2.5. Specification of the core CIF dictionary definition language (DDL1)

By S. R. Hall and A. P. F. Cook

2.5.1. Introduction

The CIF approach to data representation described in Chapter 2.2 is based on the STAR File universal data language (Hall, 1991; Hall & Spadaccini, 1994) detailed in Chapter 2.1. An important advantage of the CIF approach is the self-identification of data items through the use of tag–value tuples. This syntax removes the need for preordained data ordering in a file or stream of data and enables appropriate parsing tools to automate access independently of the data source. In this chapter, we will show that the CIF syntax also provides a higher level of abstraction for managing data storage and exchange – that of defining the meaning of data items (*i.e.* their properties and characteristics) as attribute descriptors linked to the identifying data tag.

Each attribute descriptor specifies a particular characteristic of a data item, and a collection of these attributes can be used to provide a unique definition of the data item. Moreover, placing the definitions of a selected set of data items into a CIF-like file provides an electronic dictionary for a particular subject area. In the modern parlance of knowledge management and the semantic web, such a data dictionary represents a *domain ontology*.

In most respects, data dictionaries serve a role similar to spoken-word dictionaries and as such are an important adjunct to the CIF data-management approach by providing semantic information that is necessary for automatic validation and compliance procedures. That is, prior lexical knowledge of the nature of individual items ensures that each item in a CIF can be read and interpreted correctly *via* the unique tag that is the link to descriptions in a data dictionary file. Because the descriptions in the dictionary are machine-parsable, the semantic information they contain forms an integral part of a data-handling process. In other words, machine-interpretable semantic knowledge embedded in data dictionaries leads directly to the automatic validation and manipulation of the relevant items stored in any CIF.

2.5.1.1. The concept of a dictionary definition language (DDL)

The structure or arrangement of data in a CIF is well understood and predictable because the CIF syntax may be specified succinctly (see Chapter 2.2 for CIF syntax expressed using extended Backus–Naur form). In contrast, the meaning of individual data values in a file is only known if the nature of these items is understood. For CIF data this critical link between the value and the meaning of an item is achieved using an electronic 'data dictionary' in which the definitions of relevant data items are catalogued according to their data name and expressed as attribute values, one set of attributes per item.

The dictionary definitions describe the characteristics of each item, such as data type, enumeration states and relationships between data. The more precise the definitions, the higher the level of semantic knowledge of defined items and the better the efficiency achievable in their exchange. To a large degree, the precision achievable hinges on the attributes selected for use in dictionary definitions, these being the semantic vocabulary of a dictionary. Within this context, attribute descriptors constitute a dictionary definition language (DDL). Definitions of the attributes described in this chapter are provided in Chapter 4.9.

The main purpose of this chapter is to describe the DDL attributes used to construct the core, powder, modulated structures and electron density CIF dictionaries detailed in Chapters 3.2–3.5 and 4.1–4.4. We shall see that each item definition in these dictionaries is constructed separately by appropriate choice from the attribute descriptors available, and that a sequence of definition blocks (one block per item) constitutes a CIF dictionary file. The organization of attributes and definition blocks in a dictionary file need not be related to the syntax of a data CIF, but in practice there are significant advantages if they are. Firstly, a common syntax for data and dictionary files enables the same software to be used to parse both. Secondly, and of equal importance, data descriptions and dictionaries need continual updating and additions, and the CIF syntax provides a high level of extensibility and flexibility, whereas most other formats do not. Finally, the use of a consistent syntax permits the dictionary attribute descriptors themselves to be described in their own DDL dictionary file.

While the basic functionality and flexibility of a dictionary is governed by the CIF syntax, the precision of the data definitions contained within it is determined entirely by the scope and number of the attribute descriptors representing the DDL. Indeed, the ability of a DDL to permit the simple and seamless evolution of data definitions and the scope of the DDL to precisely define data items both play absolutely pivotal roles *via* the supporting dictionaries in determining the power of the CIF data-exchange process.

2.5.2. The organization of a CIF dictionary

The precision and efficiency of a data definition language are directly related to the scope of the attribute vocabulary. In other words, the lexical richness of the DDL depends on the number and the specificity of the available language attributes. The breadth of these attributes, in terms of the number of separate data characteristics that can be specified, largely controls the precision of data definitions. However, it is the functionality of attributes that determines the information richness and enables higher-level definition complexity. For example, the attributes that define child–parent relationships between data and key pointers to items in list packets are essential to understanding the data hierarchy and to its validation. Attributes provide the semantic tools of a dictionary.

The choice and scope of attributes in the DDL are governed by both semantic and technical considerations. Attributes need to have a clear purpose to facilitate easy definition and comprehension, and their routine application in automatic validation processes. A CIF dictionary is much more than a list of unrelated data definitions. Each definition conforms to the CIF syntax, which requires each data block in the dictionary to be unique. However, the functionality of a dictionary involves elements of both relational and object-oriented processes. For example, attributes in one definition may refer to another definition *via* `_list_link_parent` or `_list_link_child` attributes, so as to indicate the dependency

Affiliations: Sydney R. Hall, School of Biomedical and Chemical Sciences, University of Western Australia, Crawley, Perth, WA 6009, Australia; Anthony P. F. Cook, BCI Ltd, 46 Uppergate Road, Stannington, Sheffield S6 6BX, England.

of data items in lists. In this way the DDL, and consequently the dictionaries constructed from the DDL, invoke aspects of relational and object-oriented database paradigms. It is therefore useful to summarize these briefly here.

