DDLM: *NEXT GENERATION* DICTIONARY DEFINITION LANGUAGE

by

Syd Hall, Nick Spadaccini and John Westbrook

(Version: 13 August 2008)

1. INTRODUCTION

This document provides the specifications of *DDLm*, a next generation dictionary definition language designed to support the definition capabilities of the existing dictionary languages, *DDL1* (Hall and Cook, 1995) and *DDL2* (Westbrook and Hall, 1995), as well as provide a higher level of semantic content for domain dictionaries. While providing important new functionality to the dictionary developer, *DDLm* fully supports the semantic capabilities of existing IUCr data dictionaries. No changes are required in existing archival data files in order to apply domain dictionaries written in *DDLm*.

Data dictionaries contain precise, machine-parseable definitions of data items, and consequently play a vital role in the rapid and reliable exchange of electronic scientific data. Increasingly data dictionaries play a major role in the automatic interpretation and validation of data. In crystallography, the exchange and deposition of CIF data (Hall, Allen & Brown, 1991) is supported by CIF dictionaries for each of the different sub-disciplines (Hall & McMahon, 2005). At present CIF dictionaries are constructed using *DDL1* and *DDL2*; the latter having evolved from the former to extend the scope and precision of dictionary definitions for macromolecular structural data.

The automatic and seamless exchange of electronic data depends implicitly on the level of usable semantic information provided by dictionaries, as this largely determines the precision of the validation and knowledge management processes. For example, the ability of a dictionary definition language to express the algorithmic methods that relate derived data items enables data definitions to assume a much more fundamental role in data validation and evaluation; a role currently performed by customized software. The advantage of using semantic knowledge stored in a dictionary for this purpose (particularly if this information has been adopted internationally by an official scientific body) is the assurance that standard, well-understood algorithms are uniformly adhered to.

Scientific data, by its very nature, are in a state of continual change and this necessitates efficient exchange processes that are in step with the evolution and creation of new data definitions. Semantically-rich and readily-updatable data dictionaries allow this to happen. *DDLm* is a next generation dictionary definition language. It encompasses the particular strengths of *DDL1* and *DDL2*, and draws upon many attributes used in *StarDDL*, to provide a more hierarchical, concise and implicit approach to data definitions. In particular, it incorporates a methods expression language, *dREL* (Spadaccini, Hall and Castleden, 2000) for relating derived data items. *dREL* applies data items computationally in relational expressions as object-oriented variables. This, with the facility to import externally-sourced data definitions, provides dictionaries written in *DDLm* with a higher level of functionality which encourages modular dictionaries with shared common definitions.

For those unfamiliar with the construction of data dictionaries, we start with a brief introduction to the terminology used in this document. The term "DDL", already used above, refers to a *dictionary definition language* in which the *lingua* is composed of a prescribed set of *attributes*. Each *attribute* serves to identify a particular characteristic of a data item. The definition of a particular data item is achieved by assigning values to the appropriate attributes that describe its precise

nature, and its relationship to other items. Which attributes are used in a definition, and which are not, is detailed later in 6.2. The collection of definitions for set of data items is referred to as a *domain* (or *discipline)* dictionary. It is also important to appreciate that the attributes themselves are defined within their own dictionary using the same DDL (i.e. each attribute is specified using a collection of the appropriate attributes) and this is referred to as the *DDL dictionary*. Finally, when a collection of data items is ascribed actual values, this is referred to as a *data file* or an *instance document*. These terms are used throughout this document.

2. OVERVIEW OF *DDLM*

2.1 MAIN DIFFERENCES TO *DDL1* AND *DDL2*

The attributes of *DDLm* encompass those in *DDL1* and *DDL2*, albeit with a richer syntax designed to increase the semantic content of definitions while keeping definition repetition to a minimum. The main difference to *DDL1* and *DDL2* are the addition of:

- methods using a symbolic algorithmic language, dREL, to relate derivable items,
- stricter and multi-level attribute inheritance between data categories,
- increased precision and flexibility in data typing,
- implicit child-parent relationship between equivalent data items, and
- procedures for importing common definitions from external dictionaries.

2.2 IMPLICIT CHILD RELATIONSHIPS OF LINKED ITEMS

Parent category relationships (i.e. categories of items that are derived from or members of another category) are explicitly identified in DDLm but not the child categories. Child category relationships can readily be derived on instantiation of the dictionary from the parent links. Removing the need to identify child categories in a dictionary eliminates a cumbersome and error-prove task for dictionary developers, and enhances the readability of the dictionary by reducing redundant information which obscures more critical aspects of definitions.

2.3 STRONGER DATA TYPING.

DDLm provides an expanded attribute set for specifying data types with the introduction of attributes for establishing the *nature* and the *origin* of items. The data typing attributes now cover four basic classes of information; known simply as the *container*, *contents*, *purpose* and *dimension* attributes. The *container* attribute identifies the *composition* of the item value. Allowed container types are *Single, Multiple, Array*, *List*, *Tuple* and *Table*. The *contents* attribute identifies the nature of the individual value(s) within the container, in terms of numerical and character descriptors. The contents attribute types are enumerated but can be expanded in the auxiliary file `com_val.dic`. The *purpose* attribute identifies the *origin* and *function* of a data item. This information is important at many levels; it indicates, for example, if a numerical value is expected to have a standard uncertainty value or not; if the value is an active filename, or a code used to import definitions; if the value has been measured, observed or assigned. This detail is needed for manipulating data within a method expression using the algorithmic language *dREL*. Also, by specifying that an item is *measured*, *observed* or *assigned,* associated validation and evaluation processes may be handled consistently and, in some cases, automatically within the method expressions.

2.4 CATEGORY INHERITANCE

In dictionaries written in *DDLm* the hierarchical parent relationships between categories of data items is specified explicitly. By default, each category inherits properties of its parent. The inheritance process is detailed later in §4.1.

2.5 DATA TAG CONSTRUCTION

The naming of individual items, and groups of items, is more flexible in *DDLm*. The name of a category is in most cases the leading string in an item tag. This is no longer a requirement, however, because the category tag and *object* tag (identified by the trailing characters) are now specified separately in a dictionary using the attributes `_name.category_id` and `_name.object_id`. Identifying the two parts of an item tag with separate attributes:

- permits the tag constructions that do not comply to the *DDL2* name rules,
- avoids progressive elongation of tags for deeply nested families of categories, and
- removes the need for a period to be used to separate the category and object names.

3. DICTIONARY ORGANISATION

The construction and organization of a dictionary written in *DDLm* conforms to the *Star File* syntax. Each item definition and each category definition is contained within a *save_* frame, known as the "definition frame". These frames contain a sequence of "attributes" as tag-value pairs. All definition frames in a dictionary file are contained with a single *data_* block, known as the "dictionary block". Attributes defining dictionary-only information are contained within the data block but not within a save frame.

3.1 DEFINITION FRAME

Each item definition and category definition in a dictionary is contained within a *save_* frame i.e. the definition starts with a `save_`*`frame_name`* statement and ends with a `save_` statement. The "*`frame_name`*" usually matches the tag of the category or item being defined within the frame. Here an example of a *category* definition of `ATOM_SITE` from the *Core_Structure* dictionary.

```
save_ATOM_SITE
    _definition.id              atom_site
    _definition.scope           Category
    _definition.class           List
    _definition.update          2007-02-06
    _description.text
;
     The CATEGORY of data items used to describe atomic site information
     used in crystallographic structure studies.
;
    _description.common         'Atom Site List'
    _category.parent_id          core_structure
    _category_key.generic      '_atom_site.key'
    _category_key.primitive    (_atom_site.label)
     save_
```

Definition 3.1.1

Here is an example of an *item* definition from the *Core_Structure* dictionary.

```
save_atom_site.description
    _definition.id             '_atom_site.description'
```

```
    _definition.update            2006-06-29
    _description.text
;
     A description of special aspects of this site. See also
     _atom_site.refinement_flags.
;
    _description.common          'Atom Site Details'
    _name.category_id             atom_site
    _name.object_id               description
    _type.purpose                 Describe
    _type.container               Single
    _type.contents                Text
     loop_
    _description_example.case    'Ag/Si disordered'
     save_
```

