JEDEC PUBLICATION

ECXML Guidelines for Electronic Thermal System Level Models – XML Requirements

JEP181A

(Revision of JEP181, September 2020)

NOVEMBER 2023

JEDEC SOLID STATE TECHNOLOGY ASSOCIATION

NOTICE

JEDEC standards and publications contain material that has been prepared, reviewed, and approved through the JEDEC Board of Directors level and subsequently reviewed and approved by the JEDEC legal counsel.

JEDEC standards and publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for use by those other than JEDEC members, whether the standard is to be used either domestically or internationally.

JEDEC standards and publications are adopted without regard to whether or not their adoption may involve patents or articles, materials, or processes. By such action JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC standards or publications.

The information included in JEDEC standards and publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC standard or publication may be further processed and ultimately become an ANSI standard.

No claims to be in conformance with this standard may be made unless all requirements stated in the standard are met.

Inquiries, comments, and suggestions relative to the content of this JEDEC standard or publication should be addressed to JEDEC at the address below, or refer to www.jedec.org under Standards and Documents for alternative contact information.

Published by
©JEDEC Solid State Technology Association 2023
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107

PRICE: Contact JEDEC

Printed in the U.S.A.

DO NOT VIOLATE
THE
LAW!

JEDEC Solid State Technology Association
3103 North 10th Street
Suite 240 South
Arlington, VA 22201-2107
https://www.jedec.org/contact

This page intentionally left blank

ECXML GUIDELINES FOR ELECTRONIC THERMAL SYSTEM LEVEL MODELS – XML REQUIREMENTS

Contents

Contents (cont'd)

Page

Figures

ECXML GUIDELINES FOR ELECTRONIC THERMAL SYSTEM LEVEL MODELS – XML REQUIREMENTS

(From JEDEC Board Ballot JCB-23-31, formulated under the cognizance of the JC-15 Committee on Thermal Characterization Techniques for Semiconductor Packages)

1 Scope

This publication establishes the requirements for the exchange of electronic thermal system level simulation models between supplier and end user in a single neutral file format. The types of model data supported are a sub-set of what is possible to define in any one simulation tool, but with the intention of allowing model data that is common between tools to be exchanged. The data represents three-dimensional model descriptions of an electronics assembly intended to be solved so as to predict thermal and hydrodynamic behaviour using computational fluid dynamics approaches. The data is held in an XML format, conforming to an XML schema that this document describes.

2 Normative References

The following normative documents contain provisions that, through reference in this text, constitute provisions of this publication. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this publication are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

JESD51-12, *Guidelines for Reporting and Using Electronic Package Thermal Information.*

JESD15-3, *Two-Resistor Compact Thermal Model Guideline*.

JESD15-1, *Compact Thermal Model Overview.*

3 Terms and Definitions

The following terms and definitions are applicable to this XML Schema.

3.1 Data Terms and Definitions

ECXML: '*E*lectronics *C*ooling' *XML*, named so as to indicate the scope and intention of the XML information.

Producer: The name of the tool used to author an instance of ECXML, or the name of the tool that the instance of the ECXML complies with.

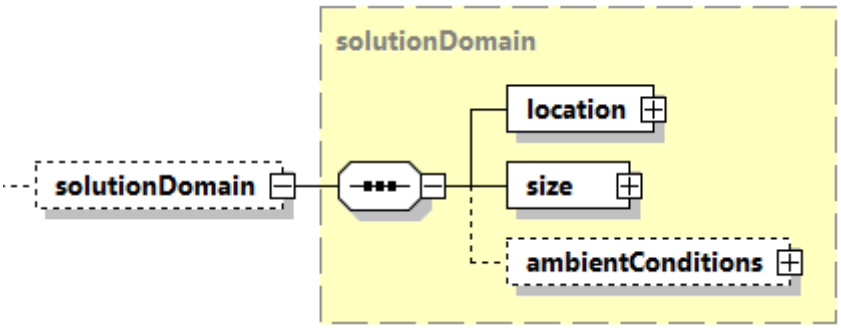
Tool: A generic name to represent any thermal simulation software that may import or export an instance of the ECXML.

Material: The physical material that an object is constructed from, including its surface properties.

3.2 XML Schema Key Terms and Definitions

An XML structure is simply a set of rules to define how to organize text into a standard structure within a file. This structure is defined by an XML schema file (.xsd). This schema file is the XML 'dictionary' and is distinct from an example, or instance, of an XML file that conforms to the schema. The structure of the schema is best represented by images. Software tools call schema viewers can enable you to see this structure as well and walk up and down the structured representation of the data. XML validators can be used to check if an XML file example conforms to its schema definition, e.g., it contains only the information that the schema allows for. Such validation, often performed on import of an XML file into a software, is useful for ensuring the integrity of the XML data and thus error free interpretation of it.

The following image shows a graphical representation of one section of the ECXML schema. A description of this image is provided so that all subsequent image descriptions of the schema may be understood. The XML element name will be indicated in *italics* in this document.



The line to the left of the *solutionDomain* element indicates that this element has a parent element. The dotted line box around *solutionDomain* indicates that this element is optional. If *solutionDomain* is included in the XML, then a *location* and *size* must also be included, but its *ambientConditions* definition is optional.

An element is a named structural entity in the XML. Using the *solutionDomain* example, this would be represented in the XML instance file as:

```
<solutionDomain>
        <location>
                <…..>
        </location>
        <size>
                <…..>
        </size>
</solutionDomain>
```

where …. represents the additional elements required by the *location* or *size* elements.

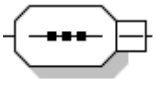
3.2 XML Schema Key Terms and Definitions (cont'd)

Many elements will contain a type. Simple types convey valuable information about what you are allowed to put under the element, summarised in the following table:

Type	Description
xs:string	A string containing characters, line feeds, carriage returns and tab characters
xs:boolean	Either true, false, 1 (which indicates true) or 0 (which indicates false).
xs:double	64-bit floating-point number
xs:integer	Numeric value without a fractional component

Elements with types beginning with xs: are standard available types that limits the format of the data, and hence improves the integrity of the data within the xml. Subsequent descriptions of ECXML elements will indicate what types are to be used.

The graphical representation of the schema presented in this guideline uses the following symbols to indicate the number and relationships of the XML data.

Graphic Descriptor	Type	Description
0..∞	Cardinality	Shown below an element. The cardinality defines if the element can be repeated or not. In this case there must be at least 0 and as many as needed. When 0 is the first value it means that this element is optional and can be excluded. When there is no cardinality there should only be one element of that type under the parent element.
	Sequence	Any of the elements to the right of the sequence can be added under the element to the left of the node. A sequence further retrains the XML by requiring the element be provided in the order specified by the schema which is also the order shown in the images.
	Choice	Only a single element to the right of the choice can be added under the element to the left of the node.
	All	All elements to the right of the node must be added under the element to the left of the node, unless the element to the right is optional. The node "All" reduces the constraints enforced by Sequence, insofar that there is no restriction on the order of the element within the xml file.

4 ECXML Schema Definition

The ECXML schema is described by use of graphical representations of the hierarchical order of the XML elements.

4.1 Units

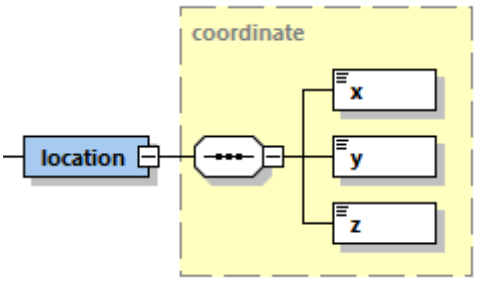
All numeric values are implicitly assumed to be in SI units. The schema does not allow for specification of a non-SI unit. The following table lists the SI units that are used by the ECXML schema:

Unit Type	SI Unit
Location/Size	Meter (m)
Temperature	Kelvin (K)
Pressure	Pascal (Pa)
Power Dissipation	Watts (W)
Thermal Conductivity	(W/mK)
Thermal Resistance	(K/W)
Specific Heat	(J/kgK)
Density	(kg/m^3)
Temperature dependent thermal conductivity coefficient	(W/mK^2)
Flow Rate	(m^3/s)
Free area ratio, Loss Coefficient, Surface emissivity	Dimensionless

4.2 ECXML Common Elements

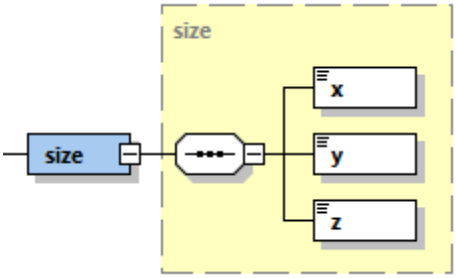
Elements that are used twice or more by other elements, e.g., low level elements, are described first.

4.2.1 Location



The *location* element defines the location (position) of an object by 3 Cartesian coordinates; X, Y and Z as xs:double types. These may be positive or negative values and refer to a single 0,0,0 global origin point. These 3 coordinates are the smallest X, Y, Z values of all eight corner locations of an object's bounding box. Other coordinate systems such as Polar are not supported.

4.2.2 Size



The *size* element defines the size of an object by the extents of its bounding box in the 3 coordinate directions. Size values are of positive xs:double types. Note that rotation of objects is not supported.

4.2.3 Active

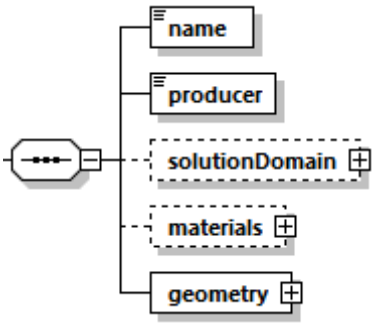
The *active* element is an xs:boolean setting that indicates whether the object should be considered by a simulation or not. For the object to be active and considered, set = true or 1. For the object to be in-active and thus ignored by a simulation, set = false or 0.

4.2.4 Plane

The *plane* element is used by two dimensional objects to indicate both an orientation and a direction (normal). The available settings are; +XY, -XY, +YZ, -YZ, +XZ, -XZ. The two letters refer to the plane orientation. The sign refers to the direction in which the object 'points'; positive in the increasing axis direction, negative in the decreasing axis direction. Other non-Cartesian axis aligned directions are not supported.

4.3 ECXML Top Level Elements

Elements at the top level of the schema are described.



4.3.1 Name

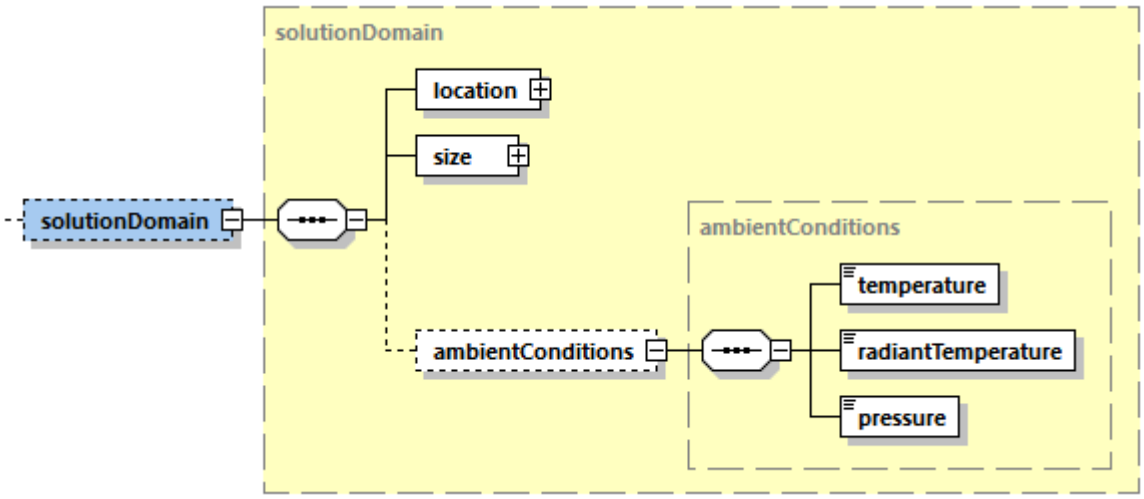
The *name* element is an xs:string that, at this level, defines the name of the model.

