CellML Specification
Draft — 30 September 2003

5 Units

5.1 Introduction

One of the key features ensuring robustness and re-usability of CellML components and models is the requirement that units be associated with all variables and numbers in a CellML document. This allows components and models that declare variables with different units to be connected, as long as variables that are mapped to one another have the same dimensions. For instance, it is possible to map a variable declared with units of "pound/foot" to a variable declared with units of "kilogram/metre", but not to a variable declared with units of "mole/litre" or "kilogram-squared/metre". The explicit declaration of units also allows CellML processing software to check the consistency of each equation in a model.

5.2 Basic Structure

5.2.1 Dictionary of standard units

CellML provides a dictionary of standard units that may be used in variable declarations or attached to bare numbers in mathematics. References to these units should make use of the actual name of the units, rather than the standard abbreviation, thus avoiding confusion between units (e.g., metre) and prefixes (e.g., milli). The full list of units that any CellML processing application is expected to recognise is given in Table 2. The keywords in the table comprise the SI base units, the SI derived units with special names and symbols, and some additional units commonly used in the types of biological models likely to be defined using CellML.

ampere	farad	katal	lux	pascal	tesla
becquerel	*gram*	**kelvin**	**meter**	radian	volt
candela	gray	**kilogram**	**metre**	**second**	watt
celsius	henry	*liter*	**mole**	siemens	weber
coulomb	hertz	*litre*	newton	sievert	
dimensionless	joule	lumen	ohm	steradian	

TABLE 2: The dictionary of units keywords that CellML processing applications are expected to recognise. Base SI units are printed in bold text, derived SI units are printed in plain text, and additions to the standard units defi ned purely for the convenience of model authors are italicised.

The SI base units are the foundation of the units system in CellML. The conversion of a variable's value between two sets of units involves the expansion of all units definitions to linear combinations of the SI base units and user-defined base units (described in Section 5.2.3). The list of SI base and derived units is taken from The International System of Units (SI)[1], including the Year 2000 Supplement[2]. The American spellings of `meter` and `liter` are taken from the NIST Guide for the Use of the International System of UNITS (SI)[3]. The SI standard defines the mathematical relationships between the SI derived units and the SI

[1] http://www.bipm.fr/pdf/si-brochure.pdf
[2] http://www.bipm.fr/pdf/si-supplement2000.pdf
[3] http://physics.nist.gov/Pubs/SP811/contents.html

base units. These relationships are given in the right hand column of Table 3 in the Year 2000 Supplement, with the exception of **celsius**, which is related to **kelvin** as described in Section 2.1.1.5 of the SI standard.

The CellML units dictionary includes four non-SI units definitions for the convenience of modellers: *dimensionless*, *gram*, *liter* and *litre*. The only unfamiliar name on this list is *dimensionless*, which is used to indicate that a number or variable has no units associated with it. The mathematical relationships between *gram* and *litre* and the base SI units are given in Section 5.2.5.

5.2.2 User defined units

CellML also provides a facility whereby new units can be defined in terms of the units provided in the dictionary. This functionality allows the definition of units which are expressed as a scaled version of other units (as is the case for most imperial units), the definition of units which are made up of the product of other units, and even the creation of units that require an offset, such as degrees Fahrenheit. This allows model authors to work in whatever set of units they feel most comfortable, while still ensuring that their models can be integrated with those of other authors using different units.

New units are defined or declared using the **<units>** element, which may be placed inside **<model>**, **<import>**, and **<component>** elements. When a **<units>** element is placed inside the **<model>** or **<import>** element, the units definition may be referenced from within any component in the model. When a **<units>** element is placed inside a **<component>** element, the units definition may only be referenced from within that component.