Definition 3.1.2

3.2 DICTIONARY BLOCK

Each dictionary written in *DDLm* must start with a data block statement which identifies the dictionary. For example, in the *core_structure* dictionary this statement is

```
data_CORE_STRUCT
```

whereas in the dictionary defining the *DDLm* attributes themselves, it is

```
data_DDL_DIC
```

3.3 DICTIONARY-SPECIFIC ATTRIBUTES

Each dictionary block contains attributes specifying the properties of the dictionary file as a whole. These attributes are not contained within a save frame and are inherited by all other categories and items, *except where the same attribute in individual definitions overrides the inherited attribute value*. For example, the start of the *core_structure* dictionary is as follows.

```
data_CORE_STRUC

    _dictionary.title             CORE_STRUC
    _dictionary.class             Instance
    _dictionary.version           1.1.05
    _dictionary.date              2008-02-12
    _dictionary.uri               www.iucr.org/cif/dic/core_struc.dic
    _dictionary.ddl_conformance   3.7.09
    _dictionary.namespace         CoreStruc:
    _description.text
;
     This dictionary contains the definitions of data items
     as considered CORE to the description of STRUCTURE data.
;
```

Definition 3.3.1

The dictionary block must also contain a "head" category definition frame that is the parent of all other categories in the dictionary. For example, the *core_structure* dictionary block contains the head category definition as follows.

```
save_CORE_STRUCTURE
    _definition.id                core_structure
    _definition.scope             Category
    _definition.class             Head
    _definition.update            2008-02-12
    _description.text
;
    The DICTIONARY group encompassing the CORE STRUCTURE data items defined
    and used with in the Crystallographic Information Framework (CIF).
;
    _category.parent_id           cif_core
```

```
    save_
```

Definition 3.3.2

The dictionary block also contains dictionary audit information. For convenience, this is usually placed at end of the dictionary, following the last item definition frame. Here is a typical audit list.

```
     loop_
    _dictionary_audit.version
    _dictionary_audit.date
    _dictionary_audit.revision

      1.0.01   2005-12-12
;
   Initial version of the TEMPLATES dictionary created from the
   definitions used in CORE_3 dictionary version 3.5.02
;
      1.0.02   2006-02-12
;
   Remove dictionary attributes from a save frame.
   Change category core_templates to template
;
```

Example 3.3.1

4. CATEGORY AND ITEM RELATIONSHIPS

4.1 CATEGORY HIERARCHY

As stated in section §3.3, each dictionary block contains a head category definition that is the "parent" for all other categories in the dictionary. The head category is at the top of a hierarchical tree of categories in the dictionary, and all other categories in the dictionary file are, directly or indirectly, its children. In a dictionary written in *DDLm*, lower categories in the hierarchy inherit attribute values from their parent categories, except when these are superceded by the local specification of the same attribute.

Example 4.1.1 below shows is a typical list of categories in a dictionary (category tags in blue and green), followed by the contained items (item tags in red) and categories. In this example, the items describing the crystal *unit cell* in the *core_crystal* dictionary are listed.

```
CORE_CRYSTAL
           CELL
          _cell.atomic_mass
          _cell.formula_units_Z
          _cell.metric_tensor
          _cell.orthogonal_matrix
          _cell.special_details
          _cell.volume
           CELL_ANGLE
          _cell_angle.alpha
          _cell_angle.beta
          _cell_angle.gamma
           CELL_LENGTH
          _cell_length.a
          _cell_length.b
          _cell_length.c
           CELL_VECTOR
          _cell_vector.a
          _cell_vector.b
          _cell_vector.c
           CELL_MEASUREMENT
          _cell_measurement.pressure
          _cell_measurement.radiation
```

```
          _cell_measurement.reflns_used
          _cell_measurement.temperature
          _cell_measurement.theta_max
          _cell_measurement.theta_min
          _cell_measurement.wavelength
                    CELL_MEASUREMENT_REFLN (parent: cell )
                   _cell_measurement_refln.hkl
                    _cell_measurement_refln.theta
                    _cell_measurement_refln.index_h
                    _cell_measurement_refln.index_k
                    _cell_measurement_refln.index_l
```

Example 4.1.1

The parent of each category is specified in its definition with the attribute `_category.parent_id`. For example, in the CELL category definition this attribute may have the value `'core_crystal'`; whereas in the CELL_ANGLE category definition this attribute would have the value `'cell'`. The list shown in Example 4.1.1 is indented according to the hierarchical relationship of items to the parent category. Part of this relationship is the assumption that all categories of items can be subsumed (i.e. merged or joined) into the parent category. This property is however modified by the *category scope*, which may have either the state code *List* or *Set* (the scope state *Head* also exists but will not be discussed here). Whereas items in an instance document belonging to a category of scope *List* must appear as a looped list (see §4.2), items with scope *Set* need not (they may, however, be referenced as a class of objects in methods expressions - examples of this will be shown later). More importantly, items in a *Set* category are seen to be automatic members of the parent category whereas *List* items are not unless the attribute `_category.parent_join` is specified as *Yes* in the category definition.

We shall see later that the ability to specify relationships between categories of items is particularly important in enriching the semantics of data.

4.2 *LIST* CATEGORIES AND REFERENCE KEYS

When an item can have more than one value it will appear in a data file as a *looped list*, and all items in this list will be members of a single category or of joined categories (see §4.3). For example, the coordinates of atomic sites are defined in the ATOM_SITE category (see Definition 3.1.1 above).

```
loop_
_atom_site.label
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
  o1   .5501(6)   .6371(6)   .1601(13)
  o2   .4012(6)   .5162(6)   .2290(12)
  o3   .2502(7)   .5705(7)   .6011(14)
  c1   .4170(8)   .6931(9)   .4965(18)
  c2   .3144(8)   .6702(9)   .6420(19)
  c3   .2789(9)   .7494(10)   .838(2)
```

Example 4.2.1

A looped list is a 2D table in which the data names are a *row of header tags* identifying *columns of values*. E.g., the tag `_atom_site.label` refers to the column of values "o1", "o2",...,"c3". In each table at least one item must have unique values in order that other values in a row may be unambiguously accessed. These rows of values are often referred to as "packets"or "list instances". The item that is designated to have "unique" values in a category is known as the "category key". The *key* in the ATOM_SITE *category* above is `_atom_site.label` and each value of this item provides the access to the other three items in the packet. For example, "o3" points to the values ".2502(7)", ".5705(7)" and ".6011(14)", and no others! In a dictionary the key is specified in

the category definition with the `_category_key.generic` or `_category_key.primitive` attributes (see the ATOM_SITE definition in §3.1 above).

The attribute information in the ATOM_SITE *category* definition (see Definition 3.1.1) refers all ATOM_SITE items. That is, the properties of a category apply to all items that are defined with the attribute `_name.category_id` set with the value "atom_site".

The definition of the key item `_atom_site.label` is as follows.

```
save_atom_site.label
     _definition.id              '_atom_site.label'
     _import_list.id           (('Att','atom_site_label','com_att.dic'))
     _name.category_id             atom_site
     _name.object_id               label
      save_
```

Definition 4.2.1

Compared to `_atom_site.description` (see Definition 3.1.2), the `_atom_site.label` definition is compact because the attribute `_import_list.id` is used to insert attributes from the generic definition `atom_site_label` stored in the file *com_att.dic*. Using an import attribute here is convenient because atom label items are defined in many different categories and dictionaries, and they are identically-equivalent being derived from the labels residing in the ATOM_SITE category. Placing the common attributes of derived items in *generic* definition and in a separate file, then using the import commands to expand definitions at application time, is both efficient and secure (a change to the generic definition will affect all equivalent definitions).

Here is the generic definition `atom_site_label` which resides in the file *com_att.dic*

```
save_atom_site_label
     _definition.id              'atom_site_label'
     _definition.update           2006-06-29
     _description.text
;
      This label is a unique identifier for a particular site in the
      asymmetric unit of the crystal unit cell.
;
     _description.common         'Atom Site Label'
     _type.purpose                Assigned
     _type.container              Single
     _type.contents               Label
      loop_
     _description_example.case   C12     Ca3g28     Fe3+17     H*251
                                 C_a_phe_83_a_0  Zn_Zn_301_A_0
      save_
```

Definition 4.2.2

Note that the processing of import attributes usually assumes that locally specified attributes take precedence over imported attributes (more details are given in §6.17 and §6.18).

