DDLm dictionary

By S. R. Hall, N. Spadaccini and J. R. Hester

ATTRIBUTES

This category is parent of all other categories in the DDLm dictionary.

ALIAS

The attributes used to specify the aliased names of definitions.

Category key(s): **`_alias.definition_id`**

`_alias.definition_id` (*Tag*)

Identifier tag of an aliased definition.

`_alias.deprecation_date` (*Date*)

Date that the aliased tag was deprecated as a definition tag.

`_alias.dictionary_uri` (*Uri*)

Absolute URI of dictionary to which the aliased definition belongs.

CATEGORY_KEY

The attributes used to specify (possibly multiple) keys for a given category.

Category key(s): **`_category_key.name`**

`_category_key.name` (*Tag*)

A minimal list of tag(s) that together constitute a compound key to access other items in a Loop category. In other words, the combined values of the data items listed in this loop must be unique, so that unambiguous access to a packet (row) in the table of values is possible.

DEFINITION

The attributes for classifying dictionary definitions.

`_definition.class` (*Code*)

The nature and the function of a definition or definitions.

Where no value is given, the assumed value is '`Datum`'.

The data value must be one of the following:

`Attribute`	Item used as an attribute in the definition of other data items in DDLm dictionaries. These items never appear in data instance files.
`Functions`	Category of items that are transient function definitions used only in dREL methods scripts. These items never appear in data instance files.
`Datum`	Item defined in a domain-specific dictionary. These items appear only in data instance files.
`Head`	Category of items that is the parent of all other categories in the dictionary.
`Loop`	Category of items that in a data file may reside in a loop-list with a key item defined.
`Set`	Category of items that form a set (but not a loopable list). These items may be referenced as a class of items in a dREL methods expression.

`_definition.id` (*Code*)

Identifier name of the Item or Category being defined.

`_definition.scope` (*Code*)

The extent to which a definition affects other definitions.

Where no value is given, the assumed value is '`Item`'.

The data value must be one of the following:

`Dictionary`	Applies to all defined items in the dictionary.
`Category`	Applies to all defined items in the category.
`Item`	Applies to a single item definition.

`_definition.update` (*Date*)

The date that a definition was last changed.

DEFINITION_REPLACED

Attributes used to describe deprecated and replaced definitions.

Category key(s): **`_definition_replaced.id`**

`_definition_replaced.by` (*Tag*)

Name of the data item that should be used instead of the defined data item. The defined data item is deprecated and should not be used. A value of '.' signifies that the data item is deprecated, with no replacement.

`_definition_replaced.id` (*Code*)

An opaque identifier for the replacement.

DESCRIPTION

The attributes of descriptive (non-machine parsable) parts of definitions.

`_description.common` (*Text*)

Commonly-used identifying name for the item.

`_description.key_words` (*Text*)

List of key-words categorising the item.

`_description.text` (*Text*)

The text description of the defined item, category, or dictionary.

DESCRIPTION_EXAMPLE

Descriptive (non-machine parsable) examples of values of the defined items and categories.

Category key(s): **`_description_example.case`**

`_description_example.case` (*Implied*)

An example case of the defined item or category. Category example cases present data names and values as they would appear in a CIF-formatted file. Item example cases present values only, which inherit the enumeration range, enumeration set, container, dimension, content and purpose type constraints of the defining item.

`_description_example.detail` (*Text*)

A description of an example case for the defined item or category.

Affiliation: Sydney R. Hall and Nick Spadaccini, School of Molecular Sciences, The University of Western Australia (M310), 35 Stirling Highway, 6009 Perth, Australia; James R. Hester, Australian Nuclear Science and Technology Organisation, Locked Bag 2001, Kirrawee DC, NSW 2232, Australia.

DICTIONARY

Attributes for identifying and registering the dictionary. The items in this category are *not* used as attributes of INDIVIDUAL data items.

_dictionary.class (*Code*)

The nature, or field of interest, of data items defined in the dictionary.

Where no value is given, the assumed value is ‘Instance’.