Each units element must define a **name** attribute, which is used to reference the units definition elsewhere. The value of the **name** attribute on a **<units>** element defined in a **<model>** element or declared in an **<import>** element must be unique across all **<units>** elements in the **<model>** and all **<import>** elements. For **<units>** elements defined in a **<component>** element, the value of the **name** attribute must be unique across all **<units>** elements in the **<component>** in which it is defined. If the value of the **name** attribute of a **<units>** element defined inside a **<component>** element matches the value of the **name** attribute on a **<units>** element defined inside the **<model>** or an **<import>** element, then it will redefine the units, and all references to these units within the **<component>** element refer to the new definition. Model authors must not redefine any of the standard units. Therefore, the value of the **name** attribute must not equal one of the names from the standard units dictionary in Table 2.

A units declaration appearing directly inside an **<import>** element must also define a **units_ref** attribute, which is described in Section 9[4]. A units definition appearing directly inside a **<model>** or **<component>** element may also define a **base_units** attribute, the associated behaviour of which is discussed in Section 5.2.3, and may contain a set of **<unit>** elements that reference units from the dictionary or some previously defined units.

A **<unit>** element must not contain any elements in the CellML namespace, but may have up to five attributes. The **units** attribute is the only one that is required. It is used to set the base quantity for the current **<unit>** element, and its value must correspond to a keyword from the standard CellML units dictionary or to the value of the **name** attribute of a **<units>** element in the current component or model.

The definition of new units in terms of subunits may require the use of some combination of the optional **offset**, **prefix**, **exponent**, and **multiplier** attributes.

A **multiplier** attribute can be used to pre-multiply the quantity to be converted by any real scale factor. For instance, a multiplier of `0.45359237` is used to define a pound in terms of a kilogram. The **multiplier** attribute has a default value of `"1.0"`

The **offset** attribute is used to represent the addition of a constant in the transformation between the current units and the base units. This should only be necessary for the definition of temperature scales. For

[4] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_import_model

instance, an **offset** attribute value of "32.0" is needed to define Fahrenheit in terms of Celsius. The **offset** attribute has a default value of "0.0".

The **prefix** attribute can be used to indicate a scale for the referenced units. It is included primarily for the convenience of modellers who want to define units that differ from another units definition only by an SI scale factor. Its value must be from the standard set of CellML prefix names given in Table 3 or be an integer, in which case the units are pre-multiplied by 10 to the power of this number. The default value of the **prefix** attribute is "0" (the referenced units are scaled by a factor of one).

name	factor	name	factor
yotta	10^{24}	deci	10^{-1}
zetta	10^{21}	centi	10^{-2}
exa	10^{18}	milli	10^{-3}
peta	10^{15}	micro	10^{-6}
tera	10^{12}	nano	10^{-9}
giga	10^{9}	pico	10^{-12}
mega	10^{6}	femto	10^{-15}
kilo	10^{3}	atto	10^{-18}
hecto	10^{2}	zepto	10^{-21}
deka	10^{1}	yocto	10^{-24}

TABLE 3: The set of names that may be used in the **prefix** attribute on a **<unit>** element and the corresponding scale factors that will pre-multiply the unit.

The scale factor described by the **prefix** attribute and the units referenced by the **units** attribute are raised to a power equal to the value of the **exponent** attribute. The value of the **exponent** attribute must be a floating point number, and is typically an integer. The **exponent** attribute has a default value of "1.0". Note that an **exponent** attribute value of "0.0" has the effect of removing the parent **<unit>** element from the current units definition.

A *simple units* definition occurs when units are defined as a linear function of some previously defined simple units or base units. In a simple units definition, a **<units>** element contains only a single child **<unit>** element, that **<unit>** element has an **exponent** attribute value of "1.0", and the units definition referenced by the **units** attribute is one of the SI or user-defined base units or is itself a simple units definition. These are the only conditions under which a **<unit>** element may define an **offset** attribute with a value other than "0.0". The formula that expresses how the old units (referenced by the value of the **units** attribute on the **<unit>** element) are transformed into the new units (defined by the value of the **name** attribute on the parent **<units>** element) is given below.

$$x_{new}\,[\,\text{Units}\,] = (\,multiplier\;prefix\,)\left[\frac{\text{Units}}{\text{units}}\right] x_{old}\,[\,\text{units}\,] + offset\,[\,\text{Units}\,] \tag{3}$$

Terms in square brackets represent the units associated with values in the expression, which are italicised. x_{old} is the value to be transformed from the old units, and x_{new} is the resulting value in the new units. **Units** are the units being defined, and $multiplier$, $prefix$, **units** and $offset$ correspond to the values of the appropriate attributes on the **<unit>** element.