4.3 JOINED CATEGORIES

As discussed in §4.2, the specification of an access key is essential for each *List* category. In this section we describe how, in special circumstances, list categories may be *joined* with their parent List categories at instantiation. This may apply when categories have equivalent category keys and list structures, but, for reasons of data presentation and simplification, items are instantiated as separate lists. For example, the use of separate lists can be efficient for lists in which there is significant disparity between the density of active data from packet to packets. In such cases the dense and sparse components of the packet are placed in separate list categories, with one category defined as the parent of the other (see §4.1).

The ability to join equivalent but separated list categories is specified within the definition of the "child" category using the attribute `_category.parent_join` set to *Yes*. Note that the keys of joined categories may be used interchangeably in the instance document.

In the Core_CIF dictionary, an example of joined categories is `ATOM_SITE` and `ATOM_SITE_ANISO`. We have seen in §3.2 that the key item of the first is `_atom_site.label` and the definition below shows that the key item of the second is `_atom_site_aniso.label`. These are interchangeable.

```
save_ATOM_SITE_ANISO
    _definition.id               atom_site_aniso
    _definition.scope            Category
    _definition.class            List
    _definition.update           2007-02-06
    _description.text
;
     The CATEGORY of data items used to describe atomic site information
     used in crystallographic structure studies.
;
    _description.common          'Atom Site Anisotropic List'
    _category.parent_id           atom_site
    _category.parent_join         Yes
    _category_key.generic       '_atom_site_aniso.key'
    _category_key.primitive     (_atom_site_aniso.label)
     save_
```

Definition 4.3.1

Here is a simple data file showing items in these categories expressed as separate lists.

```
loop_
_atom_site.label
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
  o1   .5501(6)   .6371(6)   .1601(13)
  o2   .4012(6)   .5162(6)   .2290(12)
  o3   .2502(7)   .5705(7)   .6011(14)

loop_
_atom_site_aniso.label
_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
  o1   .035   .012   .003 .043   .001   .022
  o3   .048   .011   .021 .034   .009   .032
```

Example 4.3.1

Because the categories `ATOM_SITE` and `ATOM_SITE_ANISO` are defined as joinable, these lists may be considered to be one list by parsers. That is, as follows.

```
loop_
_atom_site.label
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
_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
  o1   .5501(6) .6371(6) .1601(13) .035   .012   .003 .043   .001   .022
  o2   .4012(6) .5162(6) .2290(12) ? ? ? ? ? ?
  o3   .2502(7) .5705(7) .6011(14) .048   .011   .021 .034   .009   .032
```

Example 4.3.2

The concept of joined lists is important to the generality of data relational languages, such as *dREL*, used in method expressions. Joinable lists are assumed to have equivalent key items, so that above, either of the items `_atom_site.label` and `_atom_site_aniso.label` may be used interchangeably as the key to the joined list. This allows items in joined lists be addressed simply in terms of the extension names *of the parent category name*. That is, reference to `ATOM_SITE.U_11`, point to the value of `_atom_site_aniso.U_11`.

4.4 LINKED LIST KEYS

The values of category keys uniquely identify data packets in lists. If any item in a List category has the same values as a key, within or without the category, this is identified as a list *link*. For example, in the instance list below the item `_atom_site.calc_attached_atom` has the same label values as the key to the same list.

```
loop_
_atom_site.label
_atom_site.type_symbol
_atom_site.fract_x
_atom_site.fract_y
_atom_site.fract_z
_atom_site.calc_attached_atom
  C1   C   .41520   .69430   .49560   .
  C2   C   .31850   .66960   .63180   H1
  C3   C   .27660   .75080   .84370   .
  H1   H   .41000   .72000   .47000   .

loop_
_atom_type.symbol
_atom_type_scat.dispersion_real
_atom_type_scat.dispersion_imag
_atom_type_scat.source
  O    .047   .032   'Int Tables Vol IV Tables 2.2B and 2.3.1'
  C    .017   .009   'Int Tables Vol IV Tables 2.2B and 2.3.1'
  H     0     0      'Stewart and Davidson'
```

Example 4.4.1

This relationship is specified in the definition of `_atom_site.calc_attached_atom` with the attribute `_name.linked_item_id` indicating that `_atom_site.label` is the derivative parent.

```
save_atom_site.calc_attached_atom
    _definition.id              '_atom_site.calc_attached_atom'
    _definition.update           2006-11-03
    _description.text
;
     The _atom_site.label of the atom site to which the 'geometry-
     calculated' atom site is attached.
;
    _description.common          'Atom Site Parent Atom'
    _name.category_id            atom_site
    _name.object_id              calc_attached_atom
    _name.linked_item_id        '_atom_site.label'
    _type.purpose                Link
    _type.container              Single
    _type.contents               Label
save_
```

Definition 4.4.1

In the example 4.4.1, the values of `_atom_site.type_symbol` are the element symbols stored in the `ATOM_TYPE` category list as the key item `_atom_type.symbol`. Again, this link is specified in the definition of `_atom_site.type_symbol` using the attribute `_name.linked_item_id.`

```
save_atom_site.type_symbol
```

```
    _definition.id              '_atom_site.type_symbol'
    _definition.update           2006-11-03
    _description.text
;
     A code to identify the atom specie(s) occupying this site.
     This code must match a corresponding _atom_type.symbol. The
     specification of this code is optional if component_0 of the
     _atom_site.label is used for this purpose. See _atom_type.symbol.
;
    _description.common         'Atom Site Type Symbol'
    _name.category_id            atom_site/
    _name.object_id              type_symbol
    _name.linked_item_id        '_atom_type.symbol'
    _type.purpose                Link
    _type.container              Single
    _type.contents               Code
     save_
```

Definition 4.4.2

4.5 LINKED KEYS OF DERIVATIVE ITEMS

The origin of data items is considered to be *primitive* or *derivative*. If a data value is measured, observed or assigned it is classified as "primitive". Primitive items cannot be derived from other data. All other data are classified as "derivative". Derivative data values may be *evaluated* (i.e. calculated) from their relationships to other data items, primitive and derivative. Identifying the origin of data is important because the value of a derivative item can determined from a method expression relating it to the known values of other data items. Instance values of derivative items may be evaluated singly or as entities in a category list. In some cases the entire list category may be considered derivative, and evaluated using methods expressions.

In this section, we explain how category keys are related when one list category is derived from another. If a*n entire* list category is derivative, its list key will be dependent, or *linked*, to the key of list from which it was derived.

This is best understood from an example. We will use the dependence of a list of geometric bond distances between the atom sites in a molecule on the list of atom site coordinates. The distances form a "derivative list" because are derived from the coordinate list. Using the definitions in the *CoreCIF* dictionary, we see that the category `GEOM_BOND` items may be derived from category `ATOM_SITE` values [*Note that for the sake of simplicity the geometry bond examples and definitions shown below have intentionally excluded the symmetry of the atomic sites*].

Here is an instance file containing these two lists.

```
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
  c1   .41520   .69430   .49560   .03000   Uiso
  c2   .31850   .66960   .63180   .03000   Uiso
  c3   .27660   .75080   .84370   .03000   Uiso
  c4   .34400   .85470   .87960   .03000   Uiso
  c5   .44470   .87990   .74290   .03000   Uiso
  c6   .47570   .79210   .53701   .03000   Uiso
  c7   .45300   .61239   .27920   .03000   Uiso

loop_
_geom_bond.id
_geom_bond.distance
  [c1,c2]    1.3373(10)
  [c1,c6]    1.3134(10)
```

```
  [c1,c7]    1.4764(12)
  [c2,c3]    1.4707(10)
  [c3,c4]    1.4094(10)
  [c4,c5]    1.3786(11)
  [c5,c6]    1.4604(10)
```

Example 4.5.1

These two lists are linked through their category keys. In the ATOM_SITE list, the defined key is _atom_site.label (see the definition in §3.1). In the GEOM_BOND list the key is _geom_bond.id which is a *tuple* made up of two ATOM_SITE keys identifying the "bonded" atoms in the site list. That is, the GEOM_BOND key is dependent on the existence of the ATOM_SITE keys.