The data value must be one of the following:

Reference	DDLm reference attribute definitions.
Instance	Domain-specific data instance definitions.
Template	Domain-specific attribute/enumeration templates.
Function	Domain-specific method function scripts.

_dictionary.date (*Date*)

The date that the last dictionary revision took place.

_dictionary.ddl_conformance (*Version*)

The version number of the DDL dictionary that this dictionary conforms to.

_dictionary.formalism (*Text*)

The definitions contained in this dictionary are associated with the value of this attribute. Data items may only be redefined if the value of this attribute is also changed, and any such redefinitions must include the original behaviour as a particular case.

_dictionary.namespace (*Code*)

The namespace code that may be prefixed (with a trailing colon ‘:’) to an item tag defined in the defining dictionary when used in particular applications. Because tags must be unique, namespace codes are unlikely to be used in data files.

_dictionary.title (*Code*)

The common title of the dictionary. Will usually match the name attached to the **data_** statement of the dictionary file.

_dictionary.uri (*Uri*)

An absolute uniform resource identifier (URI) for this dictionary.

_dictionary.version (*Version*)

A unique version identifier for the dictionary.

DICTIONARY_AUDIT

Attributes for identifying and registering the dictionary. The items in this category are *not* used as attributes of individual data items.

Category key(s): **_dictionary_audit.version**

_dictionary_audit.date (*Date*)

The date of each dictionary revision.

_dictionary_audit.revision (*Text*)

A description of the revision applied for the **_dictionary_audit.version**.

_dictionary_audit.version (*Version*)

A unique version identifier for each revision of the dictionary.

DICTIONARY_VALID

Data items which are used to specify the contents of definitions in the dictionary in terms of the **_definition.scope** and the required and prohibited attributes. Validation rules described by data items in this category apply only to

Reference and Instance dictionaries.

Category key(s): **_dictionary_valid.scope**
_dictionary_valid.option

_dictionary_valid.application (*Code*[2])

Deprecated. Provides the information identifying the definition scope (from the **_definition.scope** enumeration list) and the validity options (from the **_dictionary_valid.option** enumeration list), as a two element list.

_dictionary_valid.attributes (*Code*[])

A list of the attribute names and categories that are assessed for application in the item, category and dictionary definitions. A parent attribute category implicitly recursively includes all child categories.

_dictionary_valid.option (*Code*)

Option codes for applicability of attributes in definitions. Attributes not listed as ‘Prohibited’ for a given scope are allowed in that scope.

Where no value is given, the assumed value is ‘Recommended’.

The data value must be one of the following:

Mandatory	Attribute must be present in definition frame.
Recommended	Attribute is usually in definition frame.
Prohibited	Attribute must not be used in definition frame.

_dictionary_valid.scope (*Code*)

The scope to which the specified restriction on usable attributes applies.

The data value must be one of the following:

Dictionary	Restriction applies to dictionary definition.
Category	Restriction applies to a category definition.
Item	Restriction applies to an item definition.

ENUMERATION

The attributes for restricting the values of defined data items.

_enumeration.def_index_id (*Tag*)

Specifies the data name of the item with a value used as an index to the DEFAULT enumeration list (in category ENUMERATION_DEFAULT) in order to select the default enumeration value for the defined item. The value of the identified data item must match one of the **_enumeration_default.index** values.

_enumeration.default (*Implied*)

The default value for the defined item if it is not specified explicitly. Value of this attribute inherits the enumeration range, enumeration set, container, dimension, content and purpose type constraints of the defining item.

_enumeration.mandatory (*Code*)

Yes or No flag on whether the enumerate states specified for an item in the current definition (in which item appears) *must* be used on instantiation.

Where no value is given, the assumed value is ‘Yes’.

The data value must be one of the following:

Yes	Use of state is mandatory.
No	Use of state is unnecessary.

_enumeration.range (*Range*)

The inclusive range of numerical values allowed for the defined item. If the defined item has associated SU values, the reported data values may fall outside these limits.

