NOTE: Use these buttons to print or save the form. DO NOT use the browser tool bar. SAVE PRINT

LA-UR-11-04368

Approved for public release; distribution is unlimited.

Title:	User's Guide For Exodus-II File To Abaqus '.inp' File Translator Version 1.0 (U)
Author(s):	Kevin Marshall
Intended for:	General release with MCNP6

Los Alamos NATIONAL LABORATORY EST.1943

Form 836 (7/06) SAVE PRINT CLEAR FORM

LA-UR-11-04368

User's Guide For The Exodus-II File To Abaqus '.inp' File Translator Version 1.0

Kevin Marshall 7/18/2011

Abstract

This document is a guide to using the exo_2_Abaqus_v1.0.x translator utility program for converting exodus files, which here will be assumed to have been generated using the CUBIT meshing tool, into Abaqus '.inp'-file-format text files for subsequent use in MCNP6.

The development of this tool was carried out using CUBIT version 11.1 and Abaqus version 10-EF1.

This document will describe the process for creating and exporting an ExodusII geometry model using CUBIT, such that it is then ready for conversion using the exo_2_Abaqus_v1.0.x translator into an Abaqus '.inp' file. An exhaustive description of all of the geometry and meshing features of the CUBIT meshing program will not be given here. Only the processes and operations required in order to generate a convertible Exodus geometry file will be described.

Although this report focuses on Exodus file creation using the CUBIT meshing application, the general principles of geometry creation and exporting are expected to be consistent across most applications that allow meshing of a CAD-type solid-model geometry, followed by its export as an Exodus file format.

At this point a note should be made about the terminology that will be used throughout this report. In a number of places references are made to carrying out CUBIT operations, which include specifying the widgets or menu options that should be used. When this is done the particular CUBIT tool or command will be indicated using Courier font. For example, `Create Brick.` In other places references will be made to CUBIT, Exodus or Abaqus – specific terminology. There are a number of occasions when these three formats use synonyms for similar concepts, and it can be important to understand the importance and read-across for these terms.. In this report format specific terms with a very defined meaning will appear as italic, e.g. *Block.*

Creating the exodus geometry - Overview

For typical 3-D CAD-like geometries that will form the basis of geometries to be used in MCNP6, CUBIT uses the concept of *volume bodies* as the basic constituent of any model. These volume bodies can then be merged, cut, etc. to form useful geometries that are representative of the problem in hand. Creation of these bodies is achieved using the volume creation commands such as `Brick`, `Sphere` and `Cone` for simple bodies. Or by using swept or extruded surfaces for more complicated bodies. Any two *volume bodies* that are merged will become one single new *volume body*, with the option of retaining the two original *volume bodies*. Similarly, if one *volume body* is cut from another *volume body* then a single *volume body* is created, with the other two being retained if so required.

Once the 3-D model is created each body can then be meshed. Meshing a *volume body* typically takes the form of assigning a mesh-type to a the *volume body*, then assigning the mesh size to the *volume body* and finally executing the mesh generation. A *volume body* must be meshed before it can be exported via the Exodus format.

In order to export to an Exodus file-format, the *volume body* within the CUBIT model must be assigned to *Blocks*. The concept of a *Block* is central to the Exodus mesh-geometry format and it is by creating these *Blocks* that meshes may be exported from CUBIT in Exodus format. These *Blocks* may comprise multiple volume *bodies*, which allows for effective partitioning of a model within the CUBIT model. *Blocks* within the Exodus format are essentially the equivalent of *Parts* within the Abaqus format. Exodus *Blocks* are made up of one or more individual mesh elements that are uniquely assigned to that *Block*, which is why the CUBIT volume bodies must be meshed before the model can be exported to an Exodus file.

Once the volumes have been created, meshed and assigned to *Blocks*, then the geometry can then be exported to an Exodus file. That Exodus file should then be ready for translation into an Abaqus '.inp' file.

The following sections will describe the steps for creating a simple model and exporting it to an Exodus file. Only the aspects relevant to creating the overall model for subsequent translation, and ultimately for use in MCNP6, will be emphasized. Much of the basic knowledge for geometry creation and advanced meshing will be assumed.

Creating The Solid Model

<u>Step 1: Create the geometry:</u>

Figure 1(a) shows a simple abstract model that is comprised of 7 simple *volume bodies* that have been created using the `Create Brick` and `Create Cylinder` commands. These have then been translated into the required positions, and any overlaps between bodies have been removed using the Boolean `Cut` command. *Volume bodies* may also be created using any of the options from within the `Entity-Volume Action-Create` tools, as shown in Figure 1(b). The thin plate to which the cylinders are attached was initially created as one single plate before being split into three using the `CUT Plane` command. This splitting of the plate allows the potential for higher resolution meshing to be implemented around the region where the cylinders intersect.

