International Tables for Crystallography (2006). Vol. G, Chapter 5.5, pp. 539–543.

5.5. The use of mmCIF architecture for PDB data management

BY J. D. WESTBROOK, H. YANG, Z. FENG AND H. M. BERMAN

5.5.1. Introduction

The Protein Data Bank (PDB) is an archive for macromolecular structures (Bernstein *et al.*, 1977; Berman *et al.*, 2000) and a major component of a global resource for macromolecular structural science (Berman *et al.*, 2003). The scale of its data handling operations is large, and depends on the effective exploitation of the latest developments in the science and technology of informatics. A significant component of its data storage and retrieval strategy is the management of structural data in mmCIF format with appropriate extensions.

Over its 30-year history, the PDB archive has grown from seven entries in 1973 to a collection of over 30 000 structures as of May 2005. The growth in the size of the archive has been accompanied by increases in both data content and in the structural complexity of individual entries. As the PDB has grown, there has been a significant broadening of its user community. In response to this change, the role of the PDB has expanded from being simply a provider of structure data files to providing a key information resource for the structural biology community.

Looking forward, an acceleration in the growth of the PDB archive is anticipated owing to developments in high-throughput structural determination methodologies and worldwide structural genomics efforts. To support the continued growth and evolution of the PDB archive, a framework is required that supports automation and scalability, and that can adapt to changes in both data content and delivery technology.

At the core of the PDB informatics infrastructure is an ontology of data definitions which electronically encode domain information in the form of precise definitions, examples and controlled vocabularies. In addition to domain information, data definitions also encode information such as data type, data relationships, range restrictions and presentation units.

The software-accessible PDB exchange data dictionary (Appendix 3.6.2) is the key part of the PDB informatics infrastructure. The exchange dictionary is an extension of the macromolecular Crystallographic Information File (mmCIF) data dictionary (Bourne *et al.*, 1997). The dictionary provides the foundation for software tools which exchange and validate data, create and load databases, translate data formats, and serve application program interfaces. The components of the informatics infrastructure developed by the PDB are being used to build a data pipeline to support high-throughput structure determination.

5.5.2. Representing macromolecular structure data

Macromolecular structure data have historically been represented in a simple record-oriented format developed by the PDB; this format has been widely used in structural and computational biology.

Affiliations: JOHN D. WESTBROOK, HUANWANG YANG, ZUKANG FENG and HELEN M. BERMAN, Protein Data Bank, Research Collaboratory for Structural Bioinformatics, Rutgers, The State University of New Jersey, Department of Chemistry and Chemical Biology, 610 Taylor Road, Piscataway, NJ 08854-8087, USA.

```
CRYST1  129.230   60.440   56.630  90.00 119.05  90.00 C 1 2 1       4
ATOM      1  N   ASP A   1      23.482  -0.621  -1.419  1.00 35.27           N
ATOM      2  CA  ASP A   1      24.897  -0.728  -1.885  1.00 32.46           C
ATOM      3  C   ASP A   1      25.573   0.515  -1.339  1.00 28.22           C
ATOM      4  O   ASP A   1      24.918   1.359  -0.744  1.00 29.11           O
ATOM      5  CB  ASP A   1      24.976  -0.729  -3.427  1.00 38.24           C
```

Fig. 5.5.2.1. Excerpt of records from a PDB data file.

While this PDB format has in general been adequate for representing coordinate data, it has proved less satisfactory for the description of related information such as chemical and biological features and experimental methodology. To provide a more rigorous data encoding that includes all of this related information, the Protein Data Bank has in recent years adopted a comprehensive ontology of structure and experiment based on the content of the mmCIF data dictionary.

5.5.2.1. PDB format

For the past 30 years, the PDB has served as the single central repository for macromolecular structure data. The data format used to store archival entries in the PDB is a column-oriented data format resembling many data formats developed to accommodate the limitations of paper punched-card technology (see Chapter 1.1). An example of the data format is shown in Fig. 5.5.2.1.

Many of the data records in this format are prefixed with a record tag (*e.g.* CRYST1, ATOM) followed by individual items of data. The specifications for the records in this data format are described informally by Callaway *et al.* (1996). In addition to the labelled records as in Fig. 5.5.2.1, many data records in the PDB format are presented as unstructured or only semi-structured remark records.

