International Tables for Crystallography (2006). Vol. G, Section 3.1.8, pp. 85–88.

associated with the group of three values in some well defined format (*e.g.* comma separated, as h, k, l) could have been defined instead. However, this would require a parser to understand the internal structure of the value so that it could parse out the separate values for h, k and l.

On the other hand, there are many examples of data values that are stored as string values parsable into distinct components. An extreme example is the reference list mentioned in Section 3.1.7.1. More common are dates (`_audit_creation_date`), chemical formulae (*e.g.* `_chemical_formula_moiety`), symmetry operations (`_symmetry_equiv_pos_as_xyz`) or symmetry transformation codes (`_geom_bond_site_symmetry_1`). There is no definitive answer as to which approach is preferred in a specific case. In general, the separation of the components of a compound value is preferred when a known application will make use of the separate components individually. For instance, applications may list structure factors according to a number of ordering conventions on individual Miller indices. As an extreme example of separating the components of a compound value, the mmCIF dictionary defines data names for the standard uncertainty values of most of the measurable quantities it describes, while the core dictionary just uses the convention that a standard uncertainty is specified by appending an integer in parentheses to a numeric value.

When compound values are left as parsable strings, the parsing rules for individual data items need to be made known to applications. The DDL1 attribute `_type_construct` was envisaged as a mechanism for representing the components of a data value with a combination of regular expressions and reference to primitive data items, but this has not been implemented in existing CIF dictionaries (or in dictionary utility software). An alternative approach used in DDL2-based dictionaries defines within the dictionaries a number of extended data types (expressed in regular-expression notation through the attribute `_item_type_list.code`).

A related problem is how to handle data names that describe an indeterminate number of parameters. For example, in the modulated structures dictionary an extra eight Miller indices are defined to span a reciprocal space of dimension up to 11. In principle, the dimensionality could be extended without limit. According to the practice of defining a unique data name for each modulation dimension, new data names would need to be defined as required to describe higher-dimensional systems. Beyond a certain point this will become unwieldy, as will the set of data names required to describe the n^2 components of the W matrix for a modulated structure of dimensionality n (`_cell_subsystem_matrix_W_1_1` *etc.*).

The modulated structures dictionary was constrained to define extended Miller indices in this way for compatibility with the core dictionary. Data names describing new quantities that are subject to similar unbounded extensibility should perhaps refer to values that are parsable into vector or matrix components of arbitrary dimension.

3.1.7.5. Consistency of abbreviations

One further consideration when constructing a data name is the use of consistent abbreviations within the components of the data name. This is of course a matter of style, since if a data name is fully defined in a dictionary with a machine-readable attribute set, the data name itself can be anything. Nonetheless, to help to find and group similar data names it is best to avoid too many different abbreviations.

Table 3.1.7.1 lists the abbreviations used in the current public dictionaries. Note that there are already cases where different abbreviations are used for the same term.

3.1.8. Management of multiple dictionaries

So far this chapter has discussed the mechanics of writing dictionary definitions and of assembling a collection of definitions in a single global or local dictionary file. In practice, the set of data names in a CIF data file may include names defined in several dictionary files. A mechanism is required to identify and locate the dictionaries relevant to an individual data file. In addition, because dictionaries are suitable for automated validation of the contents of a data file, it is convenient to be able to overlay the attributes listed in a dictionary with an alternative set that permit validation against modified local criteria. This section describes protocols for identifying, locating and overlaying dictionary files and fragments of dictionary files.

3.1.8.1. Identification of dictionaries relevant to a data file

A CIF data file should declare within each of its data blocks the names, version numbers and, where appropriate, locations of the global and local dictionaries that contain definitions of the data names used in that block. For DDL1 dictionaries, the relevant identifiers are the items `_audit_conform_dict_name`, `_audit_conform_dict_version` and `_audit_conform_dict_location`, defined in the core dictionary. DDL2 dictionaries are identified by the equivalent items `_audit_conform.dict_name`, `*.dict_version` and `*.dict_location`. For convenience, the DDL1 versions will be used in the following discussion.

The values of the items `_audit_conform_dict_name` and `_audit_conform_dict_version` are character strings that match the values of the `_dictionary_name` and `_dictionary_version` identifiers in the dictionary that defines the relevant data names. Validation against the latest version of a dictionary should always be sufficient, since every effort is made to ensure that a dictionary evolves only by extension, not by revising or removing parts of previous versions of the dictionary. Nevertheless, including `_audit_conform_dict_version` is encouraged: it can be useful to confirm which version of the dictionary the CIF was initially validated against.

The data item `_audit_conform_dict_location` may be used to specify a file name or uniform resource locator (URL). However, a file name on a single computer or network will be of use only to an application with the same view of the local file system, and so is not portable. A URL may be a better indicator of the location of a dictionary file on the Internet, but can go out of date as server names, addresses and file-system organization change over time. The preferred method for locating a dictionary file is to make use of a dynamic register, as described in Section 3.1.8.2. Nevertheless, `_audit_conform_dict_location` remains a valid data item that may be of legitimate use, particularly in managing local applications.

The following example demonstrates a statement of dictionary conformance in a data file describing a powder diffraction experiment with some additional local data items:

```
loop_
  _audit_conform_dict_name
  _audit_conform_dict_version
  _audit_conform_dict_location
    cif_core.dic      2.3.1   .
    cif_pd.dic        1.0.1   .
    cif_local_my.dic  1.0
       /usr/local/dics/my_local_dictionary
```

Table 3.1.7.1. *Abbreviations in CIF data names*

Terms for which abbreviations are defined are sometimes found unabbreviated.

