Table 3.1.8.1. *CIF dictionary register (maintained as a STAR File)*

```
data_validation_dictionaries
  loop_
    _cifdic_dictionary.name
    _cifdic_dictionary.version
    _cifdic_dictionary.DDL_compliance
    _cifdic_dictionary.reserved_prefix
    _cifdic_dictionary.URL
    _cifdic_dictionary.description
  cif_core.dic     .     1.4.1    .
      ftp://ftp.iucr.org/pub/cifdics/cif_core.dic
      'Core CIF Dictionary'
  cif_core.dic   1.0    .    .
      ftp://ftp.iucr.org/pub/cifdics/cifdic.C91
      'Original Core CIF Dictionary'
  cif_core.dic  2.3.1  1.4.1    .
      ftp://ftp.iucr.org/pub/cifdics/cif_core_2.3.1.dic
      'Core CIF Dictionary'
  cif_pd.dic      .      1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_pd.dic
      'Powder CIF Dictionary'
  cif_pd.dic    1.0.1    1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_pd_1.0.1.dic
      'Powder CIF Dictionary'
  cif_ms.dic      .      1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_ms.dic
      'Modulated structures CIF Dictionary'
  cif_ms.dic    1.0.1    1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_ms_1.0.1.dic
      'Modulated structures CIF Dictionary'
  cif_rho.dic     .      1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_rho.dic
      'Modulated structures CIF Dictionary'
  cif_rho.dic    1.0.1    1.4    .
      ftp://ftp.iucr.org/pub/cifdics/cif_rho_1.0.1.dic
      'Electron density CIF Dictionary'
  cif_mm.dic      .     2.1.2   .
      ftp://ftp.iucr.org/pub/cifdics/cif_mm.dic
      'Macromolecular CIF Dictionary'
  cif_mm.dic    1.0    2.1.2   .
      ftp://ftp.iucr.org/pub/cifdics/cif_mm_1.0.dic
      'Macromolecular CIF Dictionary'
  mmcif_std.dic     .    2.1.6   .
      ftp://ftp.iucr.org/pub/cifdics/mmcif_std.dic
      'Macromolecular CIF Dictionary'
  mmcif_std.dic    2.0.09   2.1.6   .
      ftp://ftp.iucr.org/pub/cifdics/cif_mm_2.0.09.dic
      'Macromolecular CIF Dictionary'
  cif_img.dic     .     2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_img.dic
      'Image CIF Dictionary'
  cif_img.dic    1.0    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_img_1.0.dic
      'Image CIF Dictionary'
  cif_img.dic    1.3.2    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_img_1.3.2.dic
      'Image CIF Dictionary'
  cif_sym.dic     .     2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_sym.dic
      'Symmetry CIF Dictionary'
  cif_sym.dic   1.0.1   2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/cif_sym_1.0.1.dic
      'Symmetry CIF Dictionary'
  cif_compat.dic    .    1.4   .
      ftp://ftp.iucr.org/pub/cifdics/cif_compat.dic
      'Legacy CIF Dictionary of deprecated terms'
  ddl_core.dic    .    1.4.1   .
      ftp://ftp.iucr.org/pub/cifdics/ddl_core.dic
      'Non-relational dictionary definition language'
  ddl_core_2.1.3.dic    .    2.1.3   .
      ftp://ftp.iucr.org/pub/cifdics/ddl_core_2.1.3.dic
      'Relational dictionary definition language'
  mmcif_ddl.dic    .    2.1.6   .
      ftp://ftp.iucr.org/pub/cifdics/mmcif_ddl.dic
      'Relational dictionary definition language'
```

Table 3.1.8.1 is an extract from the current register. The latest version of the register will always be available from the URL given above.

The entries for each dictionary include one with the version string set to '.', representing the current version; this is the version that should be retrieved unless a data file specifies otherwise.

