International Tables for Crystallography (2006). Vol. G, Section 3.1.7.1, pp. 83–84.

Example 3.1.6.8. *Definition of a child identifier.*

```
save__struct_site_gen.id
  _item_description.description
;     The value of _struct_site_gen.id must uniquely
      identify a record in the STRUCT_SITE_GEN list.

      Note that this item need not be a number;
      it can be any unique identifier.
;
  _item.name                   '_struct_site_gen.id'
  _item.category_id              struct_site_gen
  _item.mandatory_code           yes
  _item_type.code                line

save_
```

Example 3.1.6.9. *DDL2 definition of a physical quantity.*

```
save__diffrn.ambient_temp
    _item_description.description
;       The mean temperature in kelvins at which the
        intensities were measured.
;
    _item.name                  '_diffrn.ambient_temp'
    _item.category_id             diffrn
    _item.mandatory_code          no
    _item_aliases.alias_name
                         '_diffrn_ambient_temperature'
    _item_aliases.dictionary      cif_core.dic
    _item_aliases.version         2.0.1
     loop_
    _item_range.maximum
    _item_range.minimum            .    0.0
                                  0.0   0.0
    _item_related.related_name
                            '_diffrn.ambient_temp_esd'
    _item_related.function_code   associated_esd
    _item_type.code               float
    _item_type_conditions.code    esd
     _item_units.code             kelvins
save_
```

3.1.6.5.2. *Definitions of single quantities*

While it is important to ensure the referential integrity of the data in a CIF through proper book-keeping of links between tables, the crystallographer who wishes to create or extend a CIF dictionary will be more interested in the definitions of data items that refer to real physical quantities, the properties of a crystal or the details of the experiment. The DDL2 formalism makes it easy to create a detailed machine-readable listing of the attributes of such data.

Example 3.1.6.9 parallels the example chosen for DDL1 dictionaries of the ambient temperature during the experiment.

In the definition save frame, the category is specifically listed (although it is deducible from the DDL2 convention of separating the category name from the rest of the name by a full stop in the data name). The data type is specified as a floating-point number. (In the core dictionary there are fewer data types and the fact that the value may be a real rather than integer number must be inferred from the declared range.) The range of values is also specified with separate maximum and minimum values (unlike in DDL1 dictionaries, which give a single character string that must be parsed into its component minimum and maximum values). The assignment of the same value to a maximum and a minimum means that the absolute value is permitted; without the repeated '0.0' line the range in this example would be constrained to be positive definite; the equal value of 0.0 for maximum and minimum means that it may be identically zero.

The `_item_units.code` value must be one of the entries in the units table for the dictionary and can thus be converted into other units as specified in the units conversion table.

The aliases entries identify the corresponding quantity defined in the DDL1 core dictionary.

3.1.6.6. Units

As with data files described by DDL1 dictionaries, the physical unit associated with a quantitative value in a DDL2-based file is specified in the relevant dictionary. There is no option to express the quantity in other units. However, DDL2 permits a dictionary file to store not only a table of the units referred to in the dictionary (listed under `_item_units_list.code` and the accompanying descriptive item `_item_units_list.detail`), but also a table specifying the conversion factors between individual codes in the `_item_units_list.code` list. In principle, this allows a program to combine or otherwise manipulate different physical quantities while handling the units properly.

3.1.7. Composing new data definitions

Preceding sections have described the framework within which CIF dictionaries exist and are used, and their individual formal structures. While this is important for presenting the definition of new data items, it does not address what is often the most difficult question: what quantities, concepts or relationships merit separate data items? On the one hand, the extensibility of CIF provides great freedom of choice: anything that can be characterized as a separate idea may be assigned a new data name and set of attributes. On the other hand, there are practical constraints on designing software to write and read a format that is boundless in principle, and some care must be taken to organize new definitions economically and in an ordered way.

3.1.7.1. Granularity

Perhaps the most obvious decision that needs to be made is the level of detail or granularity chosen to describe the topic of interest. CIF data items may be very specific (the deadtime in microseconds of the detector used to measure diffraction intensities in an experiment) or very general (the text of a scientific paper). In general, a data name should correspond to a single well defined quantity or concept within the area of interest of a particular application. It can be seen that the level of granularity is determined by the requirements of the end application.

A practical example of determining an appropriate level of granularity is given by the core dictionary definitions for bibliographic references cited in a CIF. The dictionary originally contained a single character field, `_publ_section_references`, which was intended to contain the complete reference list for an article as undifferentiated text. *Notes for Authors* in journals accepting articles in CIF format advised authors to separate the references within the field with blank lines, but otherwise no structure was imposed upon the field. In a subsequent revision to the core dictionary, the much richer CITATION category was introduced to allow the structured presentation of references to journal articles and chapters of books. This was intended to aid queries to bibliographic databases. However, a full structured markup of references with multiple authors or editors in CIF requires additional categories, so that the details of the reference may be spread across three tables corresponding to the CITATION, CITATION_AUTHOR and CITATION_EDITOR categories. Populating several disjoint tables greatly complicates the author's task of writing a reference list. Moreover, the CITATION category does not yet cover all the many different types of bibliographic reference that it is possible to specify, and is therefore suitable only for references to journal articles and chapters of books. However, it is pos-

Copyright © 2006 International Union of Crystallography

sible to write a program that can deduce the structure of a standard reference within an undifferentiated reference list (provided the journal guidelines have been followed by the author) to the extent that enough information can be extracted to add hyperlinks to references using a cross-publisher reference linking service such as CrossRef (CrossRef, 2004). Therefore, in practice, IUCr journals still ask the author of an article to supply their reference list in the `_publ_section_references` field, rather than using the apparently more useful `_citation_` fields. It remains to be seen whether this is the best strategy in the long term.