4.3.2 Producer

The *producer* element is an xs:string indicating the authoring tool. The values available are defined in the schema. Indication of which tool authored the XML, or what tool the XML is compliant with, is important when determining how to interpret certain situations such as when geometry is defined as overlapping (see 4.5).

4.3.3 Solution Domain

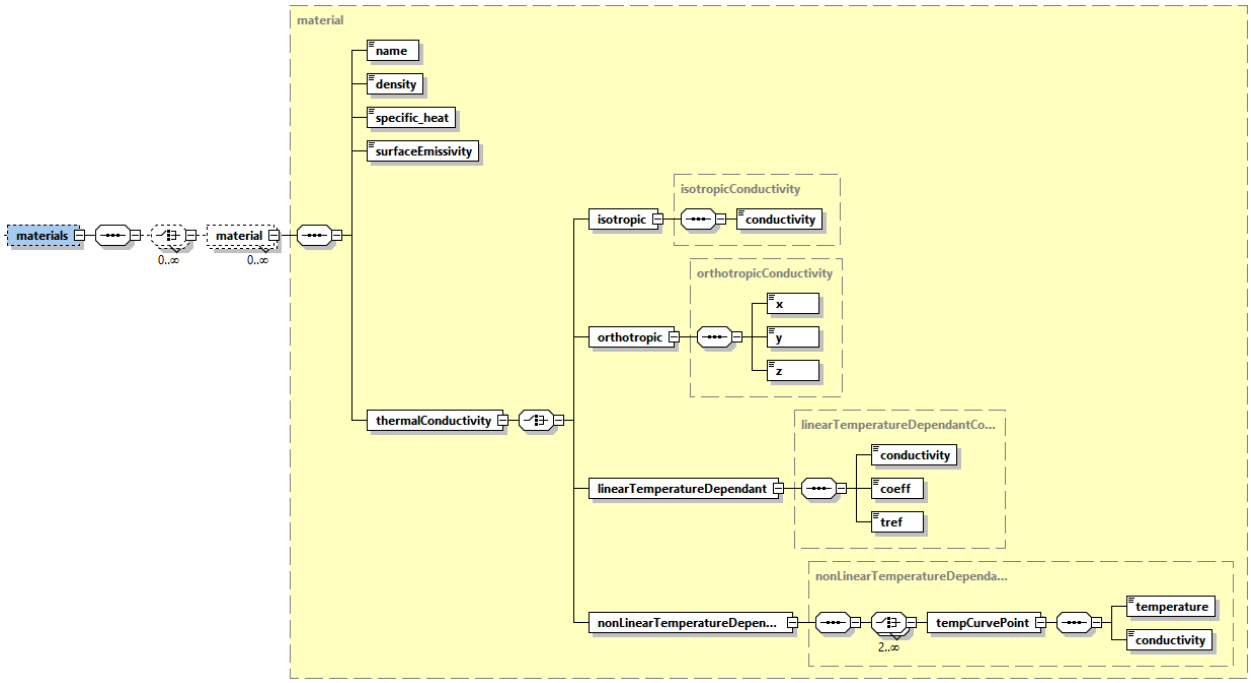
The *solutionDomain* element describes the computational domain within which the model geometry is defined.



This is an optional element. Electronics thermal models that represent just a part, or sub-assembly, need not have a *solutionDomain* defined in the XML file. If *solutionDomain* is defined then *location* and *size* are required. The *ambientConditions* values, representing the condition outside of the computational domain, are optionally specified. These are the external ambient *temperature*, an external *radiantTemperature* that is used for radiative exchange of heat from within the computational domain and the external ambient gauge *pressure*. All 3 types being xs:double.

4.3.4 Materials

The optional *materials* element holds a list of zero or more *material* elements that can be referenced by (attached to) relevant model objects.



For each *material*, a *name* is required as well as a definition of *density*, *specific_heat*, *surfaceEmissivity* and *thermalConductivity*.

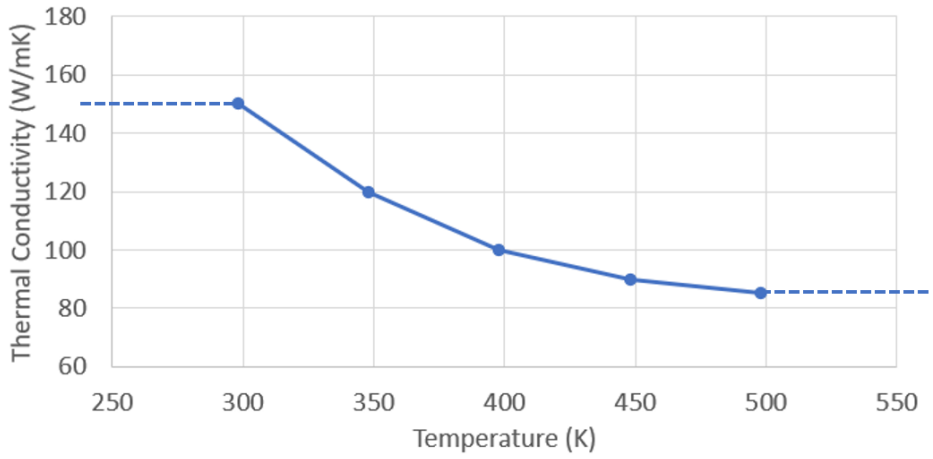
density and *specific_heat* are of type xs:double. *surfaceEmissivity* is of type xs:ratio, i.e., a value between 0.0 and 1.0.

There are 4 options for the definition of thermal conductivity:

- *isotropic*. A single value of type xs:double
- *orthotropic*. 3 values, each of type xs:double, representing the thermal conductivity in each of the 3 coordinate directions.
- *linearTemperatureDependent*. A linear relationship between thermal conductivity and temperature defined by: thermal conductivity = ***conductivity*** + ***coeff*** (Temperature – ***tref***). All 3 values defining this linear relationship are of type xs:double
- *nonLinearTemperatureDependent*. A piece-wise linear definition of the relationship between temperature and thermal conductivity. Defined by 2 or more *temperature*/*conductivity tempCurvePoint* point pairs of type xs:double.

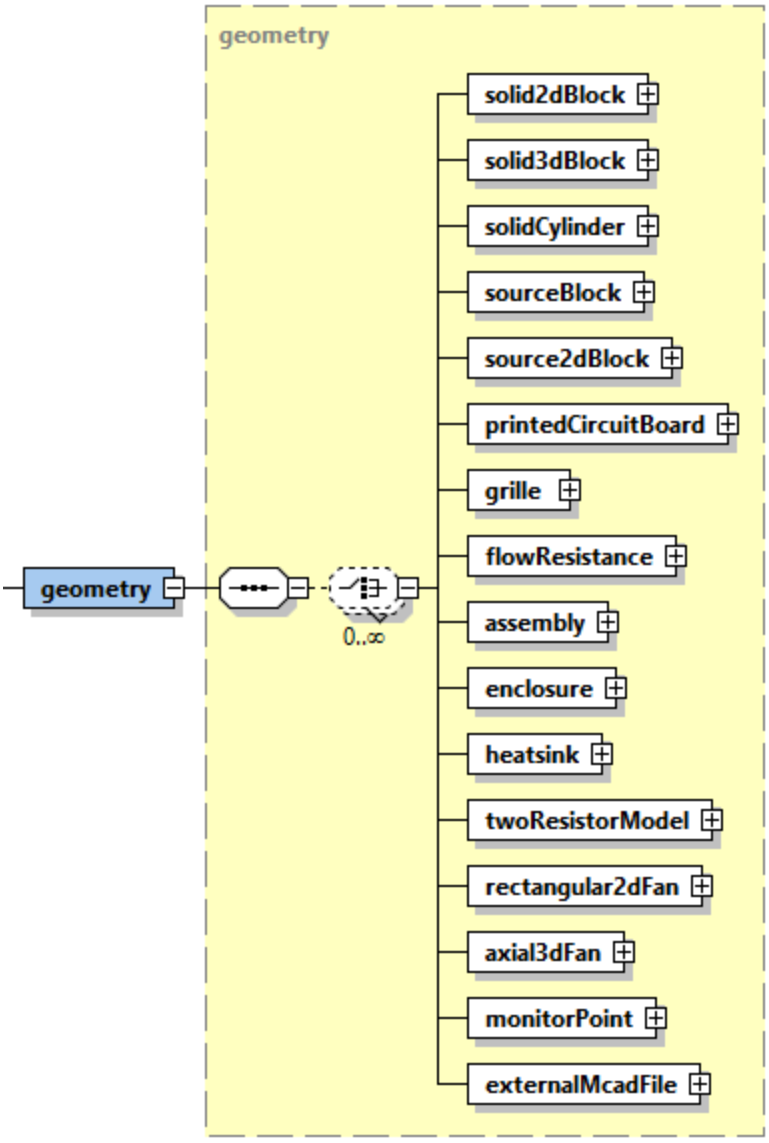
4.3.4 Materials (cont'd)

The 2 extrema points of a *nonLinearTemperatureDependent* description should define a range big enough so that expected operation occurs within it. If operation occurs outside of this range, then the temperature dependent thermal conductivity is assumed constant beyond the 2 extrema points:



4.3.5 Geometry

The *geometry* element holds a list of all model elements that go to make up construction of the system level thermal model.



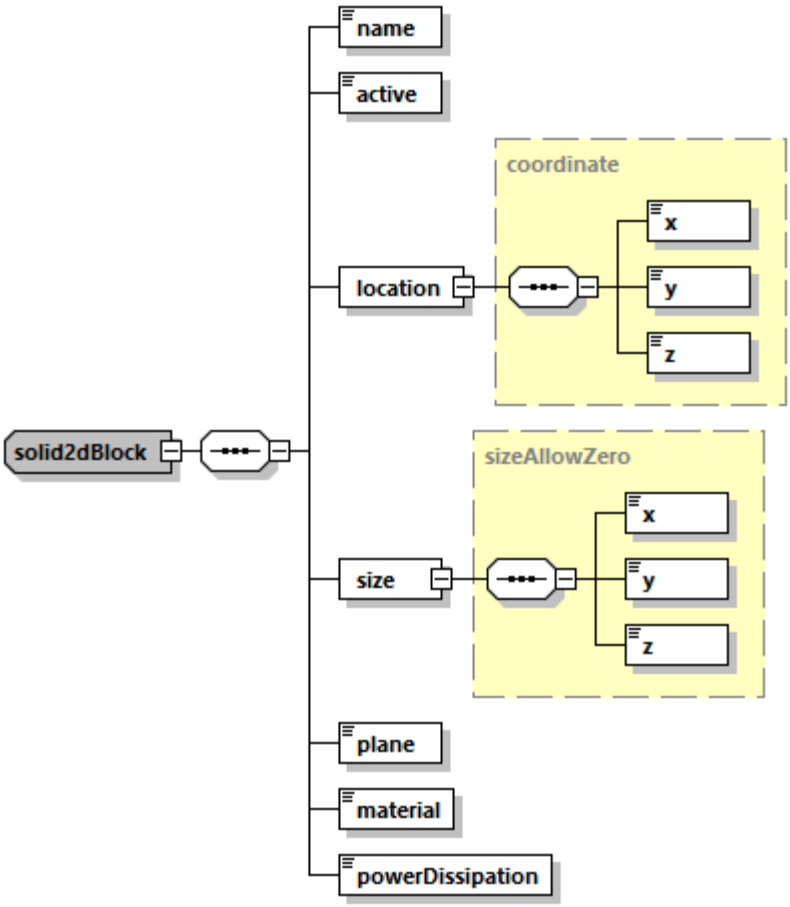
Zero or more child model objects may be present in the XML file but the parent *geometry* element must always be present.

4.4 ECXML Geometry Elements

Each model geometry element that can be defined are now described, with reference to any preceding object that has a similar data structure.

4.4.1 Solid 2D Block

The *solid2dBlock* element defines a solid object that is to be represented as a two dimensional ‘plate’ or ‘sheet’ in the simulation.



It has a *name* and an *active* setting.

