ANNOTATION

ELEMENTS

abundance*[el.abundance]*

The abundance of an isotope.

The abundance of an isotope in an isotopeList. Values are expressed in percentages.
Example

Content Model of element

[xsd:float]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`min`*[att.min]*
The minimum value allowed for an element or attribute.
[link]

`max`*[att.max]*
Maximum value allowed for an element or attribute.

[link]

units*[att.units]*
Scientific units on an element.
[link]

action*[el.action]*

An action which might occur in scientific data or narrative.

An action which might occur in scientific data or narrative. The definition is deliberately vague, intending to collect examples of possible usage. Thus an action could be addition of materials, measurement, application of heat or radiation. The content model is unrestricted. _action_ iself is normally a child of _actionList_.

The start, end and duration attributes should be interpreted as

- XSD dateTimes and XSD durations. This allows precise recording of time of day, etc, or duration after start of actionList. A `convention="xsd"` attribute should be used to enforce XSD.
- a numerical value, with a units attribute linked to a dictionary.
- a human-readable string (unlikely to be machine processable)

`startCondition` and `endCondition` values are not constrained, which allows XSL-like `test` attribute values. The semantics of the conditions are yet to be defined and at present are simply human readable.

The order of the `action` elements in the document may, but will not always, define the order that they actually occur in.

A delay can be shown by an `action` with no content. Repeated actions or actionLists are indicated through the count attribute.
Example

Content Model of element

(ANY [lax])
*

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

units*[att.units]*
Scientific units on an element.
[link]

start*[att.start]* ATTGROUP2 The start time. The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date. REF maybe

The start time.

The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date.

startCondition*[att.startCondition]* ATTGROUP2 The start condition. This can describe the condition(s) that has to be met before an action can begin, such as in a recipe. Semantics are unexplored but could be used to control robotic operations. REF maybe

The start condition.

This can describe the condition(s) that has to be met before an action can begin, such as in a recipe. Semantics are unexplored but could be used to control robotic operations.

duration*[att.duration]* ATTGROUP2 The duration of the action. Semantics undefined. REF maybe

The duration of the action.

Semantics undefined.

end*[att.end]* ATTGROUP2 The end time. The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date. REF maybe

The end time.

The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date.

endCondition*[att.endCondition]* ATTGROUP2 The end condition. At present a human-readable string describing some condition when the ac tion should end. As XML develops it may be possible to add machine-processable semantics in this field. REF maybe

The end condition.

At present a human-readable string describing some condition when the ac tion should end. As XML develops it may be possible to add machine-processable semantics in this field.

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.

A qualifier which may affect the semantics of the object.

order*[att.actionOrder]*

Describes whether child elements are sequential or parallel.

count*[att.count]*

The count of the object.

[link]

ref*[att.ref]*

A reference to an element of given type.

[link]

actionList*[el.actionList]*

A container for a group of actions.

`ActionList` contains a series of`action`s or nested`actionList`s.

Example

Content Model of element

(ANY [lax])
*

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

start*[att.start]* ATTGROUP2 The start time. The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date. REF maybe

The start time.
The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date.

startCondition*[att.startCondition]* ATTGROUP2 The start condition. This can describe the condition(s) that has to be met before an action can begin, such as in a recipe. Semantics are unexplored but could be used to control robotic operations. REF maybe

The start condition.
This can describe the condition(s) that has to be met before an action can begin, such as in a recipe. Semantics are unexplored but could be used to control robotic operations.

duration*[att.duration]* ATTGROUP2 The duration of the action. Semantics undefined. REF maybe

The duration of the action.
Semantics undefined.

end*[att.end]* ATTGROUP2 The end time. The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date. REF maybe

The end time.
The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date.

endCondition*[att.endCondition]* ATTGROUP2 The end condition. At present a human-readable string describing some condition when the ac tion should end. As XML develops it may be possible to add machine-processable semantics in this field. REF maybe

The end condition.

At present a human-readable string describing some condition when the ac tion should end. As XML develops it may be possible to add machine-processable semantics in this field.

units*[att.units]*
Scientific units on an element.
[link]

count*[att.count]*
The count of the object.
[link]

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.

A qualifier which may affect the semantics of the object.

order*[att.actionOrder]*
Describes whether child elements are sequential or parallel.

alternative*[el.alternative]*

An alternative name for an entry.

At present a child of _entry_ which represents an alternative string that refers to the concept. There is a partial controlled vocabulary in _alternativeType_ with values such as :

- synonym
- acronym
- abbreviation

Example

Content Model of element

[xsd:string]

id*[att.id]*

An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

type*[att.alternativeType]*
The type of an alternative.

amount*[el.amount]*

The amount of a substance.

The `units` attribute is mandatory and can be customised to support mass, volumes, moles, percentages, or rations (e.g. ppm).
Example

Content Model of element

[xsd:float]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

units*[att.units]*
Scientific units on an element.
[link]

angle*[el.angle]*

An angle between three atoms.

It can be used for:

- Recording experimentally determined bond angles (e.g. in a crystallographic paper).
- Providing the angle component for internal coordinates (e.g. z-matrix).

Example

Content Model of element

nonNegativeAngleType[st.nonNegativeAngleType]

A non-signed angle.

Re-used by _angle_. Note that we also provide positiveAngleType (e.g. for cell angles) and torsionAngleType for _torsion_.

Example

[xsd:float]
minInclusive: 0.0
maxInclusive: 180.0

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

atomRefs3*[att.atomRefs3]*

A list of three references to atoms.

[link]

units*[att.angleUnits]*
Restricts units to radians or degrees.
[link]

errorValue*[att.errorValue]*
Value of the error.
[link]

errorBasis*[att.errorBasis]*
Basis of the error estimate.
[link]

min*[att.min]*
The minimum value allowed for an element or attribute.
[link]

max*[att.max]*
Maximum value allowed for an element or attribute.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

annotation*[el.annotation]*

A documentation container similar to annotation in XML Schema.

A documentation container similar to annotation in XML Schema. At present this is experimental and designed to be used for dictionaries, units, etc. One approach is to convert these into XML Schemas when the documentation and appinfo children will emerge in their correct position in the derived schema.

It is possible that this may develop as a useful tool for annotating components of complex objects such as molecules.
Example

Content Model of element

(documentation|appinfo)
*

appinfo*[el.appinfo]*

A container similar to appinfo in XML Schema.

A container for machine processable documentation for an entry. This is likely to be platform and/or language specific. It is possible that XSLT, RDF or XBL will emerge as generic languages. See _annotation_ and _documentation_ for further information.

Example

```
An example in XSLT where an element _foo_ calls a bespoke
template
.
```

Content Model of element

(ANY [lax])
*

Attributes of element

`role`*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

`arg`*[el.arg]*

An argument for a function.

Arguments can be typed and have explicit or free values.

Example

Content Model of element

(
(atom|scalar|array|matrix|expression)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

dataType*[att.dataType]*
The data type of the object.
[link]

array*[el.array]*

A homogenous 1 dimensional array of similar object.

These can be encoded as strings (i.e. XSD-like datatypes) and are concatenated as string content. The size of the array should always be >= 1. The default delimiter is whitespace. The _normalize-space()_ function of XSLT could be used to normalize all whitespace to single spaces and this should not affect the value of the array elements. To extract the elements __java.lang.StringTokenizer__ could be used. If the elements themselves contain whitespace then a different delimiter must be used and is identified through the `delimiter` attribute. This method is mandatory if it is required to represent empty strings. If a delimiter is used it MUST start and end the array - leading and trailing whitespace is ignored. Thus `size+1` occurrences of the delimiter character are required. If non-normalized whitespace is to be encoded (e.g. newlines, tabs, etc) you are recommended to translate it character-wise to XML character entities.

Note that normal Schema validation tools cannot validate the elements of **array** (they

are defined as `string`) However if the string is split, a temporary schema can be constructed from the type and used for validation. Also the type can be contained in a dictionary and software could decide to retrieve this and use it for validation.

When the elements of the `array` are not simple scalars (e.g. **scalar**s with a value and an error, the `scalar`s should be used as the elements. Although this is verbose, it is simple to understand. If there is a demand for more compact representations, it will be possible to define the syntax in a later version.
Example

```
the size attribute is not mandatory but provides a useful
validity
         check):
```

Content Model of element

[xsd:string]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`dataType`*[att.dataType]*
The data type of the object.
[link]

`errorValueArray`*[att.errorValueArray]*
Array of error values.
[link]

`errorBasis`*[att.errorBasis]*

Basis of the error estimate.
[link]

minValueArray*[att.minValueArray]*
Minimum values for numeric _matrix_ or _array.
[link]

maxValueArray*[att.maxValueArray]*
Maximum values for numeric _matrix_ or _array.
[link]

units*[att.units]*
Scientific units on an element.
[link]

delimiter*[att.delimiter]*
A delimiter character for arrays and matrices.
[link]

size*[att.size]*
The size of an array or matrix.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

atom*[el.atom]*

An atom.
Usually within an _atomArray_.
Example

Content Model of element

(
(

The main content model of the atom.

- **name** can be used for atom labels, etc. More than one name can be used if required.
- **scalar** contains any scalar properties of the atom (examples are chemical shift, B-value, etc.) linked through `dictRef` (CmlDictRefType).
- **array** contains any properties of the atom describable by a homogeneous array

linked through `dictRef` (CmlDictRefType).
- **matrix** contains any properties of the atom describable by a homogeneous matrix linked through `dictRef` (CmlDictRefType). An example is the polarizability tensor
- **atomParity** (CmlAtomParityElement) the required way of defining atom-based chirality
- **electron** a away of associating electron(s) with the atom

|name|label|atomType|array|matrix|scalar|atomParity|electron)
*)

Attributes of element

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`count`*[att.count]*
The count of the object.
[link]

`elementType`*[att.elementType]*
The identity of a chemical element.
[link]

`formalCharge`*[att.formalCharge]*
The formalCharge on the object.
[link]

`hydrogenCount`*[att.hydrogenCount]*
Number of hydrogens.
[link]

`isotope`*[att.isotope]* ATTGROUP2 The isotope for an element. A real number describing the isotope. Probably obsolet. REF maybe

The isotope for an element.

A real number describing the isotope. Probably obsolet.

`isotopeNumber`*[att.isotopeNumber]* ATTGROUP2 The integer number for an isotope. The number representing the isotope. By default it does not point to a fuller description of the isotope (use isotopeRef). REF maybe

The integer number for an isotope.

The number representing the isotope. By default it does not point to a fuller description of the isotope (use isotopeRef).

isotopeRef*[att.isotopeRef]* ATTGROUP2 Reference to a fuller description of the isotope. The description may be found in an external collection (e.g. IUPAC) or within the current document. REF maybe

Reference to a fuller description of the isotope.

The description may be found in an external collection (e.g. IUPAC) or within the current document.

Example

isotopeListRef*[att.isotopeListRef]* ATTGROUP2 Reference to a description of the isotopic composition of an atom. Used when more than one atom shares the same isotopic composition (e.g. when H/D have been scrambled over some or all of the atoms in a molecule.. REF maybe

Reference to a description of the isotopic composition of an atom.

Used when more than one atom shares the same isotopic composition (e.g. when H/D have been scrambled over some or all of the atoms in a molecule..

Example

occupancy*[att.occupancy]*

Occupancy for an atom.

[link]

spinMultiplicity*[att.spinMultiplicity]* ATTGROUP2 Spin multiplicity. Normally for a molecule. This attribute gives the spin multiplicity of the molecule and is independent of any atomic information. No default, and it may take any positive integer value (though values are normally between 1 and 5. REF maybe

Spin multiplicity.

Normally for a molecule. This attribute gives the spin multiplicity of the molecule and is independent of any atomic information. No default, and it may take any positive integer value (though values are normally between 1 and 5.

x2*[att.x2]* ATTGROUP2 x2 coordinate for an object. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object. REF maybe

x2 coordinate for an object.

Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object.

y2*[att.y2]* ATTGROUP2 y2 coordinate for an object. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object. REF maybe

y2 coordinate for an object.

Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object.

x3*[att.x3]* ATTGROUP2 The x coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF maybe

The x coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

y3*[att.y3]* ATTGROUP2 The y coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF maybe

The y coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

z3*[att.z3]* ATTGROUP2 The z coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF maybe

The z coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

xFract*[att.xFract]* ATTGROUP2 Fractional x coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF maybe

Fractional x coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

yFract*[att.yFract]* ATTGROUP2 Fractional y coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF maybe

Fractional y coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

zFract*[att.zFract]* ATTGROUP2 Fractional y coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF maybe

Fractional y coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`ref`*[att.ref]*

A reference to an element of given type.

[link]

`role`*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

`atomArray`*[el.atomArray]*

A container for a list of atoms.

A child of _molecule_ and contains _atom_ information. There are two strategies:

- Create individual _atom_ elements under _atomArray_ (in any order). This gives the greatest flexibility but is the most verbose.
- Create `*Array` attributes (e.g. of _elementTypeArrayType_ under _atomArray_. This requires all arrays to be of identical lengths with explicit values for all atoms in every array. This is NOT suitable for complexType atom children such as _atomParity_. It also cannot be checked as easily by schema- and schematron validation. The _atomIDArray_ attribute is mandatory. It is allowed (though not yet recommended) to add _*Array_ children such as _floatArray_

The attributes are directly related to the scalar attributes under _atom_ which should be consulted for more info.

Example

```
Example - these are exactly equivalent representations
```

Content Model of element

(atom*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`elementType`*[att.elementTypeArray]*
The identity of a chemical element.
[link]

`countArray`*[att.countArray]*
Array of object counts.
[link]

`formalCharge`*[att.formalChargeArray]*
An array of formalCharges.
[link]

`hydrogenCount`*[att.hydrogenCountArray]*

Array of hydrogenCounts.
[link]

occupancy*[att.occupancyArray]*
Array of occupancies.
[link]

x2*[att.x2Array]*
array of x2 coordinate.
[link]

y2*[att.y2Array]*
array of y2 coordinate.
[link]

x3*[att.x3Array]*
An array of x3 coordinate.
Normally used in CML2 array mode.
[link]

y3*[att.y3Array]*
An array of y3 coordinate.
Normally used in CML2 array mode.
[link]

z3*[att.z3Array]*
An array of z3 coordinate.
Normally used in CML2 array mode.
[link]

xFract*[att.xFractArray]*
Array of fractional x coordinate.
[link]

yFract*[att.yFractArray]*
Array of fractional y coordinate.
[link]

zFract*[att.zFractArray]*
Array of fractional z coordinate.
[link]

atomID*[att.atomIDArray]*
An array of atom IDs.

[link]

atomicBasisFunction *[el.atomicBasisFunction]*

An atomicBasisFunction.

An atomic atomicBasisFunction which can be linked to atoms, eigenvalues/vectors etc. Normally contained within _basisSet_

Normally these are atom-centered functions, but they can also serve as "ghost" functions which are centered on points. IN CCML these can be dummy atoms so that the atomRef mechanism can still be used.

This information is required to interpret the eignevector components and map them onto the atom list. However this mapping is normally implicit in the program and so it may be necessary to generate `basisSet` information for some programs before XML technology can be automatically used to link the components of the CCML document.
Example

Content Model of element

(
()
*,
(gradient)
?)

Attributes of element

atomRef *[att.atomRef]*
A reference to an atom.
[link]

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id *[att.id]*
An attribute providing a unique ID for an element.
[link]

convention *[att.convention]*
A reference to a convention.

[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

n*[att.n]* ATTGROUP2 The principal quantum number. Takes values 1, 2, 3, etc. REF maybe

The principal quantum number.
Takes values 1, 2, 3, etc.

l*[att.l]* ATTGROUP2 The secondary quantum number. takes values 0, 1, etc. REF maybe

The secondary quantum number.
takes values 0, 1, etc.

m*[att.m]* ATTGROUP2 The azimuthal quantum number. takes values -1, 0, 1, etc. REF maybe

The azimuthal quantum number.
takes values -1, 0, 1, etc.

symbol*[att.symbol]* ATTGROUP2 A symbol. Currently only used on _atomicBasisFunction_. REF maybe

A symbol.
Currently only used on _atomicBasisFunction_.

lm*[att.lm]*
symbolic represention of l amd m.

atomParity*[el.atomParity]*

The stereochemistry round an atom centre.
It follows the convention of the MIF format, and uses 4 distinct atoms to define the chirality. These can be any atoms (though they are normally bonded to the current atom). There is no default order and the order is defined by the atoms in the atomRefs4 attribute. If there are only 3 ligands, the current atom should be included in the 4 atomRefs.

The value of the parity is a signed number. (It can only be zero if two or more atoms are coincident or the configuration is planar). The sign is the sign of the chiral volume

created by the four atoms (a1, a2, a3, a4):

```
|  1  1  1  1 |
| x1 x2 x3 x4 |
| y1 y2 y3 y4 |
| z1 z2 z3 z4 |
```

Note that `atomParity` cannot be used with the *Array syntax for atoms.
Example

Content Model of element

[xsd:float]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`atomRefs4`*[att.atomRefs4]*

A list of 4 references to atoms.

[link]

`atomSet` *[el.atomSet]*

A set of references to atoms.

An atomSet consists of a number of unique references to atoms throught their ids. atomSets need not be related to molecules (which are generally created by aggregation of explicit atoms). Two or more atomSets may reference the same atom, and atomSets may be empty.

atomSets have many potential uses such as:

- identifying functional groups
- results of substructure matching
- identifying atoms with particular roles in a calculation

The atomSet may be referenced from elsewhere in the document and you are encouraged to use locally unique id attributes on atomSets.

Example

Content Model of element

atomRefArrayType[st.atomRefArrayType]

An array of atomRefs.

The atomRefs cannot be schema- or schematron-validated. Instances of this type will be used in array-style representation of bonds and atomParitys. It can also be used for arrays of atomIDTypes such as in complex setereochemistry, geometrical definitions, atom groupings, etc.

Example

XSD:LIST of atomIDType

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.

Example

[xsd:string]

Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.
[link]

---

dictRef *[att.dictRef]*
A reference to a dictionary entry.
[link]

---

size *[att.size]*
The size of an array or matrix.
[link]

atomType *[el.atomType]*

An atomType.

atomTypes are used in a wide variety of ways in computational chemistry. They are normally labels added to existing atoms (or dummy atoms) in the molecule and have a number of defined properties. These properties are usually in addition to those deducible from the elementType of the atom. AtomTypes usually depend on the chemical or geometrical environment of the atom and are frequently assigned by algorithms with chemical perception. However they are often frequently set or "tweaked" by humans initiating a program run.

AtomTypes on an atom have no formal relation to its `elementType`, which only describe the number of protons in the nucleus. It is not unknown (though potentially misleading) to use an "incompatible" atomType to alter the computational properties of an atom (e.g. pretend this K+ is a Ca++ to increase its effective charge). `atomTypes` will also be required to describe pseudoAtoms such as "halogen" (generic) or "methyl group" (unified atom). Atoms in computations can therefore have an `atomTypeRef` attribute.

An atomType contains numeric or other quantities associated with it (charges, masses, use in force-fields, etc.) and also description of any perception algorithms (chemical and/or geometrical) which could be used to compute or constrain it. This is still experimental.

atomTypes are referred to by their mandatory `name` attribute. An atom referes to one or more atomTypes through atomType/@ref children
Example
examples not yet teste.

Content Model of element

(
(molecule|atom|label)
*,

(scalar|array|matrix|property)
*)

Attributes of element

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

ref*[att.ref]*
A reference to an element of given type.
[link]

atomRef*[att.atomRef]*
A reference to an atom.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

atomTypeList*[el.atomTypeList]*

A container for one or more atomTypes.

It can contain several atomTypes.

Example

Content Model of element

(metadataList*,name*,atomType*)

Attributes of element

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

band*[el.band]*

A band or Brillouin zone.
Not yet finalise.
Example

Content Model of element

(array)

Attributes of element

`kpoint`*[att.kpoint]*
The k vector.

[link]

weight*[att.weight]* ATTGROUP2 Weight of the element. Currently the weight of the kPoint, derived from the symmetry such as the inverse of the multiplicity in real space. Thus a point at 0,0,0 in monoclinic space might be 0.25. The lowest value possible is probably 1/48.0 (in m3m). 2003-09-15 (added at suggestion of Jon Wakelin). REF maybe

Weight of the element.

Currently the weight of the kPoint, derived from the symmetry such as the inverse of the multiplicity in real space. Thus a point at 0,0,0 in monoclinic space might be 0.25. The lowest value possible is probably 1/48.0 (in m3m).

2003-09-15 (added at suggestion of Jon Wakelin).

label*[att.label]* ATTGROUP2 A label. The semantics of label are not defined in the schema but are normally commonly used standard or semi-standard text strings. This attribute has the the same semantics as the more common _label_ elemen. REF maybe

A label.

The semantics of label are not defined in the schema but are normally commonly used standard or semi-standard text strings. This attribute has the the same semantics as the more common _label_ elemen.

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

bandList*[el.bandList]*

A container for bands.
Experimental.
Example

Content Model of element

(
(band)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`basisSet`*[el.basisSet]*

A container for one or more atomicBasisFunctions.
This can contain several orbitals.
Example

Content Model of element

(metadataList*,name*,
(atomicBasisFunction)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

bond*[el.bond]*

A bond between atoms, or between atoms and bonds.
bond is a child of _bondArray_ and contains bond information. Bond must refer to at least two atoms (normally using _atomRefs2_) but may also refer to more for multicentre bonds. Bond is often EMPTY but may contain _electron_, _length_ or _bondStereo_ elements.
Example
Validation

Content Model of element

(
(bondType|electron|bondStereo)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`atomRefs2`*[att.atomRefs2]*
References to two different atoms.
[link]

`atomRefs`*[att.atomRefs]*
A reference to a list of atoms.
[link]

`bondRefs`*[att.bondRefs]*
A reference to a list of bonds.
[link]

`order`*[att.order]*
The order of the bond.
[link]

`bondArray`*[el.bondArray]*

A container for a number of bonds.

bondArray is a child of _molecule_ and contains _bond_ information. There are two strategies:

- Create individual `bond` elements under `bondArray` (in any order). This gives the greatest flexibility but is the most verbose.
- Create `*Array` attributes (e.g. of `orderArrayType` under `bondArray`. This requires all arrays to be of identical lengths with explicit values for all bonds in every array. This is NOT suitable for complexType bond children such as _bondStereo_ nor can IDs be added to bonds.. It also cannot be checked as easily by schema- and schematron validation. The _atomRef1Array_ and _atomRef2Array_ attributes are then mandatory. It is allowed (though not yet recommended) to add _*Array_ children such as _floatArray_

The attributes are directly related to the scalar attributes under _atom_ which should be consulted for more info.

Example

```
Example - these are exactly equivalent representations
```

Content Model of element

(bond+|array*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`bondID`*[att.bondIDArray]*

The IDs for an array of bond.
[link]

---

atomRef1*[att.atomRef1Array]*
The first atoms in each bond.
[link]

---

atomRef2*[att.atomRef2Array]*
The second atoms in each bond.
[link]

---

order*[att.orderArray]*
The order of the bond.
[link]

bondSet*[el.bondSet]*

A set of references to bonds.

An bondSet consists of a number of unique references to bonds throught their ids. bondSets need not be related to molecules (which are generally created by aggregation of explicit bonds). Two or more bondSets may reference the same bond, and bondSets may be empty.

bondSets have many potential uses such as:
- identifying functional groups
- results of substructure matching
- identifying bonds with particular roles in a calculation

The bondSet may be referenced from elsewhere in the document and you are encouraged to use locally unique id attributes on bondSets.
Example

Content Model of element

bondRefArrayType[st.bondRefArrayType]

An array of references to bonds.

The references cannot (yet) cannot be schema- or schematron-validated. Instances of this type will be used in array-style representation of electron counts, etc. It can also be used for arrays of bondIDTypes such as in complex stereochemistry, geometrical definitions, bond groupings, etc.
XSD:LIST of bondRefType

A reference to an existing bond.

A reference to a bond may be made by atoms (e.g. for multicentre or pi-bonds),

electrons (for annotating reactions or describing electronic properties) or possibly other bonds (no examples yet). The semantics are relatively flexible.
Example
[xsd:string]
Pattern: [A-Za-z0-9_\-]+(:[A-Za-z0-9_\-]+)?

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`size`*[att.size]*
The size of an array or matrix.
[link]

`bondStereo`*[el.bondStereo]*

A container supporting cis trans wedge hatch and other stereochemistry.

An explict list of atomRefs must be given, or it must be a child of `bond`. There are no implicit conventions such as E/Z. This will be extended to other types of stereochemistry.

At present the following are supported:

- No atomRefs attribute. **Deprecated, but probably unavoidable**. This must be a child of `bond` where it picks up the two atomRefs in the `atomRefs2` attribute. Possible values are C/T (which only makes sense if there is exactly one ligand at each end of the bond) and W/H. The latter should be raplaced by `atomParity` wherever possible. Note that W/H makes no sense without 2D atom coordinates.
- **atomRefs4 attribute**. The 4 atoms represent a cis or trans configuration. This may or may not be a child of `bond`; if so the second and third atomRefs should be identical with the two atomRefs in the bond. This structure can be used to

guide processors in processing stereochemistry and is recommended, since there is general agreement on the semantics. The semantics of `bondStereo` not related to bonds is less clear (e.g. cumulenes, substituted ring nuclei) etc.It is currently an error to have more than one `bondStereo` referring to the same ordered 4-atom list

- **atomRefs attribute**. There are other stereochemical conventions such as cis/trans for metal complexes which require a variable number of reference atoms. This allows users to create their own - at present we do not see CML creating exhaustive tables. For example cis/trans square-planar complexes might require 4 (or 5) atoms for their definition, octahedral 6 or 7, etc. In principle this is very powerful and could supplement or replace the use of *cis*-, *mer*-, etc.

the `atomRefs` and `atomRefs4` attributes cannot be used simultaneously.
Example

Content Model of element

stereoType[st.stereoType]

Bond stereochemistry as a string.