We will now show how this is specified in a dictionary written in *DDLm*. Here is the definition of the GEOM_BOND category indicating that _geom_bond.id is the category key. The definition of _geom_bond.id follows and here the relationship of the key to the individual atom labels _geom_bond.atom_site_label_1 and _geom_bond.atom_site_label_2 is specified in the _method.expression attribute.

```
save_GEOM_BOND
    _definition.id               geom_bond
    _definition.scope            Category
    _definition.class            List
    _definition.update           2006-06-17
    _description.text
;
     The CATEGORY of data items used to specify the geometry bonds lengths
     in the structural model as derived from the atomic sites.
;
    _description.common          'Bond Lengths List'
    _category.parent_id           geom
    _category_key.generic       '_geom_bond.key'
    _category_key.primitive     (_geom_bond.atom_site_label_1,
                                 _geom_bond.atom_site_label_2)

    save_
```

Definition 4.5.1

```
save_geom_bond.key
    _definition.id              '_geom_bond.key'
    _definition.update           2006-06-17
    _description.text
;
     Key to the geometry bond distance list.
;
    _description.common          'Key to bond distance list'
    _name.category_id            geom_bond
    _name.object_id              key
    _type.purpose                Key
    _type.container              Single
    _type.contents               Inherited
     loop_
    _method.purpose
    _method.expression
     EVAL
;               _geom_bond.key = geom._bond.id
;
     save


save_geom_bond.id
    _definition.id              '_geom_bond.id'
    _definition.update           2008-06-24
    _description.text
;
     Identity of bond distance in terms of the atom site labels and
     symmetry operators as pairs for each of the two "bonded" atom sites.
;
    _description.common          'BondDistId'
```

```
    _name.category_id            geom_bond
    _name.object_id              id
    _type.purpose                Identify
    _type.container              Tuple
    _type.contents               Tuple[Label,Symop]
    _type.dimension              [2]
     loop_
    _method.purpose
    _method.expression
     Evaluation
;
     With  a  as  geom_bond

     _geom_bond.id =  Tuple (
                          Tuple ( a.atom_site_label_1, a.site_symmetry_1 ),
                          Tuple ( a.atom_site_label_2, a.site_symmetry_2 ) )
;
     save_
```

Definition 4.5.2

The following definition of `_geom_bond.atom_site_label_1` shows how the dependency of this label on `_atom_site.label` is specified with the attribute `_name.linked_item_id.`

```
save_geom_bond.atom_site_label_1
    _definition.id              '_geom_bond.atom_site_label_1'
    _definition.update            2006-11-03
    _description.text
;
    Label of the first site in the geometry bond.
;
    _description.common         'Atom Site Label'
    _name.category_id             geom_bond
    _name.object_id               atom_site_label_1
    _name.linked_item_id        '_atom_site.label'
    _type.purpose                 Key
    _type.container               Single
    _type.contents                Label
     save_
```

Definition 4.5.3

That is, each value of `_geom_bond.atom_site_label_1` (and `_geom_bond.atom_site_label_2`) must match a value of `_atom_site.label`, otherwise the file is in error. The link between these labels means that *dREL* may access the coordinate packets in the `ATOM_SITE` list *directly* using `_geom_bond.atom_site_label_1` or `_geom_bond.atom_site_label_2` values in the `GEOM_BOND` list.

6. GLOSSARY OF *DDLM* ATTRIBUTES

6.1 *DDLM* DICTIONARY

The purpose and function of the individual *DDLm* attributes are defined in their own *DDLM* dictionary (filename: `ddl_m.dic`) using the attributes themselves. All attributes are grouped into categories but only the category tags of *List* categories (as opposed to *Set* categories) are shown (in blue capital letters) with the parent category name in parentheses. Note that unless specified explicitly using `_category.parent_join` *List* categories in *DDLm* cannot be joined to their parent categories (i.e. their lists cannot be merged).

```
DDL_ATTR

          ALIAS  (parent: ddl_attr )
```

```
          alias.definition_id
          alias.dictionary_uri

category.parent_id
category.parent_join

category_key.generic
category_key.primitive

          CATEGORY_MANDATORY  (parent: category )
          category_mandatory.item_id

definition.class
definition.id
definition.scope
definition.update
definition.xref_code

description.key_words
description.common
description.text

          DESCRIPTION_EXAMPLE  (parent: description )
          description_example.case
          description_example.detail

dictionary.class
dictionary.date
dictionary.ddl_conformance
dictionary.namespace
dictionary.title
dictionary.uri
dictionary.version

          DICTIONARY_AUDIT  (parent: dictionary )
          dictionary_audit.date
          dictionary_audit.revision
          dictionary_audit.version

          DICTIONARY  (parent: dictionary )
          dictionary_valid.attributes
          dictionary_valid.scope

          DICTIONARY_XREF  (parent: dictionary )
          dictionary_xref.code
          dictionary_xref.date
          dictionary_xref.format
          dictionary_xref.name
          dictionary_xref.uri


enumeration.default
enumeration.def_index_id
enumeration.range
enumeration.mandatory

          ENUMERATION_DEFAULT  (parent: enumeration )
          enumeration_default.index
          enumeration_default.value

          ENUMERATION_SET  (parent: enumeration )
          enumeration_set.state
          enumeration_set.construct
          enumeration_set.detail
          enumeration_set.xref_code
          enumeration_set.xref_dictionary

          IMPORT  (parent: ddl_attr )
          import.block
          import.file
          import.if_dupl
```

```
          import.if_miss
          import.scope

import_list.id

loop.level

          METHOD  (parent: ddl_attr )
          method.purpose
          method.expression

name.object_id
name.category_id
name.linked_item_id

type.container
type.contents
type.purpose
type.dimension

units.code
```

6.2 FORMAT OF *DDLM* DEFINITION FRAMES

The basic organisation of *DDLm* definition frames is consistent across all definitions and domain dictionaries. Certain attributes are *mandatory* for the construction of a valid category or item definition frame. The appropriateness of attributes in the different frames is specified formally in the *DDLM* dictionary using the attributes `_dictionary_valid.scope` and `_dictionary_valid.attributes`. The *DDLM* dictionary is the only dictionary in which these two attributes may be invoked. The syntax of the value for `_dictionary_valid.attributes` is as follows, where:

+ <*attribute tag*> stipulates that this attribute is *mandatory,*

. <*attribute tag*> stipulates that this attribute is *encouraged,*

! <*attribute category*> stipulates that this category of attributes is *invalid*.

```
     loop_
    _dictionary_valid.scope
    _dictionary_valid.attributes

     Dictionary
;               + _dictionary.title
                + _dictionary.class
                + _dictionary.version
                + _dictionary.date
                + _dictionary.uri
                + _dictionary.ddl_conformance
                + _dictionary.namespace
                + _dictionary_audit.version
                + _dictionary_audit.date
                + _dictionary_audit.revision
                . _description.text
                ! ALIAS
                ! CATEGORY
                ! DEFINITION
                ! ENUMERATION
                ! LOOP
                ! METHOD
                ! NAME
                ! TYPE
                ! UNITS
;
     Category
```

```
;               + _definition.id
                + _definition.scope
                + _definition.class
                + _category.parent_id
                . _category_key.generic
                . _category_key.primitive
                . _category_mandatory.item_id
                . _description.text
                ! ALIAS
                ! DICTIONARY
                ! ENUMERATION
                ! IMPORT
                ! LOOP
                ! NAME
                ! TYPE
                ! UNITS
;
     Item
;               + _definition.id
                . _definition.scope
                . _definition.class
                + _definition.update
                + _name.object_id
                + _name.category_id
                + _type.purpose
                + _type.container
                + _type.contents
                . _description.text
                . _description.common
                ! CATEGORY
                ! DICTIONARY
;
```

The construction of dictionary definitions has already been illustrated in the Sections 3 and 4. For the category definitions, the attributes needed are relatively simple (see *Definitions 3.3.2, 4.3.1* and *4.5.1*). For the item definitions, the attributes required may vary considerably, but certain attributes are mandatory (see *Definitions 4.2.2, 4.4.1, 4.4.2, 4.5.2 and 4.5.3*).

