

# 11 EnSight Data Formats

This section describes the format for all readable and writable files in EnSight which you may need access to. The formats described are only for those files that are specific to EnSight. We do not describe data formats not developed by CEI (for example, data formats for various analysis codes). For information about these formats, consult the applicable creator.

*Note: If you are using this documentation to produce your own data translator, please make sure that you follow the instructions exactly as specified. In many cases, EnSight reads data in blocks to improve performance. If the format is not followed, the calculations of how much to read for a block will be thrown off. EnSight does little in the way of error checking data files when they are read. In this respect, EnSight sacrifices robustness for performance.*

## EnSight Formats

EnSight has three evolutionary file formats listed below from oldest to most recent:

### EnSight 5

- legacy format
- supported unstructured meshes only
- used a global nodal array
- used per node variables only

### EnSight 6

- support for case file
- support for both unstructured and structured meshes
- uses a global nodal array
- use per node or per element variables

### EnSight Case Gold (recommended format)

- is much faster than EnSight 6 and is more memory efficient (noticeable if you have a large number of parts or for larger models)
- uses connectivity which can be separate from the node ids
- uses a part basis rather than a global array

### Format illustration

EnSight Case Gold	EnSight 6
<b>part 1</b> node coordinates element connectivity by local node index	global nodal ids & coordinates
<b>part 2</b> node coordinates element connectivity by local node index	<b>part 1</b> element connectivity by global node ids
...	<b>part 2</b> element connectivity by global node ids
<b>part n</b> node coordinates element connectivity by local node index	...
	<b>part n</b> element connectivity by global node ids

### *Saving Gold from EnSight*

EnSight can export your model into Case Gold format (either ASCII or Binary). Activate the variables of interest, select the parts in the main part window and then go to the File menu: File->Save->Geometric Entities.

### *Tool to Check EnSight Format*

There is another advantage to using Case Gold format in that there is a debugging tool called `ens_checker` that can help you find mistakes in files as you write a translator or exporter. Just type `ens_checker file.case` and the code will echo it's progress and the problems it finds to the console. This tool will also check EnSight 6 format.

### *Detailed Description of Formats*

**Section 11.1, EnSight Gold Casefile Format** describes in detail the EnSight Gold case, geometry, and variable file formats. *This format provides the best performance for getting data into EnSight. It is readable by both the EnSight and EnSight Gold programs.*

**Section 11.2, EnSight6 Casefile Format** describes in detail the EnSight6 case, geometry, and variable file formats. *This format is still supported, but is a legacy format. It has most, but not all, of the options of the gold format - and is somewhat lower in performance.*

**Section 11.3, EnSight5 Format** describes in detail the EnSight5 geometry and variable file formats. *This format is still supported, but is a legacy format. It is for unstructured data only.*

**Section 11.4, FAST UNSTRUCTURED Results File Format** describes the “executive” .res file that can be used with FAST unstructured solution and function files.

**Section 11.5, FLUENT UNIVERSAL Results File Format** describes the “executive” .res file that can be used with FLUENT Universal files for transient models.

**Section 11.6, Movie.BYU Results File Format** describes the “executive” .res file that can be used with Movie.BYU files.

**Section 11.7, PLOT3D Results File Format** describes the “executive” .res file that can be used with PLOT3D solution and function files.

**Section 11.8, Server-of-Server Casefile Format** describes the format of the casefile used with the server-of-server capability of EnSight.

**Section 11.9, Periodic Matchfile Format** describes the format of the file which can be used to explicitly specify which nodes match from one periodic instance to the next.

**Section 11.10, XY Plot Data Format** describes the format of the file containing XY plot data.

**Section 11.11, EnSight Boundary File Format** describes the format of the file which can define unstructured boundaries of structured data.

**Section 11.12, EnSight Particle Emitter File Format** describes the format of the optional file containing particle trace emitter time and location information.

# 11.1 EnSight Gold Casefile Format

Include in this section:

**EnSight Gold General Description**

**EnSight Gold Geometry File Format**

**EnSight Gold Case File Format**

**EnSight Gold Wild Card Name Specification**

**EnSight Gold Variable File Format**

**EnSight Gold Per\_Node Variable File Format**

**EnSight Gold Per\_Element Variable File Format**

**EnSight Gold Undefined Variable Values Format**

**EnSight Gold Partial Variable Values Format**

**EnSight Gold Measured/Particle File Format**

**EnSight Gold Material Files Format**

## *EnSight Gold General Description*

EnSight Gold data consists of the following files:

- Case (required) (points to all other needed files including model geometry, variables, and possibly measured geometry and variables)

EnSight makes no assumptions regarding the physical significance of the scalar, vector, 2nd order symmetric tensor, and complex variables. These files can be from any discipline. For example, the scalar file can include such things as pressure, temperature, and stress. The vector file can be velocity, displacement, or any other vector data, etc.

In addition, EnSight Gold format handles "undefined" as well as "partial" variable values. (See appropriate subsections later in this chapter for details.)

All variable results for EnSight Gold format are contained in disk files—one variable per file. Additionally, if there are multiple time steps, there must either be a set of disk files for each time step (transient multiple-file format), or all time steps of a particular variable or geometry in one disk file each (transient single-file format).

Sources of EnSight Gold format data include the following:

- Data that can be translated to conform to the EnSight Gold data format (including being written from EnSight itself using the Save Geometric Entities option under File->Save)
- Data that originates from one of the translators supplied with the EnSight application

The EnSight Gold format supports an unstructured defined element set as shown in the figure on a following page. Unstructured data must be defined in this element set. Elements that do not conform to this set must either be subdivided or

discarded.

The EnSight Gold format also supports the same structured block data format as EnSight6, which is very similar to the PLOT3D format. *Note that for this format, the standard order of nodes is such that I's advance quickest, followed by J's, and then K's.*

A given EnSight Gold model may have either unstructured data, structured data, or a mixture of both.

This format is somewhat similar to the EnSight6 format, but differs enough to allow for more efficient reading of the data. It is intended for **3D, binary, big** data models.

*Note: While an ASCII format is available, it is not intended for use with large models and is in fact subject to limitations such as integer lengths of 10 digits. Use the binary format if your model will exceed 10 digits for node or element numbers or labels.*

Starting with version 7, EnSight writes out all model and variable files in EnSight Gold format. Thus, it can be read by all version 7 or 8 EnSight licenses (i.e. standard, gold, and custom licenses).

#### [ens\\_checker](#)

A program is supplied with EnSight which attempts to verify the integrity of the *format* of EnSight 6 and EnSight Gold files. If you are producing EnSight formatted data, this program can be very helpful, especially in your development stage, in making sure that you are adhering to the published format. It makes no attempt to verify the validity of floating point values, such as coordinates, variable values, etc. This program takes a casefile as input. Thus, it will check the format of the casefile, and all associated geometry and variable files referenced in the casefile. See [How To Use ens\\_checker](#).

## Supported EnSight Gold Elements

The elements that are supported by the EnSight Gold format are:

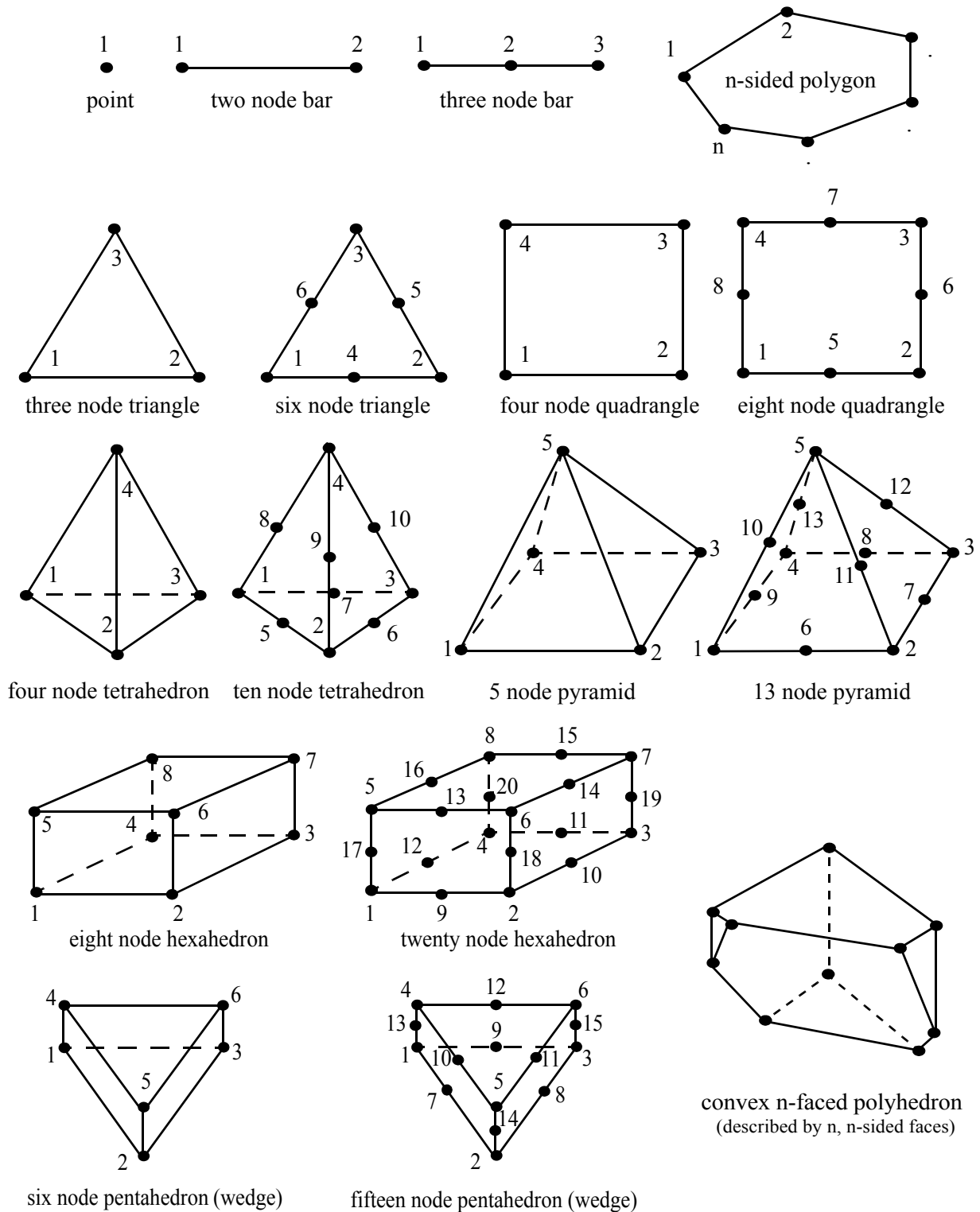


Figure 11-1  
Supported EnSight Gold Elements

## EnSight Gold Geometry File Format

The EnSight Gold format is part based for both unstructured and structured data. There is no global coordinate array that each part references, but instead - each part contains its own local coordinate array. Thus, the node numbers in element connectivities refer to the coordinate array index, not a node id or label. *This is different than the EnSight6 format!*

The EnSight Gold format consists of keywords followed by information. The following items are important when working with EnSight Gold geometry files:

1. Node ids are optional. In this format they are strictly labels and are not used in the connectivity definition. The element connectivities are based on the local implied node number of the coordinate array in each part, which is sequential starting at one. If you let EnSight assign node IDs, this implied internal numbering is used. If node IDs are set to off, they are numbered internally, however, you will not be able to display or query on them. If you have node IDs given in your data, you can have EnSight ignore them by specifying "node id ignore." Using this option may reduce some of the memory taken up by the Client and Server, but display and query on the nodes will not be available. Note, prior to EnSight 7.4, node ids could only be specified for unstructured parts. This restriction has been removed and user specified node ids are now possible for structured parts.
2. Element ids are optional. If you specify element IDs, or you let EnSight assign them, you can show them on the screen. If they are set to off, you will not be able to show or query on them. If you have element IDs given in your data you can have EnSight ignore them by specifying "element id ignore." Using this option will reduce some of the memory taken up by the Client and Server. This may or may not be a significant amount, and remember that display and query on the elements will not be available. Note, prior to EnSight 7.4, element ids could only be specified for unstructured parts. This restriction has been removed and user specified element ids are now possible for structured parts.
3. Model extents can be defined in the file so EnSight will not have to determine these while reading in data. If they are not included, EnSight will compute them, but will not actually do so until a dataset query is performed the first time.
4. The format of integers and real numbers **must be followed** (See the Geometry Example below).
5. ASCII Integers are written out using the following integer format:

From C: 10d format

From FORTRAN: i10 format

*Note: this size of integer format places a limitation on the number of nodes and the node and element labels that can make up a model. Use the binary format for large models!*

ASCII Real numbers are written out using the following floating-point format:

From C: 12.5e format

From FORTRAN: e12.5 format

**The number of integers or reals per line must also be followed!**

6. By default, a Part is processed to show the outside boundaries. This representation is loaded to the Client host system when the geometry file is read (unless other attributes have been set on the workstation, such as feature angle).
7. Coordinates for unstructured data must be defined within each part. This is normally done before any elements are defined within a part, but does not have to be. The different elements can be defined in any order (that is, you can define a hexa8 before a bar2).
8. A Part containing structured data cannot contain any unstructured element types or more than one block. **Each structured Part is limited to a single block (or some subset of that block).** A structured block is indicated by following the Part description line with a 'block' line. By default, a block will be curvilinear, non-iblankeed, non-ghost, complete range. However, by supplying one or more of the following options on the 'block' line, rectilinear or uniform blocks can be specified, nodal iblanking for the block can be used, cells within the block can be flagged as ghosts (used for computations, but not displayed), subset ranges can be specified (useful for partitioned data). The options include:

Only one of these can be used on the ‘block’ line		
curvilinear	Indicates that coordinates of all ijk locations of the block will be specified (default)	
rectilinear	Indicates that i,j,k delta vectors for a regular block with possible non-regular spacing will be specified	
uniform	Indicates that i,j,k delta values for a regular block with regular spacing will be specified	
Any, none, or all of these can be used		
iblankeed	An “iblankeed” block must contain an additional integer array of values at each node, traditionally called the iblanke array. Valid iblanke values for the EnSight Gold format are:	
	0	for nodes which are exterior to the model, sometimes called blankeed-out nodes
	1	for nodes which are interior to the model, thus in the free stream and to be used
	<0 or >1	for any kind of boundary nodes
	In EnSight’s structured Part building dialog, the iblanke option selected will control which portion of the structured block is “created”. Thus, from the same structured block, the interior flow field part as well as a symmetry boundary part could be “created”.  <i>Note: By default EnSight does not do any “partial” cell iblanke processing. Namely, only complete cells containing no “exterior” nodes are created. It is possible to obtain partial cell processing by issuing the “test:partial_cells_on” command in the Command Dialog before reading the file.</i>	

with_ghost	A block with ghosts must contain an additional integer array of flags for each cell. A flag value of zero indicates a non-ghost cell. A flag value of non-zero indicates a ghost cell.
range	<p>A block with ranges will contain an extra line, following the ijk line, which gives min and max planes for each of the ijk directions.</p> <p>Thus, normally a 6 x 5 x 1 block part would start something like:</p> <pre> part   1 description block   6    5    1 0.00000e+00 ... (The coordinate information for the 30 nodes of the block must follow.)  But if only the top 6 x 3 x 1 portion was to be represented in the file, you can use "range" like:  part   1 description for top only block range   6    5    1   1    6    3    5    1    1 0.00000e+00 ... (The coordinate information for the 18 nodes of the top portion of the block must follow. Note that the ijk line following the block line contains the size of the original block - which is needed to properly deal with node and element numbering. The next line contains the imin, imax, jmin, jmax, kmin, kmax defining the subset ranges. The actual size of the block being defined is thus computed from these ranges: size_i = imax - imin + 1 size_j = jmax - jmin + 1 size_k = kmax - kmin + 1 </pre>

*Note that for structured data, the standard order of nodes is such that I's advance quickest, followed by J's, and then K's.*



*Generic Format*

## Usage Notes:

In general an unstructured part can contain several different element types.

element type can be any of:

point	g_point
bar2	g_bar2
bar3	g_bar3
tria3	g_tria3
tria6	g_tria6
quad4	g_quad4
quad8	g_quad8
tetra4	g_tetra4
tetra10	g_tetra10
pyramid5	g_pyramid5
pyramid13	g_pyramid13
penta6	g_penta6
penta15	g_penta15
hexa8	g_hexa8
hexa20	g_hexa20
nsided	g_nsided
nfaced	g_nfaced

# = a part number

nn = total number of nodes in a part

ne = number of elements of a given type

np = number of nodes per element for a given element type

nf = number of faces per nfaced element

id\_\* = node or element id number

x\_\* = x component

y\_\* = y component

z\_\* = z component

n\*\_e\* = node number for an element

f\*\_e\* = face number for an nfaced element

ib\_\* = iblanking value

gf\_e\* = ghost flag for a structured cell

[ ] contain optional portions

< > contain choices

` indicates the beginning of an unformatted sequential FORTRAN binary write

' indicates the end of an unformatted sequential FORTRAN binary write

**C Binary form:**

C Binary	80 chars
description line 1	80 chars
description line 2	80 chars
node id <off/given/assign/ignore>	80 chars
element id <off/given/assign/ignore>	80 chars
[extents	80 chars
xmin xmax ymin ymax zmin zmax]	6 floats
part	80 chars
#	1 int
description line	80 chars
coordinates	80 chars
nn	1 int
[id_n1 id_n2 ... id_nn]	nn ints
x_n1 x_n2 ... x_nn	nn floats
y_n1 y_n2 ... y_nn	nn floats
z_n1 z_n2 ... z_nn	nn floats
element type	80 chars
ne	1 int
[id_e1 id_e2 ... id_ne]	ne ints
n1_e1 n2_e1 ... np_e1	
n1_e2 n2_e2 ... np_e2	
.	
.	
n1_ne n2_ne ... np_ne	ne*np ints
element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
description line	80 chars
block [iblancked] [with_ghost] [range]	80 chars
i j k                    # nn = i*j*k, ne = (i-1)*(j-1)*(k-1)	3 ints
[imin imax jmin jmax kmin kmax]   # if range used:	6 ints
$nn = (imax - imin + 1) * (jmax - jmin + 1) * (kmax - kmin + 1)$	
$ne = (imax - imin) * (jmax - jmin) * (kmax - kmin)$	
x_n1 x_n2 ... x_nn	nn floats
y_n1 y_n2 ... y_nn	nn floats
z_n1 z_n2 ... z_nn	nn floats
[ib_n1 ib_n2 ... ib_nn]	nn ints
[ghost_flags]	80 chars
[gf_e1 gf_e2 ... gf_ne]	ne ints
[node_ids]	80 chars
[id_n1 id_n2 ... id_nn]	nn ints
[element_ids]	80 chars
[id_e1 id_e2 ... id_ne]	ne ints
part	80 chars
#	1 int
description line	80 chars
block rectilinear [iblancked] [with_ghost] [range]	80 chars
i j k                    # nn = i*j*k, ne = (i-1)*(j-1)*(k-1)	3 ints
[imin imax jmin jmax kmin kmax]   # if range used:	6 ints
$nn = (imax - imin + 1) * (jmax - jmin + 1) * (kmax - kmin + 1)$	
$ne = (imax - imin) * (jmax - jmin) * (kmax - kmin)$	
x_1 x_2 ... x_i	i floats
y_1 y_2 ... y_j	j floats

z_1 z_2 ... z_k	k floats
[ib_n1 ib_n2 ... ib_nn]	nn ints
[ghost_flags]	80 chars
[gf_e1 gf_e2 ... gf_ne]	ne ints
[node_ids]	80 chars
[id_n1 id_n2 ... id_nn]	nn ints
[element_ids]	80 chars
[id_e1 id_e2 ... id_ne]	ne ints
part	80 chars
#	1 int
description line	80 chars
block uniform [iblancked] [with_ghost] [range]	80 chars
i j k                # nn = i*j*k, ne = (i-1)*(j-1)*(k-1)	3 ints
[imin imax jmin jmax kmin kmax]    # if range used:	6 ints
nn = (imax-imin+1) * (jmax-jmin+1) * (kmax-kmin+1)	
ne = (imax-imin) * (jmax-jmin) * (kmax-kmin)	
x_origin y_origin z_origin	3 floats
x_delta y_delta z_delta	3 floats
[ib_n1 ib_n2 ... ib_nn]	nn ints
[ghost_flags]	80 chars
[gf_e1 gf_e2 ... gf_ne]	ne ints
[node_ids]	80 chars
[id_n1 id_n2 ... id_nn]	nn ints
[element_ids]	80 chars
[id_e1 id_e2 ... id_ne]	ne ints

**Fortran Binary form:**

'Fortran Binary'	
'description line 1'	80 chars
'description line 2'	80 chars
'node id <off/given/assign/ignore>'	80 chars
'element id <off/given/assign/ignore>'	80 chars
['extents'	80 chars
'xmin xmax ymin ymax zmin zmax']	6 floats
'part'	80 chars
'#'	1 int
'description line'	80 chars
'coordinates'	80 chars
'nn'	1 int
['id_n1 id_n2 ... id_nn']	nn ints
'x_n1 x_n2 ... x_nn'	nn floats
'y_n1 y_n2 ... y_nn'	nn floats
'z_n1 z_n2 ... z_nn'	nn floats
'element type'	80 chars
'ne'	1 int
['id_e1 id_e2 ... id_ne']	ne ints
'n1_e1 n2_e1 ... np_e1	
n1_e2 n2_e2 ... np_e2	
.	
.	
n1_ne n2_ne ... np_ne'	ne*np ints
'element type'	80 chars
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars

```

`#`                                     1 int
`description line`                     80 chars
`block [iblanke] [with_ghost] [range]` 80 chars
`i j k`                               # nn = i*j*k, ne = (i-1)*(j-1)*(k-1) 3 ints
[imin imax jmin jmax kmin kmax] # if range used: 6 ints
nn = (imax-imin+1)*(jmax-jmin+1)*(kmax-kmin+1)
ne = (imax-imin)*(jmax-jmin)*(kmax-kmin)

`x_n1 x_n2 ... x_nn`                   nn floats
`y_n1 y_n2 ... y_nn`                   nn floats
`z_n1 z_n2 ... z_nn`                   nn floats
[ib_n1 ib_n2 ... ib_nn]                 nn ints
[ghost_flags]                           80 chars
[gf_e1 gf_e2 ... gf_ne]                 ne ints
[node_ids]                              80 chars
[id_n1 id_n2 ... id_nn]                 nn ints
[element_ids]                           80 chars
[id_e1 id_e2 ... id_ne]                 ne ints
`part`                                  80 chars
`#`                                     1 int
`description line`                     80 chars
`block rectilinear [iblanke] [with_ghost] [range]` 80 chars
`i j k`                               # nn = i*j*k, ne = (i-1)*(j-1)*(k-1) 3 ints
[imin imax jmin jmax kmin kmax] # if range used: 6 ints
nn = (imax-imin+1)*(jmax-jmin+1)*(kmax-kmin+1)
ne = (imax-imin)*(jmax-jmin)*(kmax-kmin)

`x_1 x_2 ... x_i`                       i floats
`y_1 y_2 ... y_j`                       j floats
`z_1 z_2 ... z_k`                       k floats
[ib_n1 ib_n2 ... ib_nn]                 nn ints
[ghost_flags]                           80 chars
[gf_e1 gf_e2 ... gf_ne]                 ne ints
[node_ids]                              80 chars
[id_n1 id_n2 ... id_nn]                 nn ints
[element_ids]                           80 chars
[id_e1 id_e2 ... id_ne]                 ne ints
`part`                                  80 chars
`#`                                     1 int
`description line`                     80 chars
`block uniform [iblanke] [with_ghost] [range]` 80 chars
`i j k`                               # nn = i*j*k, ne = (i-1)*(j-1)*(k-1) 3 ints
[imin imax jmin jmax kmin kmax] # if range used: 6 ints
nn = (imax-imin+1)*(jmax-jmin+1)*(kmax-kmin+1)
ne = (imax-imin)*(jmax-jmin)*(kmax-kmin)

`x_origin y_origin z_origin`            3 floats
`x_delta y_delta z_delta`               3 floats
[ib_n1 ib_n2 ... ib_nn]                 nn ints
[ghost_flags]                           80 chars
[gf_e1 gf_e2 ... gf_ne]                 ne ints
[node_ids]                              80 chars
[id_n1 id_n2 ... id_nn]                 nn ints
[element_ids]                           80 chars
[id_e1 id_e2 ... id_ne]                 ne ints

```

**ASCII form:**

```

description line 1                      A (max of 79 typ)
description line 2                      A
node id <off/given/assign/ignore>      A
element id <off/given/assign/ignore>   A