It is envisioned that one of the 3 sizes is considered small enough in relation to the other 2 to the extent where the object will be considered 2 dimensional by a simulation, in the specified plane. For example if the Z size was by far the smallest compared to X and Y size, the plane setting would contain XY. That Z size may even be set to 0.0 as the X, Y and Z sizes are of an xs: sizeAllowZero type that allows for values greater than or equal to zero.

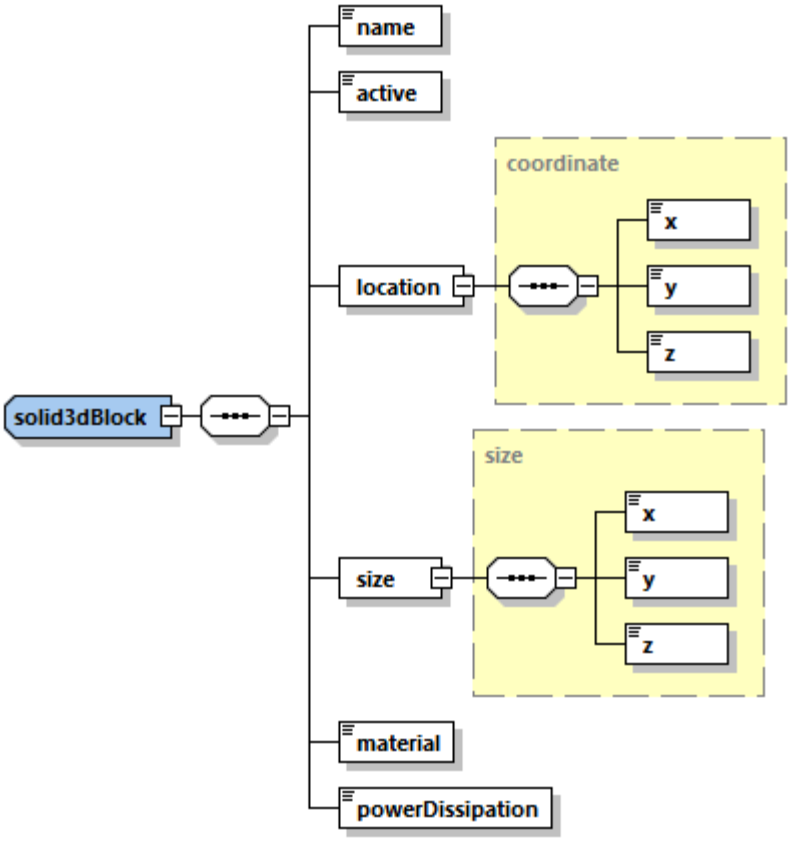
4.4.1 Solid 2D Block (cont'd)

A *material* name of type xs:string is defined relating to one of the named items in the *materials* element list (see 4.3.4).

A *powerDissipation* of type xs:double is defined. The direction that this power is applied in is taken from the sign of the plane definition.

4.4.2 Solid 3D Block

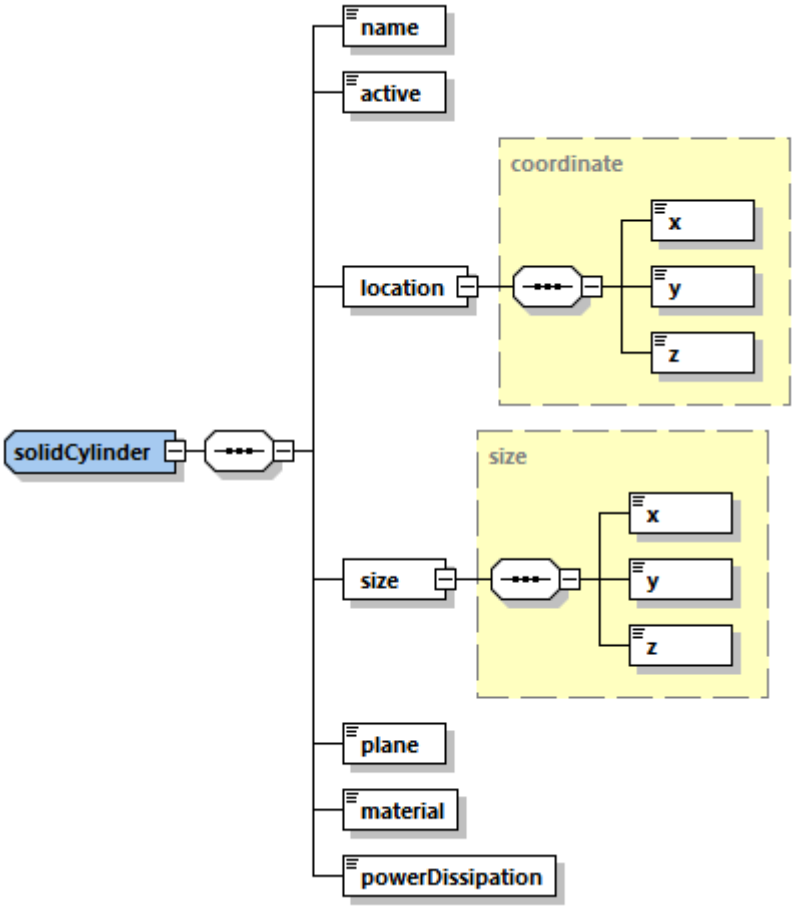
The *solid3dBlock* element defines a solid object that is to be represented as a three dimensional 'cuboid' in the simulation.



Identical to the *solid2dplate* object apart from not having to specify a plane orientation. Also, the X, Y, Z size xs:double values must all be greater than zero.

4.4.3 Solid Cylinder

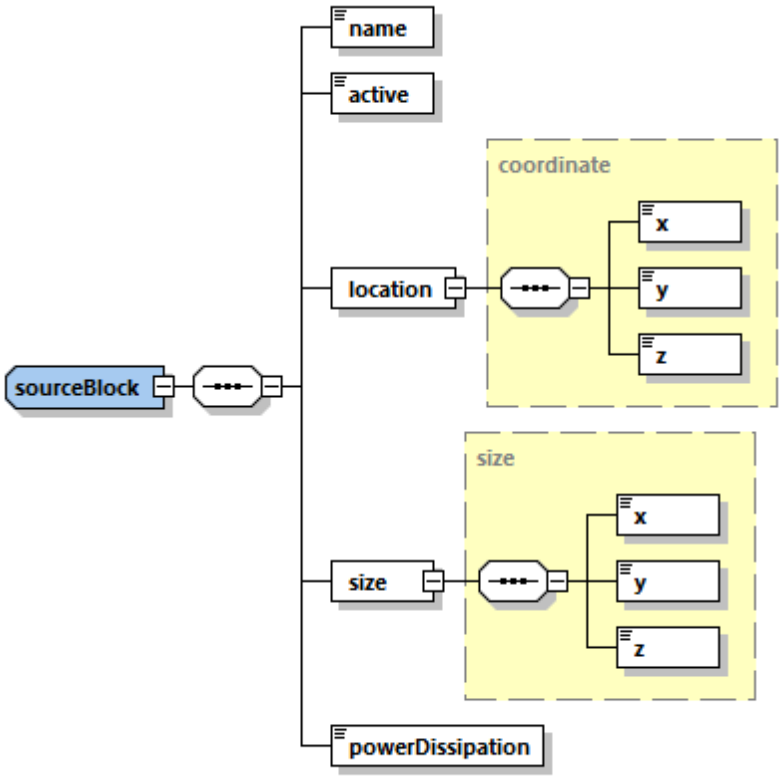
The *solidCylinder* element defines a solid cylindrically shaped object.



Similar to the *solid3dBlock* object, but having a *plane* definition that sets the orientation of the cylinder so that its central axis is normal to that plane.

4.4.4 Source Block

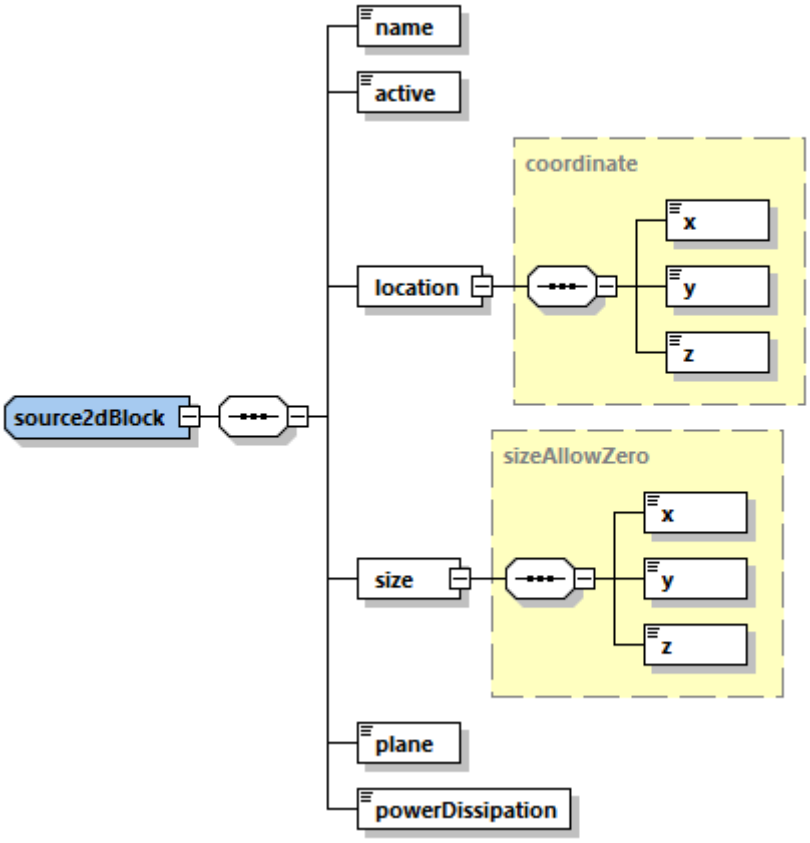
The *sourceBlock* element defines a 3-dimensional object specifying a volume over which a power is dissipated.



Similar to a *solid3dBlock*, but not requiring a *material* to be defined.

4.4.5 Source 2D Block

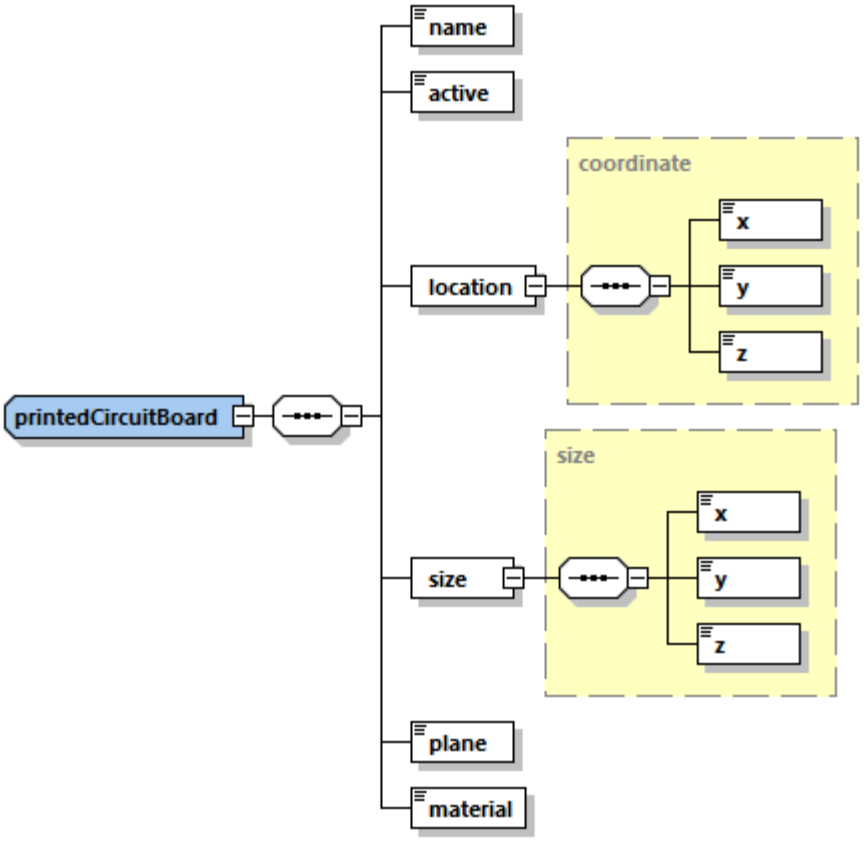
The *source2dBlock* element defines a 2-dimensional plane from which power is dissipated.