It is a requirement of the Exodus format that all components of the model that will be required as components (parts) of the radiation transport calculation **must** be created as *volume bodies* from within the `Entity-Volume Action-Create` tools.

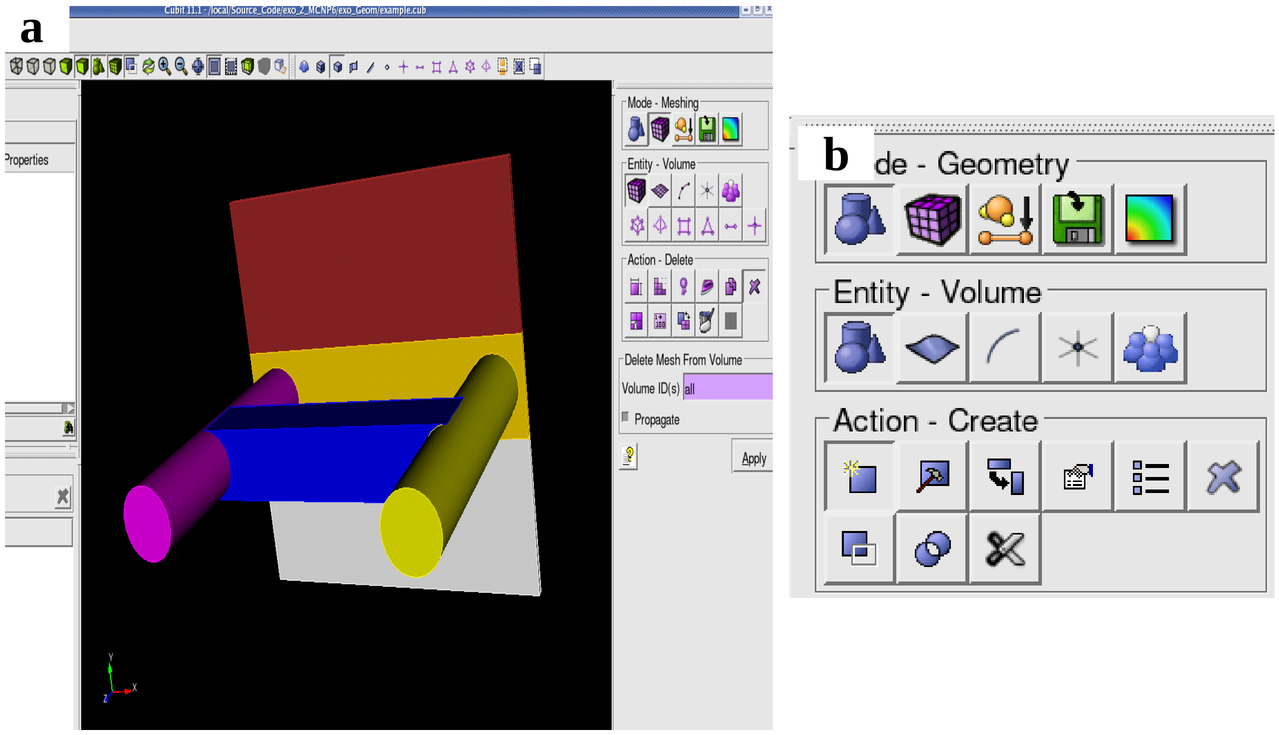
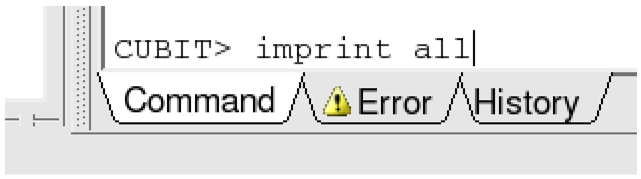


Figure 1. (a) A simple abstract geometry created using CUBIT. The overall model is made up of 6 *volume bodies*. (b) The *volume body* creation tools

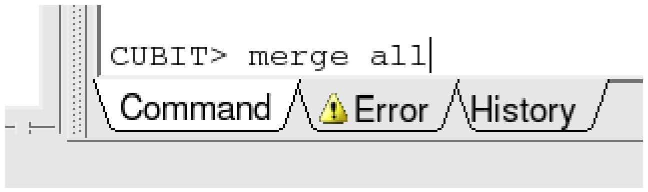
Step 2: Imprint and Merge:

CUBIT geometries are what are know as 'manifold' geometries; this means that adjacent *volume bodies* do not share surfaces. Meshing requires that *volume bodies* bodies which are adjacent and touching share a single surface. In order to rectify this discrepancy CUBIT allows the user to execute the `imprint all` command and the `merge all` command, Figure 2. The `imprint all` command essentially projects topological differences from one *volume body* onto any that are adjacent to it. The `merge` command essentially merges two coincident surfaces into one single surface, onto which a mesh can be generated.