This is purely conventional. There is no default value. The emptyString attribute can be used to indicate a bond of unknown or unspecified type. The interpretation of this is outside the scope of CML-based algorithms. It may be accompanied by a `convention` attribute which links to a dictionary.
Example

Allowed values

- C

  A cis bond.
- T

  A trans bond.
- W

  A wedge bond.
- H

  A hatch bond.
-

  empty or missing.

`atomRefs4`*[att.atomRefs4]*

A list of 4 references to atoms.

[link]

`atomRefArray`*[att.atomRefArray]*

An array of references to atoms.

[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`conventionValue`*[att.conventionValue]* ATTGROUP2 The value of an element with a _convention_. When convention is used this attribute must be present and element content must be empty. REF maybe

The value of an element with a _convention_.

When convention is used this attribute must be present and element content must be empty.

bondType*[el.bondType]*

The type of a bond.

Bond types are used to describe the behaviour of bonds in forcefields, functional groups, reactions and many other domains. They are not as well formalised as atomTypes and we provide less semantic support. BondTypes are referred to by their mandatory _name_ attribute.

Example

Content Model of element

(
(molecule|bond|label)
*,
(scalar|array|matrix|property)
*)

Attributes of element

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

ref*[att.ref]*
A reference to an element of given type.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

bondTypeList*[el.bondTypeList]*

A container for one or more bondTypes.

bondTypeList can contain several bondTypes.
Example

Content Model of element

(metadataList*,name*,
(bondType)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

cml*[el.cml]*

A general container for CML elements.
Often the root of the CML (sub)document. Has no explicit function but serves to hold the dictionaries, namespace, and can alert CML processors and search/XMLQuery tools that there is chemistry in the document. Can contain any content, but usually a list of molecules and other CML components. Can be nested.
Example

Content Model of element

(

No specific restrictions..

, ANY [lax])
*

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

conditionList*[el.conditionList]*

A container for one or more experimental condition.

This can contain several conditions. These include (but are not limited to) intensive physical properties (temperature, pressure, etc.), apparatus (test-tube, rotary evaporator, etc.). Actions can be represented elsewhere by stmml:actionList and solvents or other substances by cml:substanceList.
Example

Content Model of element

(metadataList*,name*,
(scalar|list)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.

Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

crystal*[el.crystal]*

A crystallographic cell.

Required if fractional coordinates are provided for a molecule. There are precisely SIX child scalars to represent the cell lengths and angles in that order. There are no default values; the spacegroup is also included.
Example

Content Model of element

(

All 6 cell parameters must be given, even where angles are fixed by symmetry. The order is fixed as a,b,c,alpha,gamma,beta and software can neglect any title or dictRef attributes. Error estimates can be given if required. Any units can be used, but the defaults are Angstrom (10^-10 m) and degrees.

,scalar{6,6},symmetry?)

Attributes of element

z*[att.z]* ATTGROUP2 The number of molecules per cell. Molecules are defined as the _molecule_ which directly contains the _crystal_ element. REF maybe

The number of molecules per cell.

Molecules are defined as the _molecule_ which directly contains the _crystal_ element.

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

definition*[el.definition]*

The definition for an entry.

The definition should be a short nounal phrase defining the subject of the entry. Definitions should not include commentary, implementations, equations or formulae (unless the subject is one of these) or examples.

The definition can be in any markup language, but normally XHTML will be used, perhaps with links to other XML namespaces such as CML for chemistry.

Example

```
                From the IUPAC Dictionary of Medicinal
Chemistry
```

Content Model of element

(ANY [lax])
*

description*[el.description]*

Descriptive information.

This can occur in objects which require textual comment such as entry.

Entries should have at least one separate **definition**s. `description` is then used for most of the other information, including examples. The `class` attribute has an

uncontrolled vocabulary and can be used to clarify the purposes of the `description` elements.
Example

Content Model of element

(ANY [lax])
*

Attributes of element

`convention`*[att.convention]*
A reference to a convention.
[link]

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`objectClass`*[att.objectClass]* ATTGROUP2 The class of an object. The type of this information. This is not controlled, but examples might include: label summary note usage qualifier It might be used to control display or XSL filtering. The attribute is named 'objectClass' to avoid clashes with other class attributes and inappropriate conversion to foo.getClass(). REF maybe

The class of an object.

The type of this information. This is not controlled, but examples might include:
- label
- summary
- note
- usage
- qualifier

It might be used to control display or XSL filtering.

The attribute is named 'objectClass' to avoid clashes with other class attributes and inappropriate conversion to foo.getClass().

dictionary *[el.dictionary]*

A dictionary.

A dictionary is a container for _entry_ elements. Dictionaries can also contain unit-related information. The dictRef attribute on a dictionary element sets a namespace-like prefix allowing the dictionary to be referenced from within the document. In general dictionaries are referenced from an element using the __dictRef__ attribute.

Example

Content Model of element

(unitList*,annotation*,description*,entry*)

Attributes of element

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id *[att.id]*

An attribute providing a unique ID for an element.

[link]

convention *[att.convention]*

A reference to a convention.

[link]

dictRef *[att.dictRef]*

A reference to a dictionary entry.

[link]

href *[att.href]* ATTGROUP2 address of a resource. Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present. REF maybe

address of a resource.

Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least

it works at present.

dimension*[el.dimension]*

A dimension supporting scientific unit.

This will be primarily used within the definition of units.
Example

Content Model of element

()

Attributes of element

dimensionBasis*[att.dimensionBasis]*
The basis of the dimension.
[link]

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

power*[att.power]*REQUIRED ATTGROUP2 The power to which a dimension should be raised. Normally an integer. Must be included, even if unity. REF maybe

The power to which a dimension should be raised.

Normally an integer. Must be included, even if unity.

preserve*[att.preserve]* ATTGROUP2 Is the dimension preserved during algebra. Experimental. The idea is to support concepts like volume/volume where alebraically these cancel out. preserve="yes" is intending to support preservation during derivation of new unitTypes. REF maybe

Is the dimension preserved during algebra.

Experimental. The idea is to support concepts like volume/volume where alebraically these cancel out. preserve="yes" is intending to support preservation during

derivation of new unitTypes.

`documentation` *[el.documentation]*

Documentation in the annotation of an entry.

A container similar to `documentation` in XML Schema. This is NOT part of the textual content of an entry but is designed to support the transformation of dictionary entrys into schemas for validation. This is experimental and should only be used for dictionaries, units, etc. One approach is to convert these into XML Schemas when the `documentation` and `appinfo` children will emerge in their correct position in the derived schema.

Do NOT confuse documentation with the description or the definition which are part of the content of the dictionary

If will probably only be used when there is significant appinfo in the entry or where the entry defines an XSD-like datatype of an element in the document.
Example

Content Model of element

(ANY [lax])
*

Attributes of element

`id` *[att.id]*
An attribute providing a unique ID for an element.
[link]

`title` *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

`eigen` *[el.eigen]*

An element to hold eigenstuff.

Holds an array of eigenvalues and a matrix of eigenvector.
Example

Content Model of element

(array?,matrix?)

Attributes of element

`units`*[att.units]*
Scientific units on an element.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`type`*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.
A qualifier which may affect the semantics of the object.

`electron`*[el.electron]*

An electron.
Since there is very little use of electrons in current chemical information this is a fluid concept. I expect it to be used for electron counting, input and output of theochem operations, descriptions of orbitals, spin states, oxidation states, etc. Electrons can be associated with atoms, bonds and combinations of these. At present there is no hardcoded semantics. However, _atomRef_ and similar attributes can be used to associate electrons with atoms or bond.

Example

Content Model of element

()

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`atomRef`*[att.atomRef]*
A reference to an atom.
[link]

`atomRefs`*[att.atomRefs]*
A reference to a list of atoms.
[link]

`bondRef`*[att.bondRef]*
A reference to a bond.
[link]

`bondRefs`*[att.bondRefs]*
A reference to a list of bonds.
[link]

`count`*[att.count]*
The count of the object.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

entry*[el.entry]*

A dictionary entry.

The original design for validation with attribute-based constraints is ponderous and fragile. In future constraints will be added through `appinfo` in `annotation`. We shall develop this further in the near future.

2003-03-30: added metadataList to content mode.
Example

Content Model of element

(
(metadataList|alternative|annotation|definition|description|enumeration|relatedEntry)
*)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dataType*[att.dataType]*
The data type of the object.
[link]

rows*[att.rows]*
Number of rows.
[link]

columns*[att.columns]*

Number of columns.
[link]

`unitType`*[att.unitType]* ATTGROUP2 A reference to the type of a unit. Used in defining the unit and doing symbolic algebra on the dimensionality. REF maybe

A reference to the type of a unit.
Used in defining the unit and doing symbolic algebra on the dimensionality.

`minExclusive`*[att.minExclusive]* ATTGROUP2 minimum exclusive value. by analogy with xsd:schema. REF maybe

minimum exclusive value.
by analogy with xsd:schema.

`minInclusive`*[att.minInclusive]* ATTGROUP2 minimum inclusive value. by analogy with xsd:schema. REF maybe

minimum inclusive value.
by analogy with xsd:schema.

`maxExclusive`*[att.maxExclusive]* ATTGROUP2 maximum exclusive value. by analogy with xsd:schema. REF maybe

maximum exclusive value.
by analogy with xsd:schema.

`maxInclusive`*[att.maxInclusive]* ATTGROUP2 minimum inclusive value. by analogy with xsd:schem. REF maybe

minimum inclusive value.
by analogy with xsd:schem.

`totalDigits`*[att.totalDigits]* ATTGROUP2 total digits in a scalar. based on xsd:schema. REF maybe

total digits in a scalar.
based on xsd:schema.

`fractionDigits`*[att.fractionDigits]* ATTGROUP2 Number of digits after the point. This is used in dictionaries to define precision. However it might be replaced by xsd:facet. REF maybe

Number of digits after the point.
This is used in dictionaries to define precision. However it might be replaced by xsd:facet.

length*[att.length]* ATTGROUP2 Length of a scalar. Probably will be replaced with xsd:schema tool. REF maybe

Length of a scalar.

Probably will be replaced with xsd:schema tool.

minLength*[att.minLength]* ATTGROUP2 minimum length of a scalar. by analogy with xsd:schema. REF maybe

minimum length of a scalar.

by analogy with xsd:schema.

maxLength*[att.maxLength]* ATTGROUP2 maximum length of a scalar. by analogy with xsd:schem. REF maybe

maximum length of a scalar.

by analogy with xsd:schem.

units*[att.units]*

Scientific units on an element.

[link]

whiteSpace*[att.whiteSpace]* ATTGROUP2 Whitespace. Attached to entry. This may be obsolete. REF maybe

Whitespace.

Attached to entry. This may be obsolete.

pattern*[att.pattern]* ATTGROUP2 Pattern constraint. Based on xsd:schema. REF maybe

Pattern constraint.

Based on xsd:schema.

term*[att.term]*REQUIRED ATTGROUP2 A term in a dictionary. The term should be a noun or nounal phrase, with a separate definition and further description. REF maybe

A term in a dictionary.

The term should be a noun or nounal phrase, with a separate definition and further description.

enumeration*[el.enumeration]*

An enumeration of value.

An enumeration of string values. Used where a dictionary entry constrains the possible values in a document instance. The dataTypes (if any) must all be identical and are defined by the dataType of the containing element.

Example

Content Model of element

(annotation?)

Attributes of element

`value`*[att.value]* ATTGROUP2 Value of a scalar object. The value must be consistent with the dataType of the object. REF maybe

Value of a scalar object.

The value must be consistent with the dataType of the object.

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`default`*[att.default]* ATTGROUP2 default value in an enumeration. A non-whitespace string (value is irrelevant) indicates that the content of this enumeration is the default value (usually of a scalar). It is an error to have more than one default. If the scalar in an instance document has no value (i.e. is empty or contains only whitespace) its value is given by the default. If the scalar in the instance is empty and no enumerations have a default attribute, an application may throw an error. REF maybe

default value in an enumeration.

A non-whitespace string (value is irrelevant) indicates that the content of this enumeration is the default value (usually of a scalar). It is an error to have more than one default. If the scalar in an instance document has no value (i.e. is empty or contains only whitespace) its value is given by the default. If the scalar in the instance is empty and no enumerations have a default attribute, an application may throw an error.

`expression`*[el.expression]*

An expression that can be evaluated.

Experimental. This is essentially a mathematical function, expressed currently in reverse Polish notation but we expect to move to MathML.

Example

Content Model of element

(
(parameter|operator)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`dataType`*[att.dataType]*

The data type of the object.

[link]

`formula`*[el.formula]*

A molecular formula.

It is defined by `atomArray`s each with a list of elementTypes and their counts (or default=1). All other information in the `atomArray` is ignored. `formula` are nestable so that aggregates (e.g. hydrates, salts, etc.) can be described. CML does not require that formula information is consistent with (say) crystallographic information; this allows for experimental variance.

An alternative briefer representation is also available through the `conciseForm`. This must include whitespace round all elements and their counts, which must be explicit.

Example

Content Model of element

(
(formula|atomArray)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`count`*[att.count]*

The count of the object.

[link]

`formalCharge`*[att.formalCharge]*

The formalCharge on the object.

[link]

`concise`*[att.concise]*

A concise formula.

[link]

`gradient`*[el.gradient]*

A gradient.

A container for a quantity or quantities representing the gradient of other quantities. At present just takes a scalar child.

Example

Content Model of element

(
(scalar|array|matrix|property)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`identifier` *[el.identifier]*

A structured identifier.

Supports compund identifiers such as IChI. At present uses the V0.9 IChI XML representation verbatim but will almost certainly change with future IChIs.

The inclusion of elements from other namespaces causes problems with validation. The content model is deliberately LAX but the actual elements in IChI will fail the validation as they are not declared in CML.
For simple scalar values the value attribute can be used with empty content. Where an identifier has several components a series of label elements can be used.

2003-07-10: Fixed count on identifier children..

2003-03-12: Added isotopic and atoms..

Example

Content Model of element

(ANY [lax])
*

Attributes of element

`value`*[att.value]* ATTGROUP2 Value of a scalar object. The value must be consistent with the dataType of the object. REF maybe

Value of a scalar object.

The value must be consistent with the dataType of the object.

`version`*[att.version]* ATTGROUP2 The version of the identifier. The IChI or other identifier may be dependent on the date of release and this attribute is highly recommended. REF maybe

The version of the identifier.

The IChI or other identifier may be dependent on the date of release and this attribute is highly recommended.

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`tautomeric`*[att.tautomeric]* ATTGROUP2 Indicates whether the structure is a tautomer. Currently used with IChI _identifier_ element. Semantics, vocabulary and usage are application-dependent. REF maybe

Indicates whether the structure is a tautomer.

Currently used with IChI _identifier_ element. Semantics, vocabulary and usage are

application-dependent.

isotope*[el.isotope]*

A specific isotope.

Defines an isotope in terms of exact mass and spin. Differentiate from isotopeList which defines a mixture of isotope.

Example

Content Model of element

(abundance?)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

number*[att.number]* ATTGROUP2 A number determined by context Used for isotope number in isotope, and rotational symmetry number in symmetry for calculation of entropy, etc. 2003-03-30: added number attribut. REF maybe

A number determined by context

Used for isotope number in isotope, and rotational symmetry number in symmetry for calculation of entropy, etc.

2003-03-30: added number attribut.

spin*[att.spin]*

The spin of a system.

[link]

elementType*[att.elementType]*
The identity of a chemical element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

isotopeList*[el.isotopeList]*

A container for one or more isotopes.

Can contain several isotopes. These may be related in several ways. This allows the definition of natural abundance and averged enrichment.
Example

Content Model of element

(isotope*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

label *[el.label]*

A text string qualifying an object.

A label can be used to identify or distinguish elements, add keywords or classifications and similar processes. It is usually interpretable by domain-aware humans (e.g. C3'-endo, but not a34561). It is usually either built in a semantically rich fashion (e.g. C2'-alpha-H) or belongs to a controlled vocabulary. It is possibly accessed by software in a domain-specific manner. It differs from `description` which is free text. The distinction between titles, names and labels is fuzzy, but we think this is worth making. Labels may be necesssary to identify objects within programs, while names are more likely to be reserved for database searches. Titles are likely to be freer text and not recommended for precise object retrieval.

Labels should not contain whitespace. Punctuation marks are often necessary, but should not be gratuitously used. Punctuation clashing with XML character entities should be avoided; if this is not possible it should be escaped.

Example

From IUPAC Dictionary of Medicinal Chemistry

Content Model of element

(ANY [lax])
*

Attributes of element

id *[att.id]*
An attribute providing a unique ID for an element.
[link]

dictRef *[att.dictRef]*
A reference to a dictionary entry.
[link]

value *[att.value]* ATTGROUP2 Value of a scalar object. The value must be consistent with the dataType of the object. REF maybe

Value of a scalar object.

The value must be consistent with the dataType of the object.

objectClass *[att.objectClass]* ATTGROUP2 The class of an object. The type of this information. This is not controlled, but examples might include: label summary note usage qualifier It might be used to control display or XSL filtering. The attribute is named 'objectClass' to avoid clashes with other class attributes and inappropriate

conversion to foo.getClass(). REF maybe

The class of an object.

The type of this information. This is not controlled, but examples might include:

- label
- summary
- note
- usage
- qualifier

It might be used to control display or XSL filtering.

The attribute is named 'objectClass' to avoid clashes with other class attributes and inappropriate conversion to foo.getClass().

`lattice`*[el.lattice]*

A lattice of dimension 3 or less.

Lattice is a general approach to describing periodic systems. It can have variable dimensionality or periodicity, and could be finite.

lattice is more general than _crystal_ in cmlCore which is used primarily for reporting crystallographic experiments.`A lattice can be described by latticeVectors, cell axes and angles, or metric tensors, etc. (only axes/angles are allowed under `crystal`). The dimensionality is enforced through a _system_ parent element.

Example

Content Model of element

(
(scalar{3,6}|latticeVector{1,3}|matrix)
,symmetry?)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

dictRef *[att.dictRef]*
A reference to a dictionary entry.
[link]

latticeType *[att.latticeType]*
The primitivity of a lattice.
[link]

spaceType *[att.spaceType]*
The spaceType of the lattice.
[link]

latticeVector *[el.latticeVector]*

A vector3 representing a lattice axis.

a `lattice` can be represented by 1-3 non-linearly dependent latticeVectors. If the dimensionality is less than 3 latticeVectors are the preferred method. Similarly, if the axes show a mixture of periodicity and non-periodicity latticeVectors can support this. The number of periodic vectors must correspond with the periodicity attribute on a `system` element.

The vector must not be zero and units must be given. (Zero vectors must not be used to reduce dimensionality).

A lattice vector defaults to periodic.
.
Any or all of the axes may be periodic or aperiodic. An example could be a surface where 2 periodic axes (not necessarily orthogonal) are used to describe the coordinates in the surface, perhaps representing lattice vectors of a 3D crystal or 2D layer. The third vector is orthogonal and represents coordinates normal to the surface. In this case only the direction, not the magnitude of the vector is importan.
Example

Content Model of element

vector3Type[st.vector3Type]

A vector in 3-space.

No constraints on magnitude (i.e. could be zero.
Example
BASE:
XSD:LIST of xsd:float
xsd:length
3

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

units*[att.units]*
Scientific units on an element.
[link]

periodic*[att.periodic]*Default:true ATTGROUP2 Is the axis periodic. Any or all of the axes may be periodic or aperiodic. An example could be a surface where 2 periodic axes (not necessarily orthogonal) are used to describe the coordinates in the surface, perhaps representing lattice vectors of a 3D crystal or 2D layer. The third vector is orthogonal and represents coordinates normal to the surface. In this case only the direction, not the magnitude of the vector is important. REF maybe

Is the axis periodic.

Any or all of the axes may be periodic or aperiodic. An example could be a surface where 2 periodic axes (not necessarily orthogonal) are used to describe the coordinates in the surface, perhaps representing lattice vectors of a 3D crystal or 2D layer. The third vector is orthogonal and represents coordinates normal to the surface. In this case only the direction, not the magnitude of the vector is important.

length*[el.length]*

A length between two atoms.

This is either an experimental measurement or used to build up internal coordinates (as in a z-matrix) (only one allowed). We expect to move length as a child of _molecule_ and remove it from here.
Example

Content Model of element

[xsd:float]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`atomRefs2`*[att.atomRefs2]*

References to two different atoms.

[link]

`units`*[att.units]*

Scientific units on an element.

[link]

`errorValue`*[att.errorValue]*

Value of the error.

[link]

`errorBasis`*[att.errorBasis]*

Basis of the error estimate.

[link]

`min`*[att.min]*

The minimum value allowed for an element or attribute.

[link]

`max`*[att.max]*

Maximum value allowed for an element or attribute.

[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

line3*[el.line3]*

A line in 3-space.

A line characterised by one or two endpoints.

Content Model of element

line3Type[st.line3Type]

An unbounded line in 3-space.

Defined by 6 real numbers, conventionally an arbitrary point on the line and a vector3. There is no significance to the point (i.e. it is not the "end of the line") and there are an infinite number of ways of representing the line.

Example

BASE:
XSD:LIST of xsd:float
xsd:length
6

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

units*[att.units]*

Scientific units on an element.
[link]

`link` *[el.link]*

An internal or external link to other objects.

Semantics are similar to XLink, but simpler and only a subset is implemented. This is intended to make the instances easy to create and read, and software relatively easy to implement. The architecture is:

- **A single element (`link`) used for all linking purposes.**
- **The link types are determined by the `type` attribute and can be:**.
  - **locator**. This points to a single target and must carry either a `ref` or `href` attribute. `locator` links are usually children of an extended link.
    - **arc**. This is a 1:1 link with both ends (`from` and `to`) defined.
    - **extended**. This is usually a parent of several locator links and serves to create a grouping of link ends (i.e. a list of references in documents).

    Many-many links can be built up from arcs linking extended elements

    All links can have optional `role` attributes. The semantics of this are not defined; you are encouraged to use a URI as described in the XLink specification.

    There are two address spaces:
    - The `href` attribute on locators behaves in the same way as `href` in HTML and is of type `xsd:anyURI`. Its primary use is to use XPointer to reference elements outside the document.
    - The `ref` attribute on locators and the `from` and `to` attributes on `arc`s refer to IDs (*without* the '#' syntax).

    Note: several other specific linking mechanisms are defined elsewhere in STM. **relatedEntry** should be used in dictionaries, and **dictRef** should be used to link to dictionaries. There are no required uses of `link` in STMML but we have used it to map atoms, electrons and bonds in reactions in CML

Relation to XLink. At present (2002) we are not aware of generic XLink processors from which we would benefit, so the complete implementation brings little extra value. Among the simplifications from Xlink are:

- `type` supports only `extended`, `locator` and `arc`
- `label` is not supported and `id`s are used as targets of links.
- `show` and `actuate` are not supported.
- `xlink:title` is not supported (all STM elements can have a `title` attribute).
- `xlink:role` supports any string (i.e. does not have to be a namespaced resource). This mechanism can, of course, still be used and we shall promote it where STM benefits from it
- The `to` and `from` attributes point to IDs rather than labels
- The xlink namespace is not used

- It is not intended to create independent linkbases, although some collections of links may have this property and stand outside the documents they link to

Content Model of element

(ANY)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`from`*[att.from]*
The base of a link.
[link]

`to`*[att.to]*
The target of a link.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`role`*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

href*[att.href]* ATTGROUP2 address of a resource. Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present. REF maybe

address of a resource.

Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present.

linkType*[att.linkType]*
The type of the link.

A container for locators.

A link to an element.

A labelled link.

list*[el.list]*

A generic container with no implied semantics.

A generic container with no implied semantics. It just contains things and can have attributes which bind conventions to it. It could often act as the root element in an STM document.
Example

Content Model of element

(ANY [lax])

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.

[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.

A qualifier which may affect the semantics of the object.

map*[el.map]*

A container for links

There has been some confusion between map and link. At present we are trying to develop link as the primary link and map as the container.

Content Model of element

(link)
*

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

`matrix`*[el.matrix]*

A rectangular matrix of any quantities.

By default `matrix` represents a rectangular matrix of any quantities representable as XSD or STMML dataTypes. It consists of `rows*columns` elements, where `columns` is the fasting moving index. Assuming the elements are counted from 1 they are ordered
`V[1,1],V[1,2],...V[1,columns],V[2,1],V[2,2],...V[2,columns],...V[rows,1],V[rows,2],...V[rows,columns]`

By default whitespace is used to separate matrix elements; see **array** for details. There are NO characters or markup delimiting the end of rows; authors must be careful!. The `columns` and `rows` attributes have no default values; a row vector requires a `rows` attribute of 1.

