary is in DDL2 because it was designed to operate closely alongside the macromolecular dictionary. The symmetry dictionary is an interesting case. It was constructed in DDL2 format as an exercise in supplying an extension dictionary immediately suitable for direct incorporation into other DDL2-based dictionaries and also suitable for transformation to the simpler DDL1 formalism as necessary to complement existing DDL1 dictionaries.

While the main difference between DDL1 and DDL2 lies in the rigour with which relational data structures are enforced, DDL2 also offers a larger set of attributes for specifying hierarchical relationships between data names and for typing data values, and in consequence a complete DDL2-based dictionary is richer (and correspondingly more complex to construct) than an equivalent DDL1 description.

There may be no obvious reason for selecting one formalism over the other when planning a new data dictionary, and prospective authors must give considerable thought to the merits of both formalisms. However, once the choice has been made, the structure of the dictionary and its component definitions is profoundly affected. The constructions of the two types of dictionary are discussed separately in Sections 3.1.5 and 3.1.6 below.

3.1.5. Constructing a DDL1 dictionary

Dictionaries constructed according to DDL1 have quite a simple structure. The structure is summarized in this section; Sections 3.1.5.1–3.1.5.4 provide more detail. Each definition is encapsulated within its own data block. Fig. 3.1.5.1 outlines the contents of the core CIF dictionary. The order of the data blocks has no significance, but it is common practice to start the file with the data block that describes the name, version and revision history of the dictionary itself and then to arrange data blocks in alphabetical order, sorted first on category then on names within a category. This practice is not always followed – for example, the powder dictionary is ordered by theme. The choice of order in a dictionary is only used for presentation and dictionary parsers should not assume or rely on any order of data blocks.

The name of a data block is usually constructed from the name of the data item it describes, *e.g.* `data_refln_phase_meas`. Where the data block describes an entire category instead of a single data item, the category name is followed by matching square brackets, which may contain an alphabetic code representing the dictionary name if it is an extension to the core dictionary (*e.g.* `data_refln_[]`, `data_audit_link_[ms]`). Where the data block defines several data names, the initial common portion of the names is used with a trailing underscore (*e.g.* `data_refln_`).

```
data_on_this_dictionary
     _dictionary_name            cif_core.dic
     _dictionary_version         2.3.1
     _dictionary_update          2005-06-27
     _dictionary_history
;
   1991-05-27  Created from CIF Dictionary text. SRH
   . . .
;
```

(*a*)

```
data_atom_site_[]
    _name                         '_atom_site_[]'
    _category                       category_overview
    _type                           null
    loop_ _example
    _example_detail    .   .

data_atom_site_adp_type
    _name                         '_atom_site_adp_type'
    _category                       atom_site
    _type                           char
    _definition                   'A standard code ...'

data_atom_site_aniso_B_
    loop_ _name                   '_atom_site_aniso_B_11'
                                    . . .
```

(*b*)

Fig. 3.1.5.1. Schematic structure of core CIF dictionary. (*a*) Dictionary identifiers. (*b*) Definitions of categories and data items.

A preliminary data block, by convention labelled with the header string `data_on_this_dictionary`, contains the dictionary identification information and revision history. The name of the dictionary itself (given by the data name `_dictionary_name`) is conventionally of the form `cif_`*`identifier`*`.dic`, where the *identifier* is a short code for the topic area of the dictionary (*e.g.* 'core' for the core dictionary, 'pd' for the powder dictionary, 'ms' for the modulated structures dictionary, 'rho' for the electron density dictionary).

Data names are classified by category. The `_category` attribute is a character string intended to indicate the 'natural grouping' of data items. If a data item occurs in a looped list, it must be grouped only with items from the same category. It is, however, permissible for a file to contain more than one looped list of the same category, provided that each loop has its own specific reference item identified by the `_list_reference` attribute of the data names included. Examples of this will be given below.

For each category, a data block is usually provided that contains information about the purpose of the category, generally illustrated with examples.

All other data blocks represent self-contained definitions of a single data item or a small set of closely related data items. The definition includes the physical units of and constraints on the values of the data labelled by the defined data name, and also information about relationships with other data items.