5.5.2.2. Ontology representation of macromolecular structure data

In 1998, the Research Collaboratory for Structural Bioinformatics (RCSB) assumed the management responsibilities for the PDB. One important outcome was the change in the underlying data representation used to process PDB data. The PDB now collects and processes data using a data representation based on a comprehensive ontology of macromolecular structure and experiment: the PDB exchange data dictionary. This representation is an extension of the mmCIF data dictionary, now the standard data representation for experimentally determined three-dimensional macromolecular structures. The dictionary and data files based on this data ontology (Westbrook & Bourne, 2000) are expressed using Self-defining Text Archival and Retrieval (STAR) syntax (Chapter 2.1).

Although the mmCIF dictionary was developed within the crystallographic community, the metadata model employed by mmCIF is quite general and has been adopted by other application domains including NMR, molecular modelling and molecular recognition (dictionaries are available at http://mmcif.pdb.org/). Within the crystallographic community, metadata dictionaries have also been

developed for other types of diffraction experiments, electron-microscopy data and for the general description of image data. The metadata concepts and tools that have been developed to support mmCIF are sufficiently general that they may be applied to the description of data in virtually any application.

The demands of structural genomics projects have driven the development of extensions to capture an increased level of experimental detail. These are available at http://mmcif.pdb.org/. Extensions have also been introduced to describe NMR, cryo-electron microscopy and all aspects of protein production. The ability to rapidly add extensions and incorporate these into the PDB data-processing system is an important feature for supporting the rapidly evolving technologies associated with high-throughput structure determinations.

The mmCIF metadata architecture is built from three levels as illustrated in Fig. 5.5.2.2 (see also Chapter 2.6). Individual data files are described at the top level (*e.g.* Fig. 5.5.2.2*a*). The contents of these data files are defined by a data dictionary (*e.g.* Fig. 5.5.2.2*b*) in the next lower level (see Chapters 3.6 and 4.5). The attributes used in this data dictionary to build data definitions are in turn defined in the dictionary description language (DDL) (*e.g.* Fig. 5.5.2.2*c*) in the lowest level (see Chapters 2.6 and 4.10).

The major syntactical constructs used by mmCIF are illustrated in the data file example of Fig. 5.5.2.2(*a*). Each data item or group of data items is preceded by an identifying keyword. Groups of related data items are organized into data categories. Two categories, CELL and ENTITY_POLY_SEQ, are shown in the example. CELL contains an individual instance describing a single set of crystallographic cell constants. ENTITY_POLY_SEQ contains a `loop_` (*i.e.* table) of instances describing a polymer residue sequence. Essentially all mmCIF data are described as a set of tabular data structures.

Each mmCIF data item is defined in a data dictionary. Data definitions are given between save-frame delimiters (*i.e.* `save_`); apart from this, the data definitions share the same simple syntax as used in data files. An example definition for a crystallographic cell constant is shown in Fig. 5.5.2.2(*b*). Many features of the cell constant are described in this definition, including data type, range restrictions, units of expression, dependent quantities, related definitions, necessity and related precision estimate. Although not shown in this example, dictionary definitions can also include parent–child relationships that have important consequences in maintaining data consistency.

The attributes of each data definition are defined in the DDL dictionary. Fig. 5.5.2.2(*c*) shows example DDL definitions describing data types. DDL definitions have the same syntax as definitions used in the data dictionary. Because the attributes of the DDL are also used in DDL definitions, this metadata architecture is described as self-defining.

The RCSB PDB distributes parsing tools that support all three levels of this metadata architecture (http://sw-tools.pdb.org/). The *CIFPARSE_OBJ* package (Tosic & Westbrook, 2000) provides high-level methods to read, write, validate and manage data from data files, dictionaries and DDLs. Data files can be validated relative to an input data dictionary, and dictionary files can be validated relative to an input DDL. *CIFPARSE_OBJ* stores information in a collection of table objects. Access methods are provided to search and manipulate the table objects. A companion package, *CIFOBJ* (Schirripa & Westbrook, 1996), provides an alternative representation of dictionary and DDL data. *CIFOBJ* organizes dictionary information into a collection of category and item-level objects. Access methods are provided for all dictionary attributes.

```
_cell.entry_id      W1QQQ
_cell.length_a      129.230
_cell.length_b       60.440
_cell.length_c       56.630
_cell.angle_alpha    90.00
_cell.angle_beta    119.05
_cell.angle_gamma    90.00
_cell.Z_PDB           4
loop_
  _entity_poly_seq.entity_id
  _entity_poly_seq.num
  _entity_poly_seq.mon_id
    1    1     ASP       1    2     ILE
    1    3     VAL       1    4     LEU
    1    5     THR       1    6     GLN
    1    7     SER       1    8     PRO
    1    9     ALA       1    10    SER
```