For the sake of brevity, the description of individual attribute definitions that follow will not include every attribute applied (these may be seen in the file `ddl_m.dic`). Only the characteristics that uniquely distinguish each attribute will be shown and discussed.

6.3 ALIAS Attributes

The `ALIAS` attributes identify identically-equivalent tags that may be aliased (i.e. substituted) for the defined tag. These attributes are included when equivalent items exist in this or another dictionary.

6.3.1 `ALIAS`

```
    _definition.class           List
    _category.parent_id         ddl_attr
    _category_key.generic     '_alias.definition_id'
```

6.3.2 `_alias.definition_id`

Specifies the tag of another item equivalent to the item in the current definition.

```
    _type.purpose               Key
    _type.container             Single
    _type.contents              Tag
```

6.3.3 _alias.dictionary_uri

Specifies the universal resource identifier of the dictionary containing the definition of an item aliased to the item in the current definition.

```
_type.purpose                Identify
_type.container              Single
_type.contents               Uri
```

6.4 CATEGORY Attributes

The CATEGORY attributes specify the group properties of a related set of items.

6.4.1 CATEGORY

```
_definition.class            Set
_category.parent_id          ddl_attr
```

6.4.2 _category.parent_id

Specifies the *parent* (the next highest in the category hierarchy) category tag of the category being defined (see also 4.1 above).

```
_type.purpose                Identify
_type.container              Single
_type.contents               Tag
```

6.4.3 _category.parent_join

Specifies if at instantiation time the defined List category may be joined to the parent List category (see also 4.3 above).

```
_type.purpose                Identify
_type.container              Single
_type.contents               YesorNo
_enumeration.default         No
```

6.5 CATEGORY_KEY Attributes

The CATEGORY_KEY attribute identifies the category keys of a List category. In an instance document, in order to access specific packets of item values within a looped list, these keys must have a unique value. Two keys types are provided; _category_key.generic and _category_key.primitive. The *generic* key is a tag identifying a single key item or an item containing a method specifying the key item(s). The generic key allows the key item(s) to be varied, and selected, according to instantiated values of items outside the category. The *primitive* key is a tuple containing the one or more item tags that form the basic key. The components of the primitive key are fixed.

6.5.1 CATEGORY_KEY

```
_definition.class            Set
_category.parent_id          category
```

6.5.2 _category_key.generic

Specifies the tag of the item whose value is the key to packets of items in a looped list of items in a List category. See 4.2 for an example of a generic key.

```
    _type.purpose                 Identify
    _type.container               Single
    _type.contents                Tag
```

6.5.3 _category_key.primitive

Specifies a tuple of an item or items whose composite value is the key to packets of items in a List category. See 4.2 for an example of a primitive key.

```
    _type.purpose                 Identify
    _type.container               Tuple
    _type.contents                Tag
    _type.dimension               [*]
```

6.6 CATEGORY_MANDATORY Attributes

The CATEGORY_MANDATORY attribute identifies the items that *must* be present in an instance data list of the category items.

6.6.1 CATEGORY_MANDATORY

```
    _definition.class             List
    _category.parent_id           category
    _category_key.generic      '_category_mandatory.item_id'
```

6.6.2 _category_mandatory.item_id

Specifies the tag of the item that must be present in an instance data list of the category items.

```
    _type.purpose                 Key
    _type.container               Single
    _type.contents                Tag
```

6.7 DEFINITION Attributes

The DEFINITION attributes identify the nature and purpose of definition frames in a dictionary.

6.7.1 DEFINITION

```
    _definition.class             Set
    _category.parent_id           ddl_attr
    loop_
    _category_mandatory.item_id '_definition.id'
```

6.7.2 _definition.class

Specifies the class or purpose of the dictionary, category or item being defined. The allowed definition classes are listed below.

```
    _type.purpose                 State
    _type.container               Single
    _type.contents                Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail

     Audit
;                 Item used to IDENTIFY and AUDIT dictionary properties only.
;
     Attribute
;                 Item used as an attribute in the definition
                  of other data items. Applied in dictionaries only.
;
     Head
;                 Category of items that is the parent of
                  all other categories in the dictionary.
;
```

```
      List
;                 Category of items that in a data file must
                  reside in a looped list with a key item defined.
;
      Set
;                 Category of items that form a set (but not a
                  loopable list). These items may be referenced
                  as a class of items in a dREL methods expression.
;
      Datum
;                 Item in a domain-specific dictionary.  These items
                  appear in data files.
;
      Transient
;                 Definition saveframes specifying the attributes, enumeration
                  values and functions used in dictionary definitions. These tags
                  are ONLY used in dictionary definitions.
;
    _enumeration.default          Datum
```

6.7.3 _definition.id

Specifies the tag of the *item or category* being defined within the current definition frame.

```
    _type.purpose                 Identify
    _type.container               Single
    _type.contents                Tag
```

6.7.4 _definition.scope

Specifies the *scope* of item being defined in terms of its inheritance. The allowed definition scopes are shown.

```
    _type.purpose                 State
    _type.container               Single
    _type.contents                Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail
          Dictionary          "applies to all defined items in the dictionary"
          Category            "applies to all defined items in the category"
          Item                "applies to a single item definition"
    _enumeration.default          Item
```

6.7.5 _definition.update

Specifies the calendar date (format "yyyy-mm-dd") that the item definition was last updated.

```
    _type.purpose                 Audit
    _type.container               Single
    _type.contents                Date
```

6.7.6 _definition.xref_code

Specifies a code that identifies the same item defined in another dictionary identified by the DICTIONARY_XREF category of attributes.

```
    _type.purpose                 Identify
    _type.container               Single
    _type.contents                Code
```

6.8 DESCRIPTION Attributes

The DESCRIPTION attributes provide various text descriptions of the defined data item.

6.8.1 DESCRIPTION

```
    _definition.class             Set
```

```
    _category.parent_id          ddl_attr
```

6.8.2 _description.key_words

Specifies a sequence of comma-delimited word sequences that are "key words" identifying an item for thematic searches.

```
    _type.purpose                Describe
    _type.container              List
    _type.contents               Code
```

6.8.3 _description.common

Specifies the commonly used name of the defined item.

```
    _type.purpose                Describe
    _type.container              Single
    _type.contents               Text
```

6.8.4 _description.text

Specifies the text describing of the defined item.

```
    _type.purpose                Describe
    _type.container              Single
    _type.contents               Text
```

6.9 DESCRIPTION_EXAMPLE Attributes

The DESCRIPTION_EXAMPLE attributes provide descriptive example values of the defined item. These values are not machine interpretable.