```

```

[extents                                     A
xmin xmax                                  2E12.5
ymin ymax                                  2E12.5
zmin zmax]                                2E12.5
part                                       A
#                                         I10
description line                           A
coordinates                               A
nn                                         I10
[id_n1                                     I10    1/line (nn)
 id_n2
 .
 .
 id_nn]
x_n1                                     E12.5  1/line (nn)
x_n2
 .
 .
x_nn
y_n1                                     E12.5  1/line (nn)
y_n2
 .
 .
y_nn
z_n1                                     E12.5  1/line (nn)
z_n2
 .
 .
z_nn
element type                             A
ne                                         I10
[id_e1                                     I10    1/line (ne)
 id_e2
 .
 .
 id_ne]
n1_e1 n2_e1 ... np_e1                    I10    np/line
n1_e2 n2_e2 ... np_e2                    (ne lines)
 .
 .
n1_ne n2_ne ... np_ne
element type                             A
 .
 .
part                                       A
 .
 .
part                                       A
#                                         I10
description line                           A
block [iblanke] [with_ghost] [range]      A
i j k          # nn = i*j*k, ne = (i-1)*(j-1)*(k-1)  3I10
[imin imax jmin jmax kmin kmax] # if range used:    6I10
nn = (imax-imin+1)*(jmax-jmin+1)*(kmax-kmin+1)
ne = (imax-imin)*(jmax-jmin)*(kmax-kmin)
x_n1                                     E12.5  1/line (nn)
x_n2
 .
 .
x_nn
y_n1                                     E12.5  1/line (nn)

```

## 11.1 EnSight Gold Geometry File Format

```

y_n2
.
.
y_nn
z_n1          E12.5  1/line (nn)
z_n2
.
.
z_nn
[ib_n1        I10    1/line (nn)
 ib_n2
.
.
 ib_nn]
[ghost_flags] 80 chars
[gf_e1        I10    1/line (ne)
 gf_e2
.
.
 gf_ne]
[node_ids]     80 chars
[id_n1         I10    1/line (nn)
 id_n2
.
.
 id_nn]
[element_ids]  80 chars
[id_e1         I10    1/line (ne)
 id_e2
.
.
 id_ne]
part           A
#              I10
description line A
block rectilinear [iblanke] [with_ghost] [range] A
i j k          # nn = i*j*k, ne = (i-1)*(j-1)*(k-1) 3I10
[imin imax jmin jmax kmin kmax] # if range used: 6I10
nn = (imax-imin+1)*(jmax-jmin+1)*(kmax-kmin+1)
ne = (imax-imin)*(jmax-jmin)*(kmax-kmin)
x_1            E12.5  1/line (i)
x_2
.
.
x_i
y_1            E12.5  1/line (j)
y_2
.
.
y_j
z_1            E12.5  1/line (k)
z_2
.
.
z_k
[ib_n1        I10    1/line (nn)
 ib_n2
.
.
 ib_nn]
[ghost_flags] 80 chars

```

```

[gf_e1                                I10    1/line (ne)
gf_e2
.
.
gf_ne]
[node_ids]                           80 chars
[id_n1                                I10    1/line (nn)
id_n2
.
.
id_nn]
[element_ids]                         80 chars
[id_e1                                I10    1/line (ne)
id_e2
.
.
id_ne]
part                                  A
#                                    I10
description line                      A
block uniform [iblanke] [with_ghost] [range] A
i j k                                # nn = i*j*k, ne = (i-1)*(j-1)*(k-1) 3I10
[imin imax jmin jmax kmin kmax] # if range used: 6I10
nn = (imax-imin+1)*(jmax-jmin+1)*(kmax-kmin+1)
ne = (imax-imin)*(jmax-jmin)*(kmax-kmin)

x_origin                             E12/5
y_origin                             E12/5
z_origin                             E12/5
x_delta                             E12.5
y_delta                             E12.5
z_delta                             E12.5
[ib_n1                                I10    1/line (nn)
ib_n2
.
.
ib_nn]
[ghost_flags]                        80 chars
[gf_e1                                I10    1/line (ne)
gf_e2
.
.
gf_ne]
[node_ids]                           80 chars
[id_n1                                I10    1/line (nn)
id_n2
.
.
id_nn]
[element_ids]                         80 chars
[id_e1                                I10    1/line (ne)
id_e2
.
.
id_ne]

```

## Notes:

- If `node id` is given or ignore, the `[id]` section must be there for each part.
- If `element id` is given or ignore, the `[id]` section must be there for each element type of each part
- If `iblancked` is there, the `[ib]` section must be there for the block.
- `x`, `y`, and `z` coordinates are mandatory, even if a 2D problem.
- If `block rectilinear`, then the `x`, `y`, `z` coordinates change to the `x`, `y`, and `z` delta vectors.
- If `block uniform`, then the `x`, `y`, `z` coordinates change to the `x`, `y`, `z` coordinates of the origin and the `x`, `y`, and `z` delta values.
- If `block range`, the `ijk` min/max range line must follow the `ijk` line. And the number of nodes and elements is based on the ranges. The `ijk` line indicates the size of the original block.
- If `with_ghost` is on the `block` line, then the `ghost_flag` section must be there
- Ids are just labels, the coordinate (or element) order is implied.
- The minimum needed for unstructured empty parts is the three lines:

```
part
      #                (use the actual part number)
description
```

- The minimum needed for structured empty parts is the five lines:

```
part
      #                (use the actual part number)
description
block
      0      0      0
```

- Element blocks for `nsided` elements contain an additional section - **the number of nodes in each element**. See below.

**C Binary form of element block, if nsided:**

<code>nsided</code>		80 chars
<code>ne</code>		1 int
<code>[id_n1 id_n2 ... id_ne]</code>		ne ints
<code>np1 np2 ... npne</code>	<b>This data is needed</b>	ne ints
<code>e1_n1 e1_n2 ... e1_np1</code>		
<code>e2_n1 e2_n2 ... e2_np2</code>		
<code>.</code>		
<code>.</code>		
<code>ne_n1 ne_n2 ... ne_npne</code>		<code>np1+np2+...+npne</code> ints

**Fortran Binary form of element block, if nsided:**

<code>'nsided'</code>		80 chars
<code>'ne'</code>		1 int
<code>['id_n1 id_n2 ... id_ne']</code>		ne ints
<code>'np1 np2 ... npne'</code>	<b>This data is needed</b>	ne ints
<code>'e1_n1 e1_n2 ... e1_np1</code>		



```

e2_n1 e2_n2 ... e2_np2
.
.
ne_n1 ne_n2 ... ne_npne'

```

**np1+np2+...+npne** ints

**Ascii form of element block, if nsided:**

```

nsided
ne
[id_n1
 id_n2
.
 id_ne]
np1
np2
.
.
npne
e1_n1 e1_n2 ... e1_np1
e2_n1 e2_n2 ... e2_np2
.
ne_n1 ne_n2 ... ne_npne

```

A  
I10  
I10 1/line (ne)  
  
I10 1/line (ne)  
  
I10 **np\*/line**  
(ne lines)

**This data is needed**

- Element blocks for `nfaced` elements are more involved since they are described by their `nsided` faces. Thus, there is the optional section for `ids` (`id_e*`), a section for the number of faces per element (`nf_e*`), a section for number of nodes per face per element (`np(f*_e*)`), and a section for the connectivity of each `nsided` face of each element (`n*(f*_e*)`). See below.

**C Binary form of element block, if nfaced:**

```

nfaced
ne
[id_e1 id_e2 ... id_ne]
nf_e1 nf_e2 ... nf_ne
np(f1_e1) np(f2_e1) ... np(nf_e1)
np(f1_e2) np(f2_e2) ... np(nf_e2)
.
.
np(f1_ne) np(f2_ne) ... np(nf_ne)
n1(f1_e1) n2(f1_e1) ... n(np(f1_e1))
n1(f2_e1) n2(f2_e1) ... n(np(f2_e1))
.
n1(nf_e1) n2(nf_e1) ... n(np(nf_e1))
n1(f1_e2) n2(f1_e2) ... n(np(f1_e2))
n1(f2_e2) n2(f2_e2) ... n(np(f2_e2))
.
n1(nf_e2) n2(nf_e2) ... n(np(nf_e2))
.
.
n1(f1_ne) n2(f1_ne) ... n(np(f1_ne))
n1(f2_ne) n2(f2_ne) ... n(np(f2_ne))
.
n1(nf_ne) n2(nf_ne) ... n(np(nf_ne))

```

80 chars  
1 int  
ne ints  
ne ints  
  
**nf\_e1+nf\_e2+...+nf\_ne** ints  
  
**np(f1\_e1)+np(f2\_e1)+...+np(nf\_ne)** ints

**Fortran Binary form of element block, if nfaced:**

```

'nfaced'
'ne'
['id_e1 id_e2 ... id_ne']
'nf_e1 nf_e2 ... nf_ne'

```

80 chars  
1 int  
ne ints  
ne ints

```

`np(f1_e1) np(f2_e1) ... np(nf_e1)
np(f1_e2) np(f2_e2) ... np(nf_e2)
.
.
np(f1_ne) np(f2_ne) ... np(nf_ne)`          nf_e1+nf_e2+...+nf_ne ints
`n1(f1_e1) n2(f1_e1) ... n(np(f1_e1))
n1(f2_e1) n2(f2_e1) ... n(np(f2_e1))
.
n1(nf_e1) n2(nf_e1) ... n(np(nf_e1))
n1(f1_e2) n2(f1_e2) ... n(np(f1_e2))
n1(f2_e2) n2(f2_e2) ... n(np(f2_e2))
.
n1(nf_e2) n2(nf_e2) ... n(np(nf_e2))
.
.
n1(f1_ne) n2(f1_ne) ... n(np(f1_ne))
n1(f2_ne) n2(f2_ne) ... n(np(f2_ne))
.
n1(nf_ne) n2(nf_ne) ... n(np(nf_ne))`      np(f1_e1)+np(f2_e1)+...+np(nf_ne) ints

```

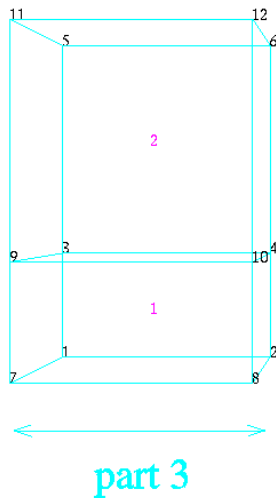
**Ascii form of element block, if nfaced:**

```

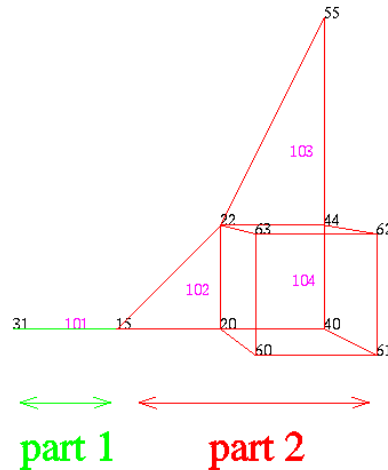
nfaced          A
ne              I10
[id_e1          I10    1/line
 id_e2          (ne lines)
.
 id_ne]
nf_e1          I10    1/line
nf_e2          (ne lines)
.
nf_ne
np(f1_e1)      I10    1/line
np(f2_e1)      (nf_e1+nf_e2+...+nf_ne lines)
.
np(nf_e1)
np(f1_e2)
np(f2_e2)
.
np(nf_e2)
.
.
np(f1_ne)
np(f2_ne)
.
np(nf_ne)
n1(f1_e1) n2(f1_e1) ... n(np(f1_e1))
n1(f2_e1) n2(f2_e1) ... n(np(f2_e1))
.
n1(nf_e1) n2(nf_e1) ... n(np(nf_e1))
n1(f1_e2) n2(f1_e2) ... n(np(f1_e2))
n1(f2_e2) n2(f2_e2) ... n(np(f2_e2))
.
n1(nf_e2) n2(nf_e2) ... n(np(nf_e2))
.
.
n1(f1_ne) n2(f1_ne) ... n(np(f1_ne))
n1(f2_ne) n2(f2_ne) ... n(np(f2_ne))
.
n1(nf_ne) n2(nf_ne) ... n(np(nf_ne))

```

### Structured Part



### Unstructured Parts



#### EnSight Gold

##### Geometry File Example

The following is an example of an ASCII EnSight Gold geometry file: This is the same example model as given in the EnSight6 geometry file section (only in Gold format) with 11 defined unstructured nodes from which 2 unstructured parts are defined, and a 2x3x2 structured part as depicted in the above diagram.