`matrix` also supports many types of square matrix, but at present we require all elements to be given, even if the matrix is symmetric, antisymmetric or banded diagonal. The `matrixType` attribute allows software to validate and process the type of matrix.

Example

Content Model of element

[xsd:string]

`dataType`*[att.dataType]*
The data type of the object.
[link]

`delimiter`*[att.delimiter]*
A delimiter character for arrays and matrices.
[link]

`rows`*[att.rows]*
Number of rows.
[link]

`columns`*[att.columns]*
Number of columns.
[link]

`units`*[att.units]*
Scientific units on an element.

[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`matrixType`*[att.matrixType]*
Type of matrix.
[link]

`errorValueArray`*[att.errorValueArray]*
Array of error values.
[link]

`errorBasis`*[att.errorBasis]*
Basis of the error estimate.
[link]

`minValueArray`*[att.minValueArray]*
Minimum values for numeric _matrix_ or _array.
[link]

`maxValueArray`*[att.maxValueArray]*
Maximum values for numeric _matrix_ or _array.
[link]

`mechanism`*[el.mechanism]*

The mechanism of a reaction.

In some cases this may be a simple textual description or reference within a controlled vocabulary. In others it may describe the complete progress of the reaction, including topological or cartesian movement of atoms, bonds and electrons and annotation with varying quantities (e.g. energies).

For named reaction mechanisms ("Diels-Alder", "ping-pong", "Claisen rearrangement", etc.) the `name` element should be used. For classification (e.g. "hydrolysis"), the `label` may be more appropriate.

In more detailed cases the mechanism refers to components of the `reaction` element. Thus bond23 might be cleaved while bond19 is transformed (mapped) to bond99. The `mechanismComponent` can be used to refer to components and add annotation. This is still experimental.

IUPAC Compendium of Chemical Terminology 2nd Edition (1997) describes a mechanism as: A detailed description of the process leading from the reactants to the products of a reaction, including a characterization as complete as possible of the composition, structure, energy and other properties of reaction intermediates, products and transition states. An acceptable mechanism of a specified reaction (and there may be a number of such alternative mechanisms not excluded by the evidence) must be consistent with the reaction stoichiometry, the rate law and with all other available experimental data, such as the stereochemical course of the reaction. Inferences concerning the electronic motions which dynamically interconvert successive species along the reaction path (as represented by curved arrows, for example) are often included in the description of a mechanism. It should be noted that for many reactions all this information is not available and the suggested mechanism is based on incomplete experimental data. It is not appropriate to use the term mechanism to describe a statement of the probable sequence in a set of stepwise reactions. That should be referred to as a reaction sequence, and not a mechanism.

CMLReact provides reactionScheme and annotions to describe the reaction sequence and both it and `mechanism` could co-occur within a reactionScheme container.

Example

Content Model of element

(name*,label*,description*,mechanismComponent*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

mechanismComponent*[el.mechanismComponent]*

An information component within a reaction mechanism.

Information components can represent both physical constituents of the reaction or abstract concepts (types of bond cleavage, thermodynamics, etc.). There are several ways that components of the reaction can be annotated and/or quantified. One approach will be to refer to specific bonds and atoms through their ids and use mechanismComponent to describe their role, properties, etc. Another is to use mechanismComponent to identify types of bond formed/broken without reference to actual atoms and bonds (initially through the name element). Yet another will be to include information on the reaction profile.

This is still experimental.
Example

Content Model of element

(ANY [lax])

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*

A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

metadata*[el.metadata]*

A general container for metadata.

A general container for metadata, including at least Dublin Core (DC) and CML-specific metadata

In its simple form each element provides a name and content in a similar fashion to the `meta` element in HTML. `metadata` may have simpleContent (i.e. a string for adding further information - this is not controlled).
Example

Content Model of element

[xsd:string]

content*[att.content]* ATTGROUP2 content of metadata. REF maybe

content of metadata.

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

name*[att.metadataType]*
The metadata type.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

metadataList *[el.metadataList]*

A general container for metadata elements.

MetadataLists can have local roles (e.g. a bibliographic reference could be a single meteadatList with, say, 3-6 components). The role attribute is used in an uncontrolled manner for this. MetadataLists can also be nested, but metadata and metadataList children should not occur on the same level of the hierarchy.

Example

Content Model of element

(metadataList|metadata)
*

Attributes of element

id *[att.id]*

An attribute providing a unique ID for an element.

[link]

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

convention *[att.convention]*

A reference to a convention.

[link]

dictRef *[att.dictRef]*

A reference to a dictionary entry.

[link]

name *[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

module*[el.module]*

A module in a calculation.

Many programs are based on discrete modules which produce chunks of output. There are also conceptual chunks such as initialisation, calculation and summary/final which often have finer submodules such as cycle, iteration, snapshot, etc. There is no controlled vocabulary but a typical structure is shown in the example. One of the challenges of CCML is to find communality between different programs and to use agreed abstractions for the modules.

Example

Content Model of element

(ANY [lax])

Attributes of element

serial*[att.serial]* ATTGROUP2 Serial number or other id. Currently only on module. Modules with the same _role_ attribute can be distinguished by _serial_. This is often an integer but other schemes may be used. REF maybe

Serial number or other id.

Currently only on module. Modules with the same _role_ attribute can be distinguished by _serial_. This is often an integer but other schemes may be used.

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

molecule*[el.molecule]*

A container for atoms, bonds and submolecules.

molecule is a container for atoms, bonds and submolecules along with properties such as crystal and non-builtin properties. It should either contain molecule or *Array for atoms and bonds. A molecule can be empty (e.g. we just know its name, id, etc.)

"Molecule" need not represent a chemically meaningful molecule. It can contain atoms with bonds (as in the solid-sate) and it could simply carry a name (e.g. "taxol") without formal representation of the structure. It can contain "sub molecules", which are often discrete subcomponents (e.g. guest-host).

Molecule can contain a <list> element to contain data related to the molecule. Within this can be string/float/integer and other nested lists
Example

Revised content model to allow any order of lengths, angles, torsions 2003-01-01..

Added role attribute 2003-03-19..

Content Model of element

The float|integer|string children are for compatibility with CML-1 and are deprecated. scalar|array|matrix should be used instead.

(metadataList*,
(formula|name|label|identifier|symmetry)
*,crystal?,
(molecule*|
(atomArray,bondArray?,electron*,
(zMatrix|length|angle|torsion)

*)
)
*,
(scalar|array|matrix|list|propertyList)
*)

Attributes of element

The float|integer|string children are for compatibility with CML-1 and are deprecated. scalar|array|matrix should be used instead.

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`formula`*[att.formula]*
Simple chemical formula.
[link]

`count`*[att.count]*
The count of the object.
[link]

`chirality`*[att.chirality]*
The chirality of a system or molecule.
[link]

formalCharge*[att.formalCharge]*
The formalCharge on the object.
[link]

---

spinMultiplicity*[att.spinMultiplicity]* ATTGROUP2 Spin multiplicity. Normally for a molecule. This attribute gives the spin multiplicity of the molecule and is independent of any atomic information. No default, and it may take any positive integer value (though values are normally between 1 and 5. REF maybe

Spin multiplicity.

Normally for a molecule. This attribute gives the spin multiplicity of the molecule and is independent of any atomic information. No default, and it may take any positive integer value (though values are normally between 1 and 5.

---

symmetryOriented*[att.symmetryOriented]* ATTGROUP2 Is the molecule oriented to the symmetry No formal default, but a molecule is assumed to be oriented according to any _symmetry_ children. This is required for crystallographic data, but some systems for isolated molecules allow specification of arbitrary Cartesian or internal coordinates, which must be fitted or refined to a prescribed symmetry. In this case the attribute value is false. REF maybe

Is the molecule oriented to the symmetry

No formal default, but a molecule is assumed to be oriented according to any _symmetry_ children. This is required for crystallographic data, but some systems for isolated molecules allow specification of arbitrary Cartesian or internal coordinates, which must be fitted or refined to a prescribed symmetry. In this case the attribute value is false.

---

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

name*[el.name]*

A string identifying a object.

`name` is used for chemical names (formal and trivial) for molecules and also for identifiers such as CAS registry and RTECS. It can also be used for labelling atoms. It should be used in preference to the `title` attribute because it is repeatable and can be linked to a dictionary.

Constraining patterns can be described in the dictionary and used to validate `name`s.

Example

Content Model of element

[xsd:string]

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`object`*[el.object]*

An object which might occur in scientific data or narrative.

Deliberately vague. Thus an instrument might be built from sub component objects, or a program could be composed of smaller modules (objects). `object` could be used to encapsulate graphical primitives (e.g. in reaction schemes, drawings of apparatus, etc.). Unrestricted content model.

Example

Content Model of element

(ANY [lax])
*

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.

[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.

A qualifier which may affect the semantics of the object.

count*[att.count]*
The count of the object.
[link]

observation*[el.observation]*

An observation or occurrence.

A container for any events that need to be recorded, whether planned or not. They can include notes, measurements, conditions that may be referenced elsewhere, etc. There are no controlled semantics.
Example

Content Model of element

(ANY [lax])
*

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.
A qualifier which may affect the semantics of the object.

count*[att.count]*
The count of the object.
[link]

operator*[el.operator]*

An operator within an expression.
Experimental. An operator acts on one or more arguments (at present the number is fixed by the type). The formulation is reverse Polish so the result (with its dataType) is put on a stack for further use.
Example

Content Model of element

(
()
*)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.
A qualifier which may affect the semantics of the object.

parameter *[el.parameter]*

A parameter describing the computation.

A parameter is a broad concept and can describe numeric quantities, objects, keywords, etc. The distinction between keywords and parameters is often fuzzy. ("MINIM" might mean "minimize", while "MINIM=3" might require three iterations to be run. It may help to think of control keywords as boolean parameters.

Numeric parameters can describe values in molecules, forcefields or other objects. Often the parameters will be refined or otherwise varied during the calculation. Some parameters may be fixed at particulat values or relaxed at different stages in the calculation. Parameters can have errors, gradients and other indications of uncertainty.

String/character parameters are often abbreviated in program input, and this is supported through the `regex` and `ignoreCase` attributes.

Parameters will usually be defined separately from the objects and use the `ref` attribute to reference them.

Parameters can be used to describe additional constraints. This will probably require the development of a microlanguage and until then may use program-specific mecxhanisms. A common approach will be to use an array of values (or objects) to represent different input values for (parts of) the calculation. Thus a conformational change could be specified by an array of several torsion angles.

A parameter will frequently have a `dictRef` pointing to a dictionary which may have

more information about how the parameter is to be used or the values it can take.

The allowable content of `parameter`s may be shown by a "template" in the `appinfo`; this is stil experimental.
Example

Content Model of element

(
(scalar|array|matrix|property|expression)
*,
(gradient)
?)

Attributes of element

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`value`*[att.value]* ATTGROUP2 Value of a scalar object. The value must be consistent with the dataType of the object. REF maybe

Value of a scalar object.
The value must be consistent with the dataType of the object.

`constraint`*[att.constraint]* ATTGROUP2 Constraint on a parameter. Semantics not yet finalised. We anticipate "fixed", "none" and symbolic relationships to other

parameters. REF maybe

Constraint on a parameter.

Semantics not yet finalised. We anticipate "fixed", "none" and symbolic relationships to other parameters.

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

parameterList*[el.parameterList]*

A container for one or more parameters.

parameterList can contain several parameters.
Example

Content Model of element

(metadataList*,name*,
(parameter)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

particle*[el.particle]*

An object in space carrying a set of properties.
particles have many of the characteristics of atoms but without at atomic nucleus. It does not have an elementType and cannot be involved in bonding, etc. It has coordinates, may carry charge and might have a mass. It represents some aspect of a computational model and should not be used for purely geometrical concepts such as centroid. Examples of particles are "shells" (e.g. in GULP) which are linked to atoms for modelling polarizability or lonepairs and approximations to multipoles. Properties such as charge, mass should be scalar/array/matrix children.

Content Model of element

(ANY [lax])

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.

A qualifier which may affect the semantics of the object.

x3*[att.x3]* ATTGROUP2 The x coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF maybe

The x coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

y3*[att.y3]* ATTGROUP2 The y coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF maybe

The y coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

z3*[att.z3]* ATTGROUP2 The z coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF maybe

The z coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

peak*[el.peak]*

A peak; annotated by human or machine.

A peak can describe:

- A single point in a spectrum. Usually a maximum but could be a shoulder,

inflexion or indeed any point of interest.
- A continuous range of values within a spectrum, defined by maximum and minimum values on either/both axes

The units should always be given. (The raw spectral data may unfortunately use different units and no assumptions should be made).

Example

Content Model of element

(metadataList?,
(atom|bond|molecule)
*)

Attributes of element

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`ref`*[att.ref]*

A reference to an element of given type.

[link]

`peakHeight`*[att.peakHeight]* ATTGROUP2 Height of a peak. For 1-dimensional data (e.g. y vs x) hould use the same units as the appropriate axis (e.g. y). REF maybe

Height of a peak.

For 1-dimensional data (e.g. y vs x) hould use the same units as the appropriate axis (e.g. y).

`peakMultiplicity`*[att.peakMultiplicity]*

Multiplicity of a peak.

A single maximum within the peak rang.

Two maxima (not necessarily equal) within the peak rang.

Three maxima (not necessarily equal) within the peak rang.

Four maxima (not necessarily equal) within the peak rang.

Five maxima (not necessarily equal) within the peak rang.

Six maxima (not necessarily equal) within the peak rang.

Several maxima (not necessarily equal) within the peak rang.

---

peakShape*[att.peakShape]*

Shape of a peak.

A sharp peak.

A broad peak.

A brodening of a peak suggesting the presence of a smaller incompletely resolved component.

User contributed vocabulary of type foo:bar.

---

integral*[att.integral]* ATTGROUP2 Area under a peak. Unfortunately units are usually arbitrary and not related to the x- and y- axis units, and in this case _peakUnits_ should be use. REF maybe

Area under a peak.

Unfortunately units are usually arbitrary and not related to the x- and y- axis units, and in this case _peakUnits_ should be use.

---

peakUnits*[att.peakUnits]*

Units for a peak or peak integral.

[link]

---

xMin*[att.xMin]* ATTGROUP2 Minimum yValue. Annotates x-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMax_ attribute but if so xMin should be less than or equals to it. REF maybe

Minimum yValue.

Annotates x-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMax_ attribute but if so xMin should be less than or equals to it.

---

xMax*[att.xMax]* ATTGROUP2 Maximum yValue. Annotates x-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMin_ attribute but if so xMax should

be greater than or equals to it. REF maybe

Maximum yValue.

Annotates x-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMin_ attribute but if so xMax should be greater than or equals to it.

xValue*[att.xValue]* ATTGROUP2 Value along an x axis. Annotates x-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. REF maybe

Value along an x axis.

Annotates x-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data.

xWidth*[att.xWidth]* ATTGROUP2 An unsigned interval along an x axis. It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses xUnits or the same units as the data. REF maybe

An unsigned interval along an x axis.

It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses xUnits or the same units as the data.

xUnits*[att.xUnits]*

Units for x axis.

[link]

yMin*[att.yMin]* ATTGROUP2 Minimum yValue. Annotates y-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMax_ attribute but if so yMin should be less than or equal to it. REF maybe

Minimum yValue.

Annotates y-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMax_ attribute but if so yMin should be less than or equal to it.

yMax*[att.yMax]* ATTGROUP2 Maximum yValue. Annotates y-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMin_ attribute but if so yMax should be greater than or equals to it. REF maybe

Maximum yValue.

Annotates y-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMin_ attribute but if so yMax should be greater than or equals to it.

yValue*[att.yValue]* ATTGROUP2 Value along a y axis. Annotates y-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. REF maybe

Value along a y axis.

Annotates y-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data.

yWidth*[att.yWidth]* ATTGROUP2 An unsigned interval along a y axis. It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses yUnits or the same units as the data. REF maybe

An unsigned interval along a y axis.

It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses yUnits or the same units as the data.

yUnits*[att.yUnits]*
Units for y axis.
[link]

peakGroup*[el.peakGroup]*

A list of closely related peaks or peakGroups.

Distinguish between `peakList` (primarily a navigational container) and `peakGroup` where the peaks (or groups) have some close relation not shared by all peaks. All descendants must use consistent units.
Example

Content Model of element

(metadataList?,
(peak|peakGroup|atom|bond|molecule)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

peakHeight*[att.peakHeight]* ATTGROUP2 Height of a peak. For 1-dimensional data (e.g. y vs x) hould use the same units as the appropriate axis (e.g. y). REF maybe

Height of a peak.
For 1-dimensional data (e.g. y vs x) hould use the same units as the appropriate axis (e.g. y).

peakMultiplicity*[att.peakMultiplicity]*
Multiplicity of a peak.
A single maximum within the peak rang.
Two maxima (not necessarily equal) within the peak rang.
Three maxima (not necessarily equal) within the peak rang.
Four maxima (not necessarily equal) within the peak rang.
Five maxima (not necessarily equal) within the peak rang.
Six maxima (not necessarily equal) within the peak rang.
Several maxima (not necessarily equal) within the peak rang.

peakShape*[att.peakShape]*
Shape of a peak.
A sharp peak.
A broad peak.
A brodening of a peak suggesting the presence of a smaller incompletely resolved component.
User contributed vocabulary of type foo:bar.

integral*[att.integral]* ATTGROUP2 Area under a peak. Unfortunately units are usually arbitrary and not related to the x- and y- axis units, and in this case _peakUnits_ should be use. REF maybe

Area under a peak.

Unfortunately units are usually arbitrary and not related to the x- and y- axis units, and in this case _peakUnits_ should be use.

peakUnits*[att.peakUnits]*
Units for a peak or peak integral.
[link]

xMin*[att.xMin]* ATTGROUP2 Minimum yValue. Annotates x-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMax_ attribute but if so xMin should be less than or equals to it. REF maybe

Minimum yValue.

Annotates x-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMax_ attribute but if so xMin should be less than or equals to it.

xMax*[att.xMax]* ATTGROUP2 Maximum yValue. Annotates x-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMin_ attribute but if so xMax should be greater than or equals to it. REF maybe

Maximum yValue.

Annotates x-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMin_ attribute but if so xMax should be greater than or equals to it.

xValue*[att.xValue]* ATTGROUP2 Value along an x axis. Annotates x-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. REF maybe

Value along an x axis.

Annotates x-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data.

xWidth*[att.xWidth]* ATTGROUP2 An unsigned interval along an x axis. It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses xUnits or the same units as the data. REF maybe

An unsigned interval along an x axis.

It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses xUnits or the same units as the data.

xUnits*[att.xUnits]*
Units for x axis.
[link]

yMin*[att.yMin]* ATTGROUP2 Minimum yValue. Annotates y-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMax_ attribute but if so yMin should be less than or equal to it. REF maybe

Minimum yValue.

Annotates y-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMax_ attribute but if so yMin should be less than or equal to it.

yMax*[att.yMax]* ATTGROUP2 Maximum yValue. Annotates y-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMin_ attribute but if so yMax should be greater than or equals to it. REF maybe

Maximum yValue.

Annotates y-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMin_ attribute but if so yMax should be greater than or equals to it.

yValue*[att.yValue]* ATTGROUP2 Value along a y axis. Annotates y-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. REF maybe

Value along a y axis.

Annotates y-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data.

yWidth*[att.yWidth]* ATTGROUP2 An unsigned interval along a y axis. It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses yUnits or the same units as the data. REF maybe

An unsigned interval along a y axis.

It is typically used for the width of a _peak_ or _peakGroup_ but could be used for

any range. It uses yUnits or the same units as the data.

yUnits*[att.yUnits]*
Units for y axis.
[link]

peakList *[el.peakList]*

A list of peaks or peakGroups.

Distinguish between peakList (primarily a navigational container) and peakGroup where the peaks (or groups) have some close relation not shared by all peaks. All peaks and peakGroups should use the same units.

Example

Content Model of element

(metadataList?,
(peak|peakGroup)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

plane3*[el.plane3]*

A plane in 3-space.

An oriented plane of indefinite extent.

Content Model of element

plane3Type[st.plane3Type]

An unbounded plane in 3-space.

Defined by 4 real numbers, conventionally a vector3 normal to the plane and a signed scalar representing the distance to the origin. The vector must not be of zero length (and need not be normalized.

Example

```
The first three numbers are the vector, followed by the
distance
```

BASE:

XSD:LIST of xsd:float

xsd:length

4

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

units*[att.units]*

Scientific units on an element.

[link]

point3 *[el.point3]*

A point in 3-space.

Content Model of element

point3Type[st.point3Type]

A point in 3-space.

The 3 components can have any signed value.

Example

BASE:

XSD:LIST of xsd:float

xsd:length

3

convention *[att.convention]*

A reference to a convention.

[link]

dictRef *[att.dictRef]*

A reference to a dictionary entry.

[link]

id *[att.id]*

An attribute providing a unique ID for an element.

[link]

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

units *[att.units]*

Scientific units on an element.

[link]

potential *[el.potential]*

An explicit potential.

This represents the actual function for the potential (i.e. with explicit values) rather than the functional form, which will normally be referenced from this.
Example

Content Model of element

(
(arg)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`form`*[att.form]*
A reference to a functional form.
[link]

`potentialForm`*[el.potentialForm]*

The functional form of a potential.
This has generic arguments and parameters rather than explicit ones. It is essentially a mathematical function, expressed currently in reverse Polish notation.
Example

Content Model of element

(
(arg)

*,
(parameter)
*,
(expression)
?)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`name`*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

`potentialList`*[el.potentialList]*

A container for explicit potentials.

Experimental.

Example

Content Model of element

(
(potential)
*)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`product` *[el.product]*

A product within a productList.

`product` describes a product species which is produced in a reaction. See `reactant` for discussion of catalysis and solvents.

Example

Content Model of element

A product will normally be identified by name(s), formula, or molecule and at least one of these should normally be given. Amount(s) of product can be given after this identification and can describe mass, volume, percent yield, etc. but not stoichiometry

(metadataList*,
(identifier?|label?|name?)
,molecule?,electron?,substance?,substanceList?,formula?,amount)

Attributes of element

A product will normally be identified by name(s), formula, or molecule and at least one of these should normally be given. Amount(s) of product can be given after this identification and can describe mass, volume, percent yield, etc. but not stoichiometry

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

count*[att.count]*
The count of the object.
[link]

state*[att.state]*
The physical state of the substance.
[link]

productList*[el.productList]*

A container for one or more products.

`productList` can contain several products. These may be related in several ways, including

- single list of products
- grouping of products of parallel reactions

. A productList can contain nested productLists. The semantics of this are currently undefined.

Example

Content Model of element

(metadataList*,name*,
(productList|product)
*)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

count*[att.count]*
The count of the object.
[link]

`property` *[el.property]*

A container for a property.

`property` can contain one or more children, usually `scalar`, `array` or `matrix`. The `dictRef` attribute is required, even if there is a single scalar child with the same dictRef. The property may have a different dictRef from the child, thus providing an extension mechanism.

Properties may have a `state` attribute to distinguish the state of matter
Example

Content Model of element

(metadataList*,name*,
(scalar|array|matrix)
*)

Attributes of element

`dictRef` *[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention` *[att.convention]*
A reference to a convention.
[link]

`title` *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

`id` *[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref` *[att.ref]*
A reference to an element of given type.
[link]

`role` *[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

`state`*[att.state]*
The physical state of the substance.
[link]

`propertyList` *[el.propertyList]*

A container for one or more properties.

`propertyList` can contain several properties. These include (but are not limited to observations, or numeric quantities.

Example

Content Model of element

(metadataList*,name*,
(property|observation)
*)

Attributes of element

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

reactant*[el.reactant]*

A reactant within a reactantList.

`reactant` describes a reactant species which takes part in a reaction. Catalysts and supports are not normally classified as reactants, but this is subjective. Enzymes (or parts of enzymes) may well be reactants, as could be substances which underwent chemical change but were restored to their original state. `reactant` is a powerful concept as it can support stoichiometry (atom and molecule counting), mapping (for mechanisms), etc. Solvents are best contained within substanceList.

Example

Content Model of element

A reactant will normally be identified by name(s), formula, or molecule and at least one of these should normally be given. Amount(s) of reactant can be given after this identification and can describe mass, volume, etc. but not stoichiometr.

(metadataList*,
(identifier|label|name)
,molecule?,electron?,substance?,substanceList?,formula?,amount)

Attributes of element

A reactant will normally be identified by name(s), formula, or molecule and at least one of these should normally be given. Amount(s) of reactant can be given after this identification and can describe mass, volume, etc. but not stoichiometr.

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

count*[att.count]*
The count of the object.
[link]

state*[att.state]*
The physical state of the substance.
[link]

reactantList*[el.reactantList]*

A container for one or more reactants.

`reactantList` can contain several reactants. These may be related in several ways, including
- lists of related reactants
- reactant schemes
- multi-step reactants
- parallel and/or coupled reactants

. A reactantList can contain nested reactantLists. The semantics of this are currently undefined.
Example

Content Model of element

(metadataList*,name?,
(reactantList|reactant)
*)

Attributes of element

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`role`*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

`count`*[att.count]*
The count of the object.
[link]

`reaction`*[el.reaction]*

A chemical reaction or reaction step.