Abbreviation	Term	Abbreviation	Term	Abbreviation	Term
abbrev	abbreviation	eqn	equation	oper	operation
abs	absolute (configuration, not structure)	esd	standard uncertainty (estimated	org	organism
absorpt	absorption		standard deviation) (*see* su)	orient	orientation
alt	alternative	expt	experiment	origx	orthogonal coordinate matrix (PDB files)
amp	amplitude	exptl	experimental	os	operating system
AN	accession number	fom	figure of merit	param	parameter
anal	analyser	fract	fractional	pd	powder diffraction
aniso	anisotropic*	Fsqd	*F* squared	PDB	Protein Data Bank
anisotrop	anisotropic*	gen	generation	PDF	Powder Diffraction File
anom	anomalous	gen	generator	perp	perpendicular
ASTM	American Society for Testing and Materials	gen	genetic	phos	phosphate
asym	asymmetric	geom	geometric	pk	peak
atten	attenuation	H-M	Hermann–Mauguin	polarisn	polarization
au	arbitrary units	ha	heavy atom	poly	polymer
auth	author	hbond	hydrogen bond	pos	position
av	average	hist	history	prep	preparation
ax	axial	horiz	horizontal	proc	processed
B	*B* form of atomic displacement	I	intensity	prof	profile
	parameter (a.d.p.)	ICSD	Inorganic Crystal Structure Database	prot	protein
backgd	background*	id	identifier	ptnr	partner
beg	begin	illum	illumination	publ	publication
bg	background*	imag	imaginary	R	agreement index
biol	biology	inc	increment	rad	radius
bkg	background*	incl	include	recd	received
bond	bonding	info	information	recip	reciprocal
Bsol	*B* form of a.d.p. for solvent	instr	instrument	ref	reference
calc	calculated	Int	international	refine	refinement
calib	calibration (pd)	ISBN	International Standard Book Number	refln	reflection
cartn	Cartesian	iso	isotropic	reflns	reflections
CAS	*Chemical Abstracts* Service	iso	isomorphous	res	resolution
char	characterization (pd)	ISSN	International Standard Serial Number	restr	restraints
chem	chemical	IUCr	International Union of Crystallography	rev	revision
chir	chirality	IUPAC	International Union of Pure and	Rmerge	agreement index of merging
clust	cluster		Applied Chemistry	rms	root mean square
coef	coefficient	len	length	rot	rotation
com	common	lim	limit	S	goodness of fit
comp	component	loc	lack of closure	samp	sample
conc	concentration	ls	least squares	scat	scattering factor
conf	conformation	max	maximum	seq	sequence
config	configuration	MDF	Metals Data File	sigI	$\sigma(I)$*
conform	conformant	meanI	mean intensity	sigmaI	$\sigma(I)$*
conn	connectivity	meas	measured	sint	$\sin\theta$
cons	constant	mid	middle (between max and min)	sint/lambda	$\sin(\theta)/\lambda$*
CSD	Cambridge Structural Database	min	minimum	sol	solvent
db	database	mod	modification	spec	specimen
defn	definition	mods	modifications	src	source
detc	detector	mon	monomer	std	standard
der	derivative	monochr	monochromator (pd)*	stol	$\sin(\theta)/\lambda$*
dev	standard deviation	mono	monochromator (pd)*	struct	structure
dict	dictionary	nat	natural	su	standard uncertainty
dif	difference*	NBS	National Bureau of Standards (now	suppl	supplementary
diff	difference*		National Institute of Standards and	sys	systematic
diffr	diffractometer		Technology)	tbar	mean path length
diffrn	diffraction	NCA	number of connected atoms	temp	temperature
displace	displacement	ncs	noncrystallographic symmetry	tor	torsion angle
dist	distance	netI	net intensity	tran	transformation*
divg	divergence	NH	number of connected hydrogen atoms	transf	transformation*
dom	domain	nha	non-hydrogen atoms	transform	transformation*
dtime	deadtime	norm	normal	tvect	translation vector (PDB files)
ens	ensemble	nst	nonstandard	vert	vertical
eq	equatorial*	nucl	nucleic acid	wR	weighted agreement index
equat	equatorial*	num	number	wt	weight
equiv	equivalent	obs	observed		

* Terms with multiple definitions.

It is clear that the location specified for the local dictionary is only meaningful for applications running on the same computer or network, and therefore the ability to validate against this local dictionary is not portable. On the other hand, it may be that the local data names used by the authors of this CIF are not intended to have meaning outside their own laboratory.

3.1.8.2. The dictionary register

COMCIFS maintains a register of dictionaries known to it, including the identifying name and version strings within those dictionaries. The register also includes the location of each dictionary, expressed at present as a URL designed to allow retrieval by file transfer protocol (ftp) from the IUCr server. Changes in the location of a particular dictionary file can be made by modifying the entry in the register, avoiding the problem of specifying a URL in a data file that would then become outdated if the dictionary was moved. Dictionary applications can consult the register (according to a protocol outlined below) to locate and retrieve the dictionaries needed for validating data files. It is of course essential that the validation software knows how to locate the register. The location is at present given by the URL ftp://ftp.iucr.org/pub/cifdics/cifdic.register.

The problem of changing URLs has therefore not disappeared completely, but is at least confined to the need to maintain one single address.

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.

references