Similar to a *solid2dBlock*, but not requiring a material to be defined.

4.4.6 Printed Circuit Board

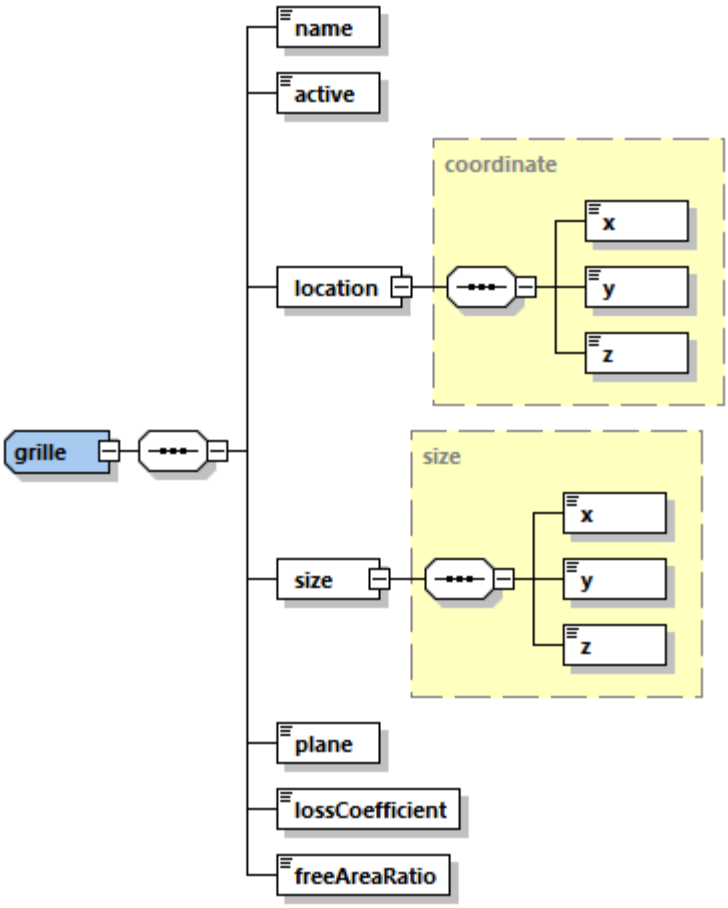
The *printedCircuitBoard* element defines a three-dimensional solid object representing a printed circuit board (PCB).



Similar to a *solid3dCuboid* but not allowing for a definition of a power dissipation. Despite the similarity, it is definable as a distinct object to allow for the importing tool to recognise it as a PCB and consider it appropriately.

4.4.7 Grille

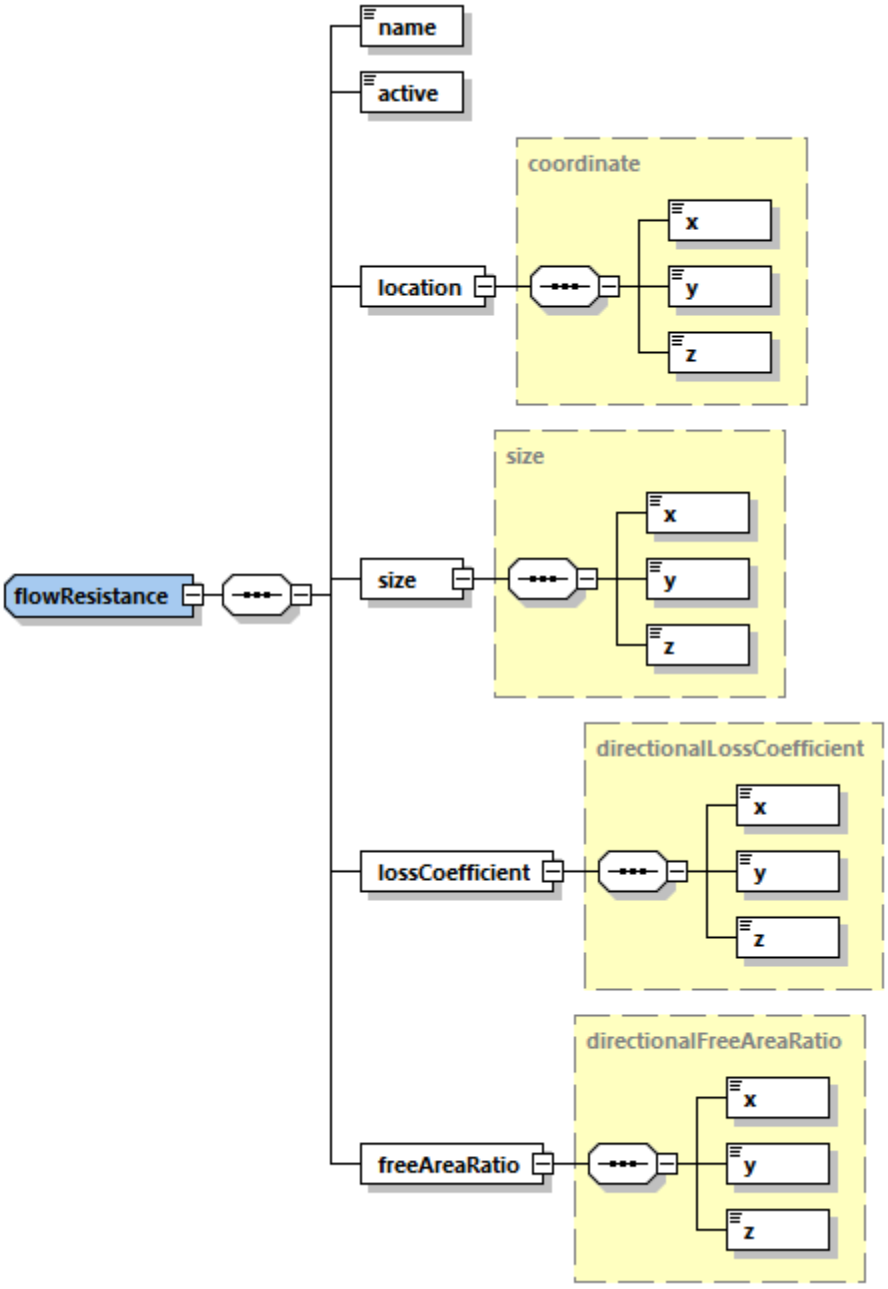
The *grille* element defines a two-dimensional object that resists air flow passing through it, e.g., a grille, louvre, or perforated plate.



The flow resistive characteristics of the grille are defined by a *lossCoefficient* (type xs: double) and a *freeAreaRatio* (type xs: ratio, value between 0.0 and 1.0). Refer to 4.5.2 for a description of how each supported authoring tool considers the definition of loss coefficient.

4.4.8 Flow Resistance

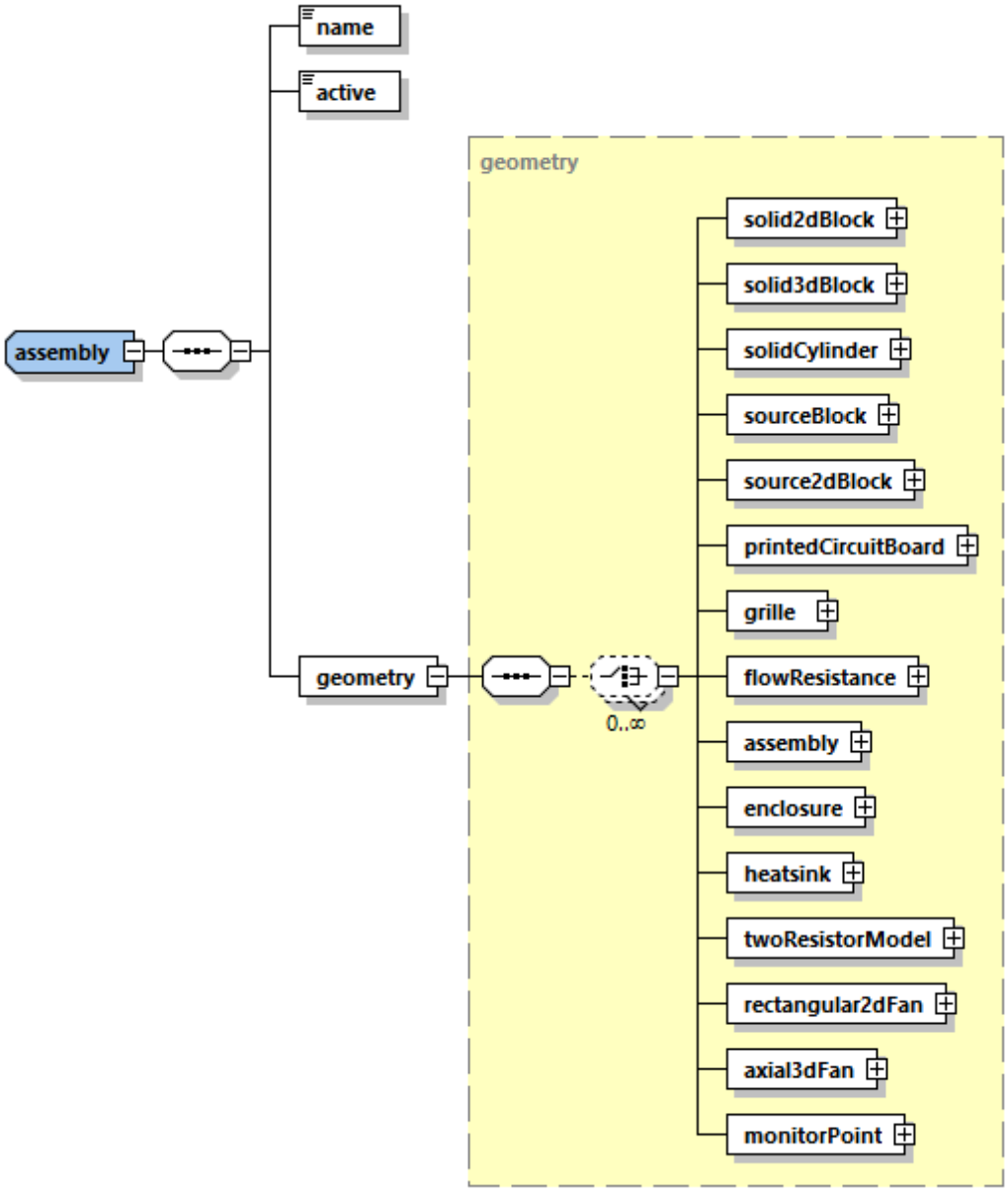
The *flowResistance* element defines a three-dimensional object that resists air flow passing through it.



Similar to a *grille* but without a plane orientation and having 3 values for both *lossCoefficient* and *freeAreaRatio*, in each of the coordinate directions. Refer to 4.5.2 for a description of how each supported authoring tool considers the definition of loss coefficient.