6.9.1 DESCRIPTION_EXAMPLE

```
    _definition.class            List
    _category.parent_id          description
    _category_key.generic      '_description_example.case'
```

6.9.2 _description_example.case

Specifies an example value for the defined item.

```
    _type.purpose                Key
    _type.container              Single
    _type.contents               Text
```

6.9.3 _description_example.detail

Specifies the text details of an example value for the defined item.

```
    _type.purpose                Describe
    _type.container              Single
    _type.contents               Text
```

6.10 DICTIONARY Attributes

The DICTIONARY attributes describe aspects of the dictionary as a whole. *These attributes are specified within the dictionary block but not within a definition frame.*

6.10.1 DICTIONARY

```
    _definition.class            Set
    _category.parent_id          ddl_attr
     loop_
    _category_mandatory.item_id
                               '_dictionary.title'
```

```
                                '_dictionary.uri'
                                '_dictionary.date'
                                '_dictionary.version'
```

6.10.1 _dictionary.class

Specifies the nature or purpose of the items defined in the dictionary.

```
    _type.purpose               State
    _type.container             Single
    _type.contents              Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail
              Attribute     'dictionary containing DDL attribute definitions'
              Instance      'dictionary containing data definitions'
              Import        'dictionary containing definitions for importation'
              Function      'dictionary containing method function definitions'
    _enumeration.default        Instance
```

6.10.2 _dictionary.date

Specifies the calendar date (format "yyyy-mm-dd") that the dictionary was last updated.

```
    _type.purpose               Audit
    _type.container             Single
    _type.contents              Date
```

6.10.3 _dictionary.ddl_conformance

Specifies the version code (nn.mm.ii) for the DDL dictionary to which all definitions in the current dictionary conform.

```
    _type.purpose               Audit
    _type.container             Single
    _type.contents              Version
```

6.10.4 _dictionary.namespace

Specifies a unique name for the dictionary that may be prefixed to an item tag (defined within the specific dictionary) with a separating colon character ":" when used in dictionary applications. Because tags must be unique, dictionary namespace prefixes are unlikely to be used in data files.

```
    _type.purpose               Identify
    _type.container             Single
    _type.contents              Code
```

6.10.5 _dictionary.uri

Specifies the URI location and filename of the current dictionary.

```
    _type.purpose               Identify
    _type.container             Single
    _type.contents              Uri
```

6.10.6 _dictionary.title

Specifies the common title for the current dictionary.

```
    _type.purpose               Identify
    _type.container             Single
    _type.contents              Code
```

6.10.7 _dictionary.version

Specifies the version code (nn.mm.ii) of the dictionary. This code must match a value for _dictionary_audit.version in the dictionary audit list (see 6.11)

```
    _type.purpose               Audit
```

```
    _type.container                 Single
    _type.contents                  Version
```

6.11 DICTIONARY_AUDIT Attributes

The DICTIONARY_AUDIT attributes describe the status and the origins of a dictionary.

6.11.1 DICTIONARY_AUDIT

```
    _definition.class               List
    _category.parent_id             dictionary
    _category_key.generic         '_dictionary_audit.version'
     loop_
    _category_mandatory.item_id '_dictionary_audit.date'
```

6.11.2 _dictionary_audit.date

Specifies the calendar date (format "yyyy-mm-dd") of the last revision of the dictionary.

```
    _type.purpose                   Audit
    _type.container                 Single
    _type.contents                  Date
```

6.11.3 _dictionary_audit.revision

Specifies the description of the revision applied.

```
    _type.purpose                   Describe
    _type.container                 Single
    _type.contents                  Text
```

6.11.4 _dictionary_audit.version

Specifies the code (nn.mm.ii) identifying the version of a dictionary (see _dictionary.version) associated with a revision.

```
    _type.purpose                   Key
    _type.container                 Single
    _type.contents                  Version
```

6.12 DICTIONARY_VALID Attributes

The DICTIONARY_VALID attributes identify when attributes are used in the different definition scopes. That is, whether specific attributes are mandatory or prohibited in the dictionary, category or item definitions. *The DICTIONARY_VALID attributes are only used in the DDL dictionary.* For the current invocation see Section 6.2.

6.12.1 DICTIONARY_VALID

```
    _definition.class               List
    _category.parent_id             dictionary
    _category_key.generic         '_dictionary_valid.scope'
    loop_
    _category_mandatory.item_id '_dictionary_valid.attributes'
```

6.12.2 _dictionary_valid.attributes

Specifies as text the names of attributes that are mandatory or prohibited.

```
    _description.text
;
     A list of the attribute names and the attribute categories that are
     either MANDATORY or PROHIBITED for the _definition.scope value
     specified in the corresponding _dictionary_valid.scope. All
     unlisted attributes are considered optional.
     MANDATORY    attributes are preceded by a "+" character.
```

```
     PROHIBITED   attributes are preceded by a "!" character.
     RECOMMENDED  attributes are preceded by a "." character.
;
    _type.purpose                 Audit
    _type.container               Single
    _type.contents                Text
```

6.12.3 _dictionary_valid.scope

Specifies the dictionary scope associated with the _dictionary_valid.attributes lists.

```
    _type.purpose                 State
    _type.container               Single
    _type.contents                Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail
          Dictionary         "applies to all defined items in the dictionary"
          Category           "applies to all defined items in the category"
          Item               "applies to a single definition"
```

6.13 DICTIONARY_XREF Attributes

The DICTIONARY_XREF attributes identify external dictionaries to which items in the current dictionary are cross-referenced using the _definition.xref_code attribute.

6.13.1 DICTIONARY_XREF

```
    _definition.class             List
    _category.parent_id           dictionary
    _category_key.generic       '_dictionary_xref.code'
```

6.13.2 _dictionary_xref.code

Specifies the key code of the cross-referenced dictionary.

```
    _type.purpose                 Key
    _type.container               Single
    _type.contents                Code
```

6.13.3 _dictionary_xref.date

Specifies the calendar date (format "yyyy-mm-dd") of the cross-referenced dictionary.

```
    _type.purpose                 Audit
    _type.container               Single
    _type.contents                Date
```

6.13.4 _dictionary_xref.format

Specifies the format description of the cross-referenced dictionary.

```
    _type.purpose                 Describe
    _type.container               Single
    _type.contents                Text
```

6.13.5 _dictionary_xref.name

Specifies the common name of the cross-referenced dictionary.

```
    _type.purpose                 Describe
    _type.container               Single
    _type.contents                Text
```

6.13.6 _dictionary_xref.uri

Specifies the URI of the cross-referenced dictionary.

```
_type.purpose                 Audit
_type.container               Single
_type.contents                Uri
```

6.14 ENUMERATION ATTRIBUTES

The ENUMERATION attributes specify any prescribed constraints on the values of defined items.

6.14.1 ENUMERATION

```
_definition.class             Set
_category.parent_id           ddl_attr
```

6.14.2 _enumeration.default

Specifies the default value the value of the defined item, which is used if a value is not present in the instance data file.

```
_type.purpose                 Limit
_type.container               Single
_type.contents                Implied
```

6.14.3 _enumeration.range

Specifies the range of values the defined item must lie within. The minimum and maximum values are separated by a colon ":" character.

```
_type.purpose                 Limit
_type.container               Single
_type.contents                Range
```

6.14.4 _enumeration.def_index_id

Specifies the tag of an item whose coded value is used as an index to select a default enumeration value from the _enumeration_default list (see 6.15). The code value must match one of the _enumeration_default.index values.

```
_type.purpose                 Identify
_type.container               Single
_type.contents                Tag
```

6.14.5 _enumeration.mandatory

Specifies if it obligatory that the enumeration constraints (set by other attributes) MUST be adhered to in any validation process. The default is *Yes*.

```
_type.purpose                 Limit
_type.container               Single
_type.contents                YesorNo
_enumeration.default          Yes
```

6.15 ENUMERATION_DEFAULT Attributes

The ENUMERATION_DEFAULT attributes specify the allowed default values for the defined item. The single default value applicable for a specific instance document is determined by the value of tag identified by the attribute _enumeration.def_index_id (see 6.14.3). The code value is used as an index to select a default enumeration value from the _enumeration_default list by matching one of the _enumeration_default.index values.

6.15.1 ENUMERATION_DEFAULT

```
_definition.class             List
_category.parent_id           enumeration
_category_key.generic       '_enumeration_default.index'
```

6.15.2 _enumeration_default.index

Specifies the key index codes to the list of eligible default values. This code is matched at instantiation time with the value of the item identified by the attribute _enumeration.def_index_id.

```
_type.purpose                Key
_type.container              Single
_type.contents               Code
```

6.15.3 _enumeration_default.value

Specifies eligible default values. The appropriate default is selected at instantiation time by matching the _enumeration_default.index code with that of the item identified by the attribute _enumeration.def_index_id.

```
_type.purpose                Limit
_type.container              Single
_type.contents               Implied
```

6.16 ENUMERATION_SET Attributes

The ENUMERATION_SET attributes specify a set of predetermined values (i.e. states) for an item.