(*a*)

```
save__cell.length_a
    _item_description.description
; Unit-cell length a corresponding to the structure
  reported.
;
    _item.name                      '_cell.length_a'
    _item.category_id                 cell
    _item.mandatory_code              no
    _item_aliases.alias_name        '_cell_length_a'
    _item_aliases.dictionary          cif_core.dic
    _item_aliases.version             2.0.1
     loop_
    _item_dependent.dependent_name
                                    '_cell.length_b'
                                    '_cell.length_c'
     loop_
    _item_range.maximum
    _item_range.minimum              .    0.0
                                    0.0   0.0
    _item_related.related_name      '_cell.length_a_esd'
    _item_related.function_code       associated_esd
    _item_sub_category.id             cell_length
    _item_type.code                   float
    _item_type_conditions.code        esd
    _item_units.code                  angstroms
    save_
```

(*b*)

```
save_ITEM_TYPE_LIST
    _category.description
;   Attributes which define each type code.
;
    _category.id                    item_type_list
    _category.mandatory_code        no
    _category_key.name            '_item_type_list.code'
     loop_
    _category_group.id              'ddl_group'
                                    'item_group'
     save_

save__item_type_list.code
    _item_description.description
;    The codes specifying the nature of the data value.
;
     loop_
    _item.name
    _item.category_id
    _item.mandatory_code
         '_item_type_list.code'   item_type_list      yes
         '_item_type.code'        item_type           yes

    _item_type.code                 code
    _item_linked.child_name       '_item_type.code'
    _item_linked.parent_name      '_item_type_list.code'

     save_
```

(*c*)

Fig. 5.5.2.2. Files at different levels of the mmCIF metadata architecture. (*a*) mmCIF data file excerpt. (*b*) Example mmCIF data dictionary definition. (*c*) Example DDL dictionary attribute definition.

5.5.2.3. Supporting other data formats and data delivery methods

One of the greatest benefits of a dictionary-based informatics infrastructure is the flexibility that it provides in supporting alternative data formats and delivery methods. Because the data and all of their defining attributes are electronically encoded, translation between data and dictionary formats can be achieved using lightweight software filters without loss of any information.

XML provides a particularly good example of the ease with which data can be converted to and from the mmCIF format. XML translations of mmCIF data files are currently provided on the Worldwide PDB ftp site (ftp://ftp.wwpdb.org/pub/pdb/data/structures/divided/XML/). These XML files use mmCIF dictionary data-item names as XML tags. These files were created by a translation tool (http://sw-tools.pdb.org/apps/MMCIF-XML-UTIL/) that translates mmCIF data files to XML in compliance with an XML schema. The XML schema is similarly software-translated from the PDB exchange data dictionary.

Other delivery methods such as Corba (http://www.omg.org/cgi-bin/doc?lifesci/00-02-02) do not require a data format, as data are exchanged using an application program interface (API). A Corba API for macromolecular structure (Greer *et al.*, 2002) based on the content of the mmCIF data dictionary has been approved by the Object Management Group (OMG). Software tools supporting this Corba API (*OpenMMS*, http://openmms.sdsc.edu, and *FILM*, http://sw-tools.pdb.org/apps/FILM) take full advantage of the data dictionary in building the interface definitions and supporting server on which the API is based (see also Section 5.3.8.2).

5.5.3. Integrated data-processing system: overview

The RCSB PDB data-processing system has been designed to take full advantage of the features of the mmCIF metadata framework. The AutoDep Input Tool (*ADIT*) is an integrated data-processing system developed to support deposition, data processing and annotation of three-dimensional macromolecular structure data.

This system, which is outlined in Fig. 5.5.3.1, accepts experimental and structural data from a user for deposition. Data are input in the form of data files or through a web-based form interface. The input data can be validated in a very basic sense for syntax compliance and internal consistency. Other computational validation can also be applied, including checking the input structure data against a variety of community standard geometrical criteria and comparing the input experimental data with the derived structure model. The suite of validation software used within *ADIT* is distributed separately (http://sw-tools.pdb.org/apps/VAL/). All of this validation information is returned to the user as a collection of HTML reports.