*Note:* The example file below (`engold.geo`) and all example variable files in the gold section (also prefixed with `engold`) may be found under your EnSight installation directory (path: `$CEI_HOME/ensight80/data/user_manual`).

*Note:* The appended “#” comment lines are for your reference only, and are not valid format lines within a geometry file as appended below. **Do NOT put these # comments in your file!!!**

```
This is the 1st description line of the EnSight Gold geometry example
This is the 2nd description line of the EnSight Gold geometry example
node id given
element id given
extents
0.00000e+00 2.00000e+00
0.00000e+00 2.00000e+00
0.00000e+00 2.00000e+00
part
```

```
1
2D uns-elements (description line for part 1)
coordinates
```

```
10          # nn          Do NOT put these # comments in your file!!
15          # node ids
20
40
22
44
55
60
61
62
63
4.00000e+00      # x components
5.00000e+00
6.00000e+00
```

## 11.1 EnSight Gold Geometry File Format

```

5.00000e+00
6.00000e+00
6.00000e+00
5.00000e+00
6.00000e+00
6.00000e+00
5.00000e+00
0.00000e+00      # y components
0.00000e+00
0.00000e+00
1.00000e+00
1.00000e+00
3.00000e+00
0.00000e+00
0.00000e+00
1.00000e+00
1.00000e+00
0.00000e+00      # z components
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00

tria3      # element type
      2      # ne
      102     # element ids
      103
      1      2      4
      4      5      6

hexa8
      1
      104
      2      3      5      4      7      8      9      10

part
      2
1D uns-elements (description line for part 2)
coordinates
      2
      15
      31
4.00000e+00
3.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
bar2
      1
      101
      2      1

part
      3
3D struct-part (description line fro part 3)
block iblanked
      2      3      2
0.00000e+00      # i components
2.00000e+00
0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00

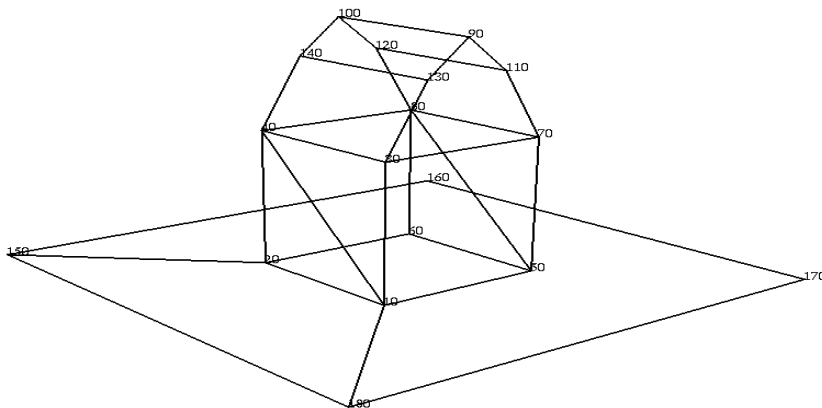
```

```
0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00
0.00000e+00      # j components
0.00000e+00
1.00000e+00
1.00000e+00
3.00000e+00
3.00000e+00
0.00000e+00
0.00000e+00
1.00000e+00
1.00000e+00
3.00000e+00
3.00000e+00
0.00000e+00      # k components
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
1      # iblanking
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
```

*Simple example  
using nsided/  
nfaced elements*

The following is an example of an ASCII EnSight Gold geometry file with nsided and nfaced data. It is a non-realistic, simple model which is intended only to illustrate the format. Two nsided elements and three nfaced elements are used, even though the model could have been represented with a single nsided and single nfaced element.

*Note:* The appended “#” comment lines are for your reference only, and are not valid format lines within a geometry file as appended below. **Do NOT put these # comments in your file!!!**



```
simple example for nsided/nfaced
element types in EnSight Gold Format
node id given
element id given
extents
-2.00000e+00 4.00000e+00
 0.00000e+00 3.50000e+00
-2.00000e+00 4.00000e+00
part
```

```
      1
barn
coordinates
```

```
      18      # nn
      10      # node ids
      20
      30
      40
      50
      60
      70
      80
      90
     100
     110
     120
     130
     140
     150
     160
     170
     180
```

**Do NOT put these # comments in your file!!**

```
0.00000e+00      # x components
2.00000e+00
0.00000e+00
2.00000e+00
```

```

0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00
0.00000e+00
2.00000e+00
4.00000e+00
4.00000e+00
-2.00000e+00
-2.00000e+00
0.00000e+00      # y components
0.00000e+00
2.00000e+00
2.00000e+00
0.00000e+00
0.00000e+00
2.00000e+00
2.00000e+00
3.50000e+00
3.50000e+00
3.00000e+00
3.00000e+00
3.00000e+00
3.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00      # x components
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
2.00000e+00
1.00000e+00
1.00000e+00
1.50000e+00
1.50000e+00
0.50000e+00
0.50000e+00
-2.00000e+00
4.00000e+00
4.00000e+00
-2.00000e+00
nsided
    2                      # 2 nsided elements
    101                   # element ids
    202
    4                      # 4 nodes in first element
    8                      # 8 nodes in second element
    2      15      18      1      15      2      6      5      # connectivity of element 1
    1      18      17      16      15      2      6      5      # connectivity of element 2
nfaced
    3                      # 3 nfaced polyhedra elements
    1001                   # element ids
    1002
    1003
    5                      # number of faces in element 1
    5                      # number of faces in element 2
    7                      # number of faces in element 3
    3                      # number of nodes in face 1 of element 1

```

## 11.1 EnSight Gold Geometry File Format

3					#	face 2 of element 1
4					#	face 3 of element 1
4					#	face 4 of element 1
4					#	face 5 of element 1
3					#	number of nodes in face 1 of element 2
3					#	face 2 of element 2
4					#	face 3 of element 2
4					#	face 4 of element 2
4					#	face 5 of element 2
5					#	number of nodes in face 1 of element 3
5					#	face 2 of element 3
4					#	face 3 of element 3
4					#	face 4 of element 3
4					#	face 5 of element 3
4					#	face 6 of element 3
4					#	face 7 of element 3
5	6	8			#	connectivity of face 1 of element 1
2	1	4			#	face 2 of element 1
6	2	4	8		#	face 3 of element 1
8	4	1	5		#	face 4 of element 1
1	2	6	5		#	face 5 of element 1
5	8	7			#	connectivity of face 1 of element 2
1	3	4			#	face 2 of element 2
7	8	4	3		#	face 3 of element 2
7	3	1	5		#	face 4 of element 2
5	1	4	8		#	face 5 of element 2
8	4	14	10	12	#	connectivity of face 1 of element 3
7	11	9	13	3	#	face 2 of element 3
7	8	12	11		#	face 3 of element 3
11	12	10	9		#	face 4 of element 3
9	10	14	13		#	face 5 of element 3
13	14	4	3		#	face 6 of element 3
7	3	4	8		#	face 7 of element 3



### Simple examples using ghost cells

The following two ASCII EnSight Gold geometry file examples show use of ghost cells in unstructured and structured models. First the geometry file for the total model, composed of four parts, is given without any ghost cells. Then, two of four separate geometry files – each containing just one of the original parts (and appropriate ghost cells) will be given. This is supposed to simulate a decomposed model, such as you might provide for EnSight’s Server-of-Servers.

#### Note:

For unstructured models, ghost cells are simply a new element type. For structured models, ghost cells are an “iblack”-like flag.

#### Unstructured model

Nodes

21	22	23	24	25
16	17	18	19	20
part 3		part 4		
11	12	13	14	15
6	7	8	9	10
part 1		part 2		
1	2	3	4	5

Elements

13	14	15	16
part 3		part 4	
9	10	11	12
5	6	7	8
part 1		part 2	
1	2	3	4

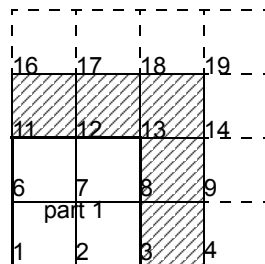
#### Total Unstructured Model Geometry File:

EnSight Model Geometry File	1.00000e+00			
EnSight 7.1.0	1.00000e+00			
node id given	1.00000e+00			
element id given	2.00000e+00			
extents	2.00000e+00			
0.00000e+00 4.00000e+00	2.00000e+00			
0.00000e+00 4.00000e+00	0.00000e+00			
0.00000e+00 0.00000e+00	0.00000e+00			
part	0.00000e+00			
1	0.00000e+00			
bottom left	0.00000e+00			
coordinates	0.00000e+00			
9	0.00000e+00			
1	0.00000e+00			
2	0.00000e+00			
3	0.00000e+00			
6	quad4			
7	4			
8	1			
11	2			
12	5			
13	6			
0.00000e+00	1	2	5	4
1.00000e+00	2	3	6	5
2.00000e+00	4	5	8	7
0.00000e+00	5	6	9	8
1.00000e+00	part			
2.00000e+00	2			
0.00000e+00	bottom right			
1.00000e+00	coordinates			
2.00000e+00	9			
0.00000e+00	3			
0.00000e+00	4			
0.00000e+00	5			
0.00000e+00	8			


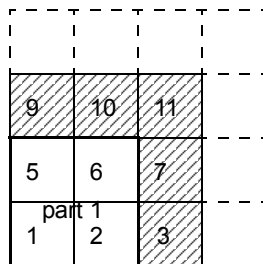
## 11-26

## EnSight 8 User Manual

Portion with part 1 containing ghost cells (other parts are empty)



## Elements

 Ghost Cell

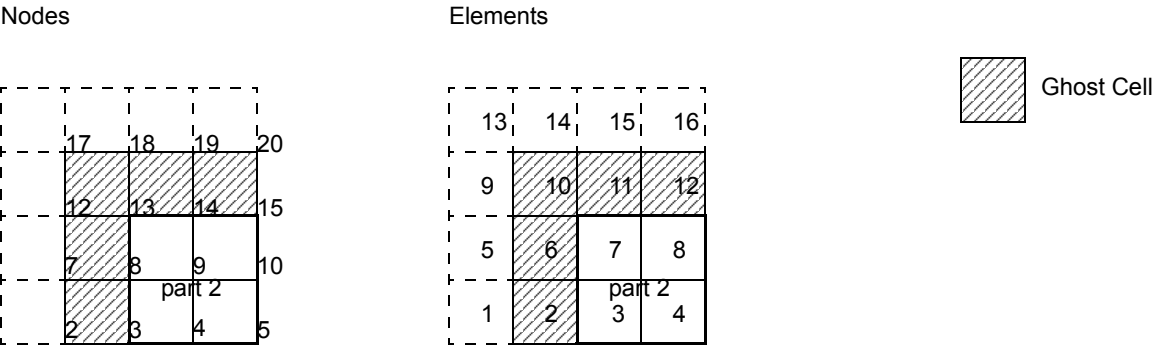
```

EnSight Model Geometry File
part 1 portion
node id given
element id given
extents
  0.00000e+00 4.00000e+00      0.00000e+00
4.00000e+00
  0.00000e+00 0.00000e+00
part
  1
bottom left
coordinates
  16
  1
  2
  3
  4
  6
  7
  8
  9
  11
  12
  13
  14
  16
  17
  18
  19
0.00000e+00
1.00000e+00
2.00000e+00
3.00000e+00
0.00000e+00
1.00000e+00
2.00000e+00
3.00000e+00
0.00000e+00
1.00000e+00
2.00000e+00
3.00000e+00
0.00000e+00
1.00000e+00
2.00000e+00
3.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
1.00000e+00
1.00000e+00
1.00000e+00
1.00000e+00
2.00000e+00

```

[illegible]

Portion with part 2 containing ghost cells (other parts are empty)