It is conventional, although not mandatory in DDL1 dictionaries, that the category name should appear as the leading component or components of a data name. For example, the data name `_exptl_crystal_colour` is a member of the core category EXPTL, while `_exptl_crystal_density_meas` is a member of the category EXPTL_CRYSTAL and `_exptl_crystal_face_perp_dist` is a member of the category EXPTL_CRYSTAL_FACE. However, it will

Example 3.1.5.1. *A DDL1 dictionary identification block.*

```
 data_on_this_dictionary
     _dictionary_name           cif_core.dic
     _dictionary_version        2.3.1
     _dictionary_update         2005-06-27
     _dictionary_history
;  1991-05-27  Created from CIF Dictionary text. SRH
   1991-05-30  Validated with CYCLOPS & CIF ms. SRH
   ...
;
```

be seen that there is no sure way of working out the category from the complete data name except by referring to its `_category` attribute in the associated dictionary. This differs from the DDL2 convention of including an explicit separator (a full stop) between the category name and the remainder of a data name.

While it is not mandatory that a data name should incorporate its category name as a leading component, authors are strongly encouraged to adopt this convention. A small number of core data items that did not conform to this convention have been deprecated in later releases of the core dictionary. However, in the powder dictionary the convention has been broken so that one can present data sets separately or merge them together. In this dictionary, some data names beginning with the strings `_pd_calc`, `_pd_meas` and `_pd_proc` all belong formally to the category PD_DATA. This allows calculated data values to be tabulated with raw and processed measurements if this is useful.

One other case where a data name does not begin with its associated category name is that of the pseudo data names such as `_exptl_[]` that appear in the dictionary to describe the purpose of a category (Section 3.1.5.3). Such data names are always assigned the category CATEGORY_OVERVIEW and are further differentiated from other data names by having a data type of 'null'.

3.1.5.1. The dictionary identification block

As mentioned above, the dictionary file must contain information that unambiguously states its identity and version. In DDL1-based CIF dictionaries, this is achieved by itemizing the full set of dictionary attributes (see Section 2.5.6.5) within a data block named `data_on_this_dictionary`, as in Example 3.1.5.1 from the core dictionary.

3.1.5.2. Irreducible sets of data items

In general, a dictionary data block defines a single data item. However, there are instances where several related data names are defined in the same data block. Sometimes this has been done for convenience, to produce a compact listing of similar data names that have common attributes and whose small differences in meaning can best be expressed by a single definition. Such groupings are discouraged, except where they represent components of a larger entity that has no sensible meaning in the absence of any of the components. For example, the data block `data_refln_index_` defines the three data items `_refln_index_h`, `_refln_index_k` and `_refln_index_l` that represent the Miller indices of a reflection. All three indices must have a value in order to specify a reflection and so each has no meaning in isolation.

Note that there is no formal method of expressing this close relationship within DDL1 except by grouping the definitions in the same data block in this way. In DDL2 dictionaries, it is common to assign the components of an irreducible set to a specific subcategory.

Example 3.1.5.2. *A category description in a DDL1 dictionary.*

```
data_exptl_[]
    _name                         '_exptl_[]'
    _category                       category_overview
    _type                           null
    loop_ _example
    _example_detail
# - - - - - - - - - - - - - - - - - - - - - - - - -
;   _exptl_absorpt_coefficient_mu     0.962
    _exptl_absorpt_correction_type    psi-scan
    _exptl_absorpt_process_details
                'North, Phillips & Mathews (1968)'
    _exptl_absorpt_correction_T_min   0.929
    _exptl_absorpt_correction_T_max   0.997
;
;   Example 1 - based on a paper by Steiner [Acta
                Cryst. (1996), C52, 2554-2556].
;
# - - - - - - - - - - - - - - - - - - - - - - - - -
    _definition
;      Data items in the EXPTL category record
       details about the experimental work prior
       to the intensity measurements and details
       about the absorption-correction technique
       employed.
;
```

3.1.5.3. Category descriptions

As discussed above, categories in DDL1 are intended as 'natural groupings' of data items. To document the purpose of a category within a dictionary, 'pseudo' data names are used. All pseudo data names are assigned a `_category` attribute of `category_overview` and have an associated `_type` value of 'null'. They are also named by convention as `_category_name_[dictionarycode]`, for example `_pd_data_[pd]` for the description of the PD_DATA category in the powder dictionary (indicated by the code 'pd' in square brackets). For the core dictionary, *dictionarycode* is not given, resulting in names like `_exptl_[]` to describe the EXPTL category.