Examples: ‘-4:10’ (Values must be no less than -4 and no greater than 10.), ‘0:’ (Values must be greater than or equal to 0.), ‘:3.1415’ (Values must be less than or equal to 3.1415.)

ENUMERATION_DEFAULT

Loop of pre-determined default enumeration values indexed to a data item by the item `_enumeration.def_index_id`.

Category key(s): `_enumeration_default.index`

`_enumeration_default.index` (*Code*)

Index key in the list default values referenced to by the value of `_enumeration.def_index_id`.

`_enumeration_default.value` (*Implied*)

Default enumeration value in the list referenced by the value of `_enumeration.def_index_id`. The reference index key is given by the value of `_enumeration_default.index` value.

ENUMERATION_SET

Attributes of data items which are used to define a set of unique pre-determined values.

Category key(s): `_enumeration_set.state`

`_enumeration_set.detail` (*Text*)

The meaning of the code (identified by `_enumeration_set.state`) in terms of the value of the quantity it describes.

`_enumeration_set.state` (*Text*)

Permitted value state for the defined item.

IMPORT

Used to import the values of specific attributes from other dictionary definitions within and without the current dictionary.

`_import.get` (*ByReference*)

A list of tables of attributes defined individually in the category IMPORT_DETAILS, used to import definitions from other dictionaries.

IMPORT_DETAILS

Items in IMPORT_DETAILS describe individual attributes of an import operation.

Category key(s): `_import_details.order`

`_import_details.file_id` (*Uri*)

A URI reference as per RFC 3986 giving the location of the source dictionary. When a relative URI is used, the base URI for the URI reference is the `_dictionary.uri` of the importing dictionary.

`_import_details.file_version` (*Version*)

The required version number for `_dictionary.version` of the imported dictionary. Dictionaries with the same major version number are compatible. If absent or null, any version is permitted.

`_import_details.frame_id` (*Code*)

The save frame code of the definition frame to be imported.

`_import_details.if_dupl` (*Code*)

Code identifying the action taken if the requested definition block already exists within the importing dictionary in ‘Full’ mode, or an attribute exists in both the importing definition block and the requested definition block in ‘Contents’ mode.

Where no value is given, the assumed value is ‘`Exit`’.

The data value must be one of the following:

`Ignore`	Ignore imported definitions if block identifiers match in ”Full” mode. Ignore imported attributes that match attributes already in the importing definition in ”Contents” mode. When importing in ”Contents” mode, if the ignored attribute belongs to a Loop category, all attributes from that category must be ignored to avoid loop mismatches.
`Replace`	Replace existing definitions with imported definitions if block identifiers match in ”Full” mode. When importing in ”Contents” mode, contents of the two save frames should be merged and any duplicate attributes replaced with those from the imported save frame. In case the replaced attribute belongs to a Loop category, all attributes from that category must first be removed from the importing save frame to avoid loop mismatches.
`Exit`	Issue an error exception and exit.

`_import_details.if_miss` (*Code*)

Code identifying the action taken if the requested definition block is missing from the source dictionary.

Where no value is given, the assumed value is ‘`Exit`’.

The data value must be one of the following:

`Ignore`	Ignore import.
`Exit`	Issue error exception and exit.

`_import_details.mode` (*Code*)

Code identifying how the definition referenced by `_import_details.frame_id` is to be imported. ‘Full’ imports the entire definition together with any child definitions (in the case of categories) found in the target dictionary. The importing definition becomes the parent of the imported definition. As such, the ‘Full’ mode must only be used in category definitions. As a special case, a ‘Head’ category importing a ‘Head’ category is equivalent to importing all children of the imported ‘Head’ category as children of the importing ‘Head’ category. A ‘Head’ category can only be imported in ‘Full’ mode and only by another ‘Head’ category. ‘Contents’ imports only the attributes found in the imported definition.

Where no value is given, the assumed value is ‘`Contents`’.

The data value must be one of the following:

`Full`	Import requested definition together with any child definitions.
`Contents`	Import contents of requested definition.