EnSight Model Geometry File	1.00000e+00			
part 2 portion	2.00000e+00			
node id given	2.00000e+00			
element id given	2.00000e+00			
extents	2.00000e+00			
0.00000e+00 4.00000e+00	3.00000e+00			
0.00000e+00 4.00000e+00	3.00000e+00			
0.00000e+00 0.00000e+00	3.00000e+00			
part	3.00000e+00			
1	0.00000e+00			
bottom left	0.00000e+00			
part	0.00000e+00			
2	0.00000e+00			
bottom right	0.00000e+00			
coordinates	0.00000e+00			
16	0.00000e+00			
2	0.00000e+00			
3	0.00000e+00			
4	0.00000e+00			
5	0.00000e+00			
7	0.00000e+00			
8	0.00000e+00			
9	0.00000e+00			
10	0.00000e+00			
12	0.00000e+00			
13	0.00000e+00			
14	quad4			
15	4			
17	3			
18	4			
19	7			
20	8			
1.00000e+00	2	3	7	6
2.00000e+00	3	4	8	7
3.00000e+00	6	7	11	10
4.00000e+00	7	8	12	11
1.00000e+00	g_quad4			
2.00000e+00	5			
3.00000e+00	1			
4.00000e+00	1			
1.00000e+00	1			
2.00000e+00	1			
3.00000e+00	1			
4.00000e+00	1			
1.00000e+00	1			
2.00000e+00	1			
3.00000e+00	1	2	6	5
4.00000e+00	5	6	10	9
1.00000e+00	9	10	14	13
2.00000e+00	10	11	15	14
3.00000e+00	11	12	16	15
4.00000e+00	part			
0.00000e+00	3			
0.00000e+00	top left			
0.00000e+00	part			
0.00000e+00	4			
1.00000e+00	top right			
1.00000e+00				
1.00000e+00				

Structured model

(using essentially the same model, but in structured format):

Nodes

21	22	23	24	25
16	17	18	19	20
part 3		part 4		
11	12	13	14	15
6	7	8	9	10
part 1		part 2		
1	2	3	4	5

Elements

13	14	15	16
part 3		part 4	
9	10	11	12
5	6	7	8
part 1		part 2	
1	2	3	4

5 x 5 x 1 Total Structured Model

EnSight Model Geometry File

Total Structured Model

node id assign

element id assign

extents

0.00000e+00 4.00000e+00

0.00000e+00 4.00000e+00

0.00000e+00 0.00000e+00

part

1

left bottom

block uniform range

5

5

1

1

3

1

3

1

1

0.00000e+00

0.00000e+00

0.00000e+00

1.00000e+00

1.00000e+00

0.00000e+00

part

2

bottom right

block uniform range

5

5

1

3

5

1

3

1

1

2.00000e+00

0.00000e+00

0.00000e+00

1.00000e+00

1.00000e+00

0.00000e+00

part

3

top left

block uniform range

5

5

1

1

3

3

5

1

1

0.00000e+00

2.00000e+00

0.00000e+00

1.00000e+00

1.00000e+00

0.00000e+00

part

4

top right

block uniform range

5

5

1

3

5

3

5

1

1

2.00000e+00

2.00000e+00

0.00000e+00

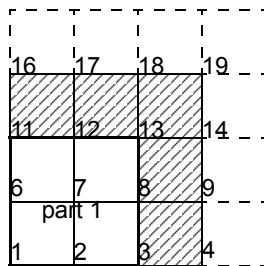
1.00000e+00

1.00000e+00

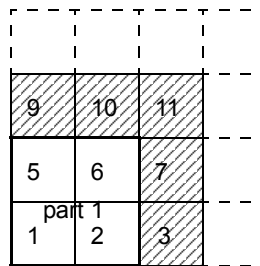
0.00000e+00

Portion with part 1 containing ghost cells (other parts are empty)

Nodes



Elements



Ghost Cell

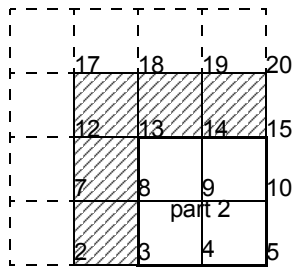
```

EnSight Model Geometry File
part 1 portion only
node id assign
element id assign
extents
  0.00000e+00 4.00000e+00
  0.00000e+00 4.00000e+00
  0.00000e+00 0.00000e+00
part
  1
left bottom
block uniform range with_ghost
  4      4      1
  1      4      1      4      1      1
  0.00000e+00
  0.00000e+00
  0.00000e+00
  1.00000e+00
  1.00000e+00
  0.00000e+00
ghost_flags
  0
  0
  1
  0
  0
  1
  1
  1
  1
  1
Part          /* Empty Part */
  2
right bottom
block
  0      0      0
part          /* Empty Part */
  3
left top
block
  0      0      0
part          /* Empty Part */
  4
right top
block
  0      0      0

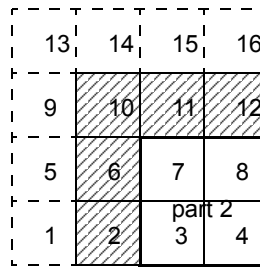
```

Portion with part 2 containing ghost cells (other parts are empty)

Nodes



Elements



```

EnSight Model Geometry File
part 2 portion only
node id assign
element id assign
extents
  0.00000e+00 4.00000e+00
  0.00000e+00 4.00000e+00
  0.00000e+00 0.00000e+00
part
  1
left bottom
block
  0      0      0
part
  2
right bottom
block uniform range with_ghost
  4      4      1      4      1      1
  2      5      1
  1.00000e+00
  0.00000e+00
  0.00000e+00
  1.00000e+00
  1.00000e+00
  0.00000e+00
ghost_flags
  1
  0
  0
  1
  0
  0
  1
  1
  1
  1
part          /* Empty Part */
  3
left top
block
  0      0      0
part          /* Empty Part */
  4
right top
block
  0      0      0

```

*Note: For both the unstructured and the structured model above, only the first two files (parts 1 and 2) are given. The portion files for parts 3 and 4 are not given, but would be similar to those for parts 1 and 2.*

## EnSight Gold Case File Format

The Case file is an ASCII free format file that contains all the file and name information for accessing model (and measured) geometry, variable, and time information. It is comprised of five sections (FORMAT, GEOMETRY, VARIABLE, TIME, FILE) as described below:

*Notes:* All lines in the Case file are limited to 79 characters.  
 The titles of each section must be in all capital letters.  
 Anything preceded by a “#” denotes a comment and is ignored. Comments may append information lines or be placed on their own lines.  
 Information following “:” may be separated by white spaces or tabs.  
 Specifications encased in “[ ]” are optional, as indicated.

### Format Section

This is a required section which specifies the type of data to be read.

Usage:

```
FORMAT
type:      ensight gold
```

### Geometry Section

This is a required section which specifies the geometry information for the model (as well as measured geometry if present, and periodic match file ([see Section 11.9, Periodic Matchfile Format](#)) if present).

Usage:

```
GEOMETRY
model:      [ts] [fs]      filename      [change_coords_only]
measured:   [ts] [fs]      filename      [change_coords_only]
match:      filename
boundary:   filename
```

where: *ts* = time set number as specified in TIME section. This is optional.

*fs* = corresponding file set number as specified in FILE section below.

**(Note, if you specify *fs*, then *ts* is no longer optional and must also be specified.)**

*filename* = The filename of the appropriate file.

-> Model or measured filenames for a static geometry case, as well as match and boundary filenames will not contain “\*” wildcards.

-> Model or measured filenames for a changing geometry case will contain “\*” wildcards.

*change\_coords\_only* = The option to indicate that the changing geometry (as indicated by wildcards in the filename) is coords only. Otherwise, changing geometry connectivity will be assumed.

### Variable Section

This is an optional section which specifies the files and names of the variables. Constant variable values can also be set in this section.

Usage:

```
VARIABLE
constant per case:      [ts]      description  const_value(s)
constant per case file: [ts]      description  cvfilename
scalar per node:        [ts] [fs]  description  filename
vector per node:        [ts] [fs]  description  filename
tensor symm per node:   [ts] [fs]  description  filename
tensor asym per node:   [ts] [fs]  description  filename
scalar per element:     [ts] [fs]  description  filename
```



```

vector per element:      [ts] [fs]  description  filename
tensor symm per element: [ts] [fs]  description  filename
tensor asym per element: [ts] [fs]  description  filename
scalar per measured node: [ts] [fs]  description  filename
vector per measured node: [ts] [fs]  description  filename
complex scalar per node:  [ts] [fs]  description  Re_fn   Im_fn   freq
complex vector per node:  [ts] [fs]  description  Re_fn   Im_fn   freq

```

```

complex scalar per element: [ts] [fs] description  Re_fn   Im_fn   freq
complex vector per element: [ts] [fs] description  Re_fn   Im_fn   freq

```

where:

`ts` = The corresponding time set number (or index) as specified in TIME section below. This is only required for transient constants and variables.

`fs` = The corresponding file set number (or index) as specified in FILE section below.

**(Note, if you specify `fs`, then `ts` is no longer optional and must also be specified.)**

`description` = The variable (GUI) name (ex. Pressure, Velocity, etc.)

`const_value(s)` = The constant value. If constants change over time, then `ns` (see TIME section below) constant values of `ts`.

`cvfilename` = The filename containing the constant values, one value per time step.

`filename` = The filename of the variable file. Note: only transient filenames contain “\*” wildcards.

`Re_fn` = The filename for the file containing the real values of the complex variable.

`Im_fn` = The filename for the file containing the imaginary values of the complex variable.

`freq` = The corresponding harmonic frequency of the complex variable. For complex variables where harmonic frequency is undefined, simply use the text string: UNDEFINED.

*Note: As many variable description lines as needed may be used.*

*Note: Variable descriptions have the following restrictions:*

*The variable description is limited to 19 characters in the current release.*

*Duplicate variable descriptions are not allowed.*

*Leading and trailing white space will be eliminated.*

*Variable descriptions must not start with a numeric digit.*

*Variable descriptions must not contain any of the following reserved characters:*

```

(   [   +   @   !   *   $
)   ]   -   space  #   ^   /

```

### Time Section

This is an optional section for steady state cases, but is required for transient cases. It contains time set information. Shown below is information for one time set. Multiple time sets (up to 16) may be specified for measured data as shown in Case File Example 3 below.

Usage:

```

TIME
time set:      ts [description]
number of steps: ns
filename start number: fs
filename increment: fi
time values:   time_1  time_2  .... time_ns

```

or

```

TIME
time set:          ts [description]
number of steps:   ns
filename numbers:  fn
time values:       time_1 time_2 .... time_ns

```

or

```

TIME
time set:          ts [description]
number of steps:   ns
filename numbers file: fnfilename
time values file:  tvfilename

```

where: *ts* = timeset number. This is the number referenced in the GEOMETRY and VARIABLE sections.

*description* = optional timeset description which will be shown in user interface.

*ns* = number of transient steps

*fs* = the number to replace the “\*” wildcards in the filenames, for the first step

*fi* = the increment to *fs* for subsequent steps

*time* = the actual time values for each step, each of which must be separated by a white space and which may continue on the next line if needed

*fn* = a list of numbers or indices, to replace the “\*” wildcards in the filenames.

*fnfilename* = name of file containing *ns* filename numbers (*fn*).

*tvfilename* = name of file containing the time values(*time\_1* ... *time\_ns*).

### File Section

This section is optional for expressing a transient case with single-file formats. This section contains single-file set information. This information specifies the number of time steps in each file of each *data entity*, i.e. each geometry and each variable (model and/or measured). Each data entity’s corresponding file set might have multiple *continuation* files due to system file size limit, i.e. ~2 GB for 32-bit and ~4 TB for 64-bit architectures. **Each file set corresponds to one and only one time set, but a time set may be referenced by many file sets.** The following information may be specified in each file set. For file sets where all of the time set data exceeds the maximum file size limit of the system, both *filename index* and *number of steps* are repeated within the file set definition for each continuation file required. Otherwise *filename index* may be omitted if there is only one file. File set information is shown in Case File Example 4 below.

Usage:

```

FILE
file set:          fs
filename index:    fi # Note: only used when data continues in other files
number of steps:   ns

```

where: *fs* = file set number. This is the number referenced in the GEOMETRY and VARIABLE sections above.

*ns* = number of transient steps

*fi* = file index number in the file name (replaces “\*” in the filenames)

## Material Section

This is an optional section for material interface part case. It contains material set information. Shown below is information for one material set. (**Note, currently only one material set is supported.**) An example of this material set information is appended to [EnSight Gold Material Files Format](#).