In addition to providing data-validation reports, *ADIT* also encodes data in archival data files and loads data into a relational database. The loading of data into the relational database is aided by an expert annotator. The *ADIT* system customizes its behaviour according to the user's requirements. One important distinction is between the behaviour of the interface provided for depositing data and that of the interface used for annotating the data. The depositor is focused only on data collection and provides the simplest possible presentation of the information to be input. The annotator sees the detail of all possible data items as well as the full functionality of the supporting data-processing software and database system.

Although the *ADIT* system was originally developed to support the centralized data deposition and annotation of macromolecular structure data, it is not limited to these particular applications. Because the architecture of the *ADIT* system derives the full scope of information to be processed from a data dictionary, the system can transparently provide data input and processing functionality for any content domain. This feature has been exploited in building a data-input tool for the BioSync project (Kuller *et al.*, 2002). The *ADIT* system can also be configured in workstation mode to provide single-user data collection and processing functionality. This version of the *ADIT* system as well as the supporting mmCIF parsing and data-management tools are currently distributed by the RCSB PDB under an open-source licence (http://sw-tools.pdb.org/apps/ADIT).

5.5.3.1. *ADIT*: functional description

The basic functions of the *ADIT* deposition system are shown in Fig. 5.5.3.2. Users interact with the *ADIT* system through a web server. The CGI components of the *ADIT* system (that is, functional software components interacting with web input data through the Common Gateway Interface protocol) dynamically build the HTML that provides the system user interface. These CGI components are currently implemented as compiled binaries from C++ source code.

User data can be provided in the form of data files or as keyboard input. Input files can be accepted in a variety of formats. *ADIT* uses a collection of format filters to convert input data to the data specification defined in a persistent data dictionary. Data in the form of data files are typically loaded first. Any input data that are not included in uploaded files can be keyed in by the user. *ADIT* builds a set of HTML forms for each category of data to be input. At any point during an input session, a user may choose to view or deposit the input data. Users who are depositing data may also use the data-validation services through the *ADIT* interface.

Comprehensive data ontologies like the PDB exchange dictionary contain vast numbers of data definitions. A data-input application may only need to access a small fraction of these definitions at any point. To address the problem of selecting only the relevant set of input data items from a data dictionary *ADIT* uses a view database. In addition to defining the scope of the data items to be edited by the *ADIT* application, an *ADIT* data view also stores

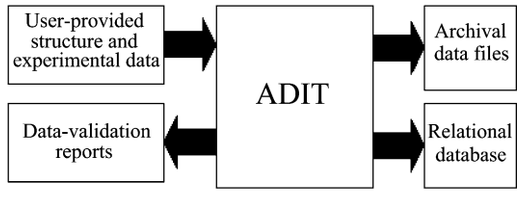

Fig. 5.5.3.1. Functional diagram of the *ADIT* system.

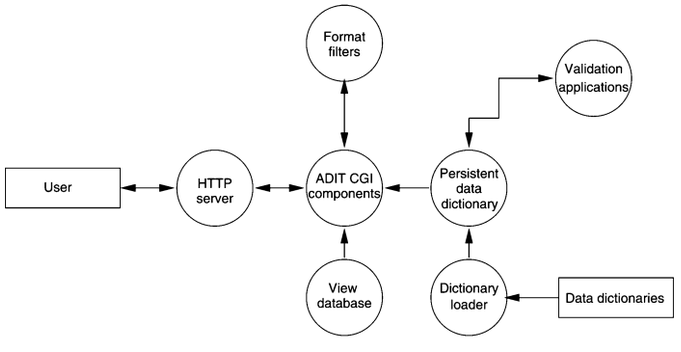

Fig. 5.5.3.2. Schematic diagram of *ADIT* editing, format translation and validation functions.

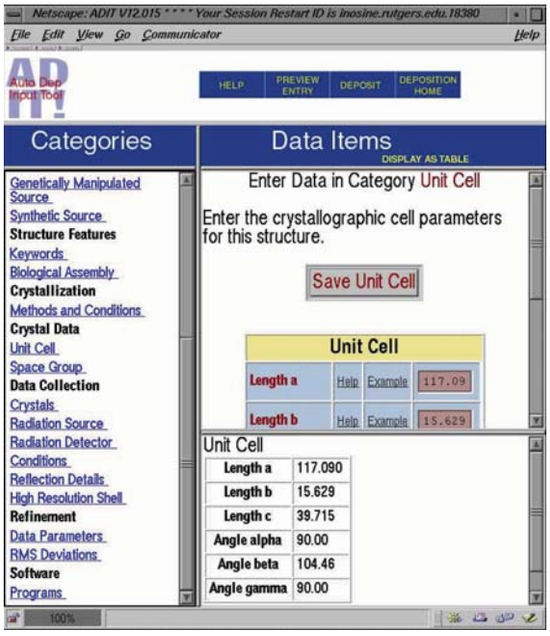

Fig. 5.5.3.3. Example *ADIT* data-input screen.