`_import_details.order` (*Integer*)

The order in which the import described by the referenced row should be executed.

`_import_details.single` (*Text*)

A Table mapping attributes defined individually in category IMPORT to their values; used to import definitions from other dictionaries.

`_import_details.single_index` (*Code*)

One of the indices permitted in the entries of values of attribute `_import_details.single`.

The data value must be one of the following:

`file`	URI reference as per RFC 3986 giving the location of the source dictionary.
`version`	Version of source dictionary.
`save`	Save frame code of source definition.
`mode`	Mode for including save frames.
`dupl`	Option for duplicate entries.
`miss`	Option for missing duplicate entries.

METHOD

Methods used for evaluating, validating and defining items.

Category key(s): **_method.purpose**

_method.expression (*Text*)

The method expression for the defined item.

_method.purpose (*Code*)

The purpose and scope of the method expression.

Where no value is given, the assumed value is ‘`Evaluation`’.

The data value must be one of the following:

`Evaluation`	Method evaluates an item from related item values.
`Definition`	Method generates attribute value(s) in the definition.
`Validation`	Method compares an evaluation with existing item value.

NAME

Attributes for identifying items and item categories.

_name.category_id (*Name*)

The name of the category in which a category or item resides. For Head categories this is the **_dictionary.title** given in the enclosing data block.

_name.linked_item_id (*Tag*)

Data name of an equivalent item which has a common set of values, or, in the definition of a type SU item is the name of the associated measurand item to which the standard uncertainty applies.

_name.object_id (*Name*)

The object name of a category or name unique within the category or family of categories.

TYPE

Attributes which specify the ‘typing’ of data items.

_type.container (*Code*)

The structure of values for the defined data item.

Where no value is given, the assumed value is ‘`Single`’.

The data value must be one of the following:

`Single`	Single value.
`List`	Ordered set of values. Elements need not be of same contents type.
`Array`	Ordered set of values of the same type. Operations across arrays are equivalent to operations across elements of the Array.
`Matrix`	Ordered set of numerical values for a tensor. Tensor operations such as dot and cross products, are valid cross matrix objects. A matrix with a single dimension is interpreted as a row or column vector as required.
`Table`	An unordered set of id:value elements.
`Implied`	**Applied ONLY in the DDLm Reference Dictionary.** The value structure is taken from **_type.container** in the definition in which the defined attribute appears.

_type.contents (*Code*)

Syntax of the value elements within the container type. Where the definition is of a ‘List’ or ‘Array’ type, this attribute describes the contents of each element. Where the definition is of a ‘Table’ container this attribute describes the construction of the value elements within those (Table) values. The CIF2 character set referenced below consists of the following Unicode code points: [U+0009], [U+000A], [U+000D], [U+0020-U+007E], [U+00A0-U+D7FF], [U+E000-U+FDCF], [U+FDF0-U+FFFD], [U+10000-U+1FFFD], [U+20000-U+2FFFD], [U+30000-U+3FFFD], [U+40000-U+4FFFD], [U+50000-U+5FFFD], [U+60000-U+6FFFD], [U+70000-U+7FFFD], [U+80000-U+8FFFD], [U+90000-U+9FFFD], [U+A0000-U+AFFFD], [U+B0000-U+BFFFD], [U+C0000-U+CFFFD], [U+D0000-U+DFFFD], [U+E0000-U+EFFFD], [U+F0000-U+FFFFD], [U+100000-U+10FFFD] Two ‘case insensitive’ strings are considered identical when they match under the Unicode canonical caseless matching algorithm. In all cases, ‘whitespace’ refers to ASCII whitespace only, that is [U+0009],[U+000A],[U+000D] and [U+0020]. Note that descriptions of text syntax are relevant only to those formats that encode data values as text.

Where no value is given, the assumed value is ‘`Text`’.