Complex units are the product of multiple units. In a complex units definition, a **<units>** element contains more than one **<unit>** element or a **<unit>** element that defines an **exponent** attribute with a value other than "1.0". The conversion between the new units and the product of the constituent units is given by the formula below.

$$x_{new}\,[\,\text{Units}\,] = (\; m_1 \;\ldots\; m_n \; {p_1}^{e_1} \;\ldots\; {p_n}^{e_n} \;) \left[\frac{\text{Units}}{{\text{u}_1}^{\text{e}_1} \;\ldots\; {\text{u}_\text{n}}^{\text{e}_\text{n}}} \right] x_{old}\,[\,{\text{u}_1}^{\text{e}_1} \;\ldots\; {\text{u}_\text{n}}^{\text{e}_\text{n}}\,] \tag{4}$$

The m_i, p_i, $\mathtt{u_i}$, and e_i terms refer to the values of the **`multiplier`**, **`prefix`**, **`units`** and **`exponent`** attributes on the *i*-th **`<unit>`** element respectively.

An **`offset`** attribute may not be defined on any **`<unit>`** elements that occur inside a complex units definition. When a complex units definition references a simple units definition, any offset associated with the simple units definition is removed. This means that conversions such as the one between degrees Fahrenheit per inch and degrees Celsius per centimetre involve only a scale factor.

5.2.3 New base units

A modeller might want to define and use units for which no simple conversion to SI units exist. A good example of this is pH, which is dimensionless, but uses a log scale. Ideally, pH should not simply be defined as dimensionless because software might then attempt to map variables defined with units of pH to any other dimensionless variables.

CellML addresses this by allowing the model author to indicate that a units definition is a new type of base unit, the definition of which cannot be resolved into simpler subunits. This is done by defining a **`base_units`** attribute value of `"yes"` on the **`<units>`** element. This element must then be left empty. The **`base_units`** attribute is optional and has a default value of `"no"`. If the **`base_units`** attribute is omitted or assigned a value of `"no"`, units are expected to be defined in terms of other units as described in Section 5.2.2.

The indiscriminate use of the **`base_units`** attribute is strongly discouraged, because it has a significant impact on the re-usability of models and components. In particular, the **`base_units`** attribute should not be used to restrict users to creating models with an application-specific dictionary of units, as this prevents the efficient exchange of CellML models with other applications.

Software that is checking the consistency of the units in an equation (described in more detail in Section 5.2.7) can stop the recursive resolution of units definitions when the only remaining units are base SI units and user-defined base units.

5.2.4 Expansion of units definitions

For interoperability, software that claims to perform units conversion when passing variables between components and/or claims to perform dimension consistency checking of equations should obtain results that are equivalent to those produced using the algorithms described in Appendix C.3.5[5] and Appendix C.3.6[6], respectively. Both of these algorithms make use of the algorithm defined in Appendix C.3.4[7] to fully expand units definitions into functions of the SI and user-defined base units.

For both simple and complex units definitions (as defined by Equation (3) and Equation (4), respectively), the algorithm recursively substitutes in equations expanding the unknown term x_{old}, stopping when the unknown term has only SI or user-defined base units.

Although this specification does not require software to implement this algorithm exactly, it is used extensively to demonstrate units conversion and dimension checking as described in Section 5.2.6 and Section 5.2.7, respectively. Appendix C.4.2[8] provides examples of units definition expansion according to the algorithm described in Appendix C.3.4[9].