4.4.9 Assembly

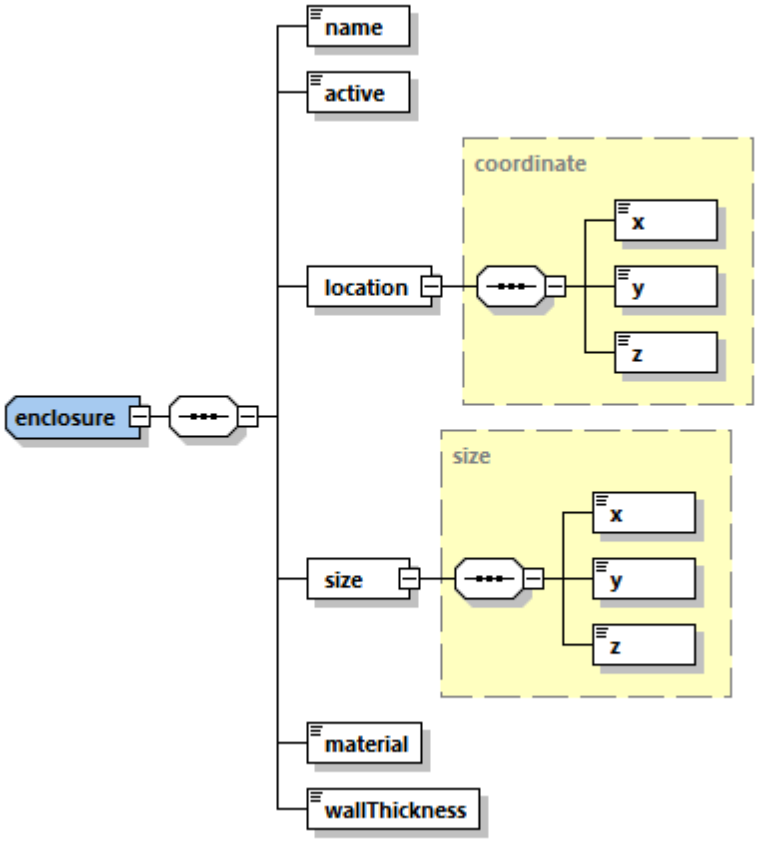
The *assembly* element is an object to group one or more objects, or other assemblies. This allows for a hierarchal description of the system level thermal model to be defined, allowing for easier interaction after import into the tool.



If defined, it must have a *name*, *active* status and a *geometry* element containing zero or more objects.

4.4.10 Enclosure

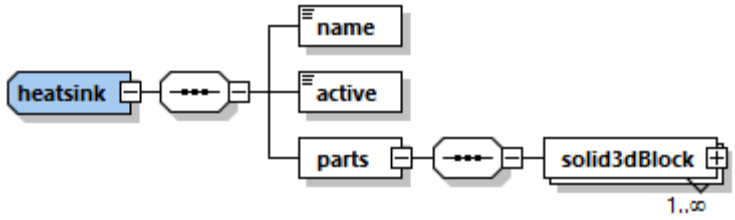
The *enclosure* element defines an object representing a box, housing, chassis etc. that has 6 sides.



The *size* represents the bounding box size of the entire object. The *wallThickness* defines the thickness of each of the 6 walls as type xs:double.

4.4.11 Heatsink

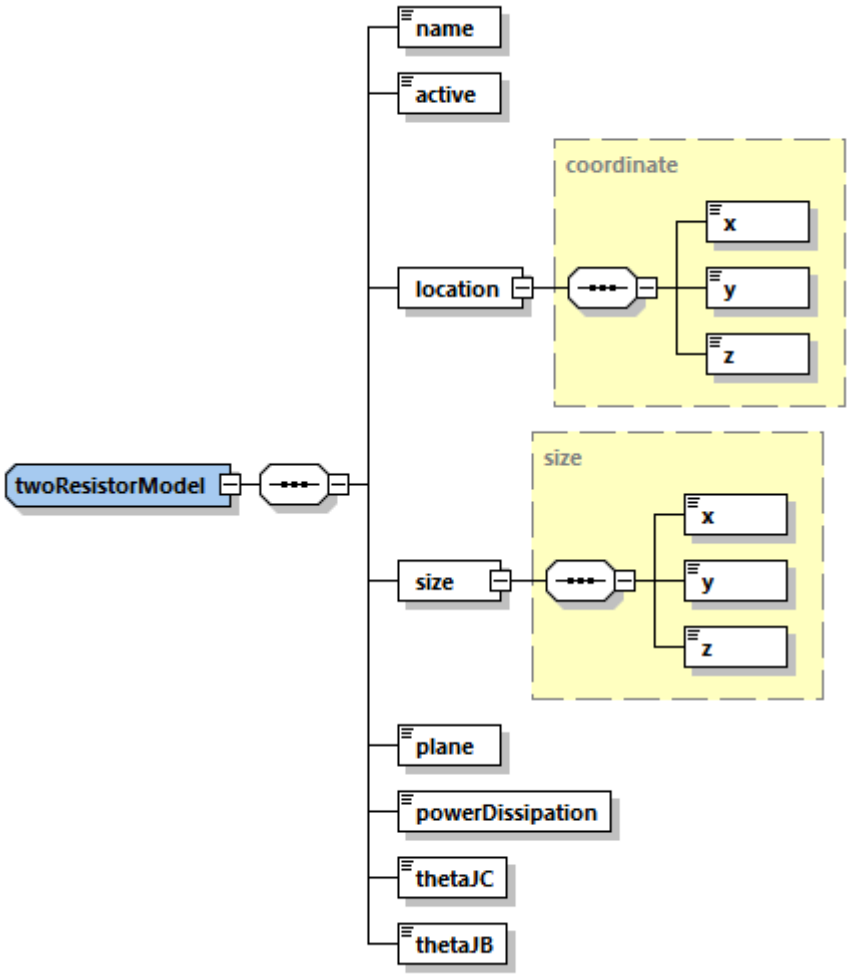
The *heatsink* element defines an object representing a heatsink by grouping one or more *solid3dBlock* objects together



Similar to an *assembly*, but only allowing solid3dBlock objects to be grouped and considered as a heatsink object by the importing tool. Separate *solid3dBlock* objects would be used for example to represent the heatsink base and each individual fin or pin. The heatsink *material* is set at the individual *solid3dBlock* level, not at the heatsink level.

4.4.12 2 Resistor Model

The *twoResistorModel* element defines an object representing a 2-resistor compact thermal model of a package.

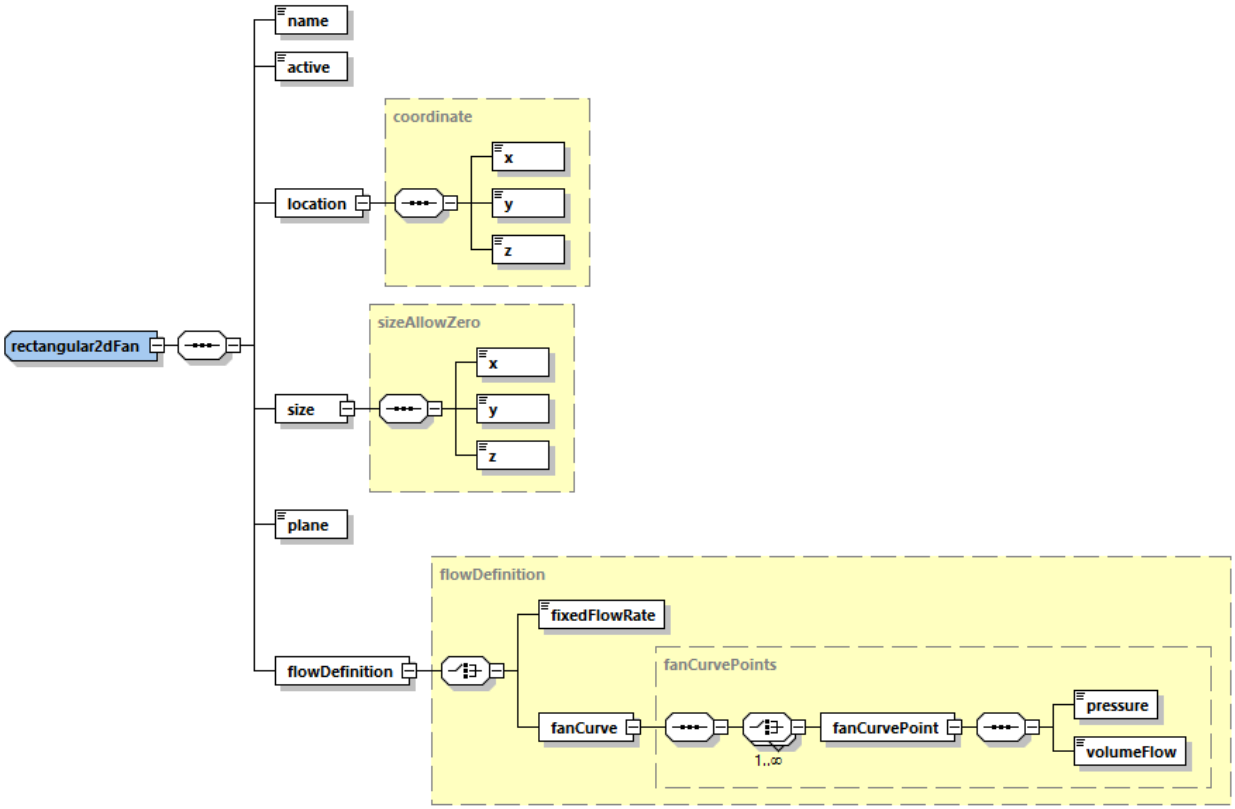


A three-dimensional shape is defined as the bounding box of the 2-resistor model. The 2 thermal resistance values, *thetaJC* (resistance between junction and case, type xs:double) and *thetaJB* (resistance between junction and board, type xs:double) are applied with respect to the plane orientation. The defined plane will determine the case face and board face of the 2-resistor model object, e.g., +XY plane will define the case face as being the top Z face and the board face as the bottom Z face.

The *powerDissipation* value will set the amount of power dissipated at the (internal) junction node.

4.4.13 Rectangular 2D Fan

The *rectangular2dFan* element defines a two-dimensional rectangular object through which air flow is induced.



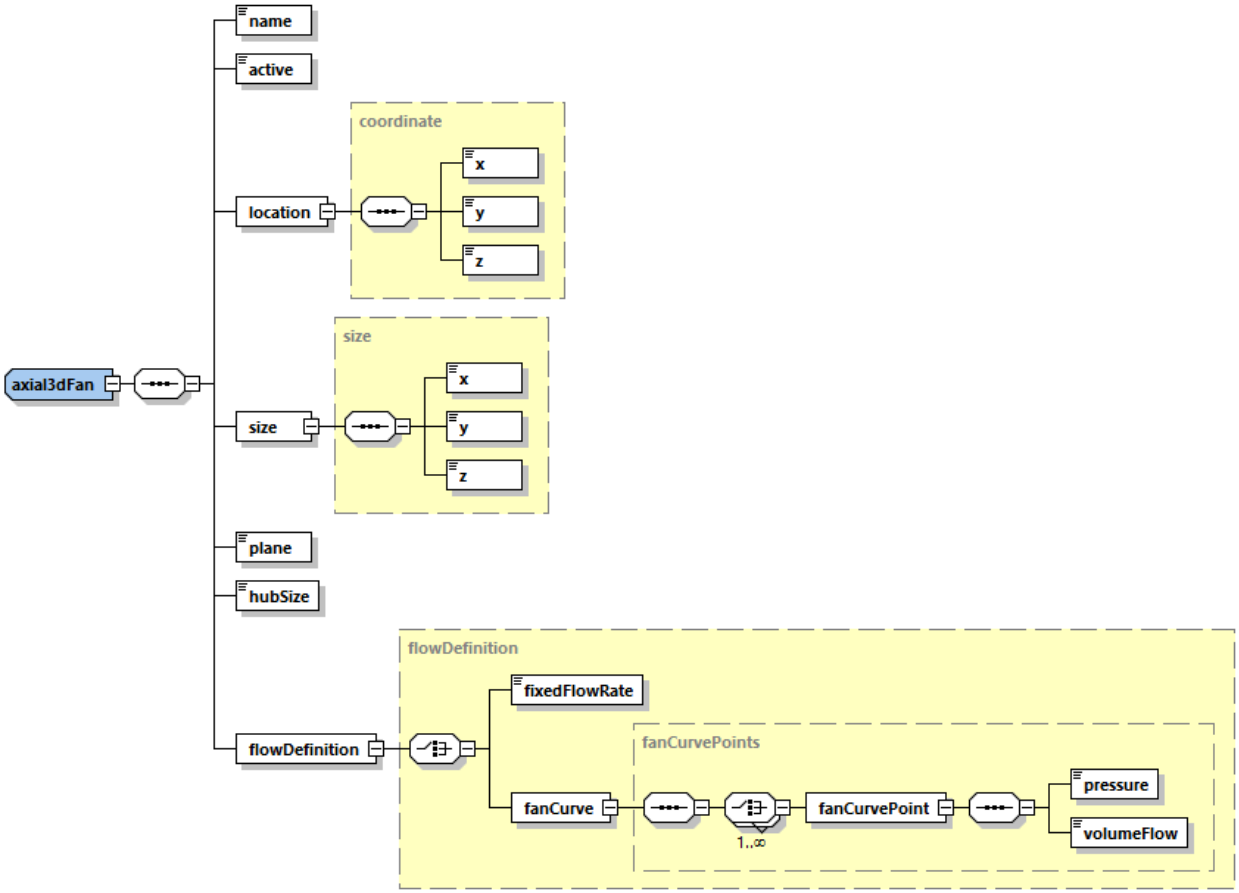
The *plane* orientation defines the fan axis orientation and the flow direction, e.g. –YZ will orient the fan in the YZ plane and direct the air flow in the decreasing X axis direction.