To execute these commands simply type and enter the following two commands separately into the command line; `imprint All` followed by `merge All`



```
CUBIT> imprint all
```



```
CUBIT> merge all
```

Figure 2 The 'imprint all' and 'merge all' commands should be executed after creating the solid geometry but prior to meshing.

Step 3: Mesh The Volume Bodies:

The *volume bodies* should then be meshed, either all together or separately, using the standard CUBIT meshing tools, Figure 3 . Any type of meshing is allowed. However, for any set of *volume bodies* that are intended to ultimately be exported as a single Exodus *Block*, *all bodies must have the same element type*, e.g. tet, map, etc. The size of the mesh in any *Block* may be set as desired, as long as the CUBIT meshing tool is able to mesh it.

In this example all of the *volume bodies* have been set to be of the same type: tetrahedral.

It can be seen in Figure 4 that the mesh size of the middle *volume bodies* of the three that make up the thin back-plate is of a higher resolution of the than that of its other neighbors. When the Exodus *Blocks* are created for export all three of these plate *volume bodies* will be assigned to a single *Block*. This will ultimately be translated in to a single Abaqus part within the '.inp' file that is created by the translator. This will mean that the part will contain regions of varying mesh resolution in the same way as can be achieved using the partitioning functionality provided by Abaqus.

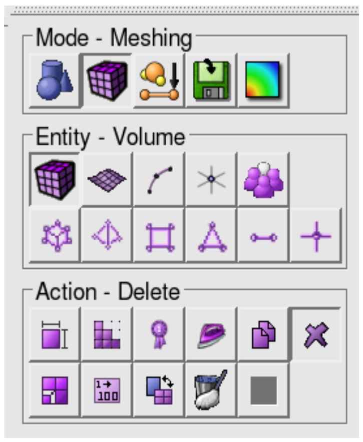
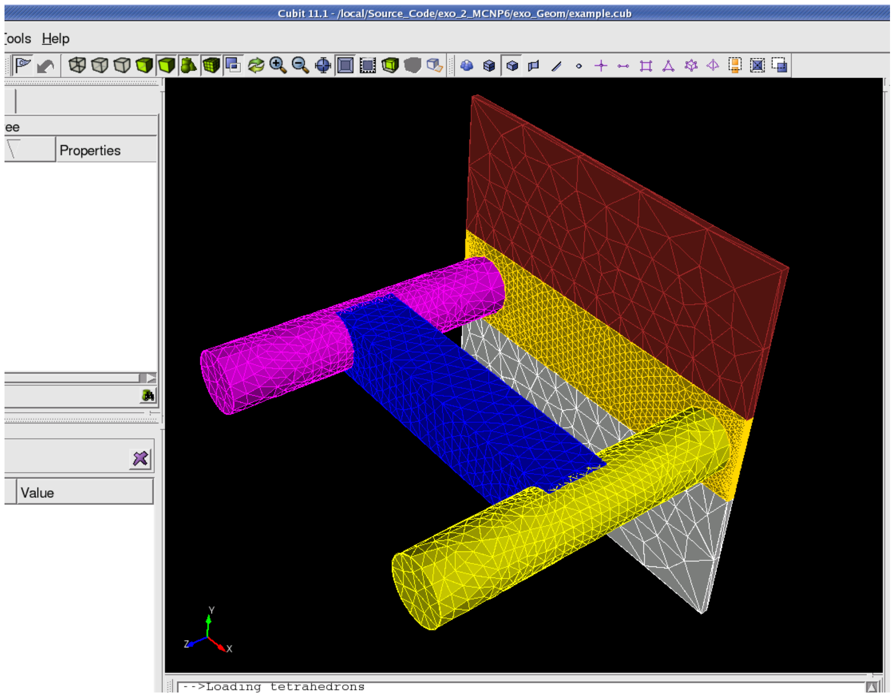


Figure 3 The CUBIT meshing tool.


Cubit 11.1 - /local/Source_Code/exo_2_MCNP6/exo_Geom/example.cub
Tools
Help
Properties
Value
Y
Z
X

```
-->Loading tetrahedrons
```

Figure 4 The meshed abstract geometry. The splitting of the thin plate into three separate *volume bodies* allows a finer mesh to be generated in the region around the intersection with the two cylinders.

Step 4: Assign The Blocks:

Once the volume bodies have been created, *imprinted* and *merged*, and meshed, then the next step is to assign them to *Blocks*. There are a number of points to understand and consider when doing this:

1) Multiple volume bodies may be assigned to a single *Block*.

2) Any given body may only be assigned to one *Block*.