6.16.1 ENUMERATION_SET

```
_definition.class            List
_category.parent_id          enumeration
_category_key.generic      '_enumeration_set.state'
```

6.16.2 _enumeration_set.state

Specifies permitted codes or "states" for a item.

```
_type.purpose                Key
_type.container              Single
_type.contents               Code
```

6.16.3 _enumeration_set.construct

Specifies the construction rules of permitted states in terms regular expression (REGEX) rules.

```
_type.purpose                Limit
_type.container              Single
_type.contents               Regex
```

6.16.4 _enumeration_set.detail

Specifies the description of a permitted enumeration state.

```
_type.purpose                Describe
_type.container              Single
_type.contents               Text
```

6.16.5 _enumeration_set.xref_code

Specifies a cross-reference code for a permitted state with respect to the codes used in the dictionary identified with the DICTIONARY_XREF category attributes.

```
_type.purpose                Identify
_type.container              Single
_type.contents               Code
```

6.16.6 _enumeration_set.xref_dictionary

Specifies the code for the dictionary identified with the DICTIONARY_XREF category attributes.

```
_type.purpose                Link
_type.container              Single
```

```
    _type.contents                Code
```

6.17 IMPORT Attributes

The IMPORT attributes faciltate the importation of definition lines from external files. These attributes do not contribute to the direct definition of an item but provide a mechanism for inserting external definition material into a dictionary.

6.17.1 IMPORT

```
    _definition.class             List
    _category.parent_id           ddl_attr
    _category_key.generic       '_import.block'
```

6.17.2 _import.block

Specifies the block of definitions to be imported from _import.file.

```
    _type.purpose                 Key
    _type.container               Single
    _type.contents                Tag
     loop_
    _description_example.case   '_atom_site.xyz'
                                 'refln'
```

6.17.3 _import.file

Specifies the URI containing the definition block specified by _import.block.

```
    _type.purpose                 Identify
    _type.container               Single
    _type.contents                Uri
```

6.17.4 _import.if_dupl

Specifies the action to be taken if the imported definition block already exists in the importing dictionary file. The actions allowed appear as enumerated states.

```
    _type.purpose                 State
    _type.container               Single
    _type.contents                Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail
                       Ignore   'ignore imported definitions if id conflict'
                       Replace  'replace existing with imported definitions'
                       Exit     'issue error exception and exit'
    _enumeration.default          Exit
```

6.17.5 _import.if_miss

Specifies the action to be taken if the imported definition block is missing from the file identified by _import.file. The actions allowed appear as enumerated states.

```
    _type.purpose                 State
    _type.container               Single
    _type.contents                Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail
                                  Ignore      'ignore import'
                                  Exit        'issue error exception and exit'
    _enumeration.default          Exit
     save_
```

6.17.6 _import.scope

Specifies the scope of imported definition block identified by `_import.block`. The scopes allowed appear as enumerated states.

```
    _type.purpose                 State
    _type.container               Single
    _type.contents                Code
     loop_
    _enumeration_set.state
    _enumeration_set.detail
                          Dic   'all saveframes in the source file'
                          Cat   'all saveframes in the specific category'
                          Grp   'all saveframes in the category with children'
                          Def   'one saveframe containing a definition'
                          Att   'import attributes within a saveframe'
                          Sta   'import enumeration state list only'
                          Val   'import enumeration default value list only'
```

6.18 IMPORT_LIST Attributes

The `IMPORT_LIST` attributes specify the importation of definition material from external files.

6.18.1 `IMPORT_LIST`

```
    _category.parent_id           import
```

6.18.2 `_import_list.id`

Specifies all of the attributes described in 6.17 for the importation of definition material from external files. This attribute is a convenient composite of the IMPORT attributes in that they are presentated as a single List () rather than as a looped list.

```
    _type.container               Tuple
    _type.contents                Tuple(Code,Tag,Uri,Code,Code)
    _type_array.dimension         [*]
     loop_
    _method.purpose
    _method.expression
     Definition
;
     With  i  as  import
     _import_list.id  = Tuple((i.scope, i.block, i.file, i.if_dupl, i.if_miss))
;
```

6.19 LOOP Attributes

The `LOOP` category attributes specify the loop level of the defined item. For CIF data this will always be 1, but for *STAR File* data nested lists to any level are permitted.

6.19.1 `LOOP`

```
    _definition.class             Set
    _category.parent_id           ddl_attr
```

6.19.2 `_loop.level`

Specifies the loop level of the defined item.

```
    _type.purpose                 Limit
    _type.container               Single
    _type.contents                Index
    _enumeration.range            1:
    _enumeration.default          1
```

6.20 METHOD Attributes

The METHOD category attributes specify methods for expressing relationships between the defined item and other defined items.

6.20.1 METHOD

```
_definition.class            List
_category.parent_id          ddl_attr
_category_key.generic      '_method.purpose'
```

6.20.2 _method.purpose

Specifies the purpose code of the method for the defined item. Three method classes exist: *Evaluation*, *Definition* and *Validation*.

```
_type.purpose                State
_type.container              Single
_type.contents               Code
 loop_
_enumeration_set.state
_enumeration_set.detail
 Evaluation    "method evaluates an item from related item values"
 Definition    "method generates attribute value(s) in the definition"
 Validation    "method compares an evaluation with existing item value"
_enumeration.default         Evaluation
```

6.20.3 _method.expression

Specifies the script, in the dREL language, relating the defined item to other items.

```
_type.purpose                Method
_type.container              Single
_type.contents               Text
```

6.21 NAME Attributes

The NAME attributes specify the name constructs of the defined item.

6.21.1 NAME

```
_category.parent_id          ddl_attr
 loop_
_category_mandatory.item_id
                           '_name.object_id    '
                           '_name.category_id'
```

6.21.2 _name.object_id

Specifies the "object tag" of the defined item. This is a unique name string identifying an object with its category.

```
_type.purpose                Identify
_type.container              Single
_type.contents               Otag
```

6.21.3 _name.category_id

Specifies the name of the category the defined item is a member of.

```
_type.purpose                Identify
_type.container              Single
_type.contents               Ctag
```

6.21.4 _name.linked_item_id

Specifies the tag that the defined item is a derivative of, and implicitly dependent on the existence of, the linked item when used in an instance document. See 4.4 and 4.5 for examples.

```
_type.purpose                 Identify
_type.container               Single
_type.contents                Tag
```

6.22 TYPE Attributes

The TYPE attributes specify the nature and origin of the defined item.

6.22.1 TYPE

```
_category.parent_id           ddl_attr
```

6.22.2 _type.container

Specifies the container type of the defined item. This is the simplest type description of the text string representing a value. Seven container types are recognised.

```
_type.purpose                 State
_type.container               Single
_type.contents                Code
 loop_
_enumeration_set.state
_enumeration_set.detail
    Single    'a single value'
    Multiple  'values related by boolean ',|&!*' or range ":" ops'
    List      'list of values bounded by []; separated by commas'
    Array     'List of fixed length and dimension'
    Tuple     'immutable List bounded by (); nested tuples allowed'
    Table     'key:value elements bounded by {}; separated by commas'
    Implied   'implied by type.container of associated value'
_enumeration.default          Single
```

6.22.3 _type.contents

Specifies the code identifying nature of the defined item. The allowed codes are specified in an enumeration list stored in the external file com_val.dic (see below).

```
_type.purpose                 State
_type.container               Multiple
_type.contents                Code
_import_list.id              (('Sta','type_contents','com_val.dic'))
 loop_
_description_example.case
_description_example.detail
 'Integer'    'all elements are integer'
 'Real,Code'  'elements are in muliples of real number and codes'
 'Real|Code'  'elements are either a real number or a code'
```