A relational database model (Kim, 1990) presents data as tables with a hierarchy of links that define the dependencies among tables. These explicit relationships enable certain properties of data to be shared, and, for related data values, to be derived. The structure of data links in a relational database is usually defined separately from the component data. This is an important strength of this approach. However, when data types and dependencies change continually, static relationships are inappropriate, and there is a need for non-relational extensions.

The object-oriented database model (Gray *et al.*, 1992) allows data items and tables to be defined without static data dependencies. A data item may be considered as a self-contained 'object' and its relationships to other objects handled by 'methods' or 'actions' defined within the objects. A database may have a base of statically defined explicit relationships with a dynamic layer of relationships provided by presenting some (or all) items as objects. Objects have well defined attributes, some of which may involve relationships with other data items, but objects need not have preordained links imposed by the static database structure.

The attributes and the functionality of CIF dictionaries incorporate aspects of both the basic relational and the object-oriented model. These provide the flexibility and extensibility associated with object-oriented data, as well as the relational links important to data validation and, ultimately, data derivation.

2.5.3. Definition attributes

Efficient data exchange depends implicitly on the prior knowledge of the data. For CIF data, this knowledge is specified in a data dictionary using definition attributes. A unique set of attribute values exists for each kind of data item, be it numerical, textual or symbolic, because these characteristics represent its identity and function. This is illustrated below with two simple examples.

2.5.3.1. Example 1: attributes of temperature

Every numerical data item has distinct properties. Consider the number 20 as a measure of temperature in degrees. To understand this number it is essential to know its measurement units. If these are degrees Celsius, one knows the item is in the *temperature* class, *degrees Celsius* sub-class, and that a lower enumeration boundary of any value can be specified at -273.15. Such a constraint can be used in data validation. More to the point, without any knowledge of both the class and subclass, a numerical value has no meaning. The number 100 is unusable unless one knows what it is a measure of (*e.g.* temperature or intensity) and, equally, unless one knows what the units are (*e.g.* degrees Celsius, Kelvin or Fahrenheit; or electrons or volts).

2.5.3.2. Example 2: attributes of Miller indices

Knowing the inter-dependency of one data item on another plays a major role in the understanding and validation of data. If a triplet of numbers $5, 3, 0$ is identified as Miller indices h, k, l, one immediately appreciates the significance of the index triplet as a vector in reciprocal crystal space. This stipulates that the three numbers form a single non-scalar data item in which the indices are non-associative (*e.g.* $3, 5, 0$ is not equivalent to $5, 3, 0$) and irreducible (*e.g.* the index 3 alone has no meaning). As a reciprocal-space vector, the triplet has other properties if is part of a list of other reciprocal-space data items; namely, its value represents a key or list pointer (*i.e.* a unique key to a row of items in a list table) to other data items in the list associated with this vector. This means that data forming a 'reflection' list are inaccessible if these indices are absent, or invalid if there is more than one occurrence of the same triplet in the list. Such interdependency and relational information is very important to the application of data, and needs to be specified in a dictionary to enable unambiguous access and validation. Other types of data dependencies will be described in Section 2.5.6.

Table 2.5.4.1. *Comparison of DDL1 and DDL2 variants*

DDL1	DDL2
Data names identify the category of data as `_<category>_<detail>`	Data names identify the category as `_<category>.<detail>`
Definitions are declared as data blocks with `data_<dataname>`	Definitions are declared starting with `save_<dataname>` and ending with `save_`
An irreducible set of items is declared within one definition *e.g.* Miller indices h, k, l	All items are defined in separate frames related by `_item_dependent.name`
Items that appear in lists are identified with the attribute `list_`	List and non-list data items are not distinguished
List dependencies are declared within each definition *e.g.* `_list_reference`	Dependencies are declared in a category definition *e.g.* `_category_key.name`
	Identifies subcategories of data within category groupings *e.g.* matrix
	Provides aliases to equivalent names, including those in DDL1 dictionaries

2.5.4. DDL versions

The capacity of a DDL to precisely define data items depends implicitly on the scope of the available attributes. It is quite possible, therefore, that a completely new data property cannot be specified using an existing DDL. In a field where data types evolve rapidly, the currently used dictionary language may be inadequate for the precise specification of an item. It is inevitable that future data items will exceed the capacity of existing dictionary attributes and methods to describe new data properties, and the dictionaries must evolve much in the same way that spoken languages continually adapt to changing modes of expression.

The first DDL used in crystallography (in 1990) was developed to compile a dictionary of 'core structural' crystallographic data items (Hall *et al.*, 1991). These data items were intended for use when submitting a manuscript to the journal *Acta Crystallographica* (and still are). The 'core' DDL is known commonly as DDL1 (Hall & Cook, 1995). Several years later, the definition of macromolecular crystallographic data items needed hierarchical descriptors for the different levels of structural entities, and an 'mmCIF' DDL, known as DDL2 (Westbrook & Hall, 1995), was developed. DDL2 was used to build the mmCIF dictionary (Bourne *et al.*, 1997). DDL1 is described in this chapter and DDL2 is described in Chapter 2.6. The DDL1 attributes have been used to construct the crystallographic core (fundamental structural), pd (powder diffraction), ms (modulated and composite structures) and rho (electron density) dictionaries. These dictionaries are discussed in Chapters