The *flowDefinition* allows for specification of a constant fixed flow rate (as type xs:double) or by one or more *fanCurvePoints* representing a piecewise linear approximation of the fan curve. Each *fanCurvePoint* requiring a *pressure* and *volumeFlow* (rate) pair of type xs:double values.

Despite intending to represent a two-dimensional fan type object, x, y and z sizes are definable, though any one can be set to zero. This is to allow for fan object mapping between tools where one tool does not support two-dimensional fan objects.

4.4.14 Axial 3D Fan

The *axial3dFan* element is similar to the *rectangular2dFan* but representing a three-dimensional axial fan.

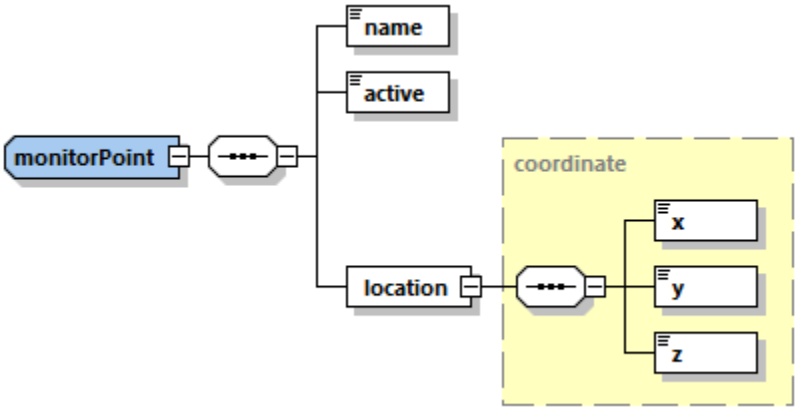


Orientation of the axial fan axis, the direction of the induced air flow and the *flowDefinition* is defined in the same was as for *rectangular2dFan*.

A *hubSize* can be defined as type xs:double, representing the diameter of the internal hub/motor of the axial fan.

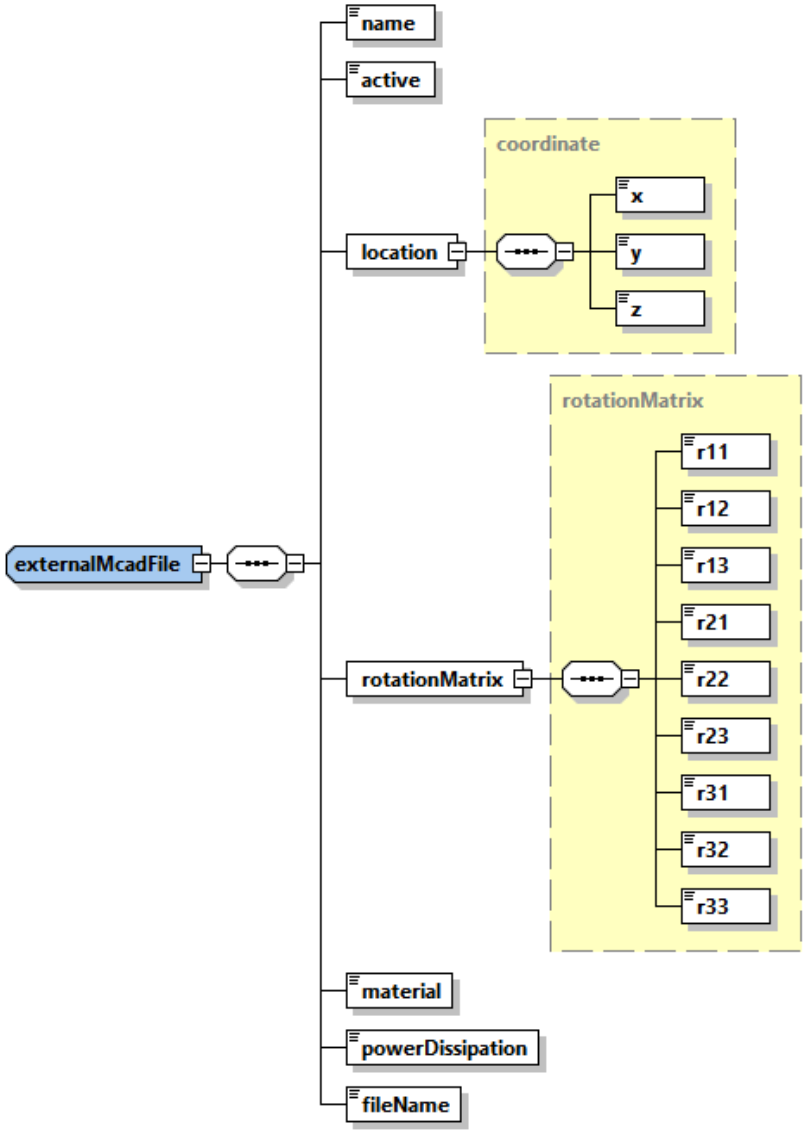
4.4.15 Monitor Point

The *monitorPoint* element defines a marked location in the model that may be used for convergence monitoring and post processing by the importing tool.



4.4.16 externalMcadFile

The *externalMcadFile* element allows for a reference to a geometry defined in an MCAD file to be included in the thermal model.



The *location* element defines the location (position) of an object by 3 Cartesian coordinates; x, y and z as xs:double types. These may be positive or negative values and refer to the translation of the 0,0,0 origin point of the referenced MCAD geometry to its location in the ECXML model.

4.4.16 ExternalMcadFile (cont'd)

The *rotationMatrix* element defines a 3x3 matrix that is used to apply a rotation transformation to the referenced MCAD geometry in the ECXML model. The 'r**' elements refer to the matrix values as follows:

$$\begin{bmatrix} r11 & r12 & r13 \\ r21 & r22 & r23 \\ r31 & r32 & r33 \end{bmatrix}$$

The 3x3 *rotationMatrix* is intended solely to represent the rotation to be applied to the referenced MCAD file geometry. Although the matrix might be defined to represent other transformations such as scaling or shearing, this is beyond scope and intention and should therefore be avoided. Examples of the application of the *rotationMatrix* are provided in 5.2.

The *powerDissipation* element defines a single power dissipation value that is assumed to be distributed over the referenced MCAD file geometry such that it has a constant power density.

The *fileName* element is an xs:string setting that can be used to define a locally referenced file name, a path to and including a local file or a URL.

4.4.16.1 Translation and Rotation Precedence

The *location* x, y, z of the referenced MCAD file is held independently from its *rotationMatrix* transformation. The referenced MCAD file geometry should be considered first by its rotation around its origin point, followed by a translation to its location.

4.5 Differences Between Authoring and Importing Tool ECXML Interpretation

Although, due to the use of an XML schema, ECXML can provide validated self-consistent data definitions of system level thermal models, there may be differences in how the importing tool will interpret the data compared to how the same data might have been interpreted by the authoring tool. The following sections clarify 2 such instances.

4.5.1 Overlapping Objects

Overlapping objects is a situation that does not invalidate the XML when checked against the schema. A given tool may interpret the overlapping condition differently however, compared to how the authoring tool considered the overlapping situation.

For situations where the ECXML does contain overlapping objects, it is important to appreciate what the authoring tool was. This can be found in the *producer* element in the ECXML file, e.g.,:

```
<producer>FloTHERM</producer>
```

or:

```
<producer>Icepak</producer>
```

4.5.1 Overlapping Objects (cont'd)

The following tables outline how overlapping conditions are handled by the authoring tools:

Simcenter Flotherm from Siemens Digital Industries Software

Overlapping condition	How the authoring tool handles the overlapping condition
Solid overlapping solid	Solid object defined lower down the model hierarchy will overwrite one defined above it
Source overlapping source	Power density added on the overlap, total power preserved
2D flow resistance overlapping 2D solid	2D flow resistance will overwrite 2D solid when the resistance is defined lower down the model hierarchy
2D fan overlapping a 2D solid	2D fan will overwrite 2D solid when the fan is defined lower down the model hierarchy

6SigmaET from Cadence

Overlapping condition	How the authoring tool handles the overlapping condition
Solid overlapping solid	Solid object defined lower down the model hierarchy will overwrite one defined above it
Source overlapping source	Power density added on the overlap, total power preserved
Source overlapping solid with heat	Power density added on overlap
Solid with heat overlapping solid with heat	Not allowed – produces critical error
Solids overlapping flow resistances	Solids will overwrite flow resistances
Solids overlapping fans	• 3D solids extending out from the fan will overwrite flow from the faces of the fan. • 2D solids and resistances on the face of the fan will be ignored (i.e. fan will overwrite always) • Solids 'inside' the fan will be ignored

scSTREAM and HeatDesigner from MSC

Overlapping condition	How the authoring tool handles the overlapping condition
Solid overlapping solid	Solid object defined lower down the model hierarchy will overwrite one defined above it
Source overlapping source	Power density added on the overlap, total power preserved
Solids overlapping flow resistances	Solids will overwrite flow resistances

4.5.1 Overlapping Objects (cont'd)

Icepak from Ansys

Overlapping condition	How the authoring tool handles the overlapping condition
Solid overlapping solid	Fully embedded solids get the higher priority by default. For partial overlaps, overlapped region belongs to the solid that has a specified higher priority.
Sheet overlapping sheet	Fully embedded sheets get the higher priority by default. For partial overlaps, overlapped region belongs to the sheet that has a specified higher priority.
Sheet overlapping solid	Sheet objects always have higher priority compared to solid objects

4.5.2 Definition of Loss Coefficient

Loss coefficient is defined as:

$$Loss\ Coefficient = \frac{dP}{\frac{1}{2}\ \rho u^2}$$

Where *dP* is the pressure drop over the object, ρ is the fluid density and *u* is a fluid velocity. In addition, a free area ratio is defined in conjunction with the loss coefficient in the ECXML instance. The fluid velocity can refer to the velocity of fluid as it passes through the device, or the velocity of the fluid as it approaches the object. The free area ratio can be used to convert the approach upstream velocity to a device velocity and vice versa. In addition, the free area ratio can be used alone to determine a loss coefficient using a tool specific function.

How each supported authoring tool considers the use of loss coefficient and free area ratio on import of a *grille* or *flowResistance* object is described as follows:

Simcenter Flotherm from Siemens Digital Industries Software
For both *grille* and *flowResistance* objects the loss coefficient is assumed based on the velocity of the fluid passing through the device and thus the free area ratio is used in conjunction with the loss coefficient to determine the pressure drop. The 3 loss coefficients for the 3D *flowResistance* object are interpreted to be a loss per unit length (1/m).

6SigmaET from Cadence
For the *grille* object the free area ratio only is used to determine the loss coefficient using a tool specific function. For the *flowResistance* object either the free area ratio only is used to determine the loss coefficient using a tool specific function or user specified loss coefficient can be set. The 3 loss coefficients for the 3D *flowResistance* object are interpreted to be a loss per unit length (1/m).

4.5.2 Definition of Loss Coefficient (cont'd)

scSTREAM and HeatDesigner from MSC
For both *grille* and *flowResistance* objects, if the loss coefficient has positive value, the pressure drop is based on the fluid approach velocity. If the loss coefficient has a zero or a negative value the loss coefficient is determined from the free area ratio using a tool specific function. The 3 loss coefficients for the 3D *flowResistance* object are interpreted to be a loss per unit length (1/m).