`reaction` is a container for reactants, products, conditions, properties and possibly other information relating to the reaction, often within a reactionList. Partial semantics exist:

- **name** the name(s) of the reaction

- **reactantList** (normally only one) the grouped reactants
- **substance** or **substanceList** substances present in the reaction but not classified as reactants. Examples might be enzymes, catalysts, solvents, supports, workup, etc.
- **condition** conditions of the reaction. These may be text strings, but ideally will have clearer semantics such as scalars for temperature, etc.
- **productList** the grouped products. This allows for parallel reactions or other semantics.
- **property** properties (often physical) associated with the reaction. Examples might be heat of formation, kinetics or equilibrium constant.

Reaction normally refers to an overall reaction or a step within a reactionList. For a complex "reaction", such as in enzymes or chain reactions, it may be best to use `reactionScheme` to hold the overall `reaction` and a `reactionList` of the individual `reaction` steps.

Example
Example
Example
Example
Example
Example

Content Model of element

The semantics of the content model are

- *metadataList for general metadata*
- *label for classifying or describing the reaction (e.g. "hydrolysis")*
- *identifier for unique identification. This could be a classification such as EC (enzyme commission) or an IChI-like string generated from the components.*
- *these are followed by the possible components of the reaction and/or a reactionList of further details.*

.

(metadataList*,label*,
(name*|identifier*)
*,
(reactiveCentre*,mechanism*,reactantList*,spectatorList*,substanceList*,conditionList*,transitionSta
*)

Attributes of element

The semantics of the content model are

- metadataList for general metadata
- label for classifying or describing the reaction (e.g. "hydrolysis")
- identifier for unique identification. This could be a classification such as EC (enzyme commission) or an IChI-like string generated from the components.
- these are followed by the possible components of the reaction and/or a reactionList of further details.

.

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

format*[att.reactionFormat]*
Format of the reaction component.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

role*[att.reactionRole]*
Role of the reaction.
[link]

type*[att.reactionType]*
Type of the reaction.
[link]

state*[att.state]*
The physical state of the substance.
[link]

atomMap*[att.atomMap]* ATTGROUP2 A reference to a map providing mappings between atoms The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking atoms. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase

"USE_IDS" assume that the sets of atoms are of equal size and have 1:1 mapping between each id. This is another way of saying that the atoms mapped by a given ID are "the same atom". REF maybe

A reference to a map providing mappings between atoms

The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking atoms. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of atoms are of equal size and have 1:1 mapping between each id. This is another way of saying that the atoms mapped by a given ID are "the same atom".

`electronMap`*[att.electronMap]* ATTGROUP2 A reference to a map providing mappings between electrons The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking electrons. The topology of the linking is defined by the application - it could be reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of electrons are of equal size and have 1:1 mapping between each id. This is another way of saying that the electrons mapped by a given ID are "the same electron". REF maybe

A reference to a map providing mappings between electrons

The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking electrons. The topology of the linking is defined by the application - it could be reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of electrons are of equal size and have 1:1 mapping between each id. This is another way of saying that the electrons mapped by a given ID are "the same electron".

`bondMap`*[att.bondMap]* ATTGROUP2 A reference to a map providing mappings between bonds The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking bonds. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of bonds are of equal size and have 1:1 mapping between each id. This is another way of saying that the bonds mapped by a given ID are "the same bond". REF maybe

A reference to a map providing mappings between bonds

The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking bonds. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of bonds are of equal size and have 1:1 mapping between each id. This is another way of saying that the bonds mapped by a given ID are "the same bond".

`yield`*[att.yield]*
Yield of a reaction or reactionStep.

[link]

reactionList *[el.reactionList]*

A container for one or more reactions or reactionSchemes with no interrelations.

A reactionList aggregates reactions and reactionSchemes but implies no semantics. The most common uses are to create small collections of reactions (e.g. databases or publications).

Example

Content Model of element

(metadataList*,
(reactionScheme*|reaction*)
)

Attributes of element

dictRef *[att.dictRef]*

A reference to a dictionary entry.

[link]

convention *[att.convention]*

A reference to a convention.

[link]

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id *[att.id]*

An attribute providing a unique ID for an element.

[link]

name *[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not

be a semi-controlled vocabulary.

ref *[att.ref]*
A reference to an element of given type.
[link]

reactionScheme *[el.reactionScheme]*

A container for two or more related reactions and their relationships.

Where reactions are closely related (and often formally dependent on each other) they should be contained within the reactionStepList of a reactionScheme. The semantics which have informed this design include:

- Steps within an organic synthesis.
- Two or more individual (primitive) steps provding the detailed mechanism for an overall reaction.
- Coupled or sequential reactions within biochemical pathways.

This design is general because "reaction" is used in several ways. A biochemical pathway (e.g. oxidation of glucose to CO2 and water) involves many coupled enzyme reactions proceeding both in parallel and in sequence. Each of these steps ("reactions" in their own right) is itself complex and can include several mechanistics steps which are themselves reactions with products, reactants, etc. `reactionScheme` can therefore include reactionStepLists (with more reactionScheme children) which provide a more detailed view of the individual components.

Where a set of reactions are primitives...
Example
Example
Example
Example

Content Model of element

The semantics of the content model are
- *metadataList for general metadata*
- *label for classifying or describing the reaction (e.g. "hydrolysis")*
- *identifier for unique identification. This could be a classification such as EC (enzyme commission) or an IChI-like string generated from the components.*
- *these are followed by the possible components of the reaction and/or a reactionList of further details.*

(metadataList*,
(label|name|identifier)
*,
(reaction|reactionStepList|reactionScheme)

*)

Attributes of element

The semantics of the content model are
- metadataList for general metadata
- label for classifying or describing the reaction (e.g. "hydrolysis")
- identifier for unique identification. This could be a classification such as EC (enzyme commission) or an IChI-like string generated from the components.
- these are followed by the possible components of the reaction and/or a reactionList of further details.

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`role`*[att.reactionRole]*
Role of the reaction.
[link]

`type`*[att.reactionType]*
Type of the reaction.
[link]

`state`*[att.state]*
The physical state of the substance.

[link]

format*[att.reactionFormat]*
Format of the reaction component.
[link]

reactionStep*[el.reactionStep]*

A child of reactionStepList and a container for reaction or reactionScheme.

`reactionStep` is always contained within reactionStepList and is designed to manage "sub-reactions" which have close relationships. These will often involve reactions which, taken together, describe a higher level reaction or reaction type. Examples are:

- biochemical pathways
- synthetic reaction schemes
- multi-step reactions
- parallel and/or coupled reactions

. A reactionStep normally contains a single reaction or reactionScheme. It can have attributes such as yield and ratio which can be used by the parent reactionStepList.
Example
Example
Example

Content Model of element

The `name` applies to the overall schema of reactions. `label` is for additional textual information and classification. `reactionStepList` normally contains `reaction`s but we make provision for nested reactionSchemes if required.

(metadataList*,
(name|label)
*,
(reactionScheme|reaction)
*)

Attributes of element

The `name` applies to the overall schema of reactions. `label` is for additional textual information and classification. `reactionStepList` normally contains `reaction`s but we make provision for nested reactionSchemes if required.

dictRef*[att.dictRef]*
A reference to a dictionary entry.

[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

yield*[att.yield]*
Yield of a reaction or reactionStep.
[link]

ratio*[att.ratio]*
A ratio in the range 0 to 1.
[link]

reactionStepList*[el.reactionStepList]*

A container for one or more related reactionSteps.

`reactionStepList` is always contained within reactionScheme and is designed to manage "sub-reactions" which have close relationships. These will often involve reactions which, taken together, describe a higher level reaction or reaction type. Examples are:

- biochemical pathways
- synthetic reaction schemes
- multi-step reactions
- parallel and/or coupled reactions

. A reactionStepList contains reactionSteps (each of which contains reactions and/or reactionSchemes (e.g. where part of the process is known in greater detail)). It may not directly contain child reactionStepLists.

The child reactionSteps can have attributes such as yield and ratio which describe

the relationship of the component steps.

Guidance on use:
- reactionScheme describes a complex of reactions with metadata, one (or more) overall reactions and a reactionStepList with the overall component reactions.
- reactionStepList aggregates and structures the individual subreactions.
- reactionList is a container for reactions and reactionSchemes with no semantics (e.g. a book or database of selected reactions).

Example
Example

Content Model of element

The `name` applies to the overall schema of reactions. `label` is for additional textual information and classification. `reactionStepList` normally contains `reactionStep`s.

(metadataList*,
(name|label)
*,
(reactionStep)
*)

Attributes of element

The `name` applies to the overall schema of reactions. `label` is for additional textual information and classification. `reactionStepList` normally contains `reactionStep`s.

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.

[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.

A qualifier which may affect the semantics of the object.

format*[att.reactionFormat]*
Format of the reaction component.
[link]

reactiveCentre*[el.reactiveCentre]*

The reactiveCentre in a reaction.

This describes the set(s) of bonds and atoms involved in the reaction. The semantics are flexible, but a common usage would be to create atomSet(s) and bondSet(s) mapping to groups which undergo changes.
Example

Content Model of element

(description?,
(atomTypeList|bondTypeList|atomSet|bondSet)
*)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.

[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

region*[el.region]*

A region of the system.

Under development. A subdivision of the system to which special protocols or properties may be attached. Typical regions could be defined by the presence of atoms belonging to an atomSet or geometrical boundaries.

A region element will not always contain other elements, but may have references from other elements. It may create a protocol, e.g. atoms within a region might be replaced by a continuum model or be subject ro a field. Semantics yet to be determined.

Regions can be created by the unions of two or more regions. This allows a region to be built from a series of (say) spheres or boxes filling space.

Example

Content Model of element

()

Attributes of element

sphere3*[att.sphere3]*
A sphere.
[link]

box3*[att.box3]*
A parallelipiped box.
[link]

atomSetRef*[att.atomSetRef]*
An atomSet describing the region.
[link]

regionRefs*[att.regionRefs]*
A list of regions creating a union.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

relatedEntry*[el.relatedEntry]*

An entry related in some way to a dictionary entry.

The range of relationships is not restricted but should include parents, aggregation, seeAlso and so on. DataCategories from ISO12620 can be referenced through the namespaced mechanism.

Example

Content Model of element

Attributes of element

type*[att.relatedEntryType]*

Type of relatedEntry.

href*[att.href]* ATTGROUP2 address of a resource. Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present. REF maybe

address of a resource.

Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present.

`sample`*[el.sample]*

An analytical or spectral sample.

The `sample` should contain information on what things were in the sample and their roles. It can include `molecule`, `substance` and `substanceList`. Typical rolos include solvent, mulling agents, salt disks, molecular supports, etc. but should not cover apparatus or conditions.

Example

Content Model of element

(metadataList*,
(molecule|substance|substanceList)
*)

Attributes of element

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`ref`*[att.ref]*

A reference to an element of given type.

[link]

`state`*[att.state]*

The physical state of the substance.

[link]

`scalar` *[el.scalar]*

An element to hold scalar data.

`scalar` holds scalar data under a single generic container. The semantics are usually resolved by linking to a dictionary. **scalar** defaults to a scalar string but has attributes which affect the type.

`scalar` does not necessarily reflect a physical object (for which **object** should be used). It may reflect a property of an object such as temperature, size, etc.

Note that normal Schema validation tools cannot validate the data type of **scalar** (it is defined as `string`), but that a temporary schema can be constructed from the type and used for validation. Also the type can be contained in a dictionary and software could decide to retrieve this and use it for validation.

Example

Content Model of element

[xsd:string]

`title` *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id` *[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention` *[att.convention]*

A reference to a convention.

[link]

`dictRef` *[att.dictRef]*

A reference to a dictionary entry.

[link]

`dataType` *[att.dataType]*

The data type of the object.

[link]

`errorValue` *[att.errorValue]*

Value of the error.

[link]

errorBasis*[att.errorBasis]*
Basis of the error estimate.
[link]

min*[att.min]*
The minimum value allowed for an element or attribute.
[link]

max*[att.max]*
Maximum value allowed for an element or attribute.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

units*[att.units]*
Scientific units on an element.
[link]

spectator *[el.spectator]*

A spectator object in a reaction.

Objects are often present during a reaction which are not formally involved in bond breaking/formation and which are not modified during the reaction. They may be catalysts, but may also be objects which in some way constrain or help the reaction to take place (surfaces, micelles, groups in enzyme active sites, etc.). In some cases molecules present in a reaction mixture may act as spectators in steps in which they are not transformed.

Example

Content Model of element

(metadataList*,
(label|molecule|object)
*)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

spectatorList*[el.spectatorList]*

A container for spectators in a reaction.
Example

Content Model of element

(spectator)
+

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*

A reference to a convention.
[link]

dictRef *[att.dictRef]*
A reference to a dictionary entry.
[link]

spectrum *[el.spectrum]*

A spectrum and relevant data or metadata.

The `spectrum` construct can hold `metadataList`, `sample` (which can contain molecule), `conditionList` (mainly for physical/chemical conditions, not instrumental), `spectrumData` for the actual data and instrumental settings/procedure and `peakList` for the assigned peaks. This approach puts the spectrum as the primary object of interest. It could also be possible to make `spectrum` a child of `molecule` (although a reference using `ref` might be preferable).

Example

Content Model of element

(metadataList*,sample*,conditionList*,spectrumData*,peakList*)

Attributes of element

dictRef *[att.dictRef]*
A reference to a dictionary entry.
[link]

convention *[att.convention]*
A reference to a convention.
[link]

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id *[att.id]*
An attribute providing a unique ID for an element.
[link]

ref *[att.ref]*

A reference to an element of given type.
[link]

moleculeRef *[att.moleculeRef]*
A reference to a molecule.
[link]

type *[att.spectrumType]*
The type of the spectrum.

An infrared spectrum.

A "simple" mass spectrum.

An NMR spectrum.

A spectrum somewhere in the UV VIS region of the spectrum.

format *[att.format]*
Format of a spectrum.

one dimensional spectru.

Two dimensional spectru.

Two dimensional spectrum with different axe.

measurement *[att.measurement]*
Type of spectral measurement.

Data are transmittance, so "peaks" are usually troughs.

Data are absorbanc.

ft *[att.ft]*Default:none
Domain of an FT spectrum.

Data are raw, so will normally require transforming.

Data have been transformed. This value indicates that an FT experiment and transformation have been performe.

This was not known to be an FT experiment. (It may have been, but the author or abstracter omitted to mention it).

state *[att.state]*
The physical state of the substance.
[link]

spectrumData *[el.spectrumData]*

Data for the spectrum.

This is primarily to record the data in interchangeable format and machine and manufacturers settings and can include other MLs in this area (AniML, SpectroML, etc.). We recommend ASCII representations of data and this is the only format that CMLSpect implementers have to support, but we also allow for the carriage of JCAMP and other data (in ML wrappers such as AniML). All numeric data should carry units and dictionary references if possible to allow for semantic interoperability.
Example

Content Model of element

(
(xaxis,yaxis+)
?)

Attributes of element

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`spectrumList` *[el.spectrumList]*

A container for one or more spectra.

`spectrumList` can contain several spectra. These may be related in several ways, including

- lists of related spectra

- bundle of common analytical spectra (NMR, IR, UV...)
- repeat measurements

. A spectrumList can contain nested spectrumLists.

Example

Content Model of element

`metadataList` contains `metadata`. `list` is for experimental and other data. `spectrumList` normally contains `spectrum`s but we make provision for nested spectrumLists if required. The `molecule`s can be a set of reference molecules which occur in the `spectrum`s and can be referenced. This makes the spectrums more readable and normalizes data when molecules are used more than once.

(metadataList*,list*,
(spectrumList*|spectrum*)
)

Attributes of element

`metadataList` contains `metadata`. `list` is for experimental and other data. `spectrumList` normally contains `spectrum`s but we make provision for nested spectrumLists if required. The `molecule`s can be a set of reference molecules which occur in the `spectrum`s and can be referenced. This makes the spectrums more readable and normalizes data when molecules are used more than once.

`dictRef` *[att.dictRef]*

A reference to a dictionary entry.

[link]

`convention` *[att.convention]*

A reference to a convention.

[link]

`title` *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id` *[att.id]*

An attribute providing a unique ID for an element.

[link]

`ref` *[att.ref]*

A reference to an element of given type.

[link]

moleculeRef*[att.moleculeRef]*
A reference to a molecule.
[link]

sphere3*[el.sphere3]*

A sphere in 3-space.

Content Model of element

sphere3Type[st.sphere3Type]

A sphere in 3-space.

Defined by 4 real numbers, conventionally a point3 at the centre of the sphere and a nonNegative scalar for the radius.

Example

BASE:
XSD:LIST of xsd:float
xsd:length
4

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

units*[att.units]*
Scientific units on an element.
[link]

stmml *[el.stmml]*

An element to hold stmml data.

stmml holds stmml data under a single generic container. Other namespaces may be present as children. No semantics implied.

Example

Content Model of element

(ANY [lax])
*

Attributes of element

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id *[att.id]*

An attribute providing a unique ID for an element.

[link]

convention *[att.convention]*

A reference to a convention.

[link]

dictRef *[att.dictRef]*

A reference to a dictionary entry.

[link]

substance *[el.substance]*

A chemical substance.

substance represents a *chemical substance* which is deliberately very general. It can represent things that may or may not be molecules, can and cannot be stored in bottles and may or may not be microscopic. Solutions and mixtures can be described by _substanceList_s of substances. The type attribute can be used to give qualitative information characterising the substance ("granular", "90%", etc.) and _role_ to describe the role in process ("desiccant", "support", etc.). There is currently no controlled vocabulary. Note that reaction is likely to have more precise

semantics. The amount of a substance is controlled by the optional _amount_ child.
Example

Added property as a child 2002-12-29

Content Model of element

(metadataList*,amount?,
(molecule*|name*|property*)
)

Attributes of element

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`type`*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF maybe

Type of the object.
A qualifier which may affect the semantics of the object.

`role`*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

`ref`*[att.ref]*
A reference to an element of given type.

[link]

count*[att.count]*
The count of the object.
[link]

state*[att.state]*
The physical state of the substance.
[link]

substanceList*[el.substanceList]*

A list of chemical substances.

Deliberately very general - see substance. substanceList is designed to manage solutions, mixtures, etc. and there is a small enumerated controlled vocabulary, but this can be extended through dictionaries.

substanceList can have an amount child. This can indicate the amount of a solution or mixture; this example describes 100 ml of 0.1M NaOH(aq). Although apparently longwinded it is precise and fully machine-interpretable

Added role attribute, 2003-03-12.
Example

Content Model of element

(metadataList*,amount?,substance*,propertyList?)

Attributes of element

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.
Example

id*[att.id]*

An attribute providing a unique ID for an element.
[link]

`type`*[att.substanceListType]*
Type of the substanceList.

`role`*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF maybe

Role of the object.
How the object functions or its position in the architecture. No controlled vocabulary.

`ref`*[att.ref]*
A reference to an element of given type.
[link]

`symmetry` *[el.symmetry]*

Molecular, crystallographic or other symmetry.

`symmetry` provides a label and/or symmetry operations for molecules or crystals. Point and spacegroups can be specified by strings, though these are not enumerated, because of variability in syntax (spaces, case-sensitivity, etc.), potential high symmetries (e.g. TMV disk is D17) and non-standard spacegroup settings. Provision is made for explicit symmetry operations through <matrix> child elements.

By default the axes of symmetry are defined by the symbol - thus C2v requires z to be the unique axis, while P21/c requires b/y. Spacegroups imply the semantics defined in International Tables for Crystallography, (Int Union for Cryst., Munksgaard). Point groups are also defined therein.

The element may also be used to give a label for the symmetry species (irreducible representation) such as "A1u" for a vibration or orbital.

The matrices should be 3x3 for point group operators and 3x4 for spacegroup operators. The use of crystallographic notation ("x,1/2+y,-z") is not supported - this would be <matrix>1 0 0 0.0 0 1 0 0.5 0 0 1 0.0<matrix>.

The default convention for point group symmetry is `Schoenflies` and for spacegroups is "H-M". Other conventions (e.g. "Hall") must be specfied through the `convention` attribute.

This element implies that the Cartesians or fractional coordinates in a molecule are oriented appropriately. In some cases it may be useful to specify the symmetry of an arbitarily oriented molecule and the <molecule> element has the attribute `symmetryOriented` for this purpose.

Example

Content Model of element

(matrix*)

Attributes of element

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`pointGroup`*[att.pointGroup]* ATTGROUP2 A point group. No fixed semantics, though Schoenflies is recommended over Hermann-Mauguin. We may provide a controlled-extensible list in the future. REF maybe

A point group.

No fixed semantics, though Schoenflies is recommended over Hermann-Mauguin. We may provide a controlled-extensible list in the future.

`spaceGroup`*[att.spaceGroup]* ATTGROUP2 A space group. No fixed semantics, though Hermann-Mauguin or Hall is recommended over Schoenflies. We may provide a controlled-extensible list in the future. REF maybe

A space group.

No fixed semantics, though Hermann-Mauguin or Hall is recommended over Schoenflies. We may provide a controlled-extensible list in the future.

`irreducibleRepresentation`*[att.irreducibleRepresentation]* ATTGROUP2 A symmetry species. No fixed semantics, though we may provide a controlled-extensible list in the future. REF maybe

A symmetry species.

No fixed semantics, though we may provide a controlled-extensible list in the future.

number*[att.number]* ATTGROUP2 A number determined by context Used for isotope number in isotope, and rotational symmetry number in symmetry for calculation of entropy, etc. 2003-03-30: added number attribut. REF maybe

A number determined by context

Used for isotope number in isotope, and rotational symmetry number in symmetry for calculation of entropy, etc.

2003-03-30: added number attribut.

system*[el.system]*

The complete system of components in a calculation.

There is no controlled vocabulary.

Content Model of element

(ANY [lax])

Attributes of element

dimensionality*[att.dimensionality]* ATTGROUP2 Dimensionality of a coordinate system. Note that this means that coordinates of higher dimensionality are ignored or an error is flagged. Thus z3 and dimensionality='2' are incompatible. At present higher dimensionalities than 3 (cf. Wondratschek) are not supported. The labelling of the axes id not controlled. ?? should we have an explicit attribute for labelling convention?. REF maybe

Dimensionality of a coordinate system.

Note that this means that coordinates of higher dimensionality are ignored or an error is flagged. Thus z3 and dimensionality='2' are incompatible. At present higher dimensionalities than 3 (cf. Wondratschek) are not supported. The labelling of the axes id not controlled. ?? should we have an explicit attribute for labelling convention?.

periodicity*[att.periodicity]* ATTGROUP2 Periodicity of the system. This represents the number of dimensions (or coordinate axes) along periodic behaviour occurs and can be supported by symmetry operators or other transformations. Periodicity must never exceed dimensionality. REF maybe

Periodicity of the system.

This represents the number of dimensions (or coordinate axes) along periodic behaviour occurs and can be supported by symmetry operators or other

transformations. Periodicity must never exceed dimensionality.

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

`table`*[el.table]*

A rectangular table of any quantities.

By default `table` represents a rectangular table of any quantities representable as XSD or STMML dataTypes. The default layout is columnwise, with `columns` columns, where each column is a (homogeneous) **array** of size `rows` data. This is the "normal" orientation of data tables but the table display could be transposed by XSLT transformation if required. Access is to columns, and thence to the data within them. DataTyping, delimiters, etc are delegated to the arrays, which must all be of the same size. For verification it is recommended that tables carry `rows` and `columns` attributes.

An alternative is to use the standard HTML table layout (tr and td). The identities of the columns (dictRef), their dataTypes and their units are given by leading `tr` elements with the `type` attribute. This is harder to process and the column-wise approach should be used where possible if dataTypes, etc. are important

Example

Content Model of element

(
(array*)
)

Attributes of element

rows*[att.rows]*
Number of rows.
[link]

columns*[att.columns]*
Number of columns.
[link]

units*[att.units]*
Scientific units on an element.
[link]

dataType*[att.dataType]*
The data type of the object.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

torsion*[el.torsion]*

A torsion angle ("dihedral") between 4 distinct atoms.

The atoms need not be formally bonded. It can be used for:
- Recording experimentally determined torsion angles (e.g. in a crystallographic

paper).
- Providing the torsion component for internal coordinates (e.g. z-matrix).

Note that the order of atoms is important.
Example

Content Model of element

torsionAngleType[st.torsionAngleType]

The type of a torsion angle.
[xsd:float]
minInclusive: -360.0
maxInclusive: 360.0

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.
No controlled value.
Example

`id`*[att.id]*
An attribute providing a unique ID for an element.
[link]

`convention`*[att.convention]*
A reference to a convention.
[link]

`dictRef`*[att.dictRef]*
A reference to a dictionary entry.
[link]

`atomRefs4`*[att.atomRefs4]*
A list of 4 references to atoms.
[link]

`units`*[att.units]*
Scientific units on an element.
[link]

`errorValue`*[att.errorValue]*
Value of the error.
[link]

`errorBasis`*[att.errorBasis]*

Basis of the error estimate.
[link]

min*[att.min]*
The minimum value allowed for an element or attribute.
[link]

max*[att.max]*
Maximum value allowed for an element or attribute.
[link]

ref*[att.ref]*
A reference to an element of given type.
[link]

transform3*[el.transform3]*

A transform in 3-space.
A 3-D transform. Conventionally a 4x4 matrix.