[5] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_conversion_between_units_definitions
[6] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_equation_dimension_checking
[7] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_units_definition_expansion
[8] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_examples_expansion
[9] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_units_definition_expansion

5.2.5 Expansion of the non-SI units definitions in the CellML dictionary

Having defined a mathematical notation in Section 5.2.2 and a technique for the expansion of units definitions, it is now possible to formally specify how the definitions of the non-SI units in Table 2 should be expanded. The CellML versions of these units definitions and the associated equations are given below. The definition of *liter* is identical to the definition of *litre*. As described in Section 5.2, *dimensionless* is not related to the SI units and cannot be expanded.

```
<units name="gram">
  <unit multiplier="0.001" units="kilogram" />
</units>
```

$$x_{new}\,[\,\text{gram}\,] = 0.001 \left[\frac{\text{gram}}{\text{kilogram}}\right] x_{old}\,[\,\text{kilogram}\,] \tag{5}$$

```
<units name="litre">
  <unit multiplier="1000" prefix="centi" units="metre" exponent="3" />
</units>
```

$$\begin{aligned} x_{new}\,[\,\text{litre}\,] &= \left(\,1000\;\left(10^{-2}\right)^{3}\,\right)\left[\frac{\text{litre}}{\text{metre}^3}\right] x_{old}\,[\,\text{metre}^3\,] \\ &= 0.001\left[\frac{\text{litre}}{\text{metre}^3}\right] x_{old}\,[\,\text{metre}^3\,] \end{aligned} \tag{6}$$

5.2.6 Conversion between units definitions

Associating units definitions with every variable declaration in a component allows variables from components that make use of different sets of units to be mapped together, as long as the variables have the same dimensions. Appendix C.3.5[10] specifies a possible method for converting a numeric value from one set of units to another. CellML processing software is not required to be capable of converting between units definitions. However, for interoperability, software that does implement this functionality should achieve the same results as if this method were used, although the exact implementation may differ.

This implementation generates an expression that relates each units definition to SI and user-defined base units. This expression is obtained by recursively expanding each units definition as described in Appendix C.3.4[11], and then simplifying the result. The expression for the input units is then inverted to give an expression that relates the appropriate base units to the input units. This inverted expression is substituted into the expression for the target units, producing a single expression that relates the quantity to be converted from the input units to a corresponding quantity in the target units. The inversion and substitution process is demonstrated by example in Appendix C.4.3[12].

5.2.7 Equation dimension checking

The association of units with every variable and bare number that appears in an equation in a CellML document provides CellML processing software the opportunity to perform equation dimension checking. Verifying that equations have consistent dimensions can potentially catch many basic mathematical errors.

[10] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_conversion_between_units_definitions
[11] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_units_definition_expansion
[12] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_examples_conversion_of_units_definitions

Appendix C.3.6[13] specifies a possible implementation of equation dimension checking. This implementation splits an equation into a tree of equation parts, in which each parent part is obtained by the application of a single operator to its children. The units definition on each leaf node (i.e., part without children) is expanded into base units, as described in Appendix C.3.4[14]. The units definition for a node at a higher level of the tree is constructed by combining the units definitions of its children. An equation has consistent dimensions if no errors are found while traversing the tree and if the fully expanded units definitions of the two nodes at the top level of the tree are equivalent, as defined in Appendix C.2.2[15].

CellML processing software is free to ignore units in mathematics and assume that equations are consistent. For interoperability, software that performs equation dimension checking should achieve the same results as if the implementation discussed in Appendix C.3.6[16] were used, although the exact implementation may differ.

This specification does not attempt to completely prevent model authors from creating invalid mathematics. Dimension consistency checking prevents modellers from adding variables with different dimensions but would not find errors in Equation (7) and Equation (8), which have different units but the same dimensions:

$$x\,[\,\text{volt}\,] = y\,[\,\text{volt}\,] + z\,[\,\text{millivolt}\,] \tag{7}$$

$$x\,[\,\text{inch}\,] = y\,[\,\text{metre}\,] + z\,[\,\text{nautical\_mile}\,] \tag{8}$$

Although it would be technically possible (and useful) to find and correct such errors, CellML processing software is not required to be able to do so.

5.3 Examples

5.3.1 User-defined units and new base units

Figure 7 demonstrates how users can extend the set of units in the CellML dictionary by defining new sets of units.

5.3.2 Advanced examples

Examples of the expansions of units definitions, conversion between units definitions and equation dimension checking are given in Appendix C.4[17].