The data value must be one of the following:

`Text`	Case-sensitive sequence of CIF2 characters.
`Word`	Case-sensitive sequence of CIF2 characters containing no ASCII whitespace.
`Code`	Case-insensitive sequence of CIF2 characters containing no ASCII whitespace.
`Name`	Case-insensitive sequence of ASCII alphanumeric characters or underscore.
`Tag`	Case-insensitive CIF2 character sequence with leading underscore and no ASCII whitespace.
`Uri`	Uniform Resource Identifier reference as defined in RFC 3986 Section 4.1.
`Date`	ISO standard date format ¡yyyy¿-¡mm¿-¡dd¿. Use DateTime for all new dictionaries.
`DateTime`	A timestamp. Text formats must use date-time or full-date productions of RFC 3339 ABNF.
`Version`	Version number string that adheres to the formal grammar provided in the Semantic Versioning specification version 2.0.0. Version strings must take the general form of <major>.<minor>.<patch> and may also contain an optional postfix with additional information such as the pre-release identifier. Reference: https://semver.org/spec/v2.0.0.html
`Dimension`	Size of an Array/Matrix/List expressed as a text string. The text string itself consists of zero or more non-negative integers separated by commas placed within bounding square brackets. Empty square brackets represent a list of unknown size.
`Range`	Inclusive range of numerical values expressed using the min:max notation in which the smallest value ’min’ and the largest value ’max’ are separated by a colon character. If ’max’ is omitted, then the range includes all values that are greater than or equal to ’min’. If ’min’ is omitted, then the range includes all values that are less than or equal to ’max’.
`Integer`	A number from the set of all integers.
`Real`	Floating-point real number.
`Imag`	Floating-point imaginary number.
`Complex`	A complex number.
`Symop`	A string composed of an integer optionally followed by an underscore or space and three or more digits.
`Implied`	The contents are described by the **_type.contents** attribute in the definition in which the defined attribute appears.
`ByReference`	The contents have the same form as those of the attribute referenced by **_type.contents_referenced_id**.

Example: ‘`Integer`’ (Content is a single or multiple integer(s).)

_type.contents_referenced_id (*Tag*)

The value of the **_definition.id** attribute of an attribute definition whose type is to be used also as the type of this item. Meaningful only when this item‘s **_type.contents** attribute has value ’ByReference’.

_type.dimension (*Dimension*)

The dimensions of a list, array or matrix of elements expressed as a text string. A Matrix with a single dimension is interpreted as a vector.

Examples: '`[3,3]`' (3x3 matrix of elements.), '`[6]`' (List of 6 elements.), '`[]`' (Unknown number of list elements.)

_type.indices (*Code*)

Used to specify the syntax construction of indices of the entries in the defined object when the defined object has 'Table' as its **_type.container** attribute. Values are a subset of the codes and constructions defined for attribute **_type.contents**, accounting for the fact that syntactically, indices are always case-sensitive quoted strings. Meaningful only when the defined item has **_type.container** 'Table'. See the definition for **_type.contents** for the character set definition.

Where no value is given, the assumed value is '`Text`'.

The data value must be one of the following:

`Text` A case-sensitive sequence of CIF2 characters.

`Code` Case-insensitive sequence of CIF2 characters containing no ASCII whitespace.

`Date` ISO date format yyyy-mm-dd.

`Uri` A Uniform Resource Identifier string, per RFC 3986.

`Version` Version digit string of the form <major>.<version>.<update>

`ByReference` Indices have the same form as the contents of the attribute identified by **_type.indices_referenced_id**.

_type.indices_referenced_id (*Tag*)

The **_definition.id** attribute of a definition whose type describes the form and construction of the indices of entries in values of the present item. Meaningful only when the defined item's **_type.container** attribute has value 'Table', and its **_type.indices** attribute has value 'ByReference'.

_type.purpose (*Code*)

The primary purpose or function the defined data item serves in a dictionary or a specific data instance.

Where no value is given, the assumed value is '`Describe`'.

The data value must be one of the following:

`Import` **Applied ONLY in the DDLm Reference Dictionary.** Used to type the SPECIAL attribute "**_import.get**" that is present in dictionaries to instigate the importation of external dictionary definitions.