3) Each *Block* that is created will generate a single Abaqus *Part*.

4) Each *Block* that is created will generate an Abaqus statistical element set for that entire part.

5) Each *Block* that is created will generate an Abaqus material set for that entire part (assigning material numbers is describe in step 5).

6) *Blocks* do not need to be physically continuous. i.e. they may be made of spatially separated *volume bodies*.

A good general principle is that *Blocks* should correspond to the conceptual parts of the model, or to regions of a conceptual part that need to be analyzed separately. In particular if there is a region of a part where a *separate statistical set* is required.

Points 4 & 5 outline that within this method of creating geometries for MCNP6, each Abaqus Part will have one and only one statistical set and only one material set. However, multiple Abaqus parts may contain the same numbered *material set*. Thus allowing multiple Abaqus parts to be assigned the same material within the MCNP6 input deck via their pseudo cells.

Blocks are assigned using the `Materials and BCs` commands, Figure 5a. To do this select `Entity- Exodus Blocks` from the `Materials and BCs` options, Figure 5b. Then select `Add` from the drop-down menu and check the `Volume` option, Figure 5b.

Here the `Block ID` field represents the number of each Exodus *Block* as it will be once exported. This number will define its *Part* number in the subsequent Abaqus '.inp' file. These numbers are user-defined, and can be any integer. However, it is usually best to number each *Block* incrementally, starting at one.