## Usage:

```
MATERIAL
material set number:      ms [description]
material id count:        nm
material id numbers:      matno_1 matno_2 ... matno_nm
material id names:        matdesc_1 mat_2 ... mat_nm
material id per element:  [ts] [fs] filename
material mixed ids:       [ts] [fs] filename
material mixed values:    [ts] [fs] filename

where:
ts = The corresponding time set number (or index) as specified in TIME section
    above. This is only required for transient materials.

fs = The corresponding file set number (or index) as specified in FILE section
    above. (Note, if you specify fs, then ts is no longer optional and must also be
    specified.)

ms = Material set number. (Note, currently there is only one, and it must be a
    positive number.)

description = Optional material set description which will be reflected in the file
    names of exported material files.

nm = Number of materials for this set.

matno = Material number used in the material and mixed-material id files. There
    should be nm of these. Non-positive numbers are grouped as the "null
    material". See EnSight Gold Material Files Format

matdesc = GUI material description corresponding to the nm matno's.

filename = The filename of the appropriate material file. Note, only transient
    filenames contain "*" wildcards. The three required files are the
    material id per element, the mixed-material ids, and the mixed-material
    values files.
```

**Note:** *Material descriptions are limited to 19 characters in the current release. Material descriptions and file names must not start with a numeric digit and must not contain any of the following reserved characters:*

```
( [ + @ ! * $
) ] - # ^ / space
```

## Case File Example 1

The following is a minimal EnSight Gold case file for a steady state model with some results.

*Note: this (engold.case) file, as well as all of its referenced geometry and variable files (along with a couple of command files) can be found under your installation directory (path: \$CEI\_HOME/ensight80/data/user\_manual). The EnSight Gold Geometry File Example and the Variable File Examples are the contents of these files.*

```
FORMAT
type: ensight gold

GEOMETRY
model: engold.geo

VARIABLE
constant per case:          Cden .8
```

## 11.1 EnSight Gold Case File Format

```

scalar per element:      Esca  engold.Esca
scalar per node:         Nsca  engold.Nsca

vector per element:      Evec  engold.Evec
vector per node:         Nvec  engold.Nvec

tensor symm per element: Eten  engold.Eten
tensor symm per node:    Nten  engold.Nten

complex scalar per element: Ecmp  engold.Ecmp_rengold.Ecmp_i2.
complex scalar per node:   Ncmp  engold.Ncmp_rengold.Ncmp_i4.

```

**Case File Example 2**    The following is a Case file for a transient model. The connectivity of the geometry is also changing.

```

FORMAT
type:  ensight gold

GEOMETRY
model:          1                      exgold2.geo**

VARIABLE
scalar per node: 1      Stress          exgold2.scl**
vector per node: 1      Displacement    exgold2.dis**

TIME
time set:        1
number of steps: 3
filename start number: 0
filename increment: 1
time values:     1.0  2.0  3.0

```

The following files would be needed for Example 2:

```

exgold2.geo00      exgold2.scl00      exgold2.dis00
exgold2.geo01      exgold2.scl01      exgold2.dis01
exgold2.geo02      exgold2.scl02      exgold2.dis02

```

**Case File Example 3**    The following is a Case file for a transient model with measured data.

*This example has pressure given per element.*

```

FORMAT
type:  ensight gold

GEOMETRY
model:          1                      exgold3.geo*
measured:       2                      exgold3.mgeo**

VARIABLE
constant per case:      Gamma          1.4
constant per case:      1      Density  .9 .9 .7 .6 .6
scalar per element      1      Pressure  exgold3.pre*
vector per node:        1      Velocity  exgold3.vel*
scalar per measured node: 2      Temperature  exgold3.mtem**
vector per measured node: 2      Velocity  exgold3.mvel**

TIME
time set:              1
number of steps:       5
filename start number: 1
filename increment:    2
time values:           .1 .2 .3          # This example shows that time

```

```

                                .4 .5          # values can be on multiple lines
time set:                       2
number of steps:                 6
filename start number:          0
filename increment:              2
time values:
.05 .15 .25 .34 .45 .55

```

The following files would be needed for Example 3:

exgold3.geo1	exgold3.pre1	exgold3.vel1
exgold3.geo3	exgold3.pre3	exgold3.vel3
exgold3.geo5	exgold3.pre5	exgold3.vel5
exgold3.geo7	exgold3.pre7	exgold3.vel7
exgold3.geo9	exgold3.pre9	exgold3.vel9
exgold3.mgeo00	exgold3.mtem00	exgold3.mvel100
exgold3.mgeo02	exgold3.mtem02	exgold3.mvel102
exgold3.mgeo04	exgold3.mtem04	exgold3.mvel104
exgold3.mgeo06	exgold3.mtem06	exgold3.mvel106
exgold3.mgeo08	exgold3.mtem08	exgold3.mvel108
exgold3.mgeo10	exgold3.mtem10	exgold3.mvel110

Case File Example 4    The following is Case File Example 3 expressed in transient single-file formats.

*In this example, the transient data for the measured velocity data entity happens to be greater than the maximum file size limit. Therefore, the first four time steps fit and are contained in the first file, and the last two time steps are 'continued' in a second file.*

```

FORMAT
type: ensight gold

GEOMETRY
model:          1      1      exgold4.geo
measured:       2      2      exgold4.mgeo

VARIABLE
constant per case:          Density          .5
scalar per element:         1      1      Pressure      exgold4.pre
vector per node:            1      1      Velocity      exgold4.vel
scalar per measured node:   2      2      Temperature   exgold4.mtem
vector per measured node:   2      3      Velocity      exgold4.mvel*

TIME
time set:          1      Model
number of steps:    5
time values:       .1 .2 .3 .4 .5

time set:          2      Measured
number of steps:    6
time values:       .05 .15 .25 .34 .45 .55

FILE
file set:          1
number of steps:    5

file set:          2

```

```

number of steps:      6

file set:             3
filename index:       1
number of steps:      4
filename index:       2
number of steps:      2

```

The following files would be needed for Example 4:

```

exgold4.geo      exgold4.pre      exgold4.vel
exgold4.mgeo     exgold4.mtem     exgold4.mvel1
exgold4.mvel2

```

#### Contents of Transient Single Files

Each file contains transient data that corresponds to the specified number of time steps. The data for each time step sequentially corresponds to the simulation time values (time values) found listed in the TIME section. In transient single-file format, the data for each time step essentially corresponds to a standard EnSight gold geometry or variable file (model or measured) as expressed in multiple file format. The data for each time step is enclosed between two *wrapper* records, i.e. preceded by a BEGIN TIME STEP record and followed by an END TIME STEP record. Time step data is not split between files. If there is not enough room to append the data from a time step to the file without exceeding the maximum file limit of a particular system, then a continuation file must be created for the time step data and any subsequent time step. Any type of user comments may be included before and/or after each transient step wrapper.

*Note 1: If transient single file format is used, EnSight expects **all** files of a dataset to be specified in transient single file format. Thus, even static files must be enclosed between a BEGIN TIME STEP and an END TIME STEP wrapper. This includes the condition where you have transient variables with static geometry. The static geometry file must have the wrapper.*

- Note 2: For binary geometry files, the first BEGIN TIME STEP wrapper must follow the <C Binary/Fortran Binary> line. Both BEGIN TIME STEP and END TIME STEP wrappers are written according to type (1) in binary. Namely: This is a write of 80 characters to the file:*

```

in C: char buffer[80];
      strcpy(buffer, "BEGIN TIME STEP");
      fwrite(buffer, sizeof(char), 80, file_ptr);

```

```

in FORTRAN: character*80 buffer
            buffer = "BEGIN TIME STEP"

```

*Note 3: Efficient reading of each file (especially binary) is facilitated by appending each file with a **file index**. A file index contains appropriate information to access the file byte positions of each time step in the file. (EnSight automatically appends a file index to each file when exporting in transient single file format.) If used, the file index must follow the last END TIME STEP wrapper in each file.*

#### File Index Usage:

ASCII	Binary	Item	Description
"%20d\n"	sizeof(int)	n	Total number of data time steps in the file.
"%20d\n"	sizeof(long)	fb <sub>1</sub>	File byte loc for contents of 1 <sup>st</sup> time step*
"%20d\n"	sizeof(long)	fb <sub>2</sub>	File byte loc for contents of 2 <sup>nd</sup> time step*

ASCII	Binary	Item	Description
...	...	...	...
"%20d\n"	sizeof(long)	fb <sub>n</sub>	File byte loc for contents of n <sup>th</sup> time step *
"%20d\n"	sizeof(int)	flag	Miscellaneous flag (= 0 for now)
"%20d\n"	sizeof(long)	fb of item n	File byte loc for Item n above
"%s\n"	sizeof(char)*80	"FILE_INDEX"	File index keyword

\* Each file byte location is the first byte that follows the BEGIN TIME STEP record.

Shown below are the contents of each of the above files, using the data files from Case File Example 3 for reference (without FILE\_INDEX for simplicity).

Contents of file exgold4.geo\_1:

```
BEGIN TIME STEP
Contents of file exgold3.geo1
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.geo3
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.geo5
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.geo7
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.geo9
END TIME STEP
```

Contents of file exgold4.pre\_1:

```
BEGIN TIME STEP
Contents of file exgold3.pre1
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.pre3
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.pre5
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.pre7
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.pre9
END TIME STEP
```

Contents of file exgold4.vel\_1:

```
BEGIN TIME STEP
Contents of file exgold3.vel1
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.vel3
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.vel5
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.vel7
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.vel9
END TIME STEP
```

Contents of file exgold4.mgeo\_1:

```
BEGIN TIME STEP
Contents of file exgold3.mgeo00
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mgeo02
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mgeo04
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mgeo06
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mgeo08
```

```

END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mgeo10
END TIME STEP

Contents of file exgold4.mtem_1:
BEGIN TIME STEP
Contents of file exgold3.mtem00
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mtem02
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mtem04
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mtem06
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mtem08
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mtem10
END TIME STEP

Contents of file exgold4.mvel1_1:
BEGIN TIME STEP
Contents of file exgold3.mvel100
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mvel102
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mvel104
END TIME STEP
BEGIN TIME STEP
Contents of file exgold3.mvel106
END TIME STEP

Contents of file exgold4.mvel2_1:
Comments can precede the beginning wrapper here.

BEGIN TIME STEP
Contents of file exgold3.mvel108
END TIME STEP

Comments can go between time step wrappers here.
BEGIN TIME STEP
Contents of file exgold3.mvel110
END TIME STEP

Comments can follow the ending time step wrapper.

```

*Note: Each of these files could (and should for efficiency reasons) have the FILE\_INDEX information following the last END TIMESTEP. See the previous discussion for its usage.*

### EnSight Gold Wild Card Name Specification

If multiple time steps are involved, the file names must conform to the EnSight wild-card specification. This specification is as follows:

- File names must include numbers that are in ascending order from beginning to end.
- Numbers in the file names must be zero filled if there is more than one significant digit.
- Numbers can be anywhere in the file name.
- When the file name is specified in the EnSight case file, you must replace the numbers in the file with an asterisk(\*). The number of asterisks specified is the number of significant digits. The asterisk must occupy the same place as the numbers in the file names.



## EnSight Gold Variable File Format

EnSight Gold variable files can either be per\_node or per\_element. They cannot be both. However, an EnSight model can have some variables which are per\_node and others which are per\_element.

### EnSight Gold Per\_Node Variable File Format

EnSight Gold variable files for per\_node variables contain values for each unstructured node and for each structured node. First comes a single description line. Second comes a part line. Third comes a line containing the part number. Fourth comes a 'coordinates' line or a 'block' line. If a 'coordinates' line, the value for each unstructured node of the part follows. If it is a scalar file, there is one value per node, while for vector files there are three values per node (output in the same component order as the coordinates, namely, all x components, then all y components, then all z components). If it is a 'block' line, the value(s) for each structured node follows. The values for each node of the structured block are output in the same IJK order as the coordinates. (The number of nodes in the part are obtained from the corresponding EnSight Gold geometry file.)