`Method` **Applied ONLY in the DDLm Reference Dictionary.** Used to type the attribute "**_method.expression**" that is present in dictionary definitions to provide the text method expressing the defined item in terms of other defined items.

`Audit` **Applied ONLY in the DDLm Reference Dictionary.** Used to type attributes employed to record the audit definition information (creation date, update version and cross reference codes) of items, categories and files.

`Identify` **Applied ONLY in the DDLm Reference Dictionary.** Used to type attributes that identify an item tag (or part thereof) or external location.

`Describe` Used to type items with values that are descriptive text intended for human interpretation.

`Encode` Used to type items with values that are text or codes that are formatted to be machine parsable.

`State` Used to type items with values that are restricted to codes present in their "enumeration_set.state" lists.

`Key` Used to type an item with a value that is unique within the looped list of these items, and does not contain encoded information.

`Link` Used to type an item that acts as a foreign key between two categories. The definition of the item must additionally contain the attribute "**_name.linked_item_id**" specifying the data name of the item with unique values in the linked category. The values of the defined item are drawn from the set of values in the referenced item. Cross referencing items from the same category is allowed.

`Composite` Used to type items with value strings composed of separate parts. These will usually need to be separated and parsed for complete interpretation and application.

`Measurand` Used to type an item with a numerically estimated value that has been recorded by measurement or derivation. A data item definition for the standard uncertainty (SU) of this item must be provided in a separate definition with **_type.purpose** of 'SU'. The value of a measurand item should be accompanied by a value of its associated SU item, either: 1) integrated with the measurand value in a manner characteristic of the data format; or 2) as a separate, explicit value for the associated SU item. These alternatives are semantically equivalent.

`SU` Used to type an item with a numerical value that is the standard uncertainty of another data item. The definition of an SU item must include the attribute "**_name.linked_item_id**" which explicitly identifies the associated measurand item. SU values must be non-negative.

`Internal` Used to type items that serve only internal purposes of the dictionary in which they appear. The particular purpose served is not defined by this state.

_type.source (*Code*)

The origin or source of the defined data item, indicating by what recording process it has been added to the domain instance.

Where no value is given, the assumed value is '`Assigned`'.

The data value must be one of the following:

`Recorded` A value (numerical or otherwise) recorded by observation or measurement during the experimental collection of data. This item is PRIMITIVE.

`Assigned` A value (numerical or otherwise) assigned as part of the data collection, analysis or modelling required for a specific domain instance. These assignments often represent a decision made that determines the course of the experiment (and therefore may be deemed PRIMITIVE) or a particular choice in the way the data was analysed (and therefore may be considered NOT PRIMITIVE).

`Related` A value or tag used in the construction of looped lists of data. Typically identifying an item whose unique value is the reference key for a loop category and/or an item which has values in common with those of another loop category and is considered a Link between these lists.

`Derived` A quantity derived from other data items within the domain instance. This item is NOT PRIMITIVE.

UNITS

The attributes for specifying units of measure.

_units.code (*Code*)

A code which identifies the units of measurement.

The data value must be one of the following:

`none`	dimensionless - e.g. a ratio, factor, weight or scale
`coulomb`	electronic charge in coulombs
`electron_volts`	electronic charge in electron volts eV
`metres`	length 'metres (metres * 10^(0))'
`centimetres`	length 'centimetres (metres * 10^(-2))'
`millimetres`	length 'millimetres (metres * 10^(-3))'
`micrometres`	length 'micrometres (metres * 10^(-6))'
`nanometres`	length 'nanometres (metres * 10^(-9))'
`angstroms`	length 'angstroms (metres * 10^(-10))'
`picometres`	length 'picometres (metres * 10^(-12))'
`femtometres`	length 'femtometres (metres * 10^(-15))'
`reciprocal_centimetres`	per-length 'reciprocal centimetres (metres * 10^(-2)^-1)'
`reciprocal_millimetres`	per-length 'reciprocal millimetres (metres * 10^(-3)^-1)'
`reciprocal_nanometres`	per-length 'reciprocal nanometres (metres * 10^(-9)^-1)'
`reciprocal_angstroms`	per-length 'reciprocal angstroms (metres * 10^(-10)^-1)'
`reciprocal_angstrom_squared`	per-area 'reciprocal angstroms^2'
`reciprocal_picometres`	per-length 'reciprocal picometres (metres * 10^(-12)^-1)'
`nanometre_squared`	length_squared 'nanometres squared (metres * 10^(-9))^2'