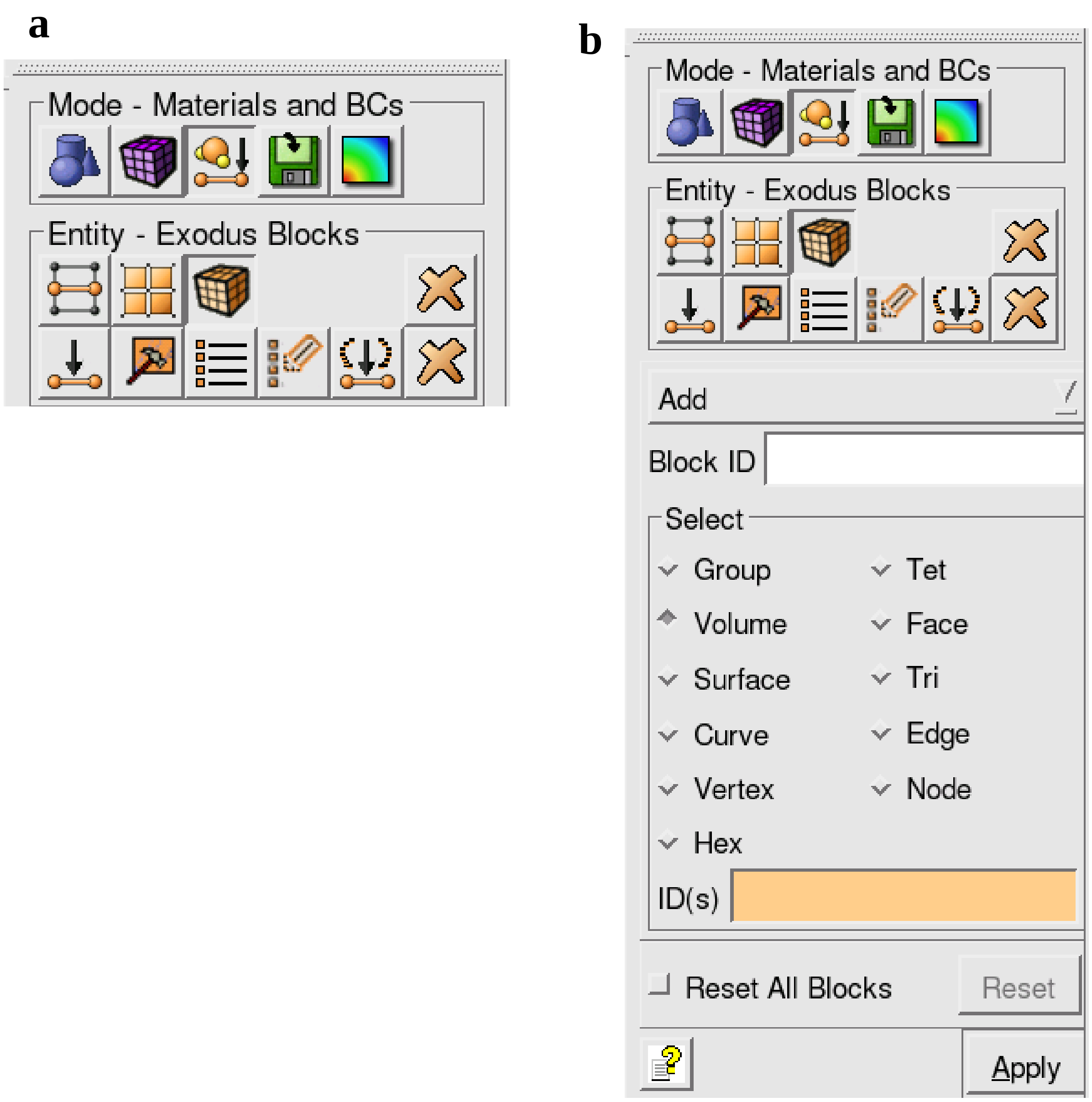
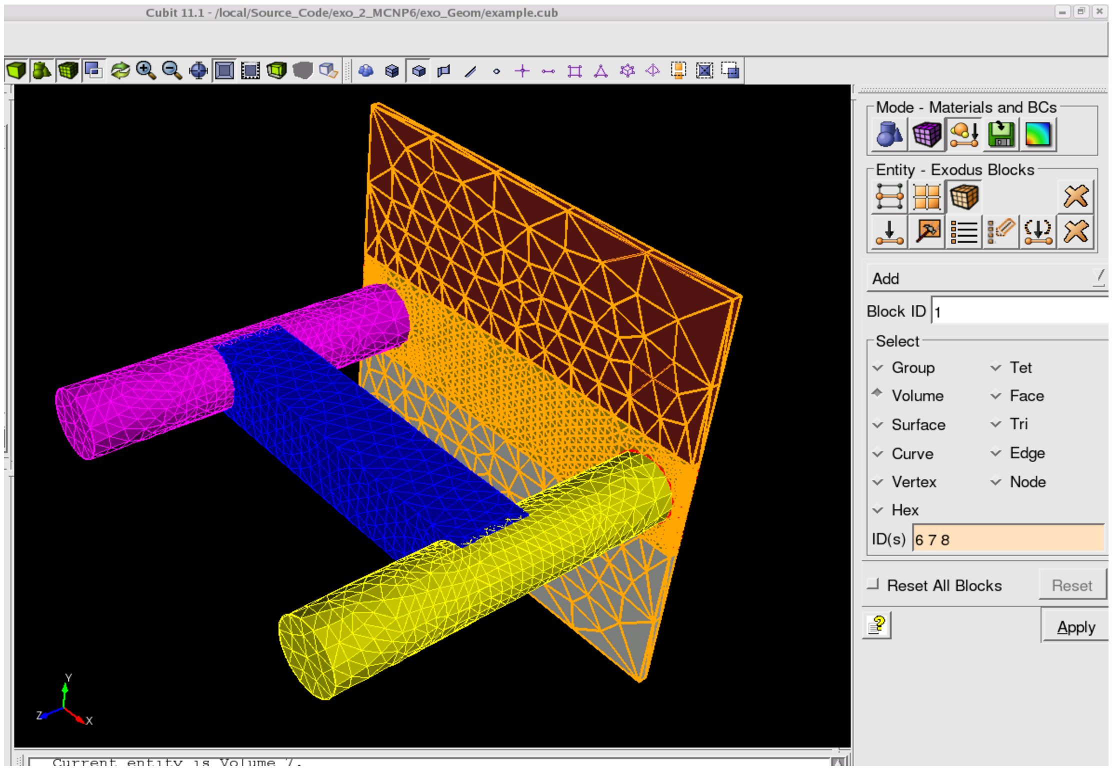


Figure 5 (a) The CUBIT Materials and Boundary Conditions tools. (b) The `Entity Exodus Blocks` option is used for creating Exodus Blocks. The `Block ID` field is the number of the Exodus *Block* that is to be created. The `ID(s)` field allows allows the `number Ids` of the *volume bodies* that will be assigned to that *Block* to be specified. The ID numbers of the *volume bodies* that are to be assigned to a given *Block* should be entered 'ID(s)' field.

So, to create a *Block*, enter its number into the 'Block ID' field, and enter the CUBIT *volume body* numbers of the bodies that will comprise the *Block* into the *ID(s)* field and select `Apply`.

Figure 6 shows this for the back-plate, which is made up of three volume bodies, 6, 7 & 8, that will eventually be *Block 1* in the Exodus file and *Part-1* in the Abaqus '.inp' file. Note, that as *volume bodies* can only be uniquely assigned to one *Block*, *volume bodies* 6, 7 & 8 cannot now be assigned to any other *Block*.

As the *Blocks* are created they will appear in the model tree under the *Boundary Condition* branch, Figure 7. Clicking on a *Block* and selecting the properties tab will allow the basic information about the *Block* to be viewed.