*Note: If the geometry of given part is **empty**, nothing for that part needs to be in the variable file.*

#### C Binary form:

##### SCALAR FILE:

description line 1	80 chars
part	80 chars
#	1 int
coordinates	80 chars
s_n1 s_n2 ... s_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block	80 chars
s_n1 s_n2 ... s_nn	nn floats

# nn = i\*j\*k

##### VECTOR FILE:

description line 1	80 chars
part	80 chars
#	1 int
coordinates	80 chars
vx_n1 vx_n2 ... vx_nn	nn floats
vy_n1 vy_n2 ... vy_nn	nn floats
vz_n1 vz_n2 ... vz_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block	80 chars
vx_n1 vx_n2 ... vx_nn	nn floats

# nn = i\*j\*k

## 11.1 EnSight Gold Per\_Node Variable File Format

vy_n1 vy_n2 ... vy_nn	nn floats
vz_n1 vz_n2 ... vz_nn	nn floats

### TENSOR FILE:

description line 1	80 chars
part	80 chars
#	1 int
coordinates	80 chars
v11_n1 v11_n2 ... v11_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v33_n1 v33_n2 ... v33_nn	nn floats
v12_n1 v12_n2 ... v12_nn	nn floats
v13_n1 v13_n2 ... v13_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block # nn = i*j*k	80 chars
v11_n1 v11_n2 ... v11_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v33_n1 v33_n2 ... v33_nn	nn floats
v12_n1 v12_n2 ... v12_nn	nn floats
v13_n1 v13_n2 ... v13_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats

### TENSOR9 FILE:

description line 1	80 chars
part	80 chars
#	1 int
coordinates	80 chars
v11_n1 v11_n2 ... v11_nn	nn floats
v12_n1 v12_n2 ... v12_nn	nn floats
v13_n1 v13_n2 ... v13_nn	nn floats
v21_n1 v21_n2 ... v21_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats
v31_n1 v31_n2 ... v31_nn	nn floats
v32_n1 v32_n2 ... v32_nn	nn floats
v33_n1 v33_n2 ... v33_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block # nn = i*j*k	80 chars
v11_n1 v11_n2 ... v11_nn	nn floats
v12_n1 v12_n2 ... v12_nn	nn floats
v13_n1 v13_n2 ... v13_nn	nn floats
v21_n1 v21_n2 ... v21_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats
v21_n1 v21_n2 ... v21_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats

### COMPLEX SCALAR FILES (Real and/or Imaginary):

description line 1	80 chars
part	80 chars
#	1 int
coordinates	80 chars
s_n1 s_n2 ... s_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = i*j*k
s_n1 s_n2 ... s_nn	nn floats

**COMPLEX VECTOR FILES (Real and/or Imaginary):**

description line 1	80 chars
part	80 chars
#	1 int
coordinates	80 chars
vx_n1 vx_n2 ... vx_nn	nn floats
vy_n1 vy_n2 ... vy_nn	nn floats
vz_n1 vz_n2 ... vz_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = i*j*k
vx_n1 vx_n2 ... vx_nn	nn floats
vy_n1 vy_n2 ... vy_nn	nn floats
vz_n1 vz_n2 ... vz_nn	nn floats

**Fortran Binary form:****SCALAR FILE:**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates'	80 chars
's_n1 s_n2 ... s_nn'	nn floats
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block'	# nn = i*j*k
's_n1 s_n2 ... s_nn'	nn floats

**VECTOR FILE:**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates'	80 chars
'vx_n1 vx_n2 ... vx_nn'	nn floats

'vy_n1 vy_n2 ... vy_nn'	nn floats
'vz_n1 vz_n2 ... vz_nn'	nn floats
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block' # nn = i*j*k	80 chars
'vx_n1 vx_n2 ... vx_nn'	nn floats
'vy_n1 vy_n2 ... vy_nn'	nn floats
'vz_n1 vz_n2 ... vz_nn'	nn floats

#### TENSOR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates'	80 chars
'v11_n1 v11_n2 ... v11_nn'	nn floats
'v22_n1 v22_n2 ... v22_nn'	nn floats
'v33_n1 v33_n2 ... v33_nn'	nn floats
'v12_n1 v12_n2 ... v12_nn'	nn floats
'v13_n1 v13_n2 ... v13_nn'	nn floats
'v23_n1 v23_n2 ... v23_nn'	nn floats
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block' # nn = i*j*k	80 chars
'v11_n1 v11_n2 ... v11_nn'	nn floats
'v22_n1 v22_n2 ... v22_nn'	nn floats
'v33_n1 v33_n2 ... v33_nn'	nn floats
'v12_n1 v12_n2 ... v12_nn'	nn floats
'v13_n1 v13_n2 ... v13_nn'	nn floats
'v23_n1 v23_n2 ... v23_nn'	nn floats

#### TENSOR9 FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates'	80 chars
'v11_n1 v11_n2 ... v11_nn'	nn floats
'v12_n1 v12_n2 ... v12_nn'	nn floats
'v13_n1 v13_n2 ... v13_nn'	nn floats
'v21_n1 v21_n2 ... v21_nn'	nn floats
'v22_n1 v22_n2 ... v22_nn'	nn floats
'v23_n1 v23_n2 ... v23_nn'	nn floats
'v31_n1 v31_n2 ... v31_nn'	nn floats
'v32_n1 v32_n2 ... v32_nn'	nn floats
'v33_n1 v33_n2 ... v33_nn'	nn floats
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block' # nn = i*j*k	80 chars
'v11_n1 v11_n2 ... v11_nn'	nn floats
'v12_n1 v12_n2 ... v12_nn'	nn floats

'v13_n1 v13_n2 ... v13_nn'	nn floats
'v21_n1 v21_n2 ... v21_nn'	nn floats
'v22_n1 v22_n2 ... v22_nn'	nn floats
'v23_n1 v23_n2 ... v23_nn'	nn floats
'v31_n1 v31_n2 ... v31_nn'	nn floats
'v32_n1 v32_n2 ... v32_nn'	nn floats
'v33_n1 v33_n2 ... v33_nn'	nn floats

**COMPLEX SCALAR FILES (Real and/or Imaginary):**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates'	80 chars
's_n1 s_n2 ... s_nn'	nn floats
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block' # nn = i*j*k	80 chars
's_n1 s_n2 ... s_nn'	nn floats

**COMPLEX VECTOR FILES (Real and/or Imaginary):**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates'	80 chars
'vx_n1 vx_n2 ... vx_nn'	nn floats
'vy_n1 vy_n2 ... vy_nn'	nn floats
'vz_n1 vz_n2 ... vz_nn'	nn floats
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block' # nn = i*j*k	80 chars
'vx_n1 vx_n2 ... vx_nn'	nn floats
'vy_n1 vy_n2 ... vy_nn'	nn floats
'vz_n1 vz_n2 ... vz_nn'	nn floats

**ASCII form:****SCALAR FILE:**

description line 1	A (max of 79 typ)
part	A
#	I10
coordinates	A
s_n1	E12.5 1/line (nn)
s_n2	
.	
.	
s_nn	
part	A
.	

.		
part		A
#		I10
block	# nn = i*j*k	A
s_n1		E12.5 1/line (nn)
s_n2		
.		
.		
s_nn		

## VECTOR FILE:

description line 1		A (max of 79 typ)
part		A
#		I10
coordinates		A
vx_n1		E12.5 1/line (nn)
vx_n2		
.		
.		
vx_nn		
vy_n1		E12.5 1/line (nn)
vy_n2		
.		
.		
vy_nn		
vz_n1		E12.5 1/line (nn)
vz_n2		
.		
.		
vz_nn		
part		A
.		
.		
part		A
#		I10
block	# nn = i*j*k	A
vx_n1		E12.5 1/line (nn)
vx_n2		
.		
.		
vx_nn		
vy_n1		E12.5 1/line (nn)
vy_n2		
.		
.		
vy_nn		
vz_n1		E12.5 1/line (nn)
vz_n2		
.		
.		
vz_nn		

## TENSOR FILE:

description line 1		A (max of 79 typ)
part		A
#		I10
coordinates		A

v11_n1		E12.5	1/line (nn)
v11_n2			
.			
.			
v11_nn			
v22_n1		E12.5	1/line (nn)
v22_n2			
.			
.			
v22_nn			
v33_n1		E12.5	1/line (nn)
v33_n2			
.			
.			
v33_nn			
v12_n1		E12.5	1/line (nn)
v12_n2			
.			
.			
v12_nn			
v13_n1		E12.5	1/line (nn)
v13_n2			
.			
.			
v13_nn			
v23_n1		E12.5	1/line (nn)
v23_n2			
.			
.			
v23_nn			
part		A	
.			
.			
part		A	
#		I10	
block	# nn = i*j*k	A	
v11_n1		E12.5	1/line (nn)
v11_n2			
.			
.			
v11_nn			
v22_n1		E12.5	1/line (nn)
v22_n2			
.			
.			
v22_nn			
v33_n1		E12.5	1/line (nn)
v33_n2			
.			
.			
v33_nn			
v12_n1		E12.5	1/line (nn)
v12_n2			
.			
.			
v12_nn			
v13_n1		E12.5	1/line (nn)
v13_n2			
.			
.			
v13_nn			

## 11.1 EnSight Gold Per\_Node Variable File Format

v23_n1	E12.5	1/line (nn)
v23_n2		
.		
.		
v23_nn		

### TENSOR9 FILE:

description line 1	A (max of 79 typ)
part	A
#	I10
coordinates	A
v11_n1	E12.5 1/line (nn)
v11_n2	
.	
.	
v11_nn	
v12_n1	E12.5 1/line (nn)
v12_n2	
.	
.	
v12_nn	
v13_n1	E12.5 1/line (nn)
v13_n2	
.	
.	
v13_nn	
v21_n1	E12.5 1/line (nn)
v21_n2	
.	
.	
v21_nn	
v22_n1	E12.5 1/line (nn)
v22_n2	
.	
.	
v22_nn	
v23_n1	E12.5 1/line (nn)
v23_n2	
.	
.	
v23_nn	
v31_n1	E12.5 1/line (nn)
v31_n2	
.	
.	
v31_nn	
v32_n1	E12.5 1/line (nn)
v32_n2	
.	
.	
v32_nn	
v33_n1	E12.5 1/line (nn)
v33_n2	
.	
.	
v33_nn	
part	A
.	
.	
part	A



```

#                                I10
block                            A
                                # nn = i*j*k
v11_n1                          E12.5  1/line (nn)
v11_n2
.
.
v11_nn
v12_n1                          E12.5  1/line (nn)
v12_n2
.
.
v12_nn
v13_n1                          E12.5  1/line (nn)
v13_n2
.
.
v13_nn
v21_n1                          E12.5  1/line (nn)
v21_n2
.
.
v21_nn
v22_n1                          E12.5  1/line (nn)
v22_n2
.
.
v22_nn
v23_n1                          E12.5  1/line (nn)
v23_n2
.
.
v23_nn
v31_n1                          E12.5  1/line (nn)
v31_n2
.
.
v31_nn
v32_n1                          E12.5  1/line (nn)
v32_n2
.
.
v32_nn
v33_n1                          E12.5  1/line (nn)
v33_n2
.
.
v33_nn

```

#### COMPLEX SCALAR FILES (Real and/or Imaginary):

```

description line 1              A (max of 79 typ)
part                            A
#                                I10
coordinates                     A
s_n1                            E12.5  1/line (nn)
s_n2
.
.
s_nn
part                            A

```

```

.
.
part
#
block
s_n1
s_n2
.
.
s_nn

```

A  
I10  
A  
E12.5 1/line (nn)

#### COMPLEX VECTOR FILES (Real and/or Imaginary):

```

description line 1
part
#
coordinates
vx_n1
vx_n2
.
.
vx_nn
vy_n1
vy_n2
.
.
vy_nn
vz_n1
vz_n2
.
.
vz_nn
part
.
.
part
#
block
vx_n1
vx_n2
.
.
vx_nn
vy_n1
vy_n2
.
.
vy_nn
vz_n1
vz_n2
.
.
vz_nn

```

A (max of 79 typ)  
A  
I10  
A  
E12.5 1/line (nn)  
  
E12.5 1/line (nn)  
  
E12.5 1/line (nn)  
  
A  
  
A  
I10  
A  
E12.5 1/line (nn)  
  
E12.5 1/line (nn)  
  
E12.5 1/line (nn)

The following variable file examples reflect scalar, vector, tensor, and complex variable values *per node* for the previously defined EnSight6 Gold Geometry File Example with 11 defined unstructured nodes and a 2x3x2 structured Part (Part number 3). The values are summarized in the following table.