8pi_angstroms_squared	length_squared ’8pi^2 * angstroms squared (metres * 10^(-10))^2’
picometre_squared	length_squared ’picometres squared (metres * 10^(-12))^2’
femtometre_squared	length_squared ’femtometres squared (metres * 10^(-15))^2’
nanometre_cubed	length_cubed ’nanometres cubed (metres * 10^(-9))^3’
angstrom_cubed	length_cubed ’angstroms cubed (metres * 10^(-10))^3’
picometre_cubed	length_cubed ’picometres cubed (metres * 10^(-12))^3’
grams_per_centimetre_cubed	density ’grams per cubic centimetre’
kilograms_per_metre_cubed	density ’kilograms per cubic metre’
megagrams_per_metre_cubed	density ’megagrams per cubic metre’
angstrom_cubed_per_dalton	density ’angstrom cubed per dalton’
millimetres_squared_per_gram	mass absorption ’square millimetres per gram’
centimetres_squared_per_gram	mass absorption ’square centimetres per gram’
kilopascals	pressure ’kilopascals’
gigapascals	pressure ’gigapascals’
hours	time ’hours’
minutes	time ’minutes’
seconds	time ’seconds’
microseconds	time ’microseconds’
degrees	angle ’degrees (of arc)’
cycles	phase ’angle in 360 degree arcs’
radians	angle ’radians’
degrees_squared	angle ’degrees (of arc)’
degree_per_minute	rotation_per-time ’degrees (of arc) per minute’
Celsius	temperature ’degrees (of temperature) Celsius’
kelvins	temperature ’temperature in kelvins’
kelvins_per_minute	cooling rate ’kelvins per minute’
electrons	electrons ’electrons’
electron_squared	electrons-squared ’electrons squared’
electrons_per_nanometre_cubed	electron-density ’electrons per nanometres cubed (electrons * (metres * 10^(-9))^(-3))’
electrons_per_angstrom_cubed	electron-density ’electrons per angstroms cubed (electrons * (metres * 10^(-10))^(-3))’
electrons_per_picometre_cubed	electron-density ’electrons per picometres cubed (electrons * (metres * 10^(-12))^(-3))’
femtometres_per_angstrom_cubed	scattering-length-density ’femtometres per angstroms cubed (10^-6 * (metres * 10^(-10))^(-2))’
dalton	standard atomic mass unit
pixels_per_millimetre	area resolution unit
pixels_per_element	area resolution unit
kilowatts	power ’kilowatts’
milliamperes	current ’milliamperes’
kilovolts	emf ’kilovolts’
volt_squared	emf ’volts squared’
Bohr_magnetons	magnetic moment
arbitrary	arbitrary ’arbitrary system of units’
counts_per_photon	measure of gain used in array detectors
counts	counts from a detector
’photons per second’	photons registered in one second
microseconds_per_angstrom	TOF coefficient ’(seconds * 10^(-6)) * (metres * 10^(-10)^(-1))’
microseconds_per_angstrom_squared	TOF coefficient ’(seconds * 10^(-6)) * (metres * 10^(-10)^(-2))’
microseconds_per_angstrom_cubed	TOF coefficient ’(seconds * 10^(-6)) * (metres * 10^(-10)^(-3))’
angstroms_per_microsecond	TOF coefficient ’(metres * 10^(-10)) * (seconds * 10^(-6)^(-1))’