Cubit 11.1 - /local/Source_Code/exo_2_MCNP6/exo_Geom/example.cub

Mode - Materials and BCs

Entity - Exodus Blocks

Add

Block ID 1

Select

Group, Volume [✓], Surface, Curve, Vertex, Hex, Tet, Face, Tri, Edge, Node

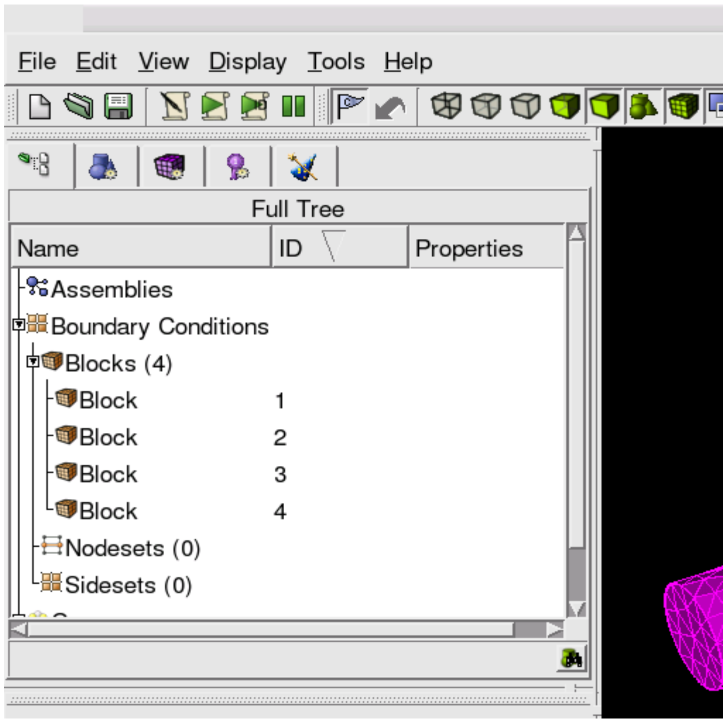
ID(s) 6 7 8

Reset All Blocks

Current entity is Volume 7.

Figure 6 Assigning volume bodies to an Exodus *Block*. Here volume bodies 6 7 and 8 are to be assigned to *Block 1*. As an Exodus *Block* will become an Abaqus *Part*, the assignment of *volume bodies* shown here means that the *Part* created from *Block 1* will

have a mesh with varying resolution.

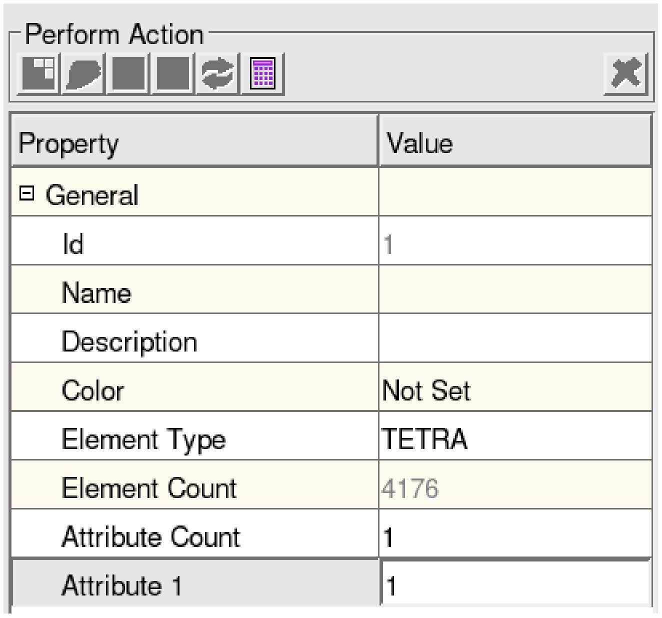


Name	ID	Properties
Assemblies		
Boundary Conditions		
Blocks (4)		
Block	1	
Block	2	
Block	3	
Block	4	
Nodesets (0)		
Sidesets (0)		

Figure 7 Once all of the required *Blocks* have been assigned, they will be shown in the `Boundary Conditions` branch of the CUBIT model tree.

Step 5: Assign The Material Numbers: MCNP6 requires that material sets be assigned to all regions of the geometry, and that materials should be created within the Abaqus model. As stated in point 5 above, a *Block* within CUBIT, which will become an Abaqus *Part*, will have a single material set encompassing all of the elements within that *Part*. The number of that *material set*, which should match the material number of the pseudo-cell corresponding to that *Part*, must therefore be assigned for each *Block* that is created within the CUBIT model. This is done using the `Attribute` feature of Blocks within CUBIT.