Content Model of element

matrix44Type[st.matrix44Type]

A 4x4 transformation matrix
...
BASE:
XSD:LIST of xsd:float
xsd:length
16

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`transitionState`*[el.transitionState]*

The transition state in a reaction.

This will normally contain a `molecule` which in its 2D representation will have partial bonds. These are yet to be formalized for the `molecule` element.

Although spectators may stabilise or otherwise interact with the transitionState they are not contained within it.

A `propertyList` is provided to capture transitionState properties.

Still experimental.

Example

Content Model of element

(molecule,propertyList?)

Attributes of element

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

`id`*[att.id]*

An attribute providing a unique ID for an element.

[link]

`convention`*[att.convention]*

A reference to a convention.

[link]

`dictRef`*[att.dictRef]*

A reference to a dictionary entry.

[link]

unit *[el.unit]*

A scientific unit.

A scientific unit. Units are of the following types:

- SI Units. These may be one of the seven fundamental types (e.g. meter) or may be derived (e.g. joule). An SI unit is identifiable because it has no parentSI attribute and will have a unitType attribute.
- nonSI Units. These will normally have a parent SI unit (e.g. calorie has joule as an SI parent).

Example

2003:04-09 Description or parentSI attribute enhance.

Content Model of element

(description|annotation)
*

Attributes of element

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

abbreviation*[att.abbreviation]* ATTGROUP2 Abbreviation. Abbreviation for units, terms, etc. REF maybe

Abbreviation.

Abbreviation for units, terms, etc.

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

parentSI*[att.parentSI]*
A dictRef like reference to the id of the parent SI unit.
[link]

unitType*[att.unitType]* ATTGROUP2 A reference to the type of a unit. Used in defining the unit and doing symbolic algebra on the dimensionality. REF maybe

A reference to the type of a unit.

Used in defining the unit and doing symbolic algebra on the dimensionality.

multiplierToSI*[att.multiplierToSI]* ATTGROUP2 Multiplier to generate SI equivalent. The factor by which the non-SI unit should be multiplied to convert a quantity to its representation in SI Units. This is applied *before* _constantToSI_. Necessarily unity for SI unit. REF maybe

Multiplier to generate SI equivalent.

The factor by which the non-SI unit should be multiplied to convert a quantity to its representation in SI Units. This is applied *before* _constantToSI_. Necessarily unity for SI unit.

constantToSI*[att.constantToSI]* ATTGROUP2 Additive constant to generate SI equivalent. The amount to add to a quantity in non-SI units to convert its representation to SI Units. This is applied *after* multiplierToSI. It is necessarily zero for SI units. REF maybe

Additive constant to generate SI equivalent.

The amount to add to a quantity in non-SI units to convert its representation to SI Units. This is applied *after* multiplierToSI. It is necessarily zero for SI units.

unitList*[el.unitList]*

A container for several unit entries.

Usually forms the complete units dictionary (along with metadata).

Example

Content Model of element

(unitType*,unit*)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*
A reference to a convention.
[link]

dictRef*[att.dictRef]*
A reference to a dictionary entry.
[link]

href*[att.href]* ATTGROUP2 address of a resource. Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present. REF maybe

address of a resource.

Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present.

unitType*[el.unitType]*

The type of a scientific unit.

Mandatory for SI Units, optional for nonSI units since they should be able to obtain this from their parent. For complex derived units without parents it may be useful.

Used within a unitList

Distinguish carefully from **unitsType** which is primarily used for attributes describing the units that elements carry
Example

Content Model of element

(alternative*,dimension*)

Attributes of element

id*[att.id]*
An attribute providing a unique ID for an element.
[link]

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled

vocabulary. REF maybe

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

vector3*[el.vector3]*

A vector in 3-space.

The vector may have magnitude but is not rooted on any points (use line3).

Content Model of element

vector3Type[st.vector3Type]

A vector in 3-space.

No constraints on magnitude (i.e. could be zero.

Example

BASE:

XSD:LIST of xsd:float

xsd:length

3

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

units*[att.units]*

Scientific units on an element.

[link]

xaxis *[el.xaxis]*

The x-axis.

A container for all information relating to the x-axis (including scales, offsets, etc.) and the data themselves (in an array). Note: AniML uses "xValues" so avoid confusion with this.

Example

Content Model of element

(array)

Attributes of element

dictRef *[att.dictRef]*

A reference to a dictionary entry.

[link]

convention *[att.convention]*

A reference to a convention.

[link]

title *[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id *[att.id]*

An attribute providing a unique ID for an element.

[link]

ref *[att.ref]*

A reference to an element of given type.

[link]

multiplierToData *[att.multiplierToData]*Default:1.0 ATTGROUP2 The scale by which to multiply the raw data. The scale is applied *before* adding the constant. REF maybe

The scale by which to multiply the raw data.

The scale is applied *before* adding the constant.

constantToData*[att.constantToData]*Default:0.0 ATTGROUP2 The constant to add to the raw data. add *after* applying any multiplier. REF maybe

The constant to add to the raw data.

add *after* applying any multiplier.

yaxis*[el.yaxis]*

The y-axis.

A container for all information relating to the y-axis (including scales, offsets, etc.) and the data themselves (in an `array`).

Example

Content Model of element

(array)

Attributes of element

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

ref*[att.ref]*

A reference to an element of given type.

[link]

multiplierToData*[att.multiplierToData]*Default:1.0 ATTGROUP2 The scale by which to multiply the raw data. The scale is applied *before* adding the constant. REF maybe

The scale by which to multiply the raw data.

The scale is applied *before* adding the constant.

constantToData*[att.constantToData]*Default:0.0 ATTGROUP2 The constant to add to the raw data. add *after* applying any multiplier. REF maybe

The constant to add to the raw data.

add *after* applying any multiplier.

zMatrix*[el.zMatrix]*

A zMatrix.

A container for `length`, `angle` and `torsion`, which must be arranged in the conventional zMatrix format.

Example

Content Model of element

(
(length|angle|torsion)
*)

Attributes of element

title*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF maybe

A title on an element.

No controlled value.

Example

id*[att.id]*

An attribute providing a unique ID for an element.

[link]

convention*[att.convention]*

A reference to a convention.

[link]

dictRef*[att.dictRef]*

A reference to a dictionary entry.

[link]

ATTRIBUTES

abbreviation*[att.abbreviation]* ATTGROUP2 Abbreviation. Abbreviation for units, terms, etc. REF no

Abbreviation.

Abbreviation for units, terms, etc.

atomID*[att.atomIDArray]*

An array of atomRefs.

ATTGROUP2 An array of atom IDs. Normally an attribute of an array-based element. REF no

An array of atom IDs.

Normally an attribute of an array-based element.

atomMap*[att.atomMap]* ATTGROUP2 A reference to a map providing mappings between atoms The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking atoms. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of atoms are of equal size and have 1:1 mapping between each id. This is another way of saying that the atoms mapped by a given ID are "the same atom". REF no

A reference to a map providing mappings between atoms

The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking atoms. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of atoms are of equal size and have 1:1 mapping between each id. This is another way of saying that the atoms mapped by a given ID are "the same atom".

atomRef*[att.atomRef]*
A reference to an existing atom.
ATTGROUP2 A reference to an atom. Used by bond, electron, etc. REF no

A reference to an atom.
Used by bond, electron, etc.

atomRef1*[att.atomRef1Array]*
An array of atomRefs.
ATTGROUP2 The first atoms in each bond. Currently only used in bondArray in CML2 array mode. REF no

The first atoms in each bond.
Currently only used in bondArray in CML2 array mode.

atomRef2*[att.atomRef2Array]*
An array of atomRefs.
ATTGROUP2 The second atoms in each bond. Only used in bondArray in CML2 array mode. REF no

The second atoms in each bond.
Only used in bondArray in CML2 array mode.

atomRefArray*[att.atomRefArray]*
An array of atomRefs.
ATTGROUP2 An array of references to atoms. Typical use would be to atoms defining a plane. REF no

An array of references to atoms.
Typical use would be to atoms defining a plane.

atomRefs*[att.atomRefs]*
An array of atomRefs.

ATTGROUP2 A reference to a list of atoms. Used by bonds, electrons, atomSets, etc. REF no

A reference to a list of atoms.

Used by bonds, electrons, atomSets, etc.

---

atomRefs2*[att.atomRefs2]*

A reference to two distinct existing atoms in order.

ATTGROUP2 References to two different atoms. Available for any reference to atoms but normally will be the normal reference attribute on the bond element. The order of atoms is preserved and may matter for some conventions (e.g. wedge/hatch or donor bonds. REF no

References to two different atoms.

Available for any reference to atoms but normally will be the normal reference attribute on the bond element. The order of atoms is preserved and may matter for some conventions (e.g. wedge/hatch or donor bonds.

---

atomRefs3*[att.atomRefs3]*

A reference to three distinct existing atoms in order.

ATTGROUP2 A list of three references to atoms. Typically used for defining angles, but could also be used to define a three-centre bond. REF no

A list of three references to atoms.

Typically used for defining angles, but could also be used to define a three-centre bond.

---

atomRefs4*[att.atomRefs4]*

A reference to four distinct existing atoms in order.

ATTGROUP2 A list of 4 references to atoms. Typically used for defining torsions and atomParities, but could also be used to define a four-centre bond. REF no

A list of 4 references to atoms.

Typically used for defining torsions and atomParities, but could also be used to define a four-centre bond.

---

atomSetRef*[att.atomSetRef]*

A reference to an existing object.

ATTGROUP2 An atomSet describing the region. Any point falling within atomOffset of any atom in the set lies within the region. This means the region could consist of disjoint fragments. REF no

An atomSet describing the region.

Any point falling within atomOffset of any atom in the set lies within the region. This means the region could consist of disjoint fragments.

---

bondID*[att.bondIDArray]*

An array of references to bonds.

ATTGROUP2 The IDs for an array of bond. Required in CML2 array mode. REF no

The IDs for an array of bond.

Required in CML2 array mode.

---

bondMap*[att.bondMap]* ATTGROUP2 A reference to a map providing mappings between bonds The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking bonds. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of bonds are of equal size and have 1:1 mapping between each id. This is another way of saying that the bonds mapped by a given ID are "the same bond". REF no

A reference to a map providing mappings between bonds

The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking bonds. The topology of the linking is defined by the application - it could be overlay of molecular fragments, reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of bonds are of equal size and have 1:1 mapping between each id. This is another way of saying that the bonds mapped by a given ID are "the same bond".

---

bondRef*[att.bondRef]*

A reference to an existing bond.

ATTGROUP2 A reference to a bond. used by electron, etc. REF no

A reference to a bond.

used by electron, etc.

bondRefs*[att.bondRefs]*

An array of references to bonds.

ATTGROUP2 A reference to a list of bonds. Used by electrons, bondSets, etc. REF no

A reference to a list of bonds.

Used by electrons, bondSets, etc.

box3*[att.box3]*

A box in 3-space.

ATTGROUP2 A parallelipiped box. By default the box uses isometric Cartesians axes but can also be linked to lattice Vector. Any point falling within the box or on a boundary is within the regio. REF no

A parallelipiped box.

By default the box uses isometric Cartesians axes but can also be linked to lattice Vector. Any point falling within the box or on a boundary is within the regio.

chirality*[att.chirality]*

The chirality of a system or molecule.

ATTGROUP2 The chirality of a system or molecule. This is being actively investigated by a IUPAC committee (2002) so the convention is likely to change. No formal default. REF no

The chirality of a system or molecule.

This is being actively investigated by a IUPAC committee (2002) so the convention is likely to change. No formal default.

columns*[att.columns]*

The size of an array.

ATTGROUP2 Number of columns. REF no

Number of columns.

concise*[att.concise]*

A concise representation for a molecular formula.

ATTGROUP2 A concise formula. The string represents an (unstructured) formula i.e. no submolecules. Recommended to use the format "H 2 O 1", etc. REF no

A concise formula.

The string represents an (unstructured) formula i.e. no submolecules. Recommended to use the format "H 2 O 1", etc.

---

constantToData*[att.constantToData]*Default:0.0 ATTGROUP2 The constant to add to the raw data. add *after* applying any multiplier. REF no

The constant to add to the raw data.

add *after* applying any multiplier.

---

constantToSI*[att.constantToSI]* ATTGROUP2 Additive constant to generate SI equivalent. The amount to add to a quantity in non-SI units to convert its representation to SI Units. This is applied *after* multiplierToSI. It is necessarily zero for SI units. REF no

Additive constant to generate SI equivalent.

The amount to add to a quantity in non-SI units to convert its representation to SI Units. This is applied *after* multiplierToSI. It is necessarily zero for SI units.

---

constraint*[att.constraint]* ATTGROUP2 Constraint on a parameter. Semantics not yet finalised. We anticipate "fixed", "none" and symbolic relationships to other parameters. REF no

Constraint on a parameter.

Semantics not yet finalised. We anticipate "fixed", "none" and symbolic relationships to other parameters.

---

content*[att.content]* ATTGROUP2 content of metadata. REF no

content of metadata.

---

convention*[att.convention]*

An XML QName with required prefix.

ATTGROUP2 A reference to a convention. There is no controlled vocabulary for conventions, but the author must ensure that the semantics are openly available and that there are mechanisms for implementation. The convention is inherited by all the subelements, so that a convention for molecule would by default extend to its bond and atom children. This can be overwritten if necessary by an explicit convention. It may be useful to create conventions with namespaces (e.g. iupac:name). Use of convention will normally require non-STMML semantics, and should be used with caution. We would expect that conventions prefixed with "ISO" would be useful, such as ISO8601 for dateTimes. There is no default, but the conventions of STMML or the related language (e.g. CML) will be assumed. REF no

A reference to a convention.

There is no controlled vocabulary for conventions, but the author must ensure that the semantics are openly available and that there are mechanisms for implementation. The convention is inherited by all the subelements, so that a convention for `molecule` would by default extend to its `bond` and `atom` children. This can be overwritten if necessary by an explicit `convention`.

It may be useful to create conventions with namespaces (e.g. `iupac:name`). Use of `convention` will normally require non-STMML semantics, and should be used with caution. We would expect that conventions prefixed with "ISO" would be useful, such as ISO8601 for dateTimes.

There is no default, but the conventions of STMML or the related language (e.g. CML) will be assumed.

Example

---

conventionValue*[att.conventionValue]* ATTGROUP2 The value of an element with a _convention_. When convention is used this attribute must be present and element content must be empty. REF no

The value of an element with a _convention_.

When convention is used this attribute must be present and element content must be empty.

---

count*[att.count]*

A positive number.

ATTGROUP2 The count of the object. No fixed semantics or default, normally integral. It is presumed that the element can be multiplied by the count value. REF no

The count of the object.

No fixed semantics or default, normally integral. It is presumed that the element can be multiplied by the count value.

---

countArray*[att.countArray]*

Array of counts.

ATTGROUP2 Array of object counts. No fixed semantics or default, normally integral. It is presumed that the element can be multiplied by the count value. REF no

Array of object counts.

No fixed semantics or default, normally integral. It is presumed that the element can be multiplied by the count value.

---

dataType*[att.dataType]*

an enumerated type for all dataTypes in STM.

ATTGROUP2 The data type of the object. Normally applied to scalar/array objects but may extend to more complex one. REF no

The data type of the object.

Normally applied to scalar/array objects but may extend to more complex one.

---

default*[att.default]* ATTGROUP2 default value in an enumeration. A non-whitespace string (value is irrelevant) indicates that the content of this enumeration is the default value (usually of a scalar). It is an error to have more than one default. If the scalar in an instance document has no value (i.e. is empty or contains only whitespace) its value is given by the default. If the scalar in the instance is empty and no enumerations have a default attribute, an application may throw an error. REF no

default value in an enumeration.

A non-whitespace string (value is irrelevant) indicates that the content of this enumeration is the default value (usually of a scalar). It is an error to have more than one default. If the scalar in an instance document has no value (i.e. is empty or contains only whitespace) its value is given by the default. If the scalar in the instance is empty and no enumerations have a default attribute, an application may throw an error.

---

delimiter*[att.delimiter]*

A single non-whitespace character to separate components in arrays.

ATTGROUP2 A delimiter character for arrays and matrices. By default array components ('elements' in the non-XML sense) are whitespace-separated. This fails for components with embedded whitespace or missing completely: Example: In the protein database ' CA' and 'CA' are different atom types, and and array could be: <array delimiter="/" dictRef="pdb:atomTypes">/ N/ CA/CA/ N/</array> Note that the array starts and ends with the delimiter, which must be chosen to avoid accidental use. There is currently no syntax for escaping delimiters. REF no

A delimiter character for arrays and matrices.

By default array components ('elements' in the non-XML sense) are whitespace-separated. This fails for components with embedded whitespace or missing completely:

```
        Example:
        In the protein database ' CA' and 'CA' are different
atom types, and and array could be:
        <array delimiter="/" dictRef="pdb:atomTypes">/ N/
CA/CA/ N/</array>
```

Note that the array starts and ends with the delimiter, which must be chosen to avoid accidental use. There is currently no syntax for escaping delimiters.

---

dictRef *[att.dictRef]*

An XML QName with required prefix.

ATTGROUP2 A reference to a dictionary entry. Elements in data instances such as _scalar_ may have a dictRef attribute to point to an entry in a dictionary. To avoid excessive use of (mutable) filenames and URIs we recommend a namespace prefix, mapped to a namespace URI in the normal manner. In this case, of course, the namespace URI must point to a real XML document containing _entry_ elements and validated against STMML Schema. Where there is concern about the dictionary becoming separated from the document the dictionary entries can be physically included as part of the data instance and the normal XPointer addressing mechanism can be used. This attribute can also be used on _dictionary_ elements to define the namespace prefix REF no

A reference to a dictionary entry.

Elements in data instances such as _scalar_ may have a `dictRef` attribute to point to an entry in a dictionary. To avoid excessive use of (mutable) filenames and URIs we recommend a namespace prefix, mapped to a namespace URI in the normal manner. In this case, of course, the namespace URI must point to a real XML document containing _entry_ elements and validated against STMML Schema.

Where there is concern about the dictionary becoming separated from the document the dictionary entries can be physically included as part of the data instance and the normal XPointer addressing mechanism can be used.

This attribute can also be used on _dictionary_ elements to define the namespace

prefix
Example

dimensionality*[att.dimensionality]* ATTGROUP2 Dimensionality of a coordinate system. Note that this means that coordinates of higher dimensionality are ignored or an error is flagged. Thus z3 and dimensionality='2' are incompatible. At present higher dimensionalities than 3 (cf. Wondratschek) are not supported. The labelling of the axes id not controlled. ?? should we have an explicit attribute for labelling convention?. REF no

Dimensionality of a coordinate system.

Note that this means that coordinates of higher dimensionality are ignored or an error is flagged. Thus z3 and dimensionality='2' are incompatible. At present higher dimensionalities than 3 (cf. Wondratschek) are not supported. The labelling of the axes id not controlled. ?? should we have an explicit attribute for labelling convention?.

dimensionBasis*[att.dimensionBasis]*
Allowed values for dimension Types in quantities.
ATTGROUP2 The basis of the dimension. Normally taken from the seven SI types but possibly expandable. REF no

The basis of the dimension.

Normally taken from the seven SI types but possibly expandable.

duration*[att.duration]* ATTGROUP2 The duration of the action. Semantics undefined. REF no

The duration of the action.

Semantics undefined.

electronMap*[att.electronMap]* ATTGROUP2 A reference to a map providing mappings between electrons The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking electrons. The topology of the linking is defined by the application - it could be reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of electrons are of equal size and have 1:1 mapping between each id.

This is another way of saying that the electrons mapped by a given ID are "the same electron". REF no

A reference to a map providing mappings between electrons

The map will normally be contained within the same document and referenced by its ID. It will contain a list of links with from and to attributes linking electrons. The topology of the linking is defined by the application - it could be reactant/product mapping, etc. The reserved phrase "USE_IDS" assume that the sets of electrons are of equal size and have 1:1 mapping between each id. This is another way of saying that the electrons mapped by a given ID are "the same electron".

---

elementType*[att.elementType]*

Allowed elementType values.

ATTGROUP2 The identity of a chemical element. Normally mandatory on _atom_, _isotope_, etc. REF no

The identity of a chemical element.

Normally mandatory on _atom_, _isotope_, etc.

---

elementType*[att.elementTypeArray]*

An array of elementTypes.

ATTGROUP2 The identity of a chemical element. Normally mandatory on _atom_, _isotope_, etc. REF no

The identity of a chemical element.

Normally mandatory on _atom_, _isotope_, etc.

---

end*[att.end]* ATTGROUP2 The end time. The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date. REF no

The end time.

The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date.

---

endCondition*[att.endCondition]* ATTGROUP2 The end condition. At present a human-readable string describing some condition when the ac tion should end. As XML develops it may be possible to add machine-processable semantics in this field.

REF no

The end condition.

At present a human-readable string describing some condition when the ac tion should end. As XML develops it may be possible to add machine-processable semantics in this field.

`errorBasis`*[att.errorBasis]*

The basis of an error value.

ATTGROUP2 Basis of the error estimate. REF no

Basis of the error estimate.

`errorValue`*[att.errorValue]*

An estimate of the error in the value of a quantity.

ATTGROUP2 Value of the error. Reports the author's estimate of the error in a scalar value. Only meaningful for dataTypes mapping to real number. REF no

Value of the error.

Reports the author's estimate of the error in a scalar value. Only meaningful for dataTypes mapping to real number.

`errorValueArray`*[att.errorValueArray]*

Array of error estimate values.

ATTGROUP2 Array of error values. Reports the author's estimate of the error in an array of values. Only meaningful for dataTypes mapping to real number. REF no

Array of error values.

Reports the author's estimate of the error in an array of values. Only meaningful for dataTypes mapping to real number.

`form`*[att.form]*

An XML QName with required prefix.

ATTGROUP2 A reference to a functional form. Currently used for potential. REF no

A reference to a functional form.

Currently used for potential.

---

formalCharge*[att.formalCharge]*

The formal charge on an object.

ATTGROUP2 The formalCharge on the object. NOT the calculated charge or oxidation state. No formal default, but assumed to be zero if omitted. It may become good practice to include it. REF no

The formalCharge on the object.

NOT the calculated charge or oxidation state. No formal default, but assumed to be zero if omitted. It may become good practice to include it.

---

formalCharge*[att.formalChargeArray]*

Array of formalCharges.

ATTGROUP2 An array of formalCharges. Used in CML2 Array mode. NOT the calculated charge or oxidation state. No formal defaults, but assumed to be zero if omitted. It may become good practice to include it. REF no

An array of formalCharges.

Used in CML2 Array mode. NOT the calculated charge or oxidation state. No formal defaults, but assumed to be zero if omitted. It may become good practice to include it.

---

format*[att.reactionFormat]*

The format of the reaction.

ATTGROUP2 Format of the reaction component. Indicates how the components of reactionScheme, reactionStepList, etc. should be processed. No controlled vocabulary. One example is format="cmlSnap" asserts that the processor can assume that the reactants and products can be rendered using the CMLSnap design. Note that the reaction can be interpreted without reference to the format, which is primarily a processing instruction. REF no

Format of the reaction component.

Indicates how the components of reactionScheme, reactionStepList, etc. should be processed. No controlled vocabulary. One example is format="cmlSnap" asserts that the processor can assume that the reactants and products can be rendered using the CMLSnap design. Note that the reaction can be interpreted without reference to the format, which is primarily a processing instruction.

format*[att.format]* ATTGROUP2 Format of a spectrum. The data structure of the spectrum. (Not the format of the data). This describes how the data structure is to be interpreted. one dimensional spectru. Data are represented by two _array_s, one representing the independent variable (e.g. wavelength, mass number) and the other the measured dependent variable (absorption, intensity, etc.). This can normally be plotted directly with the independent variable as the x-axis. The order of the points is not necessarily significant and may be increasing or decreasing. Two dimensional spectru. Data are represented by a single symmetric _matrix_ with both axes identical (i.e. the same independent variable). A typical example is a "2D 1HNMR spectrum". The dependent variable is represented by the matrix elements. This can normally be plotted as a square symmentric about a diagonal. Two dimensional spectrum with different axe. Data are represented by non-square _matrix_ with independent axes. A typical example is a "2D 1H 13C NMR spectrum". The dependent variable is represented by the matrix elements. . REF no

Format of a spectrum.

The data structure of the spectrum. (Not the format of the data). This describes how the data structure is to be interpreted.

UNION OF
Allowed values

- 1D

  one dimensional spectru.

  Data are represented by two _array_s, one representing the independent variable (e.g. wavelength, mass number) and the other the measured dependent variable (absorption, intensity, etc.). This can normally be plotted directly with the independent variable as the x-axis. The order of the points is not necessarily significant and may be increasing or decreasing.

- 2Dsymm

  Two dimensional spectru.

  Data are represented by a single symmetric _matrix_ with both axes identical (i.e. the same independent variable). A typical example is a "2D 1HNMR spectrum". The dependent variable is represented by the matrix elements. This can normally be plotted as a square symmentric about a diagonal.

- 2Dasymm

  Two dimensional spectrum with different axe.

  Data are represented by non-square _matrix_ with independent axes. A typical example is a "2D 1H 13C NMR spectrum". The dependent variable is represented by the matrix elements. .

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example
[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

formula*[att.formula]*

A concise representation for a molecular formula.