Note that the register may also contain locators for local dictionaries constructed by owners of reserved prefixes (Section 3.1.2.2) when the owner has requested that a dictionary of local names be made publicly available. An appropriate name for a local dictionary in the register (`_dictionary_name` or `_dictionary.title` for DDL1 or DDL2 dictionaries, respectively) would be `cif_local_`*`myprefix`*`.dic`, where the string indicated by *myprefix* is one of the prefixes reserved for private use by the author of the dictionary (see Section 3.1.2.2). This scheme complements the naming convention for public dictionaries.

3.1.8.3. Locating a dictionary for validation

The following protocol applies to the creation and use of software designed to locate the dictionaries referenced by a data file and validate the data file against them. The protocol is necessary to address the issues that arise because dictionaries evolve through various audited versions, because not all dictionaries referenced by a data file may be accessible, and because data files might not in practice contain pointers to their associated dictionaries.

Software source code for applications that use CIF dictionaries to validate the contents of data files should be distributed with a copy of the most recent version of the register of dictionaries, and with the URL of the master copy hard-coded. Library utilities should be provided that permit local cacheing of the register file and the ability to download and replace the cached register at regular intervals. Individual dictionary files located and retrieved through the use of the register should also be cached locally, to guard against temporary unavailability of network resources.

Each CIF data file should contain a reference to one or more dictionary files against which the file may be validated. At the very least this will be `_audit_conform_dict_name` (`_audit_conform.dict_name` for DDL2 files) (*N*). `*_version` (*V*) and `*_location` (*L*) are optional. In the event that no dictionaries are specified, the default validation dictionary should be that identified as having *N* = `cif_core.dic` and *V* = '.' (*i.e.* the most recent version of the core dictionary). Since dictionaries are intended always to be extended, it is normally enough just to specify the name (and possibly the location).

This default is appropriate for most well formed CIFs, but if it is important to provide formal validation of old CIFs conforming to the earliest printed specification, which used the now-deprecated units extension convention, the dictionary cif_compat.dic may also be added to the default list (Section 3.1.5.4.3).

There is a difficulty associated with assuming this default for CIFs containing DDL2 data names. At present, the DDL2 version of the core dictionary does not exist as a separate file. Most existing CIFs built on the DDL2 model conform to the macromolecular (mmCIF) dictionary, and so best current working practice is to assume a default validation dictionary for DDL2-style CIFs with *N* = `mmcif_std.dic` and *V* = '.' (*i.e.* the most recent version of the mmCIF dictionary), since this includes the core data names as a subset. However, to anticipate future developments, it is suggested that applications built to validate DDL2 files first search the register for a default entry with *N* = `cif_core.dic`, *V* = '.' and a value of 2 or higher for the relevant DDL version:

```
loop_
  _cifdic_dictionary.name
  _cifdic_dictionary.version
  _cifdic_dictionary.DDL_compliance
  cif_core.dic      .     2.1.2
```

A software application validating against CIF dictionaries should attempt to locate and validate against the referenced dictionaries in the order cited in the data file, according to the following procedure. The terms 'warning' and 'error' in this procedure are not necessarily messages to be delivered to a user. They may be handled as condition codes or return values delivered to calling procedures instead.

If *N*, *V* and *L* are all given, try to load the file from the location *L*, or a locally cached copy of the referenced file. If this fails, raise a warning. Then search the dictionary register for entries matching the given *N* and *V*. (An appropriate strategy would be to search a locally cached copy of the register, and to refresh that local copy with the latest version from the network if the search fails.) If a successful match is made, try to retrieve the file from the location given by the matching entry in the register (or a locally cached copy with the same *N* and *V* previously fetched from the location specified in the register). If this fails, try to load files identified from the register with the same *N* but progressively older versions *V* (version numbering takes the form *n.m.l* . . ., where *n*, *m*, *l*, . . . are integers referring to progressively less significant revision levels). Version '.' (meaning the current version) should be accessed before any other numbered version. If this fails, raise a warning indicating that the specified dictionary could not be located.

If *N* and *V* but not *L* are given, try to load locally cached or master copies of the matching dictionary files from the location specified in the register file, in the order stated above, *viz*: (i) the version number *V* specified; (ii) the version with version number indicated as '.'; (iii) progressively older versions. Success in other than the first instance should be accompanied by a warning and an indication of the revision actually loaded.