5.4 Rules for CellML Documents

Units are a fundamental part of a CellML model definition. In this section, formal rules are specified for the system of units definition introduced in Section 5.2.

5.4.1 The `<units>` element

1. **Allowed use of the `<units>` element**

   - The `<model>`, `<import>` and `<component>` elements may contain any number of `<units>` elements.

[13] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_equation_dimension_checking
[14] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_units_definition_expansion
[15] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_dimension_equivalence
[16] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_equation_dimension_checking
[17] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_adv_units_examples

```
<!-- User-defined Base Units -->
<units name="pH" base_units="yes" />

<!-- Simple Units Definitions -->
<units name="inch">
  <unit multiplier="2.54" prefix="centi" units="metre" />
</units>

<units name="fahrenheit">
  <unit multiplier="1.8" units="celsius" offset="32.0" />
</units>

<!-- Complex Units Definitions -->
<units name="celsius_per_centimetre">
  <unit units="celsius" />
  <unit prefix="centi" units="metre" exponent="-1" />
</units>

<units name="fahrenheit_per_inch">
  <unit units="fahrenheit" />
  <unit units="inch" exponent="-1" />
</units>

<units name="pH_per_celsius">
  <unit units="pH" />
  <unit units="celsius" exponent="-1" />
</units>
```

FIGURE 7: Some examples of the use of the **<units>** element demonstrating the definition of simple and complex units.

- Each **<units>** element must define a **name** attribute.
- Each **<units>** element appearing as the child of a **<model>** or **<component>** element may also define a **base_units** attribute.

  [Units declared in an **<import>** element are not new definitions, and therefore can't be defined as base units.]
- If a **<units>** element appearing as the child of a **<model>** or **<component>** element defines a **base_units** attribute with a value of "yes", then that **<units>** element must contain only the following elements, which may appear in any order:
  - **<RDF>** elements in the RDF namespace.
- If a **<units>** element appearing as the child of a **<model>** or **<component>** element does not define a **base_units** attribute with a value of "yes", then that **<units>** element must contain only the following elements, which may appear in any order:
  - **<unit>** elements in the CellML namespace,
  - **<RDF>** elements in the RDF namespace.
- Each **<units>** element appearing as the child of an **<import>** element must also define a **units_ref** attribute.
- A **<units>** element appearing as the child of an **<import>** element must contain only the following elements, which may appear in any order:
  - **<RDF>** elements in the RDF namespace.

2. **Allowed values of the name attribute**

- The value of the **name** attribute must be a valid CellML identifier as discussed in Section 2.2.1[18].
- The value of the **name** attribute must not equal one of the names defined in the standard dictionary of units in Table 2.

  [Model authors may not redefine the standard units.]
- For units defined in a **<model>** element or declared in an **<import>** element, the value of the **name** attribute must be unique across all **<units>** elements within the **<model>** and all **<import>** elements. For units defined in a **<component>** element, the value of the **name** attribute must be unique within the given **<component>** element.

  [Two **<units>** elements in the **<model>** and **<import>** elements may not have the same **name** attribute value, although a **<units>** element in a **<component>** element may share the same name as a **<units>** element in the **<model>** element or **<import>** elements. In this case, the units definition in the **<component>** element supersedes the model-wide definition when referenced inside that component.]

3. **Allowed values of the base_units attribute**

- If present, the value of the **base_units** attribute must be "yes" or "no".
- If not present, the value of the **base_units** attribute defaults to "no".

4. **Proper use of the base_units attribute**

- A **base_units** attribute must not be defined on a **<units>** element appearing as a child of an **<import>** element.

[18] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_fundamentals_identifiers

5.4.2 The `<unit>` element

1. **Allowed values of the `units_ref` attribute**

   - The value of the **`units_ref`** attribute must equal the value of the name attribute of a **`<units>`** element defined in the model at the URI given on the parent **`<import>`** element.

2. **Proper use of the `units_ref` attribute**

   - A **`units_ref`** attribute must not be defined on a **`<units>`** element appearing as a child of a **`<model>`** element or a **`<component>`** element.