The allowed codes for _type.contents are stored in the external file com_val.dic.

```
loop_
_enumeration_set.state
_enumeration_set.detail

Implied       'typing unknown; determined by referenced item'
 Achar        'an alphabetic character'
 ANchar       'an alphanumeric character'
 Pchar        'a printable character'
 Text         'a case-sensitive string/lines of text'
 Tag          'case-insensitive data item name or tag'
 Ctag         'case-insensitive category name preceding period in item tag'
 Otag         'case-insensitive object name trailing period in item tag'
 Filename     'name of an external file'
 Savename     'case insensitive name of a saveframe used as a REFERENCE tag'
 Code         'code used for indexing data or referencing data resources'
 Regex        'a REGEX conformant expression'
 Date         'ISO date format yyyy-mm-dd'
 YesorNo      'the flag with values of "yes", "y", "no" or "n".'
 Uri          'an universal resource indicator string specifying a file'
```

```
      Version       'version digit string of the form <major>.<version>.<update>'
      Dimension     'integer dimensions of an array in square brackets'
      Range         'An inclusive range of numerical values min:max'
      Digit         'a single digit unsigned number'
      Count         'an unsigned integer number'
      Index         'an unsigned non-zero integer number'
      Integer       'a positive or negative integer number'
      Float         'a floating-point real number'
      Real          'a floating-point real number'
      Imag          'a floating-point imaginary number'
      Complex       'a complex number'
      Binary        'a binary number'
      Hexadecimal   'a hexadecimal number'
      Octal         'a octal number'
      Label         'code itentifying an atom site'
      Element       'code itentifying an atom type (element symbol)'
      Formula       'code describing a chemical formula'
      Symop         'code itentifying the symmetry and lattice of an atom site'
```

6.22.4 `_type.purpose`

Specifies the purpose, origin or function code of the defined item.

```
     _type.purpose                State
     _type.container              Single
     _type.contents               Code
      loop_
     _enumeration_set.state
     _enumeration_set.detail
     _enumeration_set.construct

      Import
;                  >>> For dictionaries only <<<
                   Used within dictionaries to import definition lines
                   from other dictionaries.  In the expanded dictionary
                   the import item is replaced by the imported items.
;
      Method
;                  >>> For dictionaries only <<<
                   A text method expression in a dictionary definition relating
                   the defined item to other defined items.
;
      Audit
;                  An item used to contain audit information about the creation
                   or conformance of a file.
;
      Identify
                   An item used to identify another item or file.
;
      Describe
;                  A descriptive item intended only for human interpretation.
;
      Limit
;                  An item used to limit the values of other items.
;
      State
;                  An item with one or more codified values that must exist
                   within a discrete and countable list of enumerated states.
;
      Key
;                  An item with a codified value that is the key to identifying
                   specific packets of items in the same category.
;
      Link
;                  An item with a value that is a foreign key linking packets
                   in this category list to packets in another category.
;
      Assigned
                   An item whose value is assigned in the process of refining
```

```
                    and modeling measured and observed items.
;
     Observed
;                   An item whose value is determined by observation or deduction.
                    Numerical observed values do NOT have a standard uncertainty.
;
     Measured
;                   A numerical item whose value is measured or derived from a
                    measurement. It is expected to have a standard uncertainty
                    value which is either
                        1) appended as integers in parentheses at the
                           precision of the trailing digits, or
                        2) as a separate item with the same name as
                           defined item but with a trailing '_su'.
;
```

6.22.5 `_type.dimension`

Specifies the array dimensions (number of elements in each dimension) of the defined item.

```
    _type.purpose                Limit
    _type.container              Single
    _type.contents               Text
     loop_
    _description_example.case
    _description_example.detail  "[3,3]"   'in Array definition: 3x3 matrix'
                                 "[3]"     'in Array definition: 3 number vector'
                                 "[6]"     'in List definition: 6 values'
                                 "[*]"     'unknown number of elements'
```

6.23 UNITS Attributes

The `UNITS` attributes specify the units of measurement for a defined item.

6.23.1 `UNITS`

```
    _category.parent_id          ddl_attr
```

6.23.2 `_units.code`

Specifies the name of the units of measurement of the defined . The allowed codes are specified as an enumeration list in the external file `com_val.dic`.

```
    _type.purpose                State
    _type.container              Single
    _type.contents               Code
    _import_list.id             (('Sta','units_code','com_val.dic'))
```

The allowed `_units.code` values are specified in the external file `com_val.dic`.

```
     loop_
         _enumeration_set.state
         _enumeration_set.detail

   'centimetres'  "length    'centimetres (meters * 10^( -2))'"
   'millimetres'  "length    'millimetres (meters * 10^( -3))'"
   'nanometres'   "length    'nanometres  (meters * 10^( -9))'"
   'angstroms'    "length    'angstroms   (meters * 10^(-10))'"
   'picometres'   "length    'picometres  (meters * 10^(-12))'"
   'femtometres'  "length    'femtometres (meters * 10^(-15))'"

   'reciprocal_centimetres'
   "per_length 'reciprocal centimetres (meters * 10^( -2)^-1)'"
   'reciprocal_millimetres'
   "per_length 'reciprocal millimetres (meters * 10^( -3)^-1)'"
   'reciprocal_nanometres'
   "per-length 'reciprocal nanometres  (meters * 10^( -9)^-1)'"
   'reciprocal_angstroms'
   "per-length 'reciprocal angstroms   (meters * 10^(-10)^-1)'"
```

```
    'reciprocal_picometres'
    "per-length 'reciprocal picometres  (meters * 10^(-12)^-1)'"

    'nanometre_squared'
    "length_squared 'nanometres squared (meters * 10^( -9))^2'"
    'angstrom_squared'
    "length_squared 'angstroms squared  (meters * 10^(-10))^2'"
    '8pi_angstroms_squared'
    "length_squared '8pi^2 * angstroms squared (meters * 10^(-10))^2'"
    'picometre_squared'
    "length_squared 'picometres squared (meters * 10^(-12))^2'"

    'nanometre_cubed'
    "length_cubed 'nanometres cubed (meters * 10^( -9))^3'"
    'angstrom_cubed'
    "length_cubed 'angstroms cubed  (meters * 10^(-10))^3'"
    'picometre_cubed'
    "length_cubed 'picometres cubed (meters * 10^(-12))^3'"

    'kilopascals'      "pressure      'kilopascals'"
    'gigapascals'      "pressure      'gigapascals'"

    'hours'            "time      'hours'"
    'minutes'          "time      'minutes'"
    'seconds'          "time      'seconds'"
    'microseconds'     "time      'microseconds'"

    'degrees'          "angle      'degrees (of arc)'"

    'degree_per_minute' "rotation_per_time  'degrees (of arc) per minute'"

    'celsius'      "temperature     'degrees (of temperature) Celsius'"
    'kelvins'      "temperature     'degrees (of temperature) Kelvin'"

    'electrons'    "electrons              'electrons'"

    'electron_squared'  "electrons-squared    'electrons squared'"

    'electron_per_nanometres_cubed'
    "electron-density 'electrons per nanometres cubed (meters * 10^( -9))^3'"
    'electron_per_angstroms_cubed'
    "electron-density 'electrons per angstroms  cubed (meters * 10^(-10))^3'"
    'electron_per_picometres_cubed'
    "electron-density 'electrons per picometres cubed (meters * 10^(-12))^3'"

    'kilowatts'     "power       'kilowatts'"
    'milliamperes'  "current     'milliamperes'"
    'kilovolts'     "emf         'kilovolts'"

    'arbitrary'    "arbitary    'arbitrary system of units'"
```

7. REFERENCES

Allen, F.H., Barnard, J.M., Cook, A.F.P. & Hall, S.R. (1995). *J. Chem. Inform. Comp. Sci*. **35**, 412-427.

Cook, A.F.P. (1991) "Dictionary Definition Language in STAR File Format." *ORAC Report*.

Hall, S.R. *The STAR File:* (1991) "A New Format for Electronic Data Transfer and Archiving." *J Chemical Information and Computer Science* **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*. **A47**, 655-685.

Hall, S.R. & Cook, A.P.F. (1995) "STAR Data Definition Language: Initial Specification." *J Chemical Information and Computer Science* **35**, 819-825.

Hall, S.R. & Spadaccini, N. (1994) "The STAR File: Detailed Specifications." *J Chemical Information and Computer Science* **34**, *505-508.*

Spadaccini, N., Hall, S.R., and Castleden, I.R. (2000) "Relational Expressions in STAR File Dictionaries." *J Chemical Information and Computer Science* **40**, 1289-1301

Westbrook, J.D & Hall, S.R. (1995) "A Dictionary Description Language for Macromolecular Structure." Draft DDL V 2.1.1 (draft from *ndbserver.rutgers.edu/mmcif/ddl/*)