If only *N* is given, try to load files identified in the register by (i) the version with version number indicated as '.'; (ii) progressively older versions.

If all efforts to load a referenced dictionary fail, the validation application should raise a warning.

If all efforts to load all referenced dictionaries fail, the validation application should raise an error.

For any dictionary file successfully loaded according to this protocol, the validation application must perform a consistency check by scanning the file for internal identifiers (`_dictionary_name`, `_dictionary_version` or the DDL2 equivalents) and ensuring that they match the values of *N* and *V* (where *V* is not '.'). Failure in matching should raise an error.

3.1.9. Composite dictionaries

The dictionaries referenced by a data file are those that contain the definitions of the data names used in the data file. Typically these include or consist entirely of public dictionaries that are necessarily permissive in the range of values allowed for data items. However, the power and flexibility of validating against machine-readable dictionaries could be harnessed by applications that need to impose stricter validation criteria. For example, the core dictionary permits an enumeration range of 0 to 8 for `_atom_site_attached_hydrogens`, but one might wish to validate a data set describing well behaved organic molecules where anything above 4 is almost certainly an error. It would be helpful to have a validation dictionary identical to the core dictionary except for this enumeration range; however it would be inefficient to create an alternative dictionary of the same size simply for this one change. In Section 3.1.9.1, we consider how to build a dictionary file that includes the bulk of the content of the public dictionaries cited in the CIF, together with modifications in local dictionary files to allow alternative specifications of what constitutes a 'valid' data item.

Proper applications of this approach include restricting the enumeration range specified for an item in a public dictionary; enforcing a more strict data typing than allowed by the parent dictionary; storing a list of all data names (including local ones) permitted in a CIF; or adding to existing dictionary entries references to local data items in an extension dictionary. An example of the latter application would be the addition of a `_list_link_child` entry to a public definition to accompany the introduction of a new child category in a local dictionary. The protocol to be described does not prohibit other applications, but care must be taken to generate dictionaries that retain internal consistency and are properly parsable by standard validation tools.

3.1.9.1. A dictionary merging protocol

The following protocol describes the construction of a *composite*, or *virtual*, dictionary by merging and overlaying fragments of a local validation dictionary and the public dictionaries referenced from within a data file. The term 'dictionary fragment' refers here to a physical disk file which contains one or more data blocks or save frames (according to whether the relevant data model is DDL1 or DDL2) containing complete or partial sets of attributes associated with data names identified in the relevant dictionary data block or save frame through the item `_name` (DDL1) or `_item.name` (DDL2).

(i) Assemble and load all dictionary fragments against which the current data block will be validated. The order of presentation is important. Complete dictionaries referenced by a data file should be assembled in the order cited. A dictionary validation application may then accept a list of additional dictionary fragments to PREPEND to, REPLACE or APPEND to each file in the list of cited dictionaries. In most applications, it will be appropriate to append to or replace attributes defined in a public dictionary, and the PREPEND operation is presented only for completeness.

(ii) Define three modes in which conflicting data names in the aggregate dictionary file may be resolved, called STRICT, REPLACE and OVERLAY.

(iii) Scan the aggregate dictionary fragments in the order of loading. Assemble for each defined data name a composite definition frame (data block or save frame as appropriate) as follows, depending on the mode in which the validation application is operating:

STRICT: If a data name appears to be multiply defined, generate a fatal error. This mode permits the interpolation of local dictionaries that do not attempt to modify the attributes of public data items.

REPLACE: All attributes previously stored for the conflicting data name are deleted, and only the attributes in the later data block (or save frame) containing the definition are preserved. This mode permits the complete redefinition of public data names and is not appropriate for validation of CIFs to be archived. Its main use would be in testing modifications of individual definition frames outside the parent dictionary.

OVERLAY: New attributes are added to those already stored for the data name; conflicting attributes replace those already stored. This is the standard mechanism for modifying attributes for application-specific validation purposes.

This protocol allows the creation of a coherent virtual dictionary from several different dictionary files or fragments. Although it must be used with care, it permits different levels of validation based on dictionary-driven methods without modifying the original dictionary files themselves.