     [A **`units_ref`** attribute would have no meaning on a units definition.]

5.4.3 The `<unit>` element

1. **Allowed use of the `<unit>` element**

   - A **`<unit>`** element must contain only the following elements:
     - **`<RDF>`** elements in the RDF namespace.
   - Each **`<unit>`** element must define a **`units`** attribute. It may also define **`prefix`**, **`exponent`**, **`multiplier`**, and **`offset`** attributes.

2. **Allowed values of the `units` attribute**

   - The value of the **`units`** attribute must be taken from the standard dictionary of units listed in Table 2 or be the value of the **`name`** attribute on a **`<units>`** element defined in the current **`<component>`** or **`<model>`** element.
   - The value of the **`units`** attribute must not reference a units definition that contains **`<unit>`** elements that in turn directly or indirectly reference the current units definition.

     [This rule prevents circular units definitions. It must be possible to break down a complex units definition into SI and user-defined base units.]

3. **Allowed values of the `prefix` attribute**

   - If present, the value of the **`prefix`** attribute must be an integer or a name taken from one of the name columns of Table 3.

     [The unit is scaled by 10 raised to the power of the specified integer or the factor corresponding to the specified name. Therefore, **`prefix`** attribute values of `"centi"` and `"-2"` are equivalent.]
   - If not present, the value of the **`prefix`** attribute defaults to `"0"`.

4. **Allowed values of the `exponent` attribute**

   - If present, the value of the **`exponent`** attribute must be a real number.
   - If not present, the value of the **`exponent`** attribute defaults to `"1.0"`.

5. **Allowed values of the `multiplier` attribute**

   - If present, the value of the **`multiplier`** attribute must be a real number.
   - If not present, the value of the **`multiplier`** attribute defaults to `"1.0"`.

6. **Allowed values of the `offset` attribute**

   - If present, the value of the **`offset`** attribute must be a real number.
   - If not present, the value of the **`offset`** attribute defaults to `"0.0"`.

7. **Proper use of the `offset` attribute**

   - A **`<units>`** element containing a **`<unit>`** element that defines an **`offset`** attribute with a value other than `"0.0"` must not contain other **`<unit>`** elements.

     [The **`offset`** attribute can only be used in a simple units definition, as defined in Section 5.2.2.]
   - A **`<unit>`** element that defines an **`offset`** attribute with a value other than `"0.0"` must not define an **`exponent`** attribute with a value other than `"1.0"`.

     [The **`offset`** attribute can only be used in a simple units definition, as defined in Section 5.2.2.]

5.5 Rules for Processor Behaviour

5.5.1 Resolving references to units definitions

The **`<units>`** element may be placed inside **`<model>`**, **`<import>`**, and **`<component>`** elements. When user-defined units are referenced by a variable or number declaration inside a component, the units definition is first looked for inside the current **`<component>`** element. If a matching units definition cannot be found, then the units definition is looked for in the **`<model>`** element and **`<import>`** elements.

5.5.2 Units associated with the MathML constants elements

This section defines the units associated with the MathML elements that appear in the ***constants*** subset of the CellML set defined in Section 4.2.3[19]. These elements represent numerical values. Operators can be applied to combinations of these elements, variables and numbers in an equation. Units must be associated with these elements to allow for equation dimension checking.

The **`<true>`** and **`<false>`** elements have units of **`cellml:boolean`**, where **`cellml:boolean`** is a set of base units defined purely for use in this specification. (Note that users may not define their own **`cellml:boolean`** units, as this is not a valid CellML identifier.) **`cellml:boolean`** units are not associated with variables or numbers, but can be produced as the result of the application of relational or logical operators, as discussed in Appendix C.3.3[20].

The **`<notanumber>`**, **`<pi>`**, **`<infinity>`** and **`<exponentiale>`** elements all have units of **`dimensionless`**.

[19] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_math_cellml_subset
[20] http://www.cellml.org/public/specification/20030930/cellml_specification.html#sec_units_rules_applying_operators_to_units_definitions