Icepak from Ansys
For both the *grille* and *flowResistance* objects, the loss coefficient can be defined for each direction in both device or approach velocity formulations using tool specific functions or a user specified piecewise linear loss coefficient can be used. However, it is not defined in terms of a loss per unit length i.e., the entire resistance needs to be characterized in the specified direction.

5 ECXML Annotated Examples

5.1 A System Level ECXML Example

A simple EXCML system level example is presented in Figure 1, relating the relevant sections of the ECXML file to the corresponding objects of the 3D electronics thermal model. Not all supported objects are demonstrated, just a sub-set sufficient to enable an appreciation of the syntax of the ECXML XML instance file. A full listing of this ECXML example is shown in Annex A.

The example is of a computational wind tunnel containing a 2 Resistor compact thermal model placed on a PCB. 2 monitor points are placed within the 2 Resistor model to provide junction and case temperature indication. A rectangular 2D fan is placed on one end introducing fluid flow into the computational domain with a grille on the other end allowing that fluid to exit. The other 4 sides are covered with 2D cuboids.

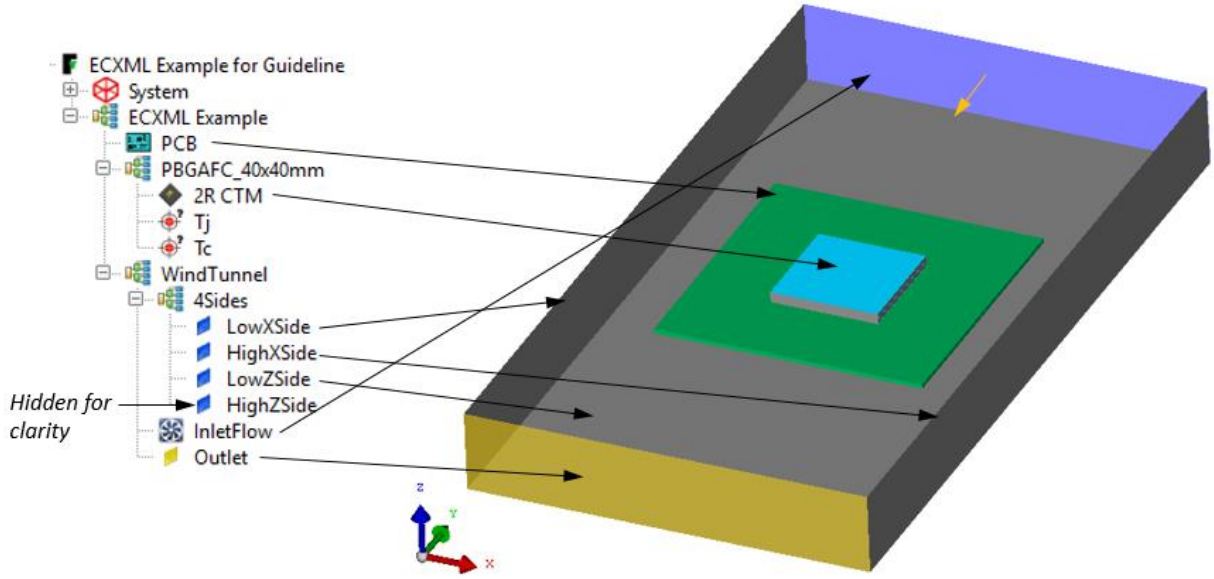


Figure 1 — EXCML System Level Example

The solution domain is the same size as the bounding box of all objects.

Due to the nested nature of the XML it can be displayed in a various ‘collapsed’ states to aid description. In this case Notepad++ is used as the XML viewer.

The top level starts with an XML verison and encoding line, required by the XML format standard. <neutralXML> … </neutralXML> tags bound all of the subsequent data:

5.1 A System Level EXCML Example (cont'd)

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<neutralXML>

  <name>ECXML Example for Guideline</name>

  <producer>FloTHERM</producer>

  <solutionDomain>

  <materials>

  <geometry>

</neutralXML>
```

The <name> element defines the name of the original model. The <producer> element specifies the name of the tool that authoured the ECXML instance file. Any tool specific conditions regarding the interpretation and intention of the ECXML data, as described in 4.5, can be referenced to the producing tool.

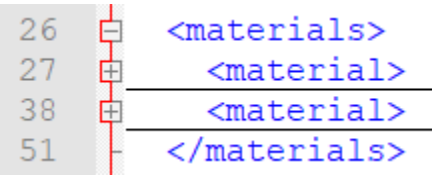
The solutionDomain element contains the location, size and ambient conditions.

```
  <solutionDomain>
    <location>
      <x>-0.07090000</x>
      <y>-0.14900000</y>
      <z>-0.00790000</z>
    </location>
    <size>
      <x>0.15000000</x>
      <y>0.28000000</y>
      <z>0.03000000</z>
    </size>
    <ambientConditions>
      <temperature>313.150</temperature>
      <radiantTemperature>313.150</radiantTemperature>
      <pressure>0.000</pressure>
    </ambientConditions>
  </solutionDomain>
```

5.1 A System Level EXCML Example (cont'd)

The location is with respect to a global origin point that, in this example lies within the solition domain, thus the solution domain location coordinates are negative. Note that SI units are used for all numeric values, including temperature in Kelvin (see 4.1).

The <materials> element contains a list of all materials in the model. They are referenced by name by subsequent model objects. Note that materials can be defined in this section that are not subsequently referenced by any model object.



5.1 A System Level EXCML Example (cont'd)

In this example there are 2 materials, 'Walls' and 'PCB Material':

```
<materials>
  <material>
    <name>Walls</name>
    <density>1200.00000000</density>
    <specific_heat>1500.00000000</specific_heat>
    <surfaceEmissivity>0.95000000</surfaceEmissivity>
    <thermalConductivity>
      <isotropic>
        <conductivity>0.90000000</conductivity>
      </isotropic>
    </thermalConductivity>
  </material>
  <material>
    <name>PCB Material</name>
    <density>1973.00000000</density>
    <specific_heat>655.95794678</specific_heat>
    <surfaceEmissivity>0.90000000</surfaceEmissivity>
    <thermalConductivity>
      <orthotropic>
        <x>38.77000046</x>
        <y>38.77000046</y>
        <z>0.33330452</z>
      </orthotropic>
    </thermalConductivity>
  </material>
</materials>
```

The 'Walls' material has a single isotropic value of thermal conductivity. The 'PCB Material' has an orthotropic thermal conductivity, defined by 3 values relating to the thermal conductivity in each of the 3 global coordinate directions. In this case identical X and Y in-plane conductivities and a lower Z through-plane conductivity.

5.1 A System Level EXCML Example (cont'd)

The <geometry> element cantain the definition of all model objects, in this case in a hierachal assembly structure. At this top level the PCB object is defined together with 2 sub-assemblies:

```
 53   <geometry>
 54     <printedCircuitBoard>
 70     <assembly>
112     <assembly>
227   </geometry>
```

The PCB is defined in terms of a location and size. The +XY plane indicates the plane in which the PCB lies so that in this example its through plane direction is in the positive Z direction, i.e. its top surface is on the high Z face of the PCB object. The material that the PCB is constructed from is a named reference to a material in the materials list as defined above:

```
54     <printedCircuitBoard>
55       <name>PCB</name>
56       <active>true</active>
57       <location>
58         <x>-0.04590000</x>
59         <y>-0.05000000</y>
60         <z>0.00195000</z>
61       </location>
62       <size>
63         <x>0.10000000</x>
64         <y>0.10000000</y>
65         <z>0.00160000</z>
66       </size>
67       <plane>+xy</plane>
68       <material>PCB Material</material>
69     </printedCircuitBoard>
```

5.1 A System Level EXCML Example (cont'd)

The 2 sub-assemblies are both named, active and contain geometry. Note that the assemblies do not have locations of their own, they are just used to group objects, all of which have their own location with repsect to a global 0,0,0 model origin point:

```
 53   <geometry>
 54     <printedCircuitBoard>
 70     <assembly>
 71       <name>PBGAFC_40x40mm</name>
 72       <active>true</active>
 73       <geometry>
111     </assembly>
112     <assembly>
113       <name>WindTunnel</name>
114       <active>true</active>
115       <geometry>
226     </assembly>
227   </geometry>
```

The first of the 2 assemblies, 'PBGAFC_40x40mm', contains the 2R component and 2 monitor points:

```
 70     <assembly>
 71       <name>PBGAFC_40x40mm</name>
 72       <active>true</active>
 73       <geometry>
 74         <twoResistorModel>
 92         <monitorPoint>
101         <monitorPoint>
110       </geometry>
111     </assembly>
```

5.1 A System Level EXCML Example (cont'd)

The 2R model is defined in terms of a location, size, active status and 2 thermal resistance values:

```
74        <twoResistorModel>
75          <name>2R CTM</name>
76          <active>true</active>
77          <location>
78            <x>-0.01590000</x>
79            <y>-0.02000000</y>
80            <z>0.00355000</z>
81          </location>
82          <size>
83            <x>0.04000000</x>
84            <y>0.04000000</y>
85            <z>0.00390000</z>
86          </size>
87          <plane>+xy</plane>
88          <powerDissipation>5.00000000</powerDissipation>
89          <thetaJC>0.40000000</thetaJC>
90          <thetaJB>1.50000000</thetaJB>
91        </twoResistorModel>
```

As with the PCB object, the plane definition, +XY, defines the directional sense of the object so that its 'top' (case) face lies on its high Z face. Conversely the 'board' face is on its low Z face. The 2 thermal metrics that define the 2R component are defined in SI units, i.e. K/W.

5.1 A System Level EXCML Example (cont'd)

The 2 monitor points are defined simply by a name, active status and location:

```
92        <monitorPoint>
93          <name>Tj</name>
94          <active>true</active>
95          <location>
96            <x>0.00410000</x>
97            <y>-0.00000000</y>
98            <z>0.00452500</z>
99          </location>
100       </monitorPoint>
101       <monitorPoint>
102         <name>Tc</name>
103         <active>true</active>
104         <location>
105           <x>0.00410000</x>
106           <y>-0.00000000</y>
107           <z>0.00650000</z>
108         </location>
109       </monitorPoint>
```

The second of the 2 assemblies, 'WindTunnel', contains another sub-assembly and 2 other objects:

```
112     <assembly>
113       <name>WindTunnel</name>
114       <active>true</active>
115       <geometry>
116         <assembly>
190         <rectangular2dFan>
208         <grille>
225       </geometry>
226     </assembly>
```

5.1 A System Level EXCML Example (cont'd)

The sub-assembly, '4sides', holds the 4 walls of the wind tunnel as *solid2dBlock* objects:

```
116        <assembly>
117          <name>4Sides</name>
118          <active>true</active>
119          <geometry>
120            <solid2dBlock>
137            <solid2dBlock>
154            <solid2dBlock>
171            <solid2dBlock>
188          </geometry>
189        </assembly>
```

Just one of the *solid2dBlock* objects is shown for brevity:

```
120            <solid2dBlock>
121              <name>LowXSide</name>
122              <active>true</active>
123              <location>
124                <x>-0.07090000</x>
125                <y>-0.14900000</y>
126                <z>-0.00790000</z>
127              </location>
128              <size>
129                <x>0.00100000</x>
130                <y>0.28000000</y>
131                <z>0.03000000</z>
132              </size>
133              <plane>+yz</plane>
134              <material>Walls</material>
135              <powerDissipation>0.000</powerDissipation>
136            </solid2dBlock>
```

This 'LowXSide' *solid2dBlock* object is defined in the +YZ plane, so its apparent 2D size is taken form the Y and Z size values. The X size value is defined and, although not used in its geometric representation, may be used by the tool to calculate its through plane thermal resistance. In this instance the object does not dissipate any power.

5.1 A System Level EXCML Example (cont'd)

The *rectangular2dFan* object defines the inlet flow to the model:

```
190        <rectangular2dFan>
191          <name>InletFlow</name>
192          <active>true</active>
193          <location>
194            <x>-0.07090001</x>
195            <y>0.13100000</y>
196            <z>-0.00790000</z>
197          </location>
198          <size>
199            <x>0.15000001</x>
200            <y>0.00000000</y>
201            <z>0.03000000</z>
202          </size>
203          <plane>-xz</plane>
204          <flowDefinition>
205            <fixedFlowRate>0.00500000</fixedFlowRate>
206          </flowDefinition>
207        </rectangular2dFan>
```

It is defined in the –XZ plane and has a zero size in the Y direction. Its directional sense is in the negative Y direction, i.e. introducing fluid in the decreasing Y coordiante direction.

5.1 A System Level EXCML Example (cont'd)

The final *grille* object is defined in the same way as the other 2D ECXML objects:

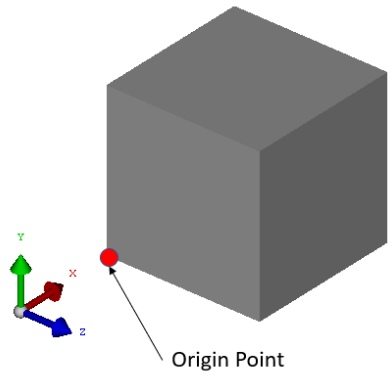
```
208         <grille>
209           <name>Outlet</name>
210           <active>true</active>
211           <location>
212             <x>-0.07090000</x>
213             <y>-0.14900000</y>
214             <z>-0.00790000</z>
215           </location>
216           <size>
217             <x>0.15000000</x>
218             <y>0.00200000</y>
219             <z>0.03000000</z>
220           </size>
221           <plane>+xz</plane>
222           <lossCoefficient>245.00000000</lossCoefficient>
223           <freeAreaRatio>1.00000000</freeAreaRatio>
224         </grille>
```

Its loss coefficient is defined in conjunction with a free area ratio of the grille. Please refer to 4.5.2 for a description of the producing tool specific treatment of the loss coefficient.

5.2 rotationMatrix Examples

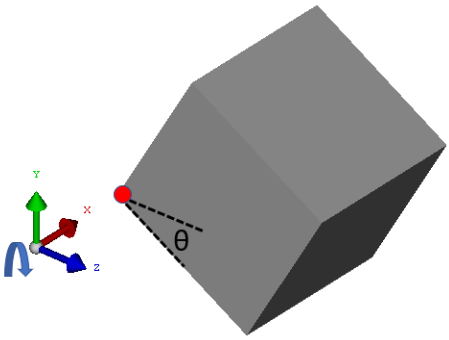
The *rotationMatrix* element is used to transform a referenced MCAD geometry to rotate it around one or more of the coordinate axes. This 3x3 matrix is described in 4.4.16. The following examples demonstrate independent rotations around each of the coordinate axes:

If there is no rotation, the *rotationMatrix* still needs to be defined, having the following elements:



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

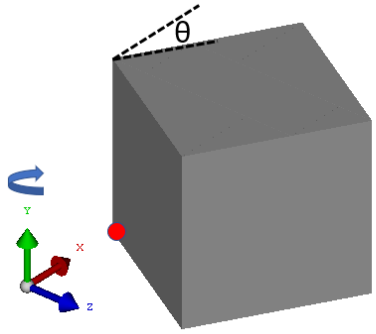
For a rotation around the x-axis:



$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\,\theta \\ 0 & sin\,\theta & cos\,\theta \end{bmatrix}$$

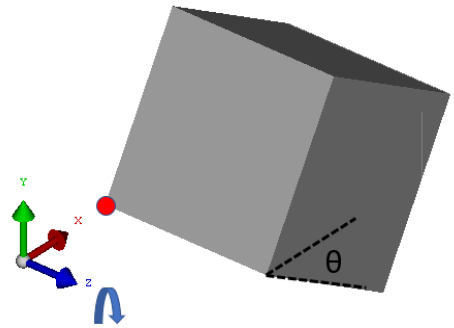
For a rotation around the y-axis:

5.2 rotationMatrix Examples (cont'd)



$$\begin{bmatrix} cos\,\theta & 0 & sin\,\theta \\ 0 & 1 & 0 \\ -sin\,\theta & 0 & cos\,\theta \end{bmatrix}$$

For a rotation around the z-axis:



$$\begin{bmatrix} cos\,\theta & -sin\,\theta & 0 \\ sin\,\theta & cos\,\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

A compound rotation may be determined by a matrix multiplication of the individual rotation matrices in an inverse sequence that the rotations are to be applied.

Annex A (informative) System Level ECXML Example Listing

A full listing of the ECXML example as outlined in 5.1.

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<neutralXML>

 <name>ECXML Example for Guideline</name>

 <producer>Flotherm</producer>

 <solutionDomain>
  <location>
   <x>-0.07090000</x>
   <y>-0.14900000</y>
   <z>-0.00790000</z>
  </location>
  <size>
   <x>0.15000000</x>
   <y>0.28000000</y>
   <z>0.03000000</z>
  </size>
  <ambientConditions>
   <temperature>313.150</temperature>
   <radiantTemperature>313.150</radiantTemperature>
   <pressure>0.000</pressure>
  </ambientConditions>
 </solutionDomain>

 <materials>
  <material>
   <name>Walls</name>
   <density>1200.00000000</density>
   <specific_heat>1500.00000000</specific_heat>
   <surfaceEmissivity>0.95000000</surfaceEmissivity>
   <thermalConductivity>
    <isotropic>
     <conductivity>0.89999998</conductivity>
    </isotropic>
   </thermalConductivity>
  </material>
  <material>
   <name>PCB Material</name>
   <density>1973.00000000</density>
   <specific_heat>655.95794678</specific_heat>
   <surfaceEmissivity>0.90000000</surfaceEmissivity>
   <thermalConductivity>
    <orthotropic>
     <x>38.77000046</x>
     <y>38.77000046</y>
     <z>0.33330452</z>
    </orthotropic>
   </thermalConductivity>
  </material>
 </materials>

 <geometry>
  <printedCircuitBoard>
   <name>PCB</name>
   <active>true</active>
   <location>
    <x>-0.04590000</x>
    <y>-0.05000000</y>
    <z>0.00195000</z>
   </location>
   <size>
    <x>0.10000000</x>
    <y>0.10000000</y>
    <z>0.00160000</z>
   </size>
   <plane>+xy</plane>
   <material>PCB Material</material>
  </printedCircuitBoard>
  <assembly>
   <name>PBGAFC_40x40mm</name>
   <active>true</active>
   <geometry>
    <twoResistorModel>
     <name>2R CTM</name>
     <active>true</active>
     <location>
      <x>-0.01590000</x>
      <y>-0.02000000</y>
      <z>0.00355000</z>
     </location>
     <size>
      <x>0.04000000</x>
      <y>0.04000000</y>
      <z>0.00390000</z>
     </size>
     <plane>+xy</plane>
     <powerDissipation>5.00000000</powerDissipation>
     <thetaJC>0.40000001</thetaJC>
     <thetaJB>1.50000000</thetaJB>
    </twoResistorModel>
    <monitorPoint>
     <name>Tj</name>
     <active>true</active>
     <location>
      <x>0.00410000</x>
      <y>-0.00000000</y>
      <z>0.00452500</z>
     </location>
    </monitorPoint>
    <monitorPoint>
     <name>Tc</name>
     <active>true</active>
     <location>
      <x>0.00410000</x>
      <y>-0.00000000</y>
      <z>0.00650000</z>
     </location>
    </monitorPoint>
   </geometry>
  </assembly>
  <assembly>
```

Annex A (cont'd)

```
<name>WindTunnel</name>
<active>true</active>
<geometry>
 <assembly>
  <name>4Sides</name>
  <active>true</active>
  <geometry>
   <solid2dBlock>
    <name>LowXSide</name>
    <active>true</active>
    <location>
     <x>-0.07090000</x>
     <y>-0.14900000</y>
     <z>-0.00790000</z>
    </location>
    <size>
     <x>0.00100000</x>
     <y>0.28000000</y>
     <z>0.03000000</z>
    </size>
    <plane>+yz</plane>
    <material>Walls</material>
    <powerDissipation>0.000</powerDissipation>
   </solid2dBlock>
   <solid2dBlock>
    <name>HighXSide</name>
    <active>true</active>
    <location>
     <x>0.07910000</x>
     <y>-0.14900000</y>
     <z>-0.00790000</z>
    </location>
    <size>
     <x>0.00100000</x>
     <y>0.28000000</y>
     <z>0.03000000</z>
    </size>
    <plane>+yz</plane>
    <material>Walls</material>
    <powerDissipation>0.000</powerDissipation>
   </solid2dBlock>
   <solid2dBlock>
    <name>LowZSide</name>
    <active>true</active>
    <location>
     <x>-0.07090000</x>
     <y>-0.14900000</y>
     <z>-0.00790000</z>
    </location>
    <size>
     <x>0.15000000</x>
     <y>0.28000000</y>
     <z>0.00100000</z>
    </size>
    <plane>+xy</plane>
    <material>Walls</material>
    <powerDissipation>0.000</powerDissipation>
   </solid2dBlock>
   <solid2dBlock>
    <name>HighZSide</name>
    <active>true</active>
    <location>
     <x>-0.07090000</x>
     <y>-0.14900000</y>
     <z>0.02210000</z>
    </location>
    <size>
     <x>0.15000000</x>
     <y>0.28000000</y>
     <z>0.00100000</z>
    </size>
    <plane>+xy</plane>
    <material>Walls</material>
    <powerDissipation>0.000</powerDissipation>
   </solid2dBlock>
  </geometry>
 </assembly>
 <rectangular2dFan>
  <name>InletFlow</name>
  <active>true</active>
  <location>
   <x>-0.07090001</x>
   <y>0.13100000</y>
   <z>-0.00790000</z>
  </location>
  <size>
   <x>0.15000001</x>
   <y>0.00000000</y>
   <z>0.03000000</z>
  </size>
  <plane>-xz</plane>
  <flowDefinition>
   <fixedFlowRate>0.00500000</fixedFlowRate>
  </flowDefinition>
 </rectangular2dFan>
 <grille>
  <name>Outlet</name>
  <active>true</active>
  <location>
   <x>-0.07090000</x>
   <y>-0.14900000</y>
   <z>-0.00790000</z>
  </location>
  <size>
   <x>0.15000000</x>
   <y>0.00200000</y>
   <z>0.03000000</z>
  </size>
  <plane>+xz</plane>
  <lossCoefficient>245.00000000</lossCoefficient>
  <freeAreaRatio>1.00000000</freeAreaRatio>
 </grille>
</geometry>
</assembly>
</geometry>

</neutralXML>
```

Annex B (informative) Differences between JEP181A and its Predecessor

This annex briefly describes most of the changes made to entries that appear in this publication. Some editorial changes that simply involves any words added or deleted or punctuations may be not included.

B.1 Differences between JEP181A compared to JEP181 (September 2020)

Clause	Description of change
4.3.4	Material: Extended to accommodate non-linear temperature dependent thermal conductivity definition.
4.4.16	ECXML Geometry Elements: New section added for external referencing of MCAD files.
4.4.16.1	Explanation of translation and rotation precedence for externally referenced MCAD files
5.2	Rotation matrix examples for externally referenced MCAD files.
Annex	Added Annex B; Differences between JEP181A and its predecessors

This page intentionally left blank.

Standard Improvement Form JEDEC JEP181A

The purpose of this form is to provide the Technical Committees of JEDEC with input from the industry regarding usage of the subject standard. Individuals or companies are invited to submit comments to JEDEC. All comments will be collected and dispersed to the appropriate committee(s).

If you can provide input, please complete this form and return to:

JEDEC
Attn: Publications Department
2500 Wilson Blvd. Suite 220
Arlington, VA 22201-3834
Fax: 703.907.7583

1. I recommend changes to the following:

☐ Requirement, clause number _____

☐ Test method number _____ Clause number _____

The referenced clause number has proven to be:

☐ Unclear ☐ Too Rigid ☐ In Error

☐ Other _____

2. Recommendations for correction:

_____

_____

_____

_____

3. Other suggestions for document improvement:

_____

_____

_____

_____

Submitted by

Name: _____ Phone: _____

Company: _____ E-mail: _____

Address: _____

City/State/Zip: _____ Date: _____


JEDEC®