ATTGROUP2 Simple chemical formula. This attribute should only be used for simple formulae (i.e. without brackets or other nesting for which a _formula_ child element should be used. The attribute might be used as a check on the child elements or for ease of representation. Essentially the same as _concise_ attribute on _formula. REF no

Simple chemical formula.

This attribute should only be used for simple formulae (i.e. without brackets or other nesting for which a _formula_ child element should be used. The attribute might be used as a check on the child elements or for ease of representation. Essentially the same as _concise_ attribute on _formula.

---

fractionDigits*[att.fractionDigits]* ATTGROUP2 Number of digits after the point. This is used in dictionaries to define precision. However it might be replaced by xsd:facet. REF no

Number of digits after the point.

This is used in dictionaries to define precision. However it might be replaced by xsd:facet.

---

from*[att.from]*

A unique ID for an element.

ATTGROUP2 The base of a link. REF no

The base of a link.

---

ft*[att.ft]*Default:none ATTGROUP2 Domain of an FT spectrum. Indicates whether a spectrum is raw FID or has been transforme. Data are raw, so will normally require transforming. Data have been transformed. This value indicates that an FT experiment and transformation have been performe. This was not known to be an FT experiment. (It may have been, but the author or abstracter omitted to mention it). REF no

Domain of an FT spectrum.

Indicates whether a spectrum is raw FID or has been transforme.

UNION OF
Allowed values

- raw

  Data are raw, so will normally require transforming.
- transformed

  Data have been transformed. This value indicates that an FT experiment and transformation have been performe.
- none

  This was not known to be an FT experiment. (It may have been, but the author or abstracter omitted to mention it).

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example
[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

href*[att.href]* ATTGROUP2 address of a resource. Links to another element in the

same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present. REF no

address of a resource.

Links to another element in the same or other file. For dictionary/@dictRef requires the prefix and the physical URI address to be contained within the same file. We can anticipate that better mechanisms will arise - perhaps through XMLCatalogs. At least it works at present.

`hydrogenCount` *[att.hydrogenCount]*

The total number of hydrogen atoms bonded to an object.

ATTGROUP2 Number of hydrogens. The total number of hydrogens bonded to the atom or molecule. It is preferable to include hydrogens explicitly, and where this is done their count represents the minimum (and may thus override this attribute). It is dangerous to use this attribute for electron-deficient molecules (e.g. diborane) or hydrogen bonds. There is NO DEFAULT and the absence of this attribute must not be given any meaning. REF no

Number of hydrogens.

The total number of hydrogens bonded to the atom or molecule. It is preferable to include hydrogens explicitly, and where this is done their count represents the minimum (and may thus override this attribute). It is dangerous to use this attribute for electron-deficient molecules (e.g. diborane) or hydrogen bonds. There is NO DEFAULT and the absence of this attribute must not be given any meaning.

`hydrogenCount` *[att.hydrogenCountArray]*

Array of hydrogenCounts.

ATTGROUP2 Array of hydrogenCounts. Normally used in CML2 array mode. The total number of hydrogens bonded to the atom or molecule. It is preferable to include hydrogens explicitly, and where this is done their count represents the minimum (and may thus override this attribute). It is dangerous to use this attribute for electron-deficient molecules (e.g. diborane) or hydrogen bonds. There is NO DEFAULT and the absence of this attribute must not be given any meaning. REF no

Array of hydrogenCounts.

Normally used in CML2 array mode. The total number of hydrogens bonded to the atom or molecule. It is preferable to include hydrogens explicitly, and where this is done their count represents the minimum (and may thus override this attribute). It is dangerous to use this attribute for electron-deficient molecules (e.g. diborane) or hydrogen bonds. There is NO DEFAULT and the absence of this attribute must not be given any meaning.

id*[att.id]*
A unique ID for an element.
ATTGROUP2 An attribute providing a unique ID for an element. REF no

An attribute providing a unique ID for an element.

integral*[att.integral]* ATTGROUP2 Area under a peak. Unfortunately units are usually arbitrary and not related to the x- and y- axis units, and in this case _peakUnits_ should be use. REF no

Area under a peak.

Unfortunately units are usually arbitrary and not related to the x- and y- axis units, and in this case _peakUnits_ should be use.

irreducibleRepresentation*[att.irreducibleRepresentation]* ATTGROUP2 A symmetry species. No fixed semantics, though we may provide a controlled-extensible list in the future. REF no

A symmetry species.

No fixed semantics, though we may provide a controlled-extensible list in the future.

isotope*[att.isotope]* ATTGROUP2 The isotope for an element. A real number describing the isotope. Probably obsolet. REF no

The isotope for an element.

A real number describing the isotope. Probably obsolet.

isotopeListRef*[att.isotopeListRef]* ATTGROUP2 Reference to a description of the isotopic composition of an atom. Used when more than one atom shares the same isotopic composition (e.g. when H/D have been scrambled over some or all of the atoms in a molecule.. REF no

Reference to a description of the isotopic composition of an atom.

Used when more than one atom shares the same isotopic composition (e.g. when

H/D have been scrambled over some or all of the atoms in a molecule..
Example

isotopeNumber*[att.isotopeNumber]* ATTGROUP2 The integer number for an isotope. The number representing the isotope. By default it does not point to a fuller description of the isotope (use isotopeRef). REF no

The integer number for an isotope.

The number representing the isotope. By default it does not point to a fuller description of the isotope (use isotopeRef).

isotopeRef*[att.isotopeRef]* ATTGROUP2 Reference to a fuller description of the isotope. The description may be found in an external collection (e.g. IUPAC) or within the current document. REF no

Reference to a fuller description of the isotope.

The description may be found in an external collection (e.g. IUPAC) or within the current document.
Example

kpoint*[att.kpoint]*
A vector in 3-space.
ATTGROUP2 The k vector. The k-vector with 3 components. REF no

The k vector.

The k-vector with 3 components.

l*[att.l]* ATTGROUP2 The secondary quantum number. takes values 0, 1, etc. REF no

The secondary quantum number.

takes values 0, 1, etc.

label*[att.label]* ATTGROUP2 A label. The semantics of label are not defined in the schema but are normally commonly used standard or semi-standard text strings. This attribute has the the same semantics as the more common _label_ elemen. REF no

A label.

The semantics of label are not defined in the schema but are normally commonly used standard or semi-standard text strings. This attribute has the the same semantics as the more common _label_ elemen.

---

latticeType*[att.latticeType]*

Allowed lattice types.

ATTGROUP2 The primitivity of a lattice. No default. The semantics of this are software-dependent (i.e. this Schema does not check for consistency between spacegroups, symmetry operators, etc. REF no

The primitivity of a lattice.

No default. The semantics of this are software-dependent (i.e. this Schema does not check for consistency between spacegroups, symmetry operators, etc.

---

length*[att.length]* ATTGROUP2 Length of a scalar. Probably will be replaced with xsd:schema tool. REF no

Length of a scalar.

Probably will be replaced with xsd:schema tool.

---

linkType*[att.linkType]* ATTGROUP2 The type of the link. A container for locators. A link to an element. A labelled link. REF no

The type of the link.

Allowed values

- extended

  A container for locators.
- locator

  A link to an element.
- arc

  A labelled link.

---

lm*[att.lm]* ATTGROUP2 symbolic represention of l amd m. takes avlues of s, p, px,

dxy, dx2y2, f, etc. REF no

symbolic represention of l amd m.
takes avlues of s, p, px, dxy, dx2y2, f, etc.

Allowed values

- s
- p
- px
- py
- pz
- d
- dxy
- dyz
- dxz
- dx2y2
- dz2
- f
- g

---

m*[att.m]* ATTGROUP2 The azimuthal quantum number. takes values -1, 0, 1, etc. REF no

The azimuthal quantum number.
takes values -1, 0, 1, etc.

---

matrixType*[att.matrixType]*

Allowed matrix types.

ATTGROUP2 Type of matrix. Mainly square, but extensible through the _xsd:union_ mechanis. REF no

Type of matrix.
Mainly square, but extensible through the _xsd:union_ mechanis.

---

max*[att.max]*

The maximum INCLUSIVE value of a quantity.

ATTGROUP2 Maximum value allowed for an element or attribute. REF no

Maximum value allowed for an element or attribute.

---

maxExclusive*[att.maxExclusive]* ATTGROUP2 maximum exclusive value. by analogy with xsd:schema. REF no

maximum exclusive value.

by analogy with xsd:schema.

---

maxInclusive*[att.maxInclusive]* ATTGROUP2 minimum inclusive value. by analogy with xsd:schem. REF no

minimum inclusive value.

by analogy with xsd:schem.

---

maxLength*[att.maxLength]* ATTGROUP2 maximum length of a scalar. by analogy with xsd:schem. REF no

maximum length of a scalar.

by analogy with xsd:schem.

---

maxValueArray*[att.maxValueArray]*

An array of floats.

ATTGROUP2 Maximum values for numeric _matrix_ or _array. A whitespace-separated list of the same length as the array in the parent element. REF no

Maximum values for numeric _matrix_ or _array.

A whitespace-separated list of the same length as the array in the parent element.

---

measurement*[att.measurement]* ATTGROUP2 Type of spectral measurement. The nature of the measured data. This is not an exhaustive list and should only be used if it affects the storage or immediate processing. Data are transmittance, so "peaks" are usually troughs. Data are absorbanc. so "peaks" are normally peaks. REF no

Type of spectral measurement.

The nature of the measured data. This is not an exhaustive list and should only be used if it affects the storage or immediate processing.

UNION OF

Allowed values

- transmittance

Data are transmittance, so "peaks" are usually troughs.

- absorbance

Data are absorbanc.

so "peaks" are normally peaks.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

min*[att.min]*

The minimum INCLUSIVE value of a quantity.

ATTGROUP2 The minimum value allowed for an element or attribute. REF no

The minimum value allowed for an element or attribute.

---

minExclusive*[att.minExclusive]* ATTGROUP2 minimum exclusive value. by analogy with xsd:schema. REF no

minimum exclusive value.

by analogy with xsd:schema.

---

minInclusive*[att.minInclusive]* ATTGROUP2 minimum inclusive value. by analogy with xsd:schema. REF no

minimum inclusive value.

by analogy with xsd:schema.

`minLength`*[att.minLength]* ATTGROUP2 minimum length of a scalar. by analogy with xsd:schema. REF no

minimum length of a scalar.

by analogy with xsd:schema.

`minValueArray`*[att.minValueArray]*

An array of floats.

ATTGROUP2 Minimum values for numeric _matrix_ or _array. A whitespace-separated lists of the same length as the array in the parent element. REF no

Minimum values for numeric _matrix_ or _array.

A whitespace-separated lists of the same length as the array in the parent element.

`moleculeRef`*[att.moleculeRef]*

A reference to an existing molecule.

ATTGROUP2 A reference to a molecule. Used by spectrum, etc. REF no

A reference to a molecule.

Used by spectrum, etc.

`multiplierToData`*[att.multiplierToData]*Default:1.0 ATTGROUP2 The scale by which to multiply the raw data. The scale is applied *before* adding the constant. REF no

The scale by which to multiply the raw data.

The scale is applied *before* adding the constant.

`multiplierToSI`*[att.multiplierToSI]* ATTGROUP2 Multiplier to generate SI equivalent. The factor by which the non-SI unit should be multiplied to convert a quantity to its representation in SI Units. This is applied *before* _constantToSI_. Necessarily unity for SI unit. REF no

Multiplier to generate SI equivalent.

The factor by which the non-SI unit should be multiplied to convert a quantity to its representation in SI Units. This is applied *before* _constantToSI_. Necessarily unity for SI unit.

---

n*[att.n]* ATTGROUP2 The principal quantum number. Takes values 1, 2, 3, etc. REF no

The principal quantum number.

Takes values 1, 2, 3, etc.

---

name*[att.name]* ATTGROUP2 Name of the object. A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary. REF no

Name of the object.

A string by which the object is known. Often a required attribute. The may or may not be a semi-controlled vocabulary.

---

name*[att.metadataType]*

The name of the metadata.

ATTGROUP2 The metadata type. This is likely to be the Dublin Core name or something similar. The use of "type" is an infelicitous misnomer and we shall try to remove it. REF no

The metadata type.

This is likely to be the Dublin Core name or something similar. The use of "type" is an infelicitous misnomer and we shall try to remove it.

---

number*[att.number]* ATTGROUP2 A number determined by context Used for isotope number in isotope, and rotational symmetry number in symmetry for calculation of entropy, etc. 2003-03-30: added number attribut. REF no

A number determined by context

Used for isotope number in isotope, and rotational symmetry number in symmetry for calculation of entropy, etc.

2003-03-30: added number attribut.

`objectClass`*[att.objectClass]* ATTGROUP2 The class of an object. The type of this information. This is not controlled, but examples might include: label summary note usage qualifier It might be used to control display or XSL filtering. The attribute is named 'objectClass' to avoid clashes with other class attributes and inappropriate conversion to foo.getClass(). REF no

The class of an object.

The type of this information. This is not controlled, but examples might include:

- label
- summary
- note
- usage
- qualifier

It might be used to control display or XSL filtering.

The attribute is named 'objectClass' to avoid clashes with other class attributes and inappropriate conversion to foo.getClass().

`occupancy`*[att.occupancy]*

A floating point number between 0 and 1 inclusive

ATTGROUP2 Occupancy for an atom. Normally only found in crystallography. Defaults to 1.0. The occupancy is required to calculate the molecular formaula from the atoms. REF no

Occupancy for an atom.

Normally only found in crystallography. Defaults to 1.0. The occupancy is required to calculate the molecular formaula from the atoms.

`occupancy`*[att.occupancyArray]*

Array of atomic occupancies.

ATTGROUP2 Array of occupancies. Normally only found in crystallography. Defaults to 1.0. The occupancy is required to calculate the molecular formula from the atoms. REF no

Array of occupancies.

Normally only found in crystallography. Defaults to 1.0. The occupancy is required to calculate the molecular formula from the atoms.

order*[att.actionOrder]* ATTGROUP2 Describes whether child elements are sequential or parallel. There is no default. REF no

Describes whether child elements are sequential or parallel.

There is no default.

Allowed values

- sequential
- parallel

---

order*[att.order]*

Bond order.

ATTGROUP2 The order of the bond. There is NO default. This order is for bookkeeping only and is not related to length, QM calculations or other experimental or theoretical calculations. REF no

The order of the bond.

There is NO default. This order is for bookkeeping only and is not related to length, QM calculations or other experimental or theoretical calculations.

---

order*[att.orderArray]*

An array of bond orders.

ATTGROUP2 The order of the bond. There is NO default. This order is for bookkeeping only and is not related to length, QM calculations or other experimental or theoretical calculations. REF no

The order of the bond.

There is NO default. This order is for bookkeeping only and is not related to length, QM calculations or other experimental or theoretical calculations.

---

parentSI*[att.parentSI]*

An XML QName with required prefix.

ATTGROUP2 A dictRef like reference to the id of the parent SI unit. This parent should occur in this or another dictionary and be accessible through the dictRef mechanism. This attribute is forbidden for SI Units themselves. The mechanism holds for base SI units (7) and all compound (derived) units made by combinations of base Units. REF no

A dictRef like reference to the id of the parent SI unit.

This parent should occur in this or another dictionary and be accessible through the dictRef mechanism. This attribute is forbidden for SI Units themselves. The

mechanism holds for base SI units (7) and all compound (derived) units made by combinations of base Units.

Example

---

pattern*[att.pattern]* ATTGROUP2 Pattern constraint. Based on xsd:schema. REF no

Pattern constraint.

Based on xsd:schema.

---

peakHeight*[att.peakHeight]* ATTGROUP2 Height of a peak. For 1-dimensional data (e.g. y vs x) hould use the same units as the appropriate axis (e.g. y). REF no

Height of a peak.

For 1-dimensional data (e.g. y vs x) hould use the same units as the appropriate axis (e.g. y).

---

peakMultiplicity*[att.peakMultiplicity]* ATTGROUP2 Multiplicity of a peak. Uses a semi-controlled vocabulary. A single maximum within the peak rang. Two maxima (not necessarily equal) within the peak rang. Three maxima (not necessarily equal) within the peak rang. Four maxima (not necessarily equal) within the peak rang. Five maxima (not necessarily equal) within the peak rang. Six maxima (not necessarily equal) within the peak rang. Several maxima (not necessarily equal) within the peak rang. User contributed vocabulary of type foo:ba. REF no

Multiplicity of a peak.

Uses a semi-controlled vocabulary.

UNION OF

Allowed values

- singlet

  A single maximum within the peak rang.
- doublet

  Two maxima (not necessarily equal) within the peak rang.
- triplet

  Three maxima (not necessarily equal) within the peak rang.
- quartet

  Four maxima (not necessarily equal) within the peak rang.

- quintet

  Five maxima (not necessarily equal) within the peak rang.
- sextuplet

  Six maxima (not necessarily equal) within the peak rang.
- multiplet

  Several maxima (not necessarily equal) within the peak rang.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example
[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

User contributed vocabulary of type foo:ba.

---

peakShape*[att.peakShape]* ATTGROUP2 Shape of a peak. Semi-controlled vocabulary such as broad or sharp. A sharp peak. A broad peak. A brodening of a peak suggesting the presence of a smaller incompletely resolved component. User contributed vocabulary of type foo:bar. REF no

Shape of a peak.

Semi-controlled vocabulary such as broad or sharp.

UNION OF
Allowed values

- sharp

  A sharp peak.
- broad

  A broad peak.
- shoulder

A brodening of a peak suggesting the presence of a smaller incompletely resolved component.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

User contributed vocabulary of type foo:bar.

---

peakUnits*[att.peakUnits]*

Scientific units.

ATTGROUP2 Units for a peak or peak integral. For 2-dimensional spectra the units represent the observation. For an integral they are usually arbitrary and not related to the x- and y- axis units. Thus NMR spectra may use hydrogen count as the units for the peak area. REF no

Units for a peak or peak integral.

For 2-dimensional spectra the units represent the observation. For an integral they are usually arbitrary and not related to the x- and y- axis units. Thus NMR spectra may use hydrogen count as the units for the peak area.

---

periodic*[att.periodic]*Default:true ATTGROUP2 Is the axis periodic. Any or all of the axes may be periodic or aperiodic. An example could be a surface where 2 periodic axes (not necessarily orthogonal) are used to describe the coordinates in the surface, perhaps representing lattice vectors of a 3D crystal or 2D layer. The third vector is orthogonal and represents coordinates normal to the surface. In this case only the direction, not the magnitude of the vector is important. REF no

Is the axis periodic.

Any or all of the axes may be periodic or aperiodic. An example could be a surface

where 2 periodic axes (not necessarily orthogonal) are used to describe the coordinates in the surface, perhaps representing lattice vectors of a 3D crystal or 2D layer. The third vector is orthogonal and represents coordinates normal to the surface. In this case only the direction, not the magnitude of the vector is important.

---

periodicity*[att.periodicity]* ATTGROUP2 Periodicity of the system. This represents the number of dimensions (or coordinate axes) along periodic behaviour occurs and can be supported by symmetry operators or other transformations. Periodicity must never exceed dimensionality. REF no

Periodicity of the system.

This represents the number of dimensions (or coordinate axes) along periodic behaviour occurs and can be supported by symmetry operators or other transformations. Periodicity must never exceed dimensionality.

---

pointGroup*[att.pointGroup]* ATTGROUP2 A point group. No fixed semantics, though Schoenflies is recommended over Hermann-Mauguin. We may provide a controlled-extensible list in the future. REF no

A point group.

No fixed semantics, though Schoenflies is recommended over Hermann-Mauguin. We may provide a controlled-extensible list in the future.

---

power*[att.power]*REQUIRED ATTGROUP2 The power to which a dimension should be raised. Normally an integer. Must be included, even if unity. REF no

The power to which a dimension should be raised.

Normally an integer. Must be included, even if unity.

---

preserve*[att.preserve]* ATTGROUP2 Is the dimension preserved during algebra. Experimental. The idea is to support concepts like volume/volume where alebraically these cancel out. preserve="yes" is intending to support preservation during derivation of new unitTypes. REF no

Is the dimension preserved during algebra.

Experimental. The idea is to support concepts like volume/volume where alebraically these cancel out. preserve="yes" is intending to support preservation during derivation of new unitTypes.

`ratio`*[att.ratio]*

A floating point number between 0 and 1 inclusive

ATTGROUP2 A ratio in the range 0 to 1. Currently used for ratios between brached reactions but re-usable for other concepts. REF no

A ratio in the range 0 to 1.

Currently used for ratios between brached reactions but re-usable for other concepts.

`ref`*[att.ref]*

A reference to an existing object.

ATTGROUP2 A reference to an element of given type. ref modifies an element into a reference to an existing element of that type within the document. This is similar to a pointer and it can be thought of a strongly typed hyperlink. It may also be used for "subclassing" or "overriding" elements. When referring to an element most of the "data" such as attribute values and element content will be on the full instantiated element. Therefore ref (and possibly id) will normally be the only attributes on the pointing element. However there may be some attributes (title, count, etc.) which have useful semantics, but these are element-specific REF no

A reference to an element of given type.

`ref` modifies an element into a reference to an existing element of that type within the document. This is similar to a pointer and it can be thought of a strongly typed hyperlink. It may also be used for "subclassing" or "overriding" elements.
When referring to an element most of the "data" such as attribute values and element content will be on the full instantiated element. Therefore ref (and possibly id) will normally be the only attributes on the pointing element. However there may be some attributes (title, count, etc.) which have useful semantics, but these are element-specific
Example

`regionRefs`*[att.regionRefs]*

A reference to an existing object.

ATTGROUP2 A list of regions creating a union. The union of a series of regions produces a larger region (possibly disjoint). Any point belonging to any of the referenced regions is a member of this region. REF no

A list of regions creating a union.

The union of a series of regions produces a larger region (possibly disjoint). Any point belonging to any of the referenced regions is a member of this region.

role*[att.role]* ATTGROUP2 Role of the object. How the object functions or its position in the architecture. No controlled vocabulary. REF no

Role of the object.

How the object functions or its position in the architecture. No controlled vocabulary.

role*[att.reactionRole]*

The role of the reaction within a reactionList.

ATTGROUP2 Role of the reaction. REF no

Role of the reaction.

rows*[att.rows]*

The size of an array.

ATTGROUP2 Number of rows. REF no

Number of rows.

serial*[att.serial]* ATTGROUP2 Serial number or other id. Currently only on module. Modules with the same _role_ attribute can be distinguished by _serial_. This is often an integer but other schemes may be used. REF no

Serial number or other id.

Currently only on module. Modules with the same _role_ attribute can be distinguished by _serial_. This is often an integer but other schemes may be used.

size*[att.size]*

The size of an array.

ATTGROUP2 The size of an array or matrix. REF no

The size of an array or matrix.

spaceGroup*[att.spaceGroup]* ATTGROUP2 A space group. No fixed semantics, though Hermann-Mauguin or Hall is recommended over Schoenflies. We may provide a controlled-extensible list in the future. REF no

A space group.

No fixed semantics, though Hermann-Mauguin or Hall is recommended over Schoenflies. We may provide a controlled-extensible list in the future.

---

spaceType*[att.spaceType]*

Signifies real or reciprocal space.

ATTGROUP2 The spaceType of the lattice. Usually real or reciprocal. No default. The semantics of this are software-dependent (i.e. this Schema does not check for consistency for unitTypes, etc. REF no

The spaceType of the lattice.

Usually real or reciprocal. No default. The semantics of this are software-dependent (i.e. this Schema does not check for consistency for unitTypes, etc.

---

sphere3*[att.sphere3]*

A sphere in 3-space.

ATTGROUP2 A sphere. Currently describes a region. Any point falling within the sphere or on its surface is within the region. REF no

A sphere.

Currently describes a region. Any point falling within the sphere or on its surface is within the region.

---

spin*[att.spin]*

A fractional representation of the spin of the nucleus.

ATTGROUP2 The spin of a system. Supports fractional values. Currently the spin of a nucleus. The normal fraction representing the spin of the isotope. REF no

The spin of a system.

Supports fractional values. Currently the spin of a nucleus. The normal fraction representing the spin of the isotope.

Example

---

spinMultiplicity*[att.spinMultiplicity]* ATTGROUP2 Spin multiplicity. Normally for a molecule. This attribute gives the spin multiplicity of the molecule and is independent of any atomic information. No default, and it may take any positive integer value (though values are normally between 1 and 5. REF no

Spin multiplicity.

Normally for a molecule. This attribute gives the spin multiplicity of the molecule and is independent of any atomic information. No default, and it may take any positive integer value (though values are normally between 1 and 5.

---

start*[att.start]* ATTGROUP2 The start time. The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date. REF no

The start time.

The start time in any allowable XSD representation of date, time or dateTime. This will normally be a clock time or date.

---

startCondition*[att.startCondition]* ATTGROUP2 The start condition. This can describe the condition(s) that has to be met before an action can begin, such as in a recipe. Semantics are unexplored but could be used to control robotic operations. REF no

The start condition.

This can describe the condition(s) that has to be met before an action can begin, such as in a recipe. Semantics are unexplored but could be used to control robotic operations.

---

state*[att.state]*

State of a substance or property.

ATTGROUP2 The physical state of the substance. No fixed semantics or default. REF no

The physical state of the substance.

No fixed semantics or default.

---

symbol*[att.symbol]* ATTGROUP2 A symbol. Currently only used on _atomicBasisFunction_. REF no

A symbol.