To assign a *material set* number to a *Block* select the desired block from the `Boundary Conditions` section of the model tree, and then select the `Properties` tab. In the table below select the value of the `Attribute Count` property and set the value to one. Another property, `Attribute 1`, should appear in a new row below. This process for Block 1 is shown in Figure 8. Set the value of `Attribute 1` to be the material number for that Block (part/pseudo cell). Repeat this procedure for each of the blocks, setting the attribute count to 1, and then setting the value of their 'Attribute 1' property to the material number for that block. Any number of *Blocks* may share the same material number. All *Blocks* must be assigned a material number. Material numbers must be integer and positive greater than 0.



Property	Value
⊟ General	
Id	1
Name	
Description	
Color	Not Set
Element Type	TETRA
Element Count	4176
Attribute Count	1
Attribute 1	1

Figure 8 Each *Block* should be assigned at least an `Attribute 1`, which has an integer values corresponding to the material for the pseudo-cell that will hold the Abaqus *Part* that is created from the *Block*.

Step 6: Export as an Exodus file: Once all of the previous step have been completed the model is now ready to be exported to an Exodus file. This is done using the `Operation – Export Mesh` command within the `Analysis Setup` options, Figure 9.

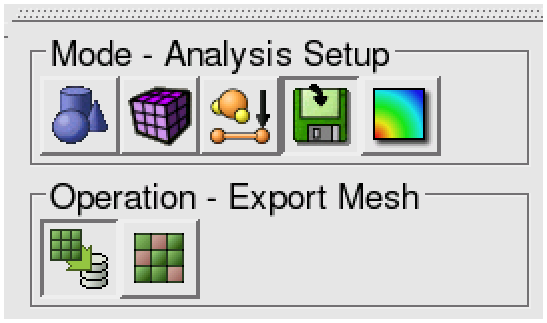


Figure 9 The Export Mesh tools in CUBIT.

In the drop-down menu select 'Genesis' and browse to a suitable directory and chose a file name, as shown in Figure 10.

In the `Block ID(s)` field enter the newly created *Block* numbers that are to be exported (order is unimportant). If all of the *Blocks* are to be exported then select enter '`all`'.
Leave all of the other options checked as default. Overwrite can be checked if required. Once this is done select '`Apply`' and the model will be exported in Exodus format.

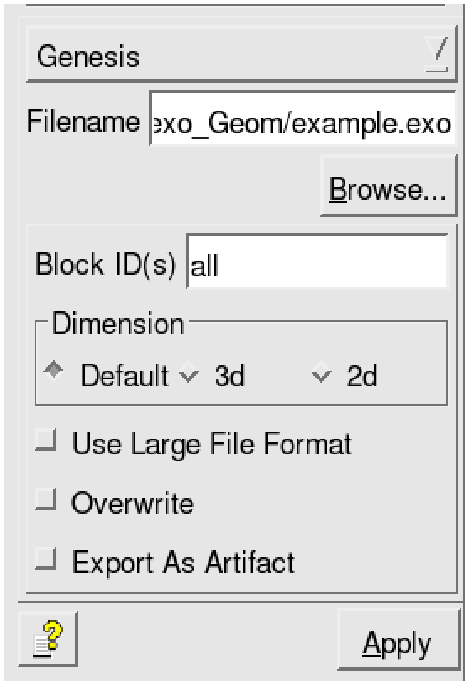

Genesis
Filename: exo_Geom/example.exo
Block ID(s): all
Dimension: [✓ Default] 3d 2d
Use Large File Format
Overwrite
Export As Artifact

Figure 10. Within the Export mesh tools the `Genesis` option should be selected and the required *Blocks* for Export listed in the `Block ID(s)` field. If all *Blocks* are required then the text '`all`' can be entered.

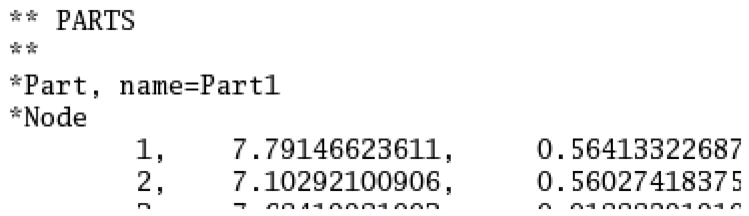
Converting The Exodus Model To Abaqus '.inp' format

To convert the Exodus file into an Abaqus '.inp' model simply execute the exo_2_Abaqus utility with the name of the exodus file as the single command line argument:

```
./exo_2abaqus example.exo
```

The resulting Abaqus .inp file will contain parts, consisting of nodes, element definitions, and material and element sets. One of these *Parts* will be created for every *Block* of the Exodus model the was exported.

For this version of the translator is not possible to label *Parts* with a meaningful name. As such the *Parts* will be identified by the number order in which their associated *Blocks* were created in the CUBIT (or other application) model, Figure 11.