Example 3.1.5.2 is a slightly edited extract from the core dictionary showing how a data block for a category description is composed, including the presence of an example.

Note that the *dictionarycode* extension allows a dictionary to include comments on items that it defines in a category already established in the core dictionary. For example, the modulated structures dictionary includes the category overview item `_audit_link_[ms]`. This describes the convention adopted to express the relationship between data blocks in a modulated structures data file using the `_audit_link_` data names already defined in the core dictionary.

3.1.5.4. Data-item definitions

The data blocks described in Sections 3.1.5.1 and 3.1.5.3 are used to identify the dictionary and to describe the nature and purpose of a category. The remaining data blocks in a dictionary provide the attributes of data values in a form suitable for machine extraction and validation. The following examples show how this is done for various types of data.

3.1.5.4.1. *Definitions of single quantities*

Example 3.1.5.3 is the core dictionary definition of the data name for the ambient temperature during the experiment. Because this is a single (non-looped) value, the relevant data name is one among several discrete items in the DIFFRN category. No further description of its relationship to other data items is required.

The type of the associated data value (*numb* for numerical) is specified, together with any constraint on its legal value. The range

Example 3.1.5.3. *A simple definition of a data item describing a physical quantity.*

```
data_diffrn_ambient_temperature
    _name               '_diffrn_ambient_temperature'
    _category                    diffrn
    _type                        numb
    _type_conditions             esd
    _enumeration_range           0.0:
    _units                       K
    _units_detail                kelvin
    _definition
;      The mean temperature in kelvins at which the
       intensities were measured.
;
```

specified (`0.0:`) indicates that it may be any non-negative real number. The physical units of the quantity are also indicated.

The `_definition` attribute is a concise human-readable documentation of the meaning associated with the data name.

Example 3.1.5.4 is taken from the powder dictionary and illustrates a data item that can have only one of a limited set of values. This data item indicates the geometry of the experiment. The associated data value is of type *char* and may legally take only one of the two possible values listed.

3.1.5.4.2. *Looped data*

Many of the attributes of looped data items, such as their physical units or valid numerical values, may be defined in exactly the same way as for non-looped data. However, more care needs to be taken to describe the relationships between different looped data items.

Consider the following example listing of some three-dimensional atom-site coordinates and displacement parameters.

```
loop_
     _atom_site_label
     _atom_site_fract_x
     _atom_site_fract_y
     _atom_site_fract_z
     _atom_site_U_iso_or_equiv
     _atom_site_thermal_displace_type
    O1 .4154(4) .56990(10) .3026000   .0600(10) Uani
    C2 .5630(5) .5087(2)   .32460(10) .060(2)   Uani
    C3 .5350(5) .4920(2)   .39970(10) .0480(10) Uani
    N4 .3570(3) .55580(10) .4167000   .0390(10) Uani
    C5 .3000(5) .6122(2)   .35810(10) .0450(10) Uani

loop_
     _atom_site_aniso_label
     _atom_site_aniso_U_11
     _atom_site_aniso_U_22
     _atom_site_aniso_U_33
     _atom_site_aniso_U_12
     _atom_site_aniso_U_13
     _atom_site_aniso_U_23
O1 .071(1) .076(1) .0342(9) .008(1)  .0051(9) -.0030(9)
C2 .060(2) .072(2) .047(1)  .002(2)  .013(1)  -.009(1)
C3 .038(1) .060(2) .044(1)  .007(1)  .001(1)  -.005(1)
N4 .037(1) .048(1) .0325(9) .0025(9) .0011(9) -.0011(9)
C5 .043(1) .060(1) .032(1)  .001(1) -.001(1)   .001(1)

loop_
     _geom_bond_atom_site_label_1
     _geom_bond_atom_site_label_2
     _geom_bond_distance
      O1  C2   1.342(4)
      O1  C5   1.439(3)
      C2  C3   1.512(4)
      C2  O21  1.199(4)
```

These loops, or tables of values, are properties of atom sites, each identified by a label such as O1. The definition of a data name such as `_atom_site_U_iso_or_equiv` expresses this by using the DDL1 `_list_reference` attribute (Example 3.1.5.5).