Currently only used on _atomicBasisFunction_.

`symmetryOriented`*[att.symmetryOriented]* ATTGROUP2 Is the molecule oriented to the symmetry No formal default, but a molecule is assumed to be oriented according to any _symmetry_ children. This is required for crystallographic data, but some systems for isolated molecules allow specification of arbitrary Cartesian or internal coordinates, which must be fitted or refined to a prescribed symmetry. In this case the attribute value is false. REF no

Is the molecule oriented to the symmetry

No formal default, but a molecule is assumed to be oriented according to any _symmetry_ children. This is required for crystallographic data, but some systems for isolated molecules allow specification of arbitrary Cartesian or internal coordinates, which must be fitted or refined to a prescribed symmetry. In this case the attribute value is false.

`tautomeric`*[att.tautomeric]* ATTGROUP2 Indicates whether the structure is a tautomer. Currently used with IChI _identifier_ element. Semantics, vocabulary and usage are application-dependent. REF no

Indicates whether the structure is a tautomer.

Currently used with IChI _identifier_ element. Semantics, vocabulary and usage are application-dependent.

`term`*[att.term]*REQUIRED ATTGROUP2 A term in a dictionary. The term should be a noun or nounal phrase, with a separate definition and further description. REF no

A term in a dictionary.

The term should be a noun or nounal phrase, with a separate definition and further description.

`title`*[att.title]* ATTGROUP2 A title on an element. No controlled value. REF no

A title on an element.

No controlled value.

Example

to*[att.to]*
A unique ID for an element.
ATTGROUP2 The target of a link. REF no

The target of a link.

totalDigits*[att.totalDigits]* ATTGROUP2 total digits in a scalar. based on xsd:schema. REF no

total digits in a scalar.

based on xsd:schema.

type*[att.type]* ATTGROUP2 Type of the object. A qualifier which may affect the semantics of the object. REF no

Type of the object.

A qualifier which may affect the semantics of the object.

type*[att.alternativeType]* ATTGROUP2 The type of an alternative. This adds semantics to an _alternative_ and might be used by an RDF or related engine. REF no

The type of an alternative.

This adds semantics to an _alternative_ and might be used by an RDF or related engine.

UNION OF
Allowed values

- synonym
- quasi-synonym
- acronym
- abbreviation
- homonym
- identifier

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

type*[att.reactionType]*

The semantic type of the reaction.

ATTGROUP2 Type of the reaction. REF no

Type of the reaction.

---

type*[att.relatedEntryType]* ATTGROUP2 Type of relatedEntry. Type represents a the type of relationship in a relatedEntry element. REF no

Type of relatedEntry.

Type represents a the type of relationship in a relatedEntry element.

UNION OF

Allowed values

- parent
- partitiveParent
- child
- partitiveChild
- related
- synonym
- quasi-synonym
- antonym
- homonym
- see
- seeAlso
- abbreviation
- acronym

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

type*[att.spectrumType]* ATTGROUP2 The type of the spectrum. An infrared spectrum. The measurement should denote transmittance or absorbanc. A "simple" mass spectrum. This excludes experiments such as GC/MS, MS/MS, etc. though these could be constructed out of individual spectra with some care. The spectrum may be continuous (data or a peakList). An NMR spectrum. This can include any experiment which creates a "1D" or "2D" data array. The symmetry of the spectrum can be specified but the details of the NMR experiment (COSY, NOESY, etc.) are not part of CMLSpect. They can be described though the normal dictRef mechanism. A spectrum somewhere in the UV VIS region of the spectrum. The measurement should denote transmittance or absorbance. REF no

The type of the spectrum.

UNION OF

Allowed values

- infrared

  An infrared spectrum.

  The measurement should denote transmittance or absorbanc.

- massSpectrum

  A "simple" mass spectrum.

  This excludes experiments such as GC/MS, MS/MS, etc. though these could be constructed out of individual spectra with some care. The spectrum may be continuous (`data` or a `peakList`).

- NMR

  An NMR spectrum.

  This can include any experiment which creates a "1D" or "2D" data array. The symmetry of the spectrum can be specified but the details of the NMR experiment (COSY, NOESY, etc.) are not part of CMLSpect. They can be described though the normal `dictRef` mechanism.

- UV/VIS

A spectrum somewhere in the UV VIS region of the spectrum.

The measurement should denote transmittance or absorbance.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

`type`*[att.substanceListType]* ATTGROUP2 Type of the substanceList. Extension is allowed through the "other" value. REF no

Type of the substanceList.

Extension is allowed through the "other" value.

Allowed values

- solution
- mixture
- other

`units`*[att.units]*

Scientific units.

ATTGROUP2 Scientific units on an element. These must be taken from a dictionary of units. There should be some mechanism for validating the type of the units against the possible values of the element. REF no

Scientific units on an element.

These must be taken from a dictionary of units. There should be some mechanism for validating the type of the units against the possible values of the element.

---

units*[att.angleUnits]*
An enumeration of allowed angle units.
ATTGROUP2 Restricts units to radians or degrees. REF no

Restricts units to radians or degrees.

---

unitType*[att.unitType]* ATTGROUP2 A reference to the type of a unit. Used in defining the unit and doing symbolic algebra on the dimensionality. REF no

A reference to the type of a unit.

Used in defining the unit and doing symbolic algebra on the dimensionality.

---

value*[att.value]* ATTGROUP2 Value of a scalar object. The value must be consistent with the dataType of the object. REF no

Value of a scalar object.

The value must be consistent with the dataType of the object.

---

version*[att.version]* ATTGROUP2 The version of the identifier. The IChI or other identifier may be dependent on the date of release and this attribute is highly recommended. REF no

The version of the identifier.

The IChI or other identifier may be dependent on the date of release and this attribute is highly recommended.

---

weight*[att.weight]* ATTGROUP2 Weight of the element. Currently the weight of the kPoint, derived from the symmetry such as the inverse of the multiplicity in real space. Thus a point at 0,0,0 in monoclinic space might be 0.25. The lowest value possible is probably 1/48.0 (in m3m). 2003-09-15 (added at suggestion of Jon Wakelin). REF no

Weight of the element.

Currently the weight of the kPoint, derived from the symmetry such as the inverse of the multiplicity in real space. Thus a point at 0,0,0 in monoclinic space might be 0.25. The lowest value possible is probably 1/48.0 (in m3m).

2003-09-15 (added at suggestion of Jon Wakelin).

whiteSpace*[att.whiteSpace]* ATTGROUP2 Whitespace. Attached to entry. This may be obsolete. REF no

Whitespace.

Attached to entry. This may be obsolete.

x2*[att.x2]* ATTGROUP2 x2 coordinate for an object. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object. REF no

x2 coordinate for an object.

Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object.

x2*[att.x2Array]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 array of x2 coordinate. Normally used in CML2 array mode. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object. REF no

array of x2 coordinate.

Normally used in CML2 array mode. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object.

x3*[att.x3]* ATTGROUP2 The x coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF no

The x coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

x3*[att.x3Array]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 An array of x3 coordinate. Normally used in CML2 array mode. REF no

An array of x3 coordinate.

Normally used in CML2 array mode.

---

xFract*[att.xFract]* ATTGROUP2 Fractional x coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF no

Fractional x coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

---

xFract*[att.xFractArray]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 Array of fractional x coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF no

Array of fractional x coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

---

xMax*[att.xMax]* ATTGROUP2 Maximum yValue. Annotates x-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMin_ attribute but if so xMax should be greater than or equals to it. REF no

Maximum yValue.

Annotates x-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMin_ attribute but if so xMax should be greater than or equals to it.

---

xMin*[att.xMin]* ATTGROUP2 Minimum yValue. Annotates x-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the

data. There may or may not be a _xMax_ attribute but if so xMin should be less than or equals to it. REF no

Minimum yValue.

Annotates x-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. There may or may not be a _xMax_ attribute but if so xMin should be less than or equals to it.

---

xUnits*[att.xUnits]*

Scientific units.

ATTGROUP2 Units for x axis. All x-axis data must have unambiguous units. Ideally the data and _xMin_ or _xValue_ should share the same units but different xUnits can be used as long as it is clear.. REF no

Units for x axis.

All x-axis data must have unambiguous units. Ideally the data and _xMin_ or _xValue_ should share the same units but different xUnits can be used as long as it is clear..

---

xValue*[att.xValue]* ATTGROUP2 Value along an x axis. Annotates x-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data. REF no

Value along an x axis.

Annotates x-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses xUnits or the same units as the data.

---

xWidth*[att.xWidth]* ATTGROUP2 An unsigned interval along an x axis. It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses xUnits or the same units as the data. REF no

An unsigned interval along an x axis.

It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses xUnits or the same units as the data.

---

y2*[att.y2]* ATTGROUP2 y2 coordinate for an object. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the

axes matters as it can affect the chirality of object. REF no

y2 coordinate for an object.

Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object.

---

y2*[att.y2Array]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 array of y2 coordinate. Normally used in CML2 array mode. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object. REF no

array of y2 coordinate.

Normally used in CML2 array mode. Used for displaying the object in 2 dimensions. Unrelated to the 3-D coordinates for the object. The orientation of the axes matters as it can affect the chirality of object.

---

y3*[att.y3]* ATTGROUP2 The y coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF no

The y coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

---

y3*[att.y3Array]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 An array of y3 coordinate. Normally used in CML2 array mode. REF no

An array of y3 coordinate.

Normally used in CML2 array mode.

---

yFract*[att.yFract]* ATTGROUP2 Fractional y coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF no

Fractional y coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a

crystal element should also occur.

---

yFract*[att.yFractArray]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 Array of fractional y coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF no

Array of fractional y coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

---

yield*[att.yield]*

A floating point number between 0 and 1 inclusive

ATTGROUP2 Yield of a reaction or reactionStep. Yields can be given on either element. They should lie in the range 0 to 1 inclusive (i.e. percentages will need to be converted). Software may use yield to calculate amounts of substances created during a reaction or series of reactions. REF no

Yield of a reaction or reactionStep.

Yields can be given on either element. They should lie in the range 0 to 1 inclusive (i.e. percentages will need to be converted). Software may use yield to calculate amounts of substances created during a reaction or series of reactions.

---

yMax*[att.yMax]* ATTGROUP2 Maximum yValue. Annotates y-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMin_ attribute but if so yMax should be greater than or equals to it. REF no

Maximum yValue.

Annotates y-axis data with a maximum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMin_ attribute but if so yMax should be greater than or equals to it.

---

yMin*[att.yMin]* ATTGROUP2 Minimum yValue. Annotates y-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used

for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMax_ attribute but if so yMin should be less than or equal to it. REF no

Minimum yValue.

Annotates y-axis data with a minimum value. This need not be algorithmically deducible from the data and is typically used for the extent of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. There may or may not be a _yMax_ attribute but if so yMin should be less than or equal to it.

---

yUnits*[att.yUnits]*
Scientific units.
ATTGROUP2 Units for y axis. All y-axis data must have unambiguous units. Ideally the data and _yMin_ or _yValue_ should share the same units but different yUnits can be used as long as it is clear. REF no

Units for y axis.

All y-axis data must have unambiguous units. Ideally the data and _yMin_ or _yValue_ should share the same units but different yUnits can be used as long as it is clear.

---

yValue*[att.yValue]* ATTGROUP2 Value along a y axis. Annotates y-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data. REF no

Value along a y axis.

Annotates y-axis data with a value. It is typically used for the location of a _peak_ or _peakGroup_. It uses yUnits or the same units as the data.

---

yWidth*[att.yWidth]* ATTGROUP2 An unsigned interval along a y axis. It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses yUnits or the same units as the data. REF no

An unsigned interval along a y axis.

It is typically used for the width of a _peak_ or _peakGroup_ but could be used for any range. It uses yUnits or the same units as the data.

---

z*[att.z]* ATTGROUP2 The number of molecules per cell. Molecules are defined as the

molecule which directly contains the _crystal_ element. REF no

The number of molecules per cell.

Molecules are defined as the _molecule_ which directly contains the _crystal_ element.

---

z3*[att.z3]* ATTGROUP2 The z coordinate of a 3 dimensional object. The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system. REF no

The z coordinate of a 3 dimensional object.

The default units are Angstrom. (The provision for other units is weak at present.) Objects are always described with a right-handed coordinate system.

---

z3*[att.z3Array]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 An array of z3 coordinate. Normally used in CML2 array mode. REF no

An array of z3 coordinate.

Normally used in CML2 array mode.

---

zFract*[att.zFract]* ATTGROUP2 Fractional y coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF no

Fractional y coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

---

zFract*[att.zFractArray]*

An array of coordinateComponents for a single coordinate.

ATTGROUP2 Array of fractional z coordinate. normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur. REF no

Array of fractional z coordinate.

normally xFract, yFract and zFract should all be present or absent. If present a _crystal_ element should also occur.

SIMPLETYPES

angleUnitsType[st.angleUnitsType]

An enumeration of allowed angle units.

May be obsolete.

Allowed values

- degrees
- radians

atomIDType[st.atomIDType]

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.

Example

[xsd:string]

Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?

atomRefArrayType[st.atomRefArrayType]

An array of atomRefs.

The atomRefs cannot be schema- or schematron-validated. Instances of this type will be used in array-style representation of bonds and atomParitys. It can also be used for arrays of atomIDTypes such as in complex setereochemistry, geometrical definitions, atom groupings, etc.

Example
XSD:LIST of atomIDType

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.
Example
[xsd:string]
Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?

---

atomRefs2Type[st.atomRefs2Type]

A reference to two distinct existing atoms in order.
Example
BASE:
XSD:LIST of atomIDType

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.
Example
[xsd:string]
Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?
xsd:length
2

---

atomRefs3Type[st.atomRefs3Type]

A reference to three distinct existing atoms in order.
Example
BASE:
XSD:LIST of atomIDType

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.
Example
[xsd:string]
Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?
xsd:length
3

---

atomRefs4Type[st.atomRefs4Type]

A reference to four distinct existing atoms in order.
Example
BASE:
XSD:LIST of atomIDType

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.
Example
[xsd:string]
Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?
xsd:length
4

atomRefType[st.atomRefType]

A reference to an existing atom.

Example

BASE: atomIDType

An identifier for an atom.

Of the form prefix:suffix where prefix and suffix are purely alphanumeric (with _ and -) and prefix is optional. This is similar to XML IDs (and we promote this as good practice for atomIDs. Other punctuation and whitespace is forbidden, so IDs from (say) PDB files are not satisfactory.

The prefix is intended to form a pseudo-namespace so that atom IDs in different molecules may have identical suffixes. It is also useful if the prefix is the ID for the molecule (though this clearly has its limitation). Atom IDs should not be typed as XML IDs since they may not validate.

Example

[xsd:string]

Pattern: [A-Za-z_][A-Za-z0-9_\-]*(:[A-Za-z0-9_\-]+)?

bondRefArrayType[st.bondRefArrayType]

An array of references to bonds.

The references cannot (yet) cannot be schema- or schematron-validated. Instances of this type will be used in array-style representation of electron counts, etc. It can also be used for arrays of bondIDTypes such as in complex stereochemistry, geometrical definitions, bond groupings, etc.

XSD:LIST of bondRefType

A reference to an existing bond.

A reference to a bond may be made by atoms (e.g. for multicentre or pi-bonds), electrons (for annotating reactions or describing electronic properties) or possibly other bonds (no examples yet). The semantics are relatively flexible.

Example

[xsd:string]

Pattern: [A-Za-z0-9_\-]+(:[A-Za-z0-9_\-]+)?

bondRefType[st.bondRefType]

A reference to an existing bond.

A reference to a bond may be made by atoms (e.g. for multicentre or pi-bonds), electrons (for annotating reactions or describing electronic properties) or possibly other bonds (no examples yet). The semantics are relatively flexible.

Example

[xsd:string]
Pattern: [A-Za-z0-9_\-]+(:[A-Za-z0-9_\-]+)?

---

box3Type[st.box3Type]

A box in 3-space.

Defined by 6 real numbers (x1 y1 z1 x2 y2 z2). By default these are Cartesian coordinates (with units specified elsewhere - responsibility of schema creator.) If there is a means of specifying oblique axes (e.g. crystallographic cell) the box may be a parallelipiped. The components are grouped in threes ans separated by a semicolon to avoid problems of guessing the convention.

Example

BASE:
XSD:LIST of xsd:float
xsd:length
6

---

chiralityType[st.chiralityType]

The chirality of a system or molecule.

This is being actively investigated by a IUPAC committee (2002) so the convention is likely to change. No formal default.

Allowed values

- enantiomer
- racemate
- unknown
- other

---

coordinateComponentArrayType[st.coordinateComponentArrayType]

An array of coordinateComponents for a single coordinate.

An array of coordinateComponents for a single coordinate where these all refer to an X-coordinate (NOT x,y,z).Instances of this type will be used in array-style representation of 2-D or 3-D coordinates. Currently no machine validation. Currently not used in STMML, but re-used by CML (see example).

Example
XSD:LIST of xsd:float

---

countArrayType[st.countArrayType]

Array of counts.
XSD:LIST of countType

A count multiplier for an object.

Many elements represent objects which can occur an arbitrary number of times in a scientific context. Examples are `action`, `object` or `molecule`s.
Example
[xsd:nonNegativeInteger]

---

countType[st.countType]

A count multiplier for an object.

Many elements represent objects which can occur an arbitrary number of times in a scientific context. Examples are `action`, `object` or `molecule`s.
Example
[xsd:nonNegativeInteger]

---

dataTypeType[st.dataTypeType]

an enumerated type for all dataTypes in STM.

`dataTypeType` represents an enumeration of allowed dataTypes (at present identical with those in XML-Schemas (Part2- datatypes). This means that implementers should be able to use standard XMLSchema-based tools for validation without major implementation problems.

It will often be used an an attribute on **scalar**, **array** or **matrix** elements.

Note: the attribute xsi:type might be used to enforce the type-checking but I haven't worked this through yet.
Example
UNION OF
Allowed values
- xsd:string
- xsd:boolean
- xsd:float
- xsd:double

- xsd:decimal
- xsd:duration
- xsd:dateTime
- xsd:time
- xsd:date
- xsd:gYearMonth
- xsd:gYear
- xsd:gMonthDay
- xsd:gDay
- xsd:gMonth
- xsd:hexBinary
- xsd:base64Binary
- xsd:anyURI
- xsd:QName
- xsd:NOTATION
- xsd:normalizedString
- xsd:token
- xsd:language
- xsd:IDREFS
- xsd:ENTITIES
- xsd:NMTOKEN
- xsd:NMTOKENS
- xsd:Name
- xsd:NCName
- xsd:ID
- xsd:IDREF
- xsd:ENTITY
- xsd:integer
- xsd:nonPositiveInteger
- xsd:negativeInteger
- xsd:long
- xsd:int
- xsd:short
- xsd:byte
- xsd:nonNegativeInteger
- xsd:unsignedLong
- xsd:unsignedInt
- xsd:unsignedShort
- xsd:unsignedByte
- xsd:positiveInteger

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example
[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'
Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

delimiterType[st.delimiterType]

A single non-whitespace character to separate components in arrays.

Some STMML elements (such as **array**) have content representing concatenated values. The default separator is whitespace (which can be normalised) and this should be used whenever possible. However in some cases the values are empty, or contain whitespace or other problematic punctuation, and a delimiter is required.

Note that the content string MUST start and end with the delimiter so there is no ambiguity as to what the components are. Only printable characters from the ASCII character set should be used, and character entities should be avoided.

When delimiters are used to separate precise whitespace this should always consist of spaces and not the other allowed whitespace characters (newline, tabs, etc.). If the latter are important it is probably best to redesign the application.

At present there is a controlled pattern of characters selected so as not to collide with common usage in XML document
Example

```
                The values in the array are
  "A", "B12", "" (empty string) and "D and   E"
 note the spaces
```

[xsd:string]
Pattern: [\!\$\%\^*\@\~\;\#\,\/\|]

---

dimensionType[st.dimensionType]

Allowed values for dimension Types in quantities.

These are the 7 types prescribed by the SI system, together with the "dimensionless" type. We intend to be somewhat uncoventional and explore enhanced values of "dimensionless", such as "angle". This may be heretical, but we find the present system impossible to implement in many cases.

Used for constructing entries in a dictionary of units
Example
Allowed values
- mass
- length
- time
- current
- amount
- luminosity
- temperature
- dimensionless
- angle

An angl.
(formally dimensionless, but useful to have units).

---

elementTypeArrayType[st.elementTypeArrayType]

An array of elementTypes.
Instances of this type will be used in array-style representation of atoms.
Example
XSD:LIST of elementTypeType

Allowed elementType values.

The periodic table (up to element number 118. In addition the following strings are allowed:
- Du. ("dummy") This does not correspond to a "real" atom and can support a point in space or within a chemical graph.
- R. ("R-group") This indicates that an atom or group of atoms could be attached at this point.

Example
UNION OF
Allowed values
- Ac
- Al
- Ag
- Am
- Ar
- As
- At
- Au
- B
- Ba
- Bh

- Bi
- Be
- Bk
- Br
- C
- Ca
- Cd
- Ce
- Cf
- Cl
- Cm
- Co
- Cr
- Cs
- Cu
- Db
- Dy
- Er
- Es
- Eu
- F
- Fe
- Fm
- Fr
- Ga
- Gd
- Ge
- H

  Any isotope of hydrogen.

  There are no special element symbols for D and T which should use the `isotope` attribute.
- He
- Hf
- Hg
- Ho
- Hs
- I
- In
- Ir
- K
- Kr
- La
- Li
- Lr
- Lu
- Md
- Mg

- Mn
- Mo
- Mt
- N
- Na
- Nb
- Nd
- Ne
- Ni
- No
- Np
- O
- Os
- P
- Pa
- Pb
- Pd
- Pm
- Po
- Pr
- Pt
- Pu
- Ra
- Rb
- Re
- Rf
- Rh
- Rn
- Ru
- S
- Sb
- Sc
- Se
- Sg
- Si
- Sm
- Sn
- Sr
- Ta
- Tb
- Tc
- Te
- Th
- Ti
- Tl
- Tm
- U
- Uun
- Uuu

- Uub
- Uut
- Uuq
- Uup
- Uuh
- Uus
- Uuo
- V
- W
- Xe
- Y
- Yb
- Zn
- Zr
- Dummy
- Du

  A point or object with no chemical semantics.

  Examples can be centroids, bond-midpoints, orienting "atoms" in small z-matrices.

  Note "Dummy" has the same semantics but is now deprecated.
- R

  A point at which an atom or group might be attached.

  Examples are abbreviated organic functional groups, Markush representations, polymers, unknown atoms, etc. Semantics may be determined by the `role` attribute on the atom.

[xsd:string]
Pattern: [A-Za-z]+:[A-Za-z][A-Za-z0-9\-]+

---

elementTypeType[st.elementTypeType]

Allowed elementType values.

The periodic table (up to element number 118. In addition the following strings are allowed:

- `Du`. ("dummy") This does not correspond to a "real" atom and can support a point in space or within a chemical graph.
- `R`. ("R-group") This indicates that an atom or group of atoms could be attached at this point.

Example
UNION OF
Allowed values

- Ac
- Al
- Ag
- Am
- Ar
- As
- At
- Au
- B
- Ba
- Bh
- Bi
- Be
- Bk
- Br
- C
- Ca
- Cd
- Ce
- Cf
- Cl
- Cm
- Co
- Cr
- Cs
- Cu
- Db
- Dy
- Er
- Es
- Eu
- F
- Fe
- Fm
- Fr
- Ga
- Gd
- Ge
- H

  Any isotope of hydrogen.

  There are no special element symbols for D and T which should use the `isotope` attribute.
- He
- Hf
- Hg
- Ho
- Hs

- I
- In
- Ir
- K
- Kr
- La
- Li
- Lr
- Lu
- Md
- Mg
- Mn
- Mo
- Mt
- N
- Na
- Nb
- Nd
- Ne
- Ni
- No
- Np
- O
- Os
- P
- Pa
- Pb
- Pd
- Pm
- Po
- Pr
- Pt
- Pu
- Ra
- Rb
- Re
- Rf
- Rh
- Rn
- Ru
- S
- Sb
- Sc
- Se
- Sg
- Si
- Sm
- Sn
- Sr

- Ta
- Tb
- Tc
- Te
- Th
- Ti
- Tl
- Tm
- U
- Uun
- Uuu
- Uub
- Uut
- Uuq
- Uup
- Uuh
- Uus
- Uuo
- V
- W
- Xe
- Y
- Yb
- Zn
- Zr
- Dummy
- Du

  A point or object with no chemical semantics.

  Examples can be centroids, bond-midpoints, orienting "atoms" in small z-matrices.

  Note "Dummy" has the same semantics but is now deprecated.
- R

  A point at which an atom or group might be attached.

  Examples are abbreviated organic functional groups, Markush representations, polymers, unknown atoms, etc. Semantics may be determined by the `role` attribute on the atom.

[xsd:string]
Pattern: [A-Za-z]+:[A-Za-z][A-Za-z0-9\-]+

---

errorBasisType[st.errorBasisType]

The basis of an error value.

Errors in values can be of several types and this simpleType provides a small controlled vocabulary.

Example

UNION OF

Allowed values

- observedRange
- observedStandardDeviation
- observedStandardError
- estimatedStandardDeviation
- estimatedStandardError

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

errorValueArrayType[st.errorValueArrayType]

Array of error estimate values.