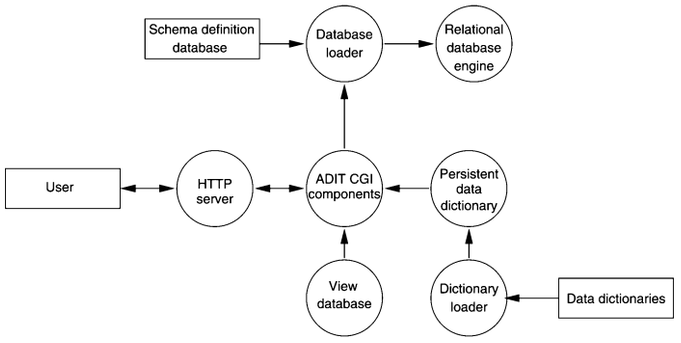

Fig. 5.5.3.4. Schematic diagram of *ADIT* database loading functions.

presentation details that are used in building the HTML input forms. An important use of the data view is to provide a simple and intuitive presentation of information for novice users which disguises the complex details of a data dictionary.

Fig. 5.5.3.3 shows an example *ADIT* editing screen for the crystallographic unit cell. The data dictionary category containing this information is named CELL, and the length of the first cell axis is defined in the dictionary as `_cell.length_a` (Fig. 5.5.2.2*b*). In this case, the data view has substituted *Unit Cell* and *Length a* for the dictionary data names. Although this example is simple, some dictionary data names are as long as 75 characters, and in these instances the ability to display a simpler name is essential.

Precise dictionary definitions and examples obtained from the data dictionary are accessible from the *ADIT* interface through buttons next to each data item. *ADIT* makes full use of the dictionary specification in data-input operations. Data items defined to assume only specific values have pulldown menus or selection boxes. Data type and range restrictions are checked when data are input and diagnostics are displayed to the user if errors are detected.

For performance reasons, the data dictionary is converted from its tabular text structure to an object representation using *CIFOBJ*. The class supporting the object representation provides efficient access functions to all of the data dictionary attributes. A dictionary loader is used to check the consistency of the data dictionary and to load the object representation from the text form of the data dictionary.

Any dictionary that complies with the dictionary description language (DDL2) can be loaded and used by *ADIT*. All *ADIT* software components gain their knowledge of the input data from the data dictionary and any associated data views. Consequently, *ADIT* can be tailored for use in virtually any data-input and data-processing application.

5.5.3.2. Generalized database support

In addition to the data editing and processing functions, *ADIT* also supports a versatile database loader (*mmCIF Loader*; http://sw-tools.pdb.org/apps/MMCIF-LOADER) that builds database schemata and extracts the processed data required to load database instances. The relation of the database loader to the central components of the *ADIT* system is shown in Fig. 5.5.3.4.

Schemata are defined in a metadata repository that is accessed by the loader application. In the simplest case, a schema can be constructed that is modelled directly from the data dictionary. Since the data model underlying the dictionary description language used to build *ADIT* data dictionaries is essentially relational, mapping a data dictionary specification to a relational schema is straightforward.

In other cases, a mapping is required between the target schema and the data dictionary specification. This mapping is encoded in the schema metadata repository. The database loader uses this mapping information to extract items from data files and translate these data into a form that can be loaded into the target database schema. The definition of the mapping operation can include: selection operations with equijoin constraints (*e.g.* the value of `_entity.type` where `_entity.id` = 1), aggregation (*e.g.* count, sum, average), collapse (*e.g.* vector to string), type conversions and existence tests.

Schema definitions are converted by the database loader into SQL instructions that create the defined tables and indices. Loadable data are produced either as SQL insert/update instructions or in the more efficient table copy formats used by popular database engines (*i.e.* DB2, Sybase, Oracle and MySQL). Loadable data can also be produced in XML.

5.5.3.3. Building a structure-determination data pipeline

One goal of high-throughput structural genomics is the automatic capture of all the details of each step in the process of structure determination. Fig. 5.5.3.5 shows a simplified structure-determination data pipeline. The essential details of each pipeline step are extracted and later assembled to make a data file for PDB deposition. The RCSB PDB data-processing infrastructure has been developed in anticipation of a data pipeline in which automated deposition would be the terminal step. The dictionary technology and software tools developed by the RCSB PDB to process and manage mmCIF data can be reused to provide the data-handling operations required to build the pipeline.