In more technical topic areas, the details of an experimental instrument could be described by a huge number of possible data names, ranging from the manufacturer's serial number to the colour of the instrument casing. However, many of these details are irrelevant to the analysis of the data generated by the instrument, so the characteristics of an instrument that are assigned individual data names are typically just those parameters that need to be entered in equations describing the calibration or interpretation of the data it generates.

3.1.7.2. Category 'special details' fields

When the specific items in a particular topic area that need to be recorded under their own data names have been decided, there is likely to be other information that could be recorded, but is felt to be irrelevant to the immediate purposes of the data collection and analysis. It is good practice to provide a place in the CIF for such additional information; it encourages an author to record the infomation and permits data mining at a later stage. Each category typically contains a data name with the suffix `_details` (or `_special_details`) which identifies a text field in which additional information relating to the category may be stored. This field often contains explanatory text qualifying the information recorded elsewhere in the same category, but it might contain additional specific items of information for which no data name is given and for which no obvious application is envisaged. This helps to guard against the loss of information that might be put to good use in the future. Of course, if a `*_details` field is regularly used to store some specific item of information *and* this information is seen to be valuable in the analysis or interpretation of data elsewhere in the file, there is a case for defining a new, separate tag for this information.

3.1.7.3. Construction of data names

Since a dictionary definition contains all the machine-readable attributes necessary for validating the contents of a data field, the data name itself may be an arbitrary tag, devoid of semantic content. However, while dictionary-driven access to a CIF is useful in many cases, there are circumstances where it is useful to browse the file. It is therefore helpful to construct a data name in a way that gives a good indication of the quantity described. From the beginning, CIF data names have been constructed from self-descriptive components in an order that reflects the hierarchical relationship of the component ideas, from highest (most general) level to lowest (most specific) level when read from left to right.

In a typical example from the core CIF dictionary, the data name `_atom_site_type_symbol` defines a code (`symbol`) indicating the chemical nature (`type`) of the occupant of a location in the crystal lattice (`atom_site`). The equivalent data name from the mmCIF dictionary, `_atom_site.type_symbol`, explicitly separates the category to which the data name belongs from its more specific qualifiers by using a full stop (`.`) instead of an underscore (`_`). While this use of a full stop is mandated in DDL2 dictionaries, it should nevertheless be considered a convenience, since the category membership is explicitly listed in the dictionary definition frame for every data name.

```
_database_code_CSD                    'VOBYUG'
```

(*a*)

```
_database_2.database_id               'PDB'
_database_2.database_code             '5HVP'
```

(*b*)

Fig. 3.1.7.1. Alternative quantities described (*a*) by data-name extension (core dictionary) or (*b*) by paired data names (mmCIF dictionary).

However, it may not always be easy to establish the best order of components when constructing a new data name. In the JOURNAL category, there was initially some uncertainty about whether to associate the telephone numbers of different contact persons by appending codes such as `_coeditor` and `_techeditor` to a common base name. In the end, the order of components was reversed to give names like `_journal_coeditor_phone` and `_journal_techeditor_phone`. Examining the JOURNAL category in the core CIF dictionary will show why this was done. Similarly, the extension of geometry categories to include details of hydrogen bonding went through a stage of discussing adding new data names to the existing categories, but with suffixes indicating that the components were participating in hydrogen bonding, before it was decided that a completely new category for describing all elements of a hydrogen bond was justified. These examples show that the correct ordering of components within a data name is closely related to the perceived classification of data names by category and subcategory.

Sometimes it is useful to differentiate alternative data items by appending a suffix to a root data name. For example, the core dictionary defines several data names for recording the reference codes associated with a data block by different databases: `_database_code_CAS`, `_database_code_CSD` *etc.* This is convenient where there are two or three alternatives, but becomes unwieldy when the number of possibilities increases, because new data names need to be defined for each new alternative case. A better solution is to have a single base name and a companion data item that defines which of the available alternatives the base item refers to. The mmCIF dictionary follows this principle: the category DATABASE_2 contains two data names, `_database_2.database_code` (the value of which is an assigned database code) and `_database_2.database_id` (the value of which identifies which of the possible databases assigned the code) (Fig. 3.1.7.1).

Note the distinction between a data name constructed with a suffix indicating a particular database, and a data name which incorporates a prefix registered for the private use of a database. The data name `_database_code_PDB` is a *public* data name specifying an entry in the Protein Data Bank, while `_pdb_database_code` is a *private* data name used for some internal purpose by the Protein Data Bank (see Section 3.1.8.2).

3.1.7.4. Parsable data values *versus* separate data names

An advantage of defining multiple data names for the individual components of a complicated quantity is that there is no ambiguity in resolving the separate components. Hence the Miller indices of a reflection in the list of diffraction measurements are specified in the core dictionary by the group of three data names `_diffrn_refln_index_h`, `_diffrn_refln_index_k` and `_diffrn_refln_index_l`. In principle, a single data name

references