An observed or calculated estimate of the error in the value of a numeric quantity. It should be ignored for dataTypes such as URL, date or string. The statistical basis of the errorValueType is not defined - it could be a range, an estimated standard deviation, an observed standard error, etc. This information can be added through _errorBasisType_.

Example

XSD:LIST of errorValueType

An estimate of the error in the value of a quantity.

An observed or calculated estimate of the error in the value of a numeric quantity. It should be ignored for dataTypes such as URL, date or string. The statistical basis of the errorValueType is not defined - it could be a range, an estimated standard deviation, an observed standard error, etc. This information can be added through _errorBasisType_.

Example

[xsd:float]

errorValueType[st.errorValueType]

An estimate of the error in the value of a quantity.

An observed or calculated estimate of the error in the value of a numeric quantity. It should be ignored for dataTypes such as URL, date or string. The statistical basis of the errorValueType is not defined - it could be a range, an estimated standard deviation, an observed standard error, etc. This information can be added through _errorBasisType_.

Example
[xsd:float]

floatArrayType[st.floatArrayType]

An array of floats.

An array of floats or other real numbers. Not used in STM Schema, but re-used by CML and other languages.

Example
XSD:LIST of xsd:float

formalChargeArrayType[st.formalChargeArrayType]

Array of formalCharges.

Used for electron-bookeeping. This has no relation to its calculated (fractional) charge or oxidation state.

Example
XSD:LIST of formalChargeType

The formal charge on an object.

Used for electron-bookeeping. This has no relation to its calculated (fractional) charge or oxidation state.

Example
[xsd:integer]

formalChargeType[st.formalChargeType]

The formal charge on an object.

Used for electron-bookeeping. This has no relation to its calculated (fractional) charge or oxidation state.

Example

[xsd:integer]

---

formulaType[st.formulaType]

A concise representation for a molecular formula.

This MUST adhere to a whitespaced syntax so that it is trivially machine-parsable. Each element is followed by its count, and the string is optionally ended by a formal charge. NO brackets or other nesting is allowed.

Example

[xsd:string]

Pattern: \s*([A-Z][a-z]?\s+[1-9][0-9]*)(\s+[A-Z][a-z]?\s+[1-9][0-9]*)*(\s+[-|+]?[0-9]+)?\s*

---

hydrogenCountArrayType[st.hydrogenCountArrayType]

Array of hydrogenCounts.

The total number of hydrogen atoms bonded to an atom or contained in a molecule, whether explicitly included as atoms or not. It is an error to have hydrogen count less than the explicit hydrogen count. There is no default value and no assumptions about hydrogen Count can be made if it is not given. If hydrogenCount is given on every atom, then the values can be summed to give the total hydrogenCount for the (sub)molecule. Because of this hydrogenCount should not be used where hydrogen atoms bridge 2 or more atoms.

Example

XSD:LIST of hydrogenCountType

The total number of hydrogen atoms bonded to an object.

The total number of hydrogen atoms bonded to an atom or contained in a molecule, whether explicitly included as atoms or not. It is an error to have hydrogen count less than the explicit hydrogen count. There is no default value and no assumptions about hydrogen Count can be made if it is not given. If hydrogenCount is given on every atom, then the values can be summed to give the total hydrogenCount for the (sub)molecule. Because of this hydrogenCount should not be used where hydrogen atoms bridge 2 or more atoms.

Example

[xsd:nonNegativeInteger]

---

hydrogenCountType[st.hydrogenCountType]

The total number of hydrogen atoms bonded to an object.

The total number of hydrogen atoms bonded to an atom or contained in a molecule, whether explicitly included as atoms or not. It is an error to have hydrogen count less than the explicit hydrogen count. There is no default value and no assumptions about hydrogen Count can be made if it is not given. If hydrogenCount is given on every atom, then the values can be summed to give the total hydrogenCount for the (sub)molecule. Because of this hydrogenCount should not be used where hydrogen atoms bridge 2 or more atoms.

Example

[xsd:nonNegativeInteger]

---

idType[st.idType]

A unique ID for an element.

This is not formally of type ID (an XML NAME which must start with a letter and contain only letters, digits and `.-_:`). It is recommended that IDs start with a letter, and contain no punctuation or whitespace. The function in XSLT will generate semantically void unique IDs.

It is difficult to ensure uniqueness when documents are merged. We suggest namespacing IDs, perhaps using the containing elements as the base. Thus `mol3:a1` could be a useful unique ID. However this is still experimental.

[xsd:QName]

---

isotopicSpinType[st.isotopicSpinType]

A fractional representation of the spin of the nucleus.

[xsd:string]

Pattern: \d{1,}(/\d)?

---

latticeType[st.latticeType]

Allowed lattice types.

Example

UNION OF

Allowed values

- primitive
- full
- aCentred

lattice with A centering.

A lattice which uses the translation operator {0, 0.5, 0.5}.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

User-defined lattice-type.

This definition must be by reference to a namespaced dictionary entry.

---

line3Type[st.line3Type]

An unbounded line in 3-space.

Defined by 6 real numbers, conventionally an arbitrary point on the line and a vector3. There is no significance to the point (i.e. it is not the "end of the line") and there are an infinite number of ways of representing the line.

Example

BASE:
XSD:LIST of xsd:float
xsd:length
6

---

matrix44Type[st.matrix44Type]

A 4x4 transformation matrix

...

BASE:
XSD:LIST of xsd:float
xsd:length

16

matrixType[st.matrixType]

Allowed matrix types.

Many are square matrices. By default all elements must be included. For symmetric, antisymmetric and diagonal matrices some compression is possible by not reporting the identical or forced zero elements. These have their own subtypes, usually with UT or LT appended. Use these with caution as there is chance of confusion and you cannot rely on standard software to read these.

The matrix type fixes the order and semantics of the elements in the XML element but does not mandate any local syntax. Thus an application may insert newline characters after each row or use a <row> element.

Example

UNION OF

Allowed values

- rectangular

  Rectangular with no semantic constraints and ordered rowwise (i.e. the column index runs fastest).

  ```
  1 2 3 4
  0 3 5 6
  ```

- square

  Square with no semantic constraints.

  ```
  1   2 78
  3   4 -1
  -34 2  7
  ```

- squareSymmetric

  Square symmetric with all elements explicit.

  ```
  1 2 3
  2 7 1
  3 1 9
  ```

- squareSymmetricLT

  Square symmetric with the diagonal and lower triangle explicit and the upper triangle omitted. Rows are of length 1, 2, 3...

  ```
  1
  2 7
  ```

```
3 1 9
```

is equivalent to

```
1  2  3
2  7  1
3  1  9
```

- squareSymmetricUT

Square symmetric with the diagonal and upper triangle explicit. Rows are of length n, n-1, ... 2, 1

```
1  7  9
   2 -1
     34
```

is equivalent to

```
1  7  9
7  2 -1
9 -1 34
```

- squareAntisymmetric

Square antisymmetric with all elements explicit. The diagonal is necessarily zero.

```
 0 -2  3
 2  0  1
-3 -1  0
```

- squareAntisymmetricLT

Square symmetric with the lower triangle explicit and diagonal and upper triangle omitted. Rows are of length 1, 2,... n-1.

```
-7
-9  1
```

is equivalent to

```
 0  7  9
-7  0 -1
-9  1  0
```

- squareAntisymmetricUT

Square symmetric with the upper triangle explicit and diagonal and lower triangle omitted. Rows are of length n-1, n-2,... 2,1.

```
 7  9
   -1
```

is equivalent to

```
 0  7  9
-7  0 -1
-9  1  0
```

- diagonal

  Symmetric. Elements are zero except on the diagonal. No compressed representation available (use `array` element).

  ```
  1 0 0
  0 3 0
  0 0 4
  ```

- upperTriangular

  Square. Elements are zero below the diagonal

  ```
  1 2 3 4
  0 3 5 6
  0 0 4 8
  0 0 0 2
  ```

- upperTriangularUT

  Square. Elements below the diagonal are zero and omitted, and rows are of length n, n-1, ... , 2, 1.

  ```
  1 2 3 4
    3 5 6
      4 8
        2
  ```

  is equivalent to

  ```
  1 2 3 4
  0 3 5 6
  0 0 4 8
  0 0 0 2
  ```

- lowerTriangular

  Square. Elements are zero above the diagonal

  ```
  1 0 0
  7 3 0
  9 2 4
  ```

- lowerTriangularLT

  Square. Elements above the diagonal are zero and omitted, and rows are of length 1, 2, ...n.

```
1
3 7
9 2 3
```

is equivalent to

```
1 0 0
3 7 0
9 2 3
```

- unit

  Square. Diagonal elements are 1 and off-diagonal are zero.

  ```
  1 0 0
  0 1 0
  0 0 1
  ```

- unitary

  Square. When multiplied by its transpose gives the unit matrix.

  ```
  0 -1  0
  1  0  0
  0  0  1
  ```

- rowEigenvectors

  Square. Each row corresponds to an eigenvector of a square matrix. Elements are real. The length of the eigenvectors is undefined, i.e. they are not required to be normalised to 1.

  ```
  0 -1  0
  1  0  0
  0  0  1
  ```

- rotation22

  The rotation is defined by the matrix premultiplyin a column vector (x, y) .

  ```
  0 -1
  1  0
  ```

  produces (-y, x), i.e. a rotation of -90 degrees.

- rotationTranslation32

  A third column defining the translation is added to a rotation22.

  ```
  0 -1 22
  1  0 33
  ```

  produces (-y + 22, x + 33), i.e. a rotation of -90 degrees followed by a translation

of (22, 33).

- homogeneous33
- rotation33
- rotationTranslation43
- homogeneous44

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

User-defined matrix-type.

This definition must be by reference to a namespaced dictionary entry.

---

maxType[st.maxType]

The maximum INCLUSIVE value of a quantity.

The maximum INCLUSIVE value of a sortable quantity such as numeric, date or string. It should be ignored for dataTypes such as URL. The use of `min` and `max` attributes can be used to give a range for the quantity. The statistical basis of this range is not defined. The value of `max` is usually an observed quantity (or calculated from observations). To restrict a value, the `maxExclusive` type in a dictionary should be used.

The type of the maximum is the same as the quantity to which it refers - numeric, date and string are currently allowed

Example

[xsd:string]

---

metadataType[st.metadataType]

The name of the metadata.

Metadata consists of name-value pairs (value is in the "content" attribute). The names are from a semi-restricted vocabulary, mainly Dublin Core. The content is unrestricted. The order of metadata has no implied semantics at present. Users can create their own metadata names using the namespaced prefix syntax (e.g. foo:institution). Ideally these names should be defined in an STMML dictionary.

2003-03-05: Added UNION to manage non-controlled name.

UNION OF
Allowed values

- dc:coverage

  The extent or scope of the content of the resource.

  Coverage will typically include spatial location (a place name or geographic coordinates), temporal period (a period label, date, or date range) or jurisdiction (such as a named administrative entity). Recommended best practice is to select a value from a controlled vocabulary (for example, the Thesaurus of Geographic Names [TGN]) and that, where appropriate, named places or time periods be used in preference to numeric identifiers such as sets of coordinates or date ranges.

- dc:description

  An account of the content of the resource.

  Description may include but is not limited to: an abstract, table of contents, reference to a graphical representation of content or a free-text account of the content.

- dc:identifier

  An unambiguous reference to the resource within a given context.

  Recommended best practice is to identify the resource by means of a string or number conforming to a formal identification system. Example formal identification systems include the Uniform Resource Identifier (URI) (including the Uniform Resource Locator (URL)), the Digital Object Identifier (DOI) and the International Standard Book Number (ISBN).

- dc:format

  The physical or digital manifestation of the resource.

  Typically, Format may include the media-type or dimensions of the resource. Format may be used to determine the software, hardware or other equipment needed to display or operate the resource. Examples of dimensions include size and duration. Recommended best practice is to select a value from a controlled vocabulary (for example, the list of Internet Media Types [MIME] defining computer media formats).

- dc:relation

  A reference to a related resource.

  Recommended best practice is to reference the resource by means of a string or

number conforming to a formal identification system.

- dc:rights

  Information about rights held in and over the resource.

  Typically, a Rights element will contain a rights management statement for the resource, or reference a service providing such information. Rights information often encompasses Intellectual Property Rights (IPR), Copyright, and various Property Rights. If the Rights element is absent, no assumptions can be made about the status of these and other rights with respect to the resource.

- dc:subject

  The topic of the content of the resource.

  Typically, a Subject will be expressed as keywords, key phrases or classification codes that describe a topic of the resource. Recommended best practice is to select a value from a controlled vocabulary or formal classification scheme.

- dc:title

  A name given to the resource.

  Typically, a Title will be a name by which the resource is formally known.

- dc:type

  The nature or genre of the content of the resource.

  Type includes terms describing general categories, functions, genres, or aggregation levels for content. Recommended best practice is to select a value from a controlled vocabulary (for example, the working draft list of Dublin Core Types [DCT1]). To describe the physical or digital manifestation of the resource, use the FORMAT element.

- dc:contributor

  An entity responsible for making contributions to the content of the resource.

  Examples of a Contributor include a person, an organisation, or a service. Typically, the name of a Contributor should be used to indicate the entity.

- dc:creator

  An entity primarily responsible for making the content of the resource.

  Examples of a Creator include a person, an organisation, or a service. Typically, the name of a Creator should be used to indicate the entity.

- dc:publisher

  An entity responsible for making the resource available.

  Examples of a Publisher include a person, an organisation, or a service. Typically, the name of a Publisher should be used to indicate the entity.

- dc:source

  A Reference to a resource from which the present resource is derived.

  The present resource may be derived from the Source resource in whole or in

part. Recommended best practice is to reference the resource by means of a string or number conforming to a formal identification system.

- dc:language

A language of the intellectual content of the resource.

Recommended best practice for the values of the Language element is defined by RFC 1766 [RFC1766] which includes a two-letter Language Code (taken from the ISO 639 standard [ISO639]), followed optionally, by a two-letter Country Code (taken from the ISO 3166 standard [ISO3166]). For example, 'en' for English, 'fr' for French, or 'en-uk' for English used in the United Kingdom.

- dc:date

A date associated with an event in the life cycle of the resource.

Typically, Date will be associated with the creation or availability of the resource. Recommended best practice for encoding the date value is defined in a profile of ISO 8601 [W3CDTF] and follows the YYYY-MM-DD format.

- cmlm:safety

Entry contains information relating to chemical safety.

Typically the content will be a reference to a handbook, MSDS, threshhold or other human-readable strin.

- cmlm:insilico

Part or whole of the information was computer-generated.

Typically the content will be the name of a method or a progra.

- cmlm:structure

3D structure included.

details include.

- cmlm:reaction
- cmlm:identifier
- other

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification

(alphanumeric, '_', '.' and '-'
Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

minType[st.minType]

The minimum INCLUSIVE value of a quantity.

The minimum INCLUSIVE value of a sortable quantity such as numeric, date or string. It should be ignored for dataTypes such as URL. The use of `min` and `min` attributes can be used to give a range for the quantity. The statistical basis of this range is not defined. The value of `min` is usually an observed quantity (or calculated from observations). To restrict a value, the `minExclusive` type in a dictionary should be used.

The type of the minimum is the same as the quantity to which it refers - numeric, date and string are currently allowed
Example
[xsd:string]

---

moleculeRefType[st.moleculeRefType]

A reference to an existing molecule.
BASE: idType

A unique ID for an element.

This is not formally of type ID (an XML NAME which must start with a letter and contain only letters, digits and `.-_:`). It is recommended that IDs start with a letter, and contain no punctuation or whitespace. The function in XSLT will generate semantically void unique IDs.

It is difficult to ensure uniqueness when documents are merged. We suggest namespacing IDs, perhaps using the containing elements as the base. Thus `mol3:a1` could be a useful unique ID. However this is still experimental.
[xsd:QName]

---

namespaceRefType[st.namespaceRefType]

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

nonNegativeAngleType[st.nonNegativeAngleType]

A non-signed angle.

Re-used by _angle_. Note that we also provide positiveAngleType (e.g. for cell angles) and torsionAngleType for _torsion_.

Example

[xsd:float]
minInclusive: 0.0
maxInclusive: 180.0

---

occupancyArrayType[st.occupancyArrayType]

Array of atomic occupancies.

Primarily for crystallography. Values outside 0-1 are not allowed.

Example

XSD:LIST of occupancyType

A floating point number between 0 and 1 inclusive

Originally for crystallographic occupancy but re-usable for fractinal yield, etc.

Example

[xsd:float]
minInclusive: 0
maxInclusive: 1

---

occupancyType[st.occupancyType]

A floating point number between 0 and 1 inclusive

Originally for crystallographic occupancy but re-usable for fractinal yield, etc.

Example
[xsd:float]
minInclusive: 0
maxInclusive: 1

orderArrayType[st.orderArrayType]

An array of bond orders.

See order.

XSD:LIST of orderType

Bond order.

This is purely conventional and used for bond/electron counting. There is no default value. The emptyString attribute can be used to indicate a bond of unknown or unspecified type. The interpretation of this is outside the scope of CML-based algorithms. It may be accompanied by a `convention` attribute on the `bond` which links to a dictionary. Example: `<bond convention="ccdc:9" atomRefs2="a1 a2"/>` could represent a delocalised bond in the CCDC convention.

UNION OF
Allowed values

- hbond

  Hydrogen bond.

  Carries no semantics but will normally be between a hydrogen atom and an element with lone pairs.
- partial01

  Partial bond.

  Can be used for a partial bond in a transition state, intermolecular interaction, etc. There is no numeric value associated and the bond order could be anywhere between 0 and single.
- S

  Single bond.

  synonymous with "1.
- 1

  Single bond.
- partial12

  Intermediate between 1 and .

  Could be used for a transition state or a delocalised system.
- D

Double bond.

- 2

Double bond.

- partial23

Intermediate between 2 and .
Could be used for a transition state or a delocalised system.

- T

Triple bond.

- 3

Triple bond.

- A

Aromatic bond.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.
Example
[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'
Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

orderType[st.orderType]

Bond order.

This is purely conventional and used for bond/electron counting. There is no default value. The emptyString attribute can be used to indicate a bond of unknown or unspecified type. The interpretation of this is outside the scope of CML-based algorithms. It may be accompanied by a `convention` attribute on the `bond` which links to a dictionary. Example: `<bond convention="ccdc:9" atomRefs2="a1 a2"/>` could represent a delocalised bond in the CCDC convention.

UNION OF
Allowed values

- hbond

  Hydrogen bond.
  Carries no semantics but will normally be between a hydrogen atom and an element with lone pairs.
- partial01

  Partial bond.
  Can be used for a partial bond in a transition state, intermolecular interaction, etc. There is no numeric value associated and the bond order could be anywhere between 0 and single.
- S

  Single bond.
  synonymous with "1.
- 1

  Single bond.
- partial12

  Intermediate between 1 and .
  Could be used for a transition state or a delocalised system.
- D

  Double bond.
- 2

  Double bond.
- partial23

  Intermediate between 2 and .
  Could be used for a transition state or a delocalised system.
- T

  Triple bond.
- 3

  Triple bond.
- A

  Aromatic bond.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

plane3Type[st.plane3Type]

An unbounded plane in 3-space.

Defined by 4 real numbers, conventionally a vector3 normal to the plane and a signed scalar representing the distance to the origin. The vector must not be of zero length (and need not be normalized.

Example

```
The first three numbers are the vector, followed by the
distance
```

BASE:

XSD:LIST of xsd:float

xsd:length

4

---

point3Type[st.point3Type]

A point in 3-space.

The 3 components can have any signed value.

Example

BASE:

XSD:LIST of xsd:float

xsd:length

3

---

positiveNumberType[st.positiveNumberType]

A positive number.

Note that we also provide nonNegativeNumber with inclusive zero. The maximum number is (quite large) since 'unbounded' is more difficult to implement.

[xsd:double]
minExclusive: 0.0
maxInclusive: 1.0E+99

---

reactionFormatType[st.reactionFormatType]

The format of the reaction.

This is provided for machine-understanding of the format of the reaction steps and components.

Semantics are semi-controlled.

UNION OF
Allowed values

- reactantProduct

  The commonest representation with reactantList and productList.

- cmlSnap

  A list of molecules representing snap shots on a reaction pathway.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

reactionRoleType[st.reactionRoleType]

The role of the reaction within a reactionList.

Semantics are semi-controlled.

UNION OF

Allowed values

- complete

  On reactionList signifies that the children are the complete description of the reaction.

- overall

  The overall reaction in a multi-step reaction. Normally this would be the first reaction in a reactionList and the individual steps are held in a following sibling reactionList.

- rateDeterminingStep

  The rate-determining step in a multi-step reaction. This implies also that the reaction has a role of step.

- step

  A step in a multi-step reaction. This reaction will normally be a child of reactionList.

- steps

  a reactionList containing steps

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

Examples could be "myDict:step1", "foo:chainPropagation", etc.

---

reactionTypeType[st.reactionTypeType]

The semantic type of the reaction.

This is provided for machine-understanding of the topology or logic of the reaction steps and components (i.e. not for a general classification for which `label` is more appropriate.)

Semantics are semi-controlled. Some terms are appropriate to multistep reactions, and can be used with or without explicit steps.

UNION OF
Allowed values

- chainReaction

  A reaction in which one or more reactive reaction intermediates (frequently radicals) are continuously regenerated, usually through a repetitive cycle of elementary steps (the 'propagation step') (IUPAC GoldBook).
- initiation

  A reaction or process generating free radicals (or some other reactive reaction intermediates) which then induce a chain reaction. For example, in the chlorination of alkanes by a radical mechanism the initiation step is the dissociation of molecular chlorine. IUPAC Compendium of Chemical Terminology 2nd Edition (1997).
- termination

  The steps in a chain reaction in which reactive intermediates are destroyed or rendered inactive, thus ending the chain. IUPAC Compendium of Chemical Terminology 2nd Edition (1997) .
- reversible

  A reaction which can proceed in the forward direction as readily as in the reverse direction (IUPAC GoldBook).

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

refType[st.refType]

A reference to an existing object.

A reference to an existing element in the document. The target of the ref attribute must exist. The test for validity will normally occur in the element's _appinfo_. Any DOM Node created from this element will normally be a reference to another Node, so that if the target node is modified a the dereferenced content is modified. At present there are no deep copy semantics hardcoded into the schema.

[xsd:QName]

---

sizeType[st.sizeType]

The size of an array.

The size of an array. Redundant, but serves as a check for processing software (useful if delimiters are used).

[xsd:nonNegativeInteger]

---

spaceType[st.spaceType]

Signifies real or reciprocal space.

Likely to be used on types such as lattice, plane, point.

Example

UNION OF

Allowed values

- real
- k-space

  A synonym for reciprocal.

- Fourier

  A synonym for reciprocal.
- reciprocal

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

User-defined space-type.

No obvious possibilities, but who know.

---

sphere3Type[st.sphere3Type]

A sphere in 3-space.

Defined by 4 real numbers, conventionally a point3 at the centre of the sphere and a nonNegative scalar for the radius.

Example

BASE:

XSD:LIST of xsd:float

xsd:length

4

---

stateType[st.stateType]

State of a substance or property.

The state(s) of matter appropriate to a substance or property. It follows a partially controlled vocabulary. It can be extended through namespace codes to dictionaries.

Example

UNION OF
Allowed values

- aqueous

  An aqueous solutio.
- gas

  Gas or vapor. The default state for computation on isolated molecule.
- glass

  A glassy stat.
- liquid

  Normally pure liquid (use solution where appropriate.
- nematic

  The nematic phas.
- smectic

  The smectic phas.
- solid

  A soli.
- solidSolution

  A solid solutio.
- solution

  A (liquid) solutio.

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example
[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

stereoType[st.stereoType]

Bond stereochemistry as a string.

This is purely conventional. There is no default value. The emptyString attribute can be used to indicate a bond of unknown or unspecified type. The interpretation of this is outside the scope of CML-based algorithms. It may be accompanied by a `convention` attribute which links to a dictionary.

Example

Allowed values

- C

  A cis bond.
- T

  A trans bond.
- W

  A wedge bond.
- H

  A hatch bond.
-

  empty or missing.

---

torsionAngleType[st.torsionAngleType]

The type of a torsion angle.

[xsd:float]
minInclusive: -360.0
maxInclusive: 360.0

---

unitsType[st.unitsType]

Scientific units.

These will be linked to dictionaries of units with conversion information, using namespaced references (e.g. `si:m`). Distinguish carefully from _unitType_ which is an element describing a type of a unit in a _unitList_.

Example

BASE: namespaceRefType

An XML QName with required prefix.

A string referencing a dictionary, units, convention or other metadata.

The purpose is to allow authors to extend the vocabulary through their own namespaces without altering the schema. The prefix is mandatory. This convention is only used within STMML and related languages; it is NOT a generic URI.

Example

[xsd:string]

The namespace prefix must start with an alpha character and can only contain alphanumeric and '_'. The suffix can have characters from the XML ID specification (alphanumeric, '_', '.' and '-'

Pattern: [A-Za-z][A-Za-z0-9_]*:[A-Za-z][A-Za-z0-9_\.\-]*

---

vector3Type[st.vector3Type]

A vector in 3-space.

No constraints on magnitude (i.e. could be zero.

Example

BASE:
XSD:LIST of xsd:float
xsd:length
3

MISC