Example 3.1.5.4. *A data item that can take only one of a discrete set of allowed values.*

```
data_pd_spec_mount_mode
    _name                         '_pd_spec_mount_mode'
    _category                      pd_spec
    _type                          char
    loop_ _enumeration             reflection
                                   transmission
    _definition
;      A code describing the beam path through
       the specimen.
;
```

Example 3.1.5.5. *Definition relating a looped data item to the item used to identify a 'loop packet', or row of entries in a table.*

```
data_atom_site_U_iso_or_equiv
    _name               '_atom_site_U_iso_or_equiv'
    _category                      atom_site
    _type                          numb
    _type_conditions               esd
    _list                          yes
    _list_reference               '_atom_site_label'
```

Example 3.1.5.6. *Definition of a mandatory item within a loop.*

```
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_bond_atom_site_label_1'
          '_geom_bond_atom_site_label_2'
```

For an entry in the table to make sense, the site identifier must be present, so the definition for `_atom_site_label` declares it a mandatory item within its list (Example 3.1.5.6).

It is common for an atom-site identifier to be used in several related tabulations in a particular crystal structure description, and in a CIF description this means that it may occur in several different looped lists. The dictionary definition gives a formal account of this by listing the data names in other looped lists which are just different manifestations of this same item. This is done using the `_list_link_child` attribute, which identifies the data names to which the one being currently defined is 'parent'. In Example 3.1.5.6 (which is a subset of the full list in the core dictionary), `_atom_site_aniso_label`, `_geom_bond_atom_site_label_1` and `_geom_bond_atom_site_label_2` are identified as children of `_atom_site_label`.

It can been seen immediately that `_atom_site_aniso_label` is the atom-site identification label appearing in the second table in the example listing above, and the `_geom_bond_` items are clearly atom-site labels in a table of bonding properties between specified sites. There is, however, a difference between the two secondary tables: the bond-properties table is described by data items in the GEOM_BOND category, but the table of anisotropic displacement parameters includes data names that have the same `_category` attribute as the coordinate data items, namely ATOM_SITE. The latter is an example of multiple lists or tables belonging to the same category, a feature permitted only in DDL1-based data files.

3.1.5.4.3. *Units*

The physical units in which a quantitative data item must be expressed are identified by the DDL1 attributes `_units` and `_units_detail`. The latter is a character field describing the units; the `_units` attribute is a code that may be interpreted by machine. In DDL1-based dictionaries, type codes are purely conventional, and there is no mechanism for converting units or relating quantities in different units. Table 3.1.5.1 lists the units codes used in the DDL1-based dictionaries described in this volume. There can be some inconsistencies: two codes ('s' and 'sec') are already in use to indicate the time unit of seconds.

The original CIF paper (Hall *et al.*, 1991) described a convention allowing physical quantities to be listed in a CIF in units other than those specified in the dictionary. Under this convention, a data name representing a value expressed in different units could be constructed by appending one of a series of known 'units extension codes' to the standard data name. Thus `_cell_length_a_pm` would represent a cell length expressed in picometres instead of the default ångströms. This approach is now deprecated, and all quantities must be expressed in the single unit permitted in their definition block. However, to allow the formal validation of old CIFs, a 'compatibility dictionary' is available which defines all data names that could have been constructed under this convention in a properly DDL1.4-compliant form. *This dictionary should only be used for validating old CIFs, and must not be used to construct new data files.* The dictionary is called cif_compat.dic in the IUCr CIF dictionary register (see Section 3.1.8.2).