Dictionary definitions have been carefully developed to describe the details of each step in the structure-determination pipeline. These data items are typically accessible in electronic form after each program step. The information is either exported directly in mmCIF format or is printed in a program output file. To deal with the latter case, a utility program, *PDB_EXTRACT* (http://sw-tools.pdb.org/apps/PDB_EXTRACT), has been developed to parse program output files and extract key data values. In either case, the results of this incremental extraction of data from each program step must be merged to build a complete mmCIF

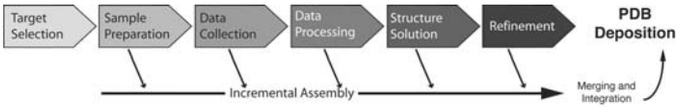

Fig. 5.5.3.5. Schematic diagram of a structure-determination data pipeline.

data file ready for deposition. The *PDB_EXTRACT* program also carrys out this merging operation.

Some steps in the structure-determination pipeline may not be driven by software. For instance, the details of protein production may be held in laboratory databases or within laboratory notebooks. A version of *ADIT* with a data view including all of the structural genomics data extensions has been created for entering these data. This *ADIT* tool can also be used to validate and check the completeness of the final data file.

5.5.4. Access

All of the software tools and libraries described in this chapter are distributed with full source under an open-source licence. Applications are also distributed in binary form for Intel/Linux, Sun/Solaris, SGI/IRIX and Dec Alpha platforms.

The RCSB/PDB is operated by Rutgers, The State University of New Jersey; the San Diego Supercomputer Center at the University of California, San Diego; and the Center for Advanced Research in Biotechnology of the National Institute of Standards and Technology. RCSB/PDB is supported by funds from the National Science Foundation (NSF), the National Institute of General Medical Sciences (NIGMS), the Department of Energy (DOE), the National Library of Medicine (NLM), the National Cancer Institute (NCI), the National Center for Research Resources (NCRR), the National Institute of Biomedical Imaging and Bioengineering (NIBIB), and the National Institute of Neurological Disorders and Stroke (NINDS).

References

Berman, H. M., Henrick, K. & Nakamura, H. (2003). *Announcing the worldwide Protein Data Bank. Nature Struct. Biol.* **10**, 980.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. (2000). *The Protein Data Bank. Nucleic Acids Res.* **28**, 235–242.

Bernstein, F. C., Koetzle, T. F., Williams, G. J. B., Meyer, E. F. Jr, Brice, M. D., Rodgers, J. R., Kennard, O., Shimanouchi, T. & Tasumi, M. (1977). *The Protein Data Bank: a computer-based archival file for macromolecular structures. J. Mol. Biol.* **112**, 535–542.

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

Callaway, J., Cummings, M., Deroski, B., Esposito, P., Forman, A., Langdon, P., Libeson, M., McCarthy, J., Sikora, J., Xue, D., Abola, E., Bernstein, F., Manning, N., Shea, R., Stampf, D. & Sussman, J. (1996). *Protein Data Bank contents guide: Atomic coordinate entry format description.* Brookhaven National Laboratory, New York, USA. Available from http://www.wwpdb.org/documentation/PDB_format_1996.pdf.

Greer, D. S., Westbrook, J. D. & Bourne, P. E. (2002). *An ontology driven architecture for derived representations of macromolecular structure. Bioinformatics*, **18**, 1280–1281.

Kuller, A., Fleri, W., Bluhm, W. F., Smith, J. L., Westbrook, J. & Bourne, P. E. (2002). *A biologist's guide to synchrotron facilities: the BioSync web resource. Trends Biochem. Sci.* **27**, 213–215.

Schirripa, S. & Westbrook, J. D. (1996). *CIFOBJ. A class library of mmCIF access tools.* Reference guide. http://sw-tools.pdb.org/apps/CIFOBJ/cifobj/index.html.

Tosic, O. & Westbrook, J. D. (2000). *CIFParse. A library of access tools for mmCIF.* Reference guide. http://sw-tools.pdb.org/apps/CIFPARSE-OBJ/cifparse/index.html.

Westbrook, J. & Bourne, P. E. (2000). *STAR/mmCIF: an ontology for macromolecular structure. Bioinformatics*, **16**, 159–168.