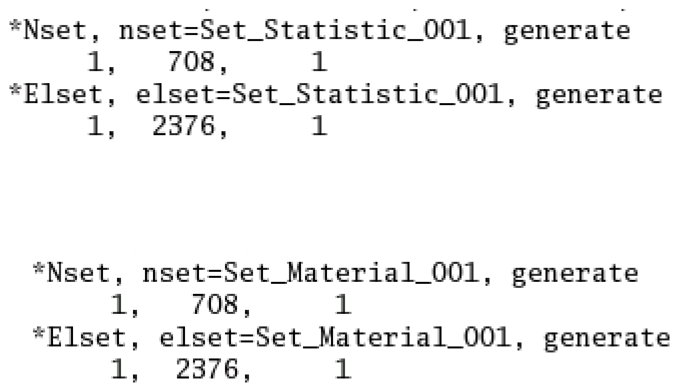
```
** PARTS
**
*Part, name=Part1
*Node
      1,    7.79146623611,     0.56413322687
      2,    7.10292100906,     0.56027418375
```

Figure 11. Each Exodus Block that is exported will result in an Abaqus part.

As with direct export from the Abaqus application itself, the nodes and element numbers are local to each *Part*.

Each part will also contain a single *element statistic set* and a single *statistic node set*, which will encompass the entire *Part*. Both of these will be assigned the number '001'. See Figure 12a.

Similarly, each part will contain a single *material element set* and *nodal material set.* The number of these set will correspond to the value given to 'Attribute 1' of the associated Exodus *Block*, as described in step 5 of the previous section. See Figure 12b



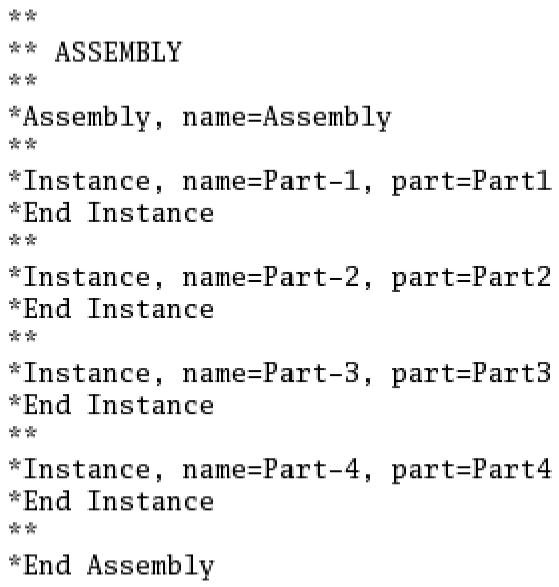
```
*Nset, nset=Set_Statistic_001, generate
     1,   708,     1
*Elset, elset=Set_Statistic_001, generate
     1,  2376,     1
```

```
*Nset, nset=Set_Material_001, generate
     1,   708,     1
*Elset, elset=Set_Material_001, generate
     1,  2376,     1
```

Figure 12. Each Abaqus *Part* that is generated from an Exodus *Block* will come with an

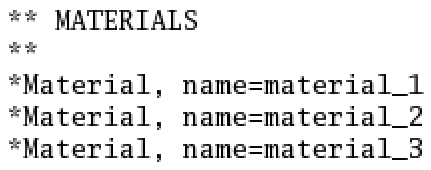
associated *statistic and material set*. Both of which contain all elements within the *Part*. Following the definitions of all of the *Parts* is the definition of the assembly. This simply consists of a list of all of the *Parts*, as per a standard Abaqus '.inp' file. Figure 13.

Finally, the file contains a list of all of the materials that were defined for the *Blocks* as described in step 5 of the previous section. As there is no way to export a named material attribute with the exodus file, materials are named according to their number, i.e. each material will have the name `material_n,` where n is the number of the material.



```
**
** ASSEMBLY
**
*Assembly, name=Assembly
**
*Instance, name=Part-1, part=Part1
*End Instance
**
*Instance, name=Part-2, part=Part2
*End Instance
**
*Instance, name=Part-3, part=Part3
*End Instance
**
*Instance, name=Part-4, part=Part4
*End Instance
**
*End Assembly
```

Figure 13 Each Exodus *Block* will generate an instance of a *Part* within the *Assembly* description towards the end of the '.inp' file.



```
** MATERIALS
**
*Material, name=material_1
*Material, name=material_2
*Material, name=material_3
```

Figure 14 a list of materials based on the numbers assigned to the 'Attribute 1' attributes of the *Blocks* within the Exodus file will be generated.

As the Exodus file contains the node positions of all elements as they were after any translational, rotational or scaling operations were carried out on their parent body, there is no need for the Abaqus '.inp' file that is created to contain a translations/rotations section.