Table 3.1.5.1. *Units codes and their interpretation in DDL1-based dictionaries*

Unit code (`_units`)	Meaning (`_units_detail`)
`A`	Ångströms
`A^-1^`	Reciprocal ångströms
`A^2^`	Ångströms squared
`A^3^`	Ångströms cubed
`Da`	Daltons
`K`	Kelvins
`Kmin^-1^`	Kelvins/minute
`Mgm^-3^`	Megagrams per cubic metre
`\ms`	Microseconds
`deg`	Degrees
`deg/min`	Degrees per minute
`eV`	Electronvolts
`e_A^-3^`	Electrons per cubic ångström
`fm`	Femtometres
`kPa`	Kilopascals
`kV`	Kilovolts
`kW`	Kilowatts
`mA`	Milliamperes
`min`	Minutes
`mm`	Millimetres
`mm^-1^`	Reciprocal millimetres
`s`	Seconds
`sec`	Seconds

3.1.6. Constructing a DDL2 dictionary

The DDL2 dictionary definition language was designed to specify a relational data model and has provision for including within a dictionary tables of relationships between data entries. Like a relational database which contains tables describing the data tables in the database, DDL2-based dictionaries contain definition blocks describing CIF categories, units and relationships as well as data items.

Unlike DDL1 dictionaries, a DDL2 dictionary is presented as a single data block. Within this data block a number of looped lists describe properties of the dictionary as a whole, or properties and relationships shared across the items defined in the dictionary. Typically these are: the dictionary name, version identifiers and revision history; the category groupings that give structure to the items defined by the dictionary; the labels that identify closely related data items; and the physical units employed in the dictionary, their definitions in terms of base units and their interconversion factors.

Definitions of individual data items and categories are contained within save frames. While the save frames are not referenced by name in any dictionary application, they permit multiple occurrences of data definition tags within the scope of a single data block and are therefore suitable for structuring a data dictionary. It is a convention that the name of a save frame defining a category is given in capitals, and the name of a save frame for a definition of a data item is given as lower-case. For example, `save_ATOM_SITE` is the name of the save frame defining the category with the `atom_site` identifier, while `save__atom_site.details` is the name of the save frame holding the definition of the individual data name `_atom_site.details` (note how the initial underscore character of the data name is preserved following the initial `save_` string of the save-frame name).

As with DDL1 dictionaries, the name of the dictionary itself (given by the data name `_dictionary.title`) is usually of the form `cif_`*`identifier`*`.dic`, where the *identifier* is a short code for the topic area of the dictionary (*e.g.* 'img' for the image dictionary, 'sym' for the symmetry dictionary).

As is invariable with DDL2 data names, the names themselves are formed from the category name separated by a full stop from the specific descriptor of the item.

Fig. 3.1.6.1 shows the structure of the macromolecular CIF dictionary. The ordering of the various looped lists and save frames is of no significance for machine parsing. The sole data block has the same name as the dictionary title string and the data block is introduced by the dictionary identification data items. The dictionary revision history introduces the file, followed by information about the extended data types and physical units used within the current dictionary. These are followed by the lists of closely related items (corresponding to 'irreducible sets' in DDL1 dictionaries and called 'subcategories' in the terminology of DDL2) and lists of category groupings. The body of the dictionary contains category and item definitions. Each category definition is followed by the definitions of its component data items. The ordering is alphabetic by category and then alphabetic by item name within categories.

3.1.6.1. Dictionary identification

Dictionary files must contain information that unambiguously states their identity and version. In DDL2-based dictionaries this is done using the dictionary attributes described in Section 2.6.6.4. The name of the data block comprising the whole content of a DDL2 dictionary is by convention the same as the dictionary identification string given as `_dictionary.title`. This value is repeated as the value of `_dictionary.datablock_id` (see Example 3.1.6.1) for use in checking the consistency of the dictionary.

The dictionary history is also an important audit record of changes to the dictionary content. Unlike in DDL1-based dictionaries where the history is contained in a single field, DDL2 provides a looped list of version labels, dates and annotations. For convenience, the history records in large DDL2-based dictionaries are sometimes placed at the end of the dictionary file.

3.1.6.2. Subcategory definitions

In the DDL1 formalism, particular relationships between data items may sometimes be stated within a text description or may be implied by the organization of the dictionary (where several data