*Note: These are the same values as listed in the EnSight6 per\_node variable file section. Subsequently, the following example files contain the same data as the example files given in the EnSight6 section - only they are listed in gold format. (No asymmetric tensor example data given)*

						ComplexScalar	
	Node	Node	Scalar	Vector	Tensor (2nd order symm.)	Real	Imaginary
	Index	Id	Value	Values	Values	Value	Value
Unstructured	1	15	(1.)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.3, 1.4, 1.5, 1.6)	(1.1)	(1.2)
	2	31	(2.)	(2.1, 2.2, 2.3)	(2.1, 2.2, 2.3, 2.4, 2.5, 2.6)	(2.1)	(2.2)
	3	20	(3.)	(3.1, 3.2, 3.3)	(3.1, 3.2, 3.3, 3.4, 3.5, 3.6)	(3.1)	(3.2)
	4	40	(4.)	(4.1, 4.2, 4.3)	(4.1, 4.2, 4.3, 4.4, 4.5, 4.6)	(4.1)	(4.2)
	5	22	(5.)	(5.1, 5.2, 5.3)	(5.1, 5.2, 5.3, 5.4, 5.5, 5.6)	(5.1)	(5.2)
	6	44	(6.)	(6.1, 6.2, 6.3)	(6.1, 6.2, 6.3, 6.4, 6.5, 6.6)	(6.1)	(6.2)
	7	55	(7.)	(7.1, 7.2, 7.3)	(7.1, 7.2, 7.3, 7.4, 7.5, 7.6)	(7.1)	(7.2)
	8	60	(8.)	(8.1, 8.2, 8.3)	(8.1, 8.2, 8.3, 8.4, 8.5, 8.6)	(8.1)	(8.2)
	9	61	(9.)	(9.1, 9.2, 9.3)	(9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	(9.1)	(9.2)
	10	62	(10.)	(10.1,10.2,10.3)	(10.1,10.2,10.3,10.4,10.5,10.6)	(10.1)	(10.2)
	11	63	(11.)	(11.1,11.2,11.3)	(11.1,11.2,11.3,11.4,11.5,11.6)	(11.1)	(11.2)
Structured	1	1	(1.)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.3, 1.4, 1.5, 1.6)	(1.1)	(1.2)
	2	2	(2.)	(2.1, 2.2, 2.3)	(2.1, 2.2, 2.3, 2.4, 2.5, 2.6)	(2.1)	(2.2)
	3	3	(3.)	(3.1, 3.2, 3.3)	(3.1, 3.2, 3.3, 3.4, 3.5, 3.6)	(3.1)	(3.2)
	4	4	(4.)	(4.1, 4.2, 4.3)	(4.1, 4.2, 4.3, 4.4, 4.5, 4.6)	(4.1)	(4.2)
	5	5	(5.)	(5.1, 5.2, 5.3)	(5.1, 5.2, 5.3, 5.4, 5.5, 5.6)	(5.1)	(5.2)
	6	6	(6.)	(6.1, 6.2, 6.3)	(6.1, 6.2, 6.3, 6.4, 6.5, 6.6)	(6.1)	(6.2)
	7	7	(7.)	(7.1, 7.2, 7.3)	(7.1, 7.2, 7.3, 7.4, 7.5, 7.6)	(7.1)	(7.2)
	8	8	(8.)	(8.1, 8.2, 8.3)	(8.1, 8.2, 8.3, 8.4, 8.5, 8.6)	(8.1)	(8.2)
	9	9	(9.)	(9.1, 9.2, 9.3)	(9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	(9.1)	(9.2)
	10	10	(10.)	(10.1,10.2,10.3)	(10.1,10.2,10.3,10.4,10.5,10.6)	(10.1)	(10.2)
	11	11	(11.)	(11.1,11.2,11.3)	(11.1,11.2,11.3,11.4,11.5,11.6)	(11.1)	(11.2)
	12	12	(12.)	(12.1,12.2,12.3)	(12.1,12.2,12.3,12.4,12.5,12.6)	(12.1)	(12.2)

**Per\_node (Scalar) Variable Example 1:** This shows an ASCII scalar file (engold.Nsca) for the gold geometry example.

```

Per_node scalar values for the EnSight Gold geometry example
part
  1
coordinates
1.00000E+00
3.00000E+00
4.00000E+00
5.00000E+00
6.00000E+00
7.00000E+00
8.00000E+00
9.00000E+00
1.00000E+01
1.10000E+01
part
  2
coordinates
1.00000E+00
2.00000E+00

```

```

part
    3
block
    1.00000E+00
    2.00000E+00
    3.00000E+00
    4.00000E+00
    5.00000E+00
    6.00000E+00
    7.00000E+00
    8.00000E+00
    9.00000E+00
    1.00000E+01
    1.10000E+01
    1.20000E+01

```

**Per\_node (Vector) Variable Example 2:** This example shows an ASCII vector file (`engold.Nvec`) for the gold geometry example.

```

Per_node vector values for the EnSight Gold geometry example
part
    1
coordinates
    1.10000E+00
    3.10000E+00
    4.10000E+00
    5.10000E+00
    6.10000E+00
    7.10000E+00
    8.10000E+00
    9.10000E+00
    1.01000E+01
    1.11000E+01
    1.20000E+00
    3.20000E+00
    4.20000E+00
    5.20000E+00
    6.20000E+00
    7.20000E+00
    8.20000E+00
    9.20000E+00
    1.02000E+01
    1.12000E+01
    1.30000E+00
    3.30000E+00
    4.30000E+00
    5.30000E+00
    6.30000E+00
    7.30000E+00
    8.30000E+00
    9.30000E+00
    1.03000E+01
    1.13000E+01
part
    2
coordinates
    1.10000E+00
    2.10000E+00
    1.20000E+00
    2.20000E+00
    1.30000E+00
    2.30000E+00
part
    3
block
    1.10000E+00

```

```

2.10000E+00
3.10000E+00
4.10000E+00
5.10000E+00
6.10000E+00
7.10000E+00
8.10000E+00
9.10000E+00
1.01000E+01
1.11000E+01
1.21000E+01
1.20000E+00
2.20000E+00
3.20000E+00
4.20000E+00
5.20000E+00
6.20000E+00
7.20000E+00
8.20000E+00
9.20000E+00
1.02000E+01
1.12000E+01
1.22000E+01
1.30000E+00
2.30000E+00
3.30000E+00
4.30000E+00
5.30000E+00
6.30000E+00
7.30000E+00
8.30000E+00
9.30000E+00
1.03000E+01
1.13000E+01
1.23000E+01

```

**Per\_node (Tensor) Variable Example 3:** This example shows an ASCII 2nd order symmetric tensor file (engold.Nten) for the gold geometry example.

Per\_node symmetric tensor values for the EnSight Gold geometry example  
part

```

1
coordinates
1.10000E+00
3.10000E+00
4.10000E+00
5.10000E+00
6.10000E+00
7.10000E+00
8.10000E+00
9.10000E+00
1.01000E+01
1.11000E+01
1.20000E+00
3.20000E+00
4.20000E+00
5.20000E+00
6.20000E+00
7.20000E+00
8.20000E+00
9.20000E+00
1.02000E+01
1.12000E+01
1.30000E+00
3.30000E+00
4.30000E+00

```

## 11.1 EnSight Gold Per\_Node Variable File Format

```
5.30000E+00
6.30000E+00
7.30000E+00
8.30000E+00
9.30000E+00
1.03000E+01
1.13000E+01
1.40000E+00
3.40000E+00
4.40000E+00
5.40000E+00
6.40000E+00
7.40000E+00
8.40000E+00
9.40000E+00
1.04000E+01
1.14000E+01
1.50000E+00
3.50000E+00
4.50000E+00
5.50000E+00
6.50000E+00
7.50000E+00
8.50000E+00
9.50000E+00
1.05000E+01
1.15000E+01
1.60000E+00
3.60000E+00
4.60000E+00
5.60000E+00
6.60000E+00
7.60000E+00
8.60000E+00
9.60000E+00
1.06000E+01
1.16000E+01
part
  2
coordinates
  1.10000E+00
  2.10000E+00
  1.20000E+00
  2.20000E+00
  1.30000E+00
  2.30000E+00
  1.40000E+00
  2.40000E+00
  1.50000E+00
  2.50000E+00
  1.60000E+00
  2.60000E+00
part
  3
block
  1.10000E+00
  2.10000E+00
  3.10000E+00
  4.10000E+00
  5.10000E+00
  6.10000E+00
  7.10000E+00
  8.10000E+00
  9.10000E+00
  1.01000E+01
  1.11000E+01
  1.21000E+01
```

1.20000E+00  
2.20000E+00  
3.20000E+00  
4.20000E+00  
5.20000E+00  
6.20000E+00  
7.20000E+00  
8.20000E+00  
9.20000E+00  
1.02000E+01  
1.12000E+01  
1.22000E+01  
1.30000E+00  
2.30000E+00  
3.30000E+00  
4.30000E+00  
5.30000E+00  
6.30000E+00  
7.30000E+00  
8.30000E+00  
9.30000E+00  
1.03000E+01  
1.13000E+01  
1.23000E+01  
1.40000E+00  
2.40000E+00  
3.40000E+00  
4.40000E+00  
5.40000E+00  
6.40000E+00  
7.40000E+00  
8.40000E+00  
9.40000E+00  
1.04000E+01  
1.14000E+01  
1.24000E+01  
1.50000E+00  
2.50000E+00  
3.50000E+00  
4.50000E+00  
5.50000E+00  
6.50000E+00  
7.50000E+00  
8.50000E+00  
9.50000E+00  
1.05000E+01  
1.15000E+01  
1.25000E+01  
1.60000E+00  
2.60000E+00  
3.60000E+00  
4.60000E+00  
5.60000E+00  
6.60000E+00  
7.60000E+00  
8.60000E+00  
9.60000E+00  
1.06000E+01  
1.16000E+01  
1.26000E+01

**Per\_node (Complex) Variable Example 4:** This example shows ASCII complex real (`engold.Ncmp_r`) and imaginary (`engold.Ncmp_i`) *scalar* files for the gold geometry example. (The same methodology would apply for complex real and imaginary *vector* files.)

Real scalar File:

```
Per_node complex real scalar values for the EnSight Gold geometry example
part
    1
coordinates
    1.10000E+00
    3.10000E+00
    4.10000E+00
    5.10000E+00
    6.10000E+00
    7.10000E+00
    8.10000E+00
    9.10000E+00
    1.01000E+01
    1.11000E+01
part
    2
coordinates
    1.10000E+00
    2.10000E+00
part
    3
block
    1.10000E+00
    2.10000E+00
    3.10000E+00
    4.10000E+00
    5.10000E+00
    6.10000E+00
    7.10000E+00
    8.10000E+00
    9.10000E+00
    1.01000E+01
    1.11000E+01
    1.21000E+01
```

Imaginary scalar File:

```
Per_node complex imaginary scalar values for the EnSight Gold geometry example
part
    1
coordinates
    1.20000E+00
    3.20000E+00
    4.20000E+00
    5.20000E+00
    6.20000E+00
    7.20000E+00
    8.20000E+00
    9.20000E+00
    1.02000E+01
    1.12000E+01
part
    2
coordinates
    1.20000E+00
    2.20000E+00
part
    3
block
```



```

1.20000E+00
2.20000E+00
3.20000E+00
4.20000E+00
5.20000E+00
6.20000E+00
7.20000E+00
8.20000E+00
9.20000E+00
1.02000E+01
1.12000E+01
1.22000E+01

```

## EnSight Gold Per\_Element Variable File Format

EnSight Gold variable files for per\_element variables contain values for each element of designated types of designated Parts. First comes a single description line. Second comes a Part line. Third comes a line containing the part number. Fourth comes an element type line and then comes the value for each element of that type and part. If it is a scalar variable, there is one value per element, while for vector variables there are three values per element. (The number of elements of the given type are obtained from the corresponding EnSight Gold geometry file.)

*Note: If the geometry of given part is **empty**, nothing for that part needs to be in the variable file.*

### C Binary form:

#### SCALAR FILE:

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
s_e1 s_e2 ... s_ne	ne floats
element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = (i-1)*(j-1)*(k-1)
s_n1 s_n2 ... s_nn	nn floats

#### VECTOR FILE:

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
vx_e1 vx_e2 ... vx_ne	ne floats
vy_e1 vy_e2 ... vy_ne	ne floats
vz_e1 vz_e2 ... vz_ne	ne floats

element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = (i-1)*(j-1)*(k-1) 80 chars
vx_n1 vx_n2 ... vx_nn	nn floats
vy_n1 vy_n2 ... vy_nn	nn floats
vz_n1 vz_n2 ... vz_nn	nn floats

**TENSOR FILE:**

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
v11_e1 v11_e2 ... v11_ne	ne floats
v22_e1 v22_e2 ... v22_ne	ne floats
v33_e1 v33_e2 ... v33_ne	ne floats
v12_e1 v12_e2 ... v12_ne	ne floats
v13_e1 v13_e2 ... v13_ne	ne floats
v23_e1 v23_e2 ... v23_ne	ne floats
element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = (i-1)*(j-1)*(k-1) 80 chars
v11_n1 v11_n2 ... v11_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v33_n1 v33_n2 ... v33_nn	nn floats
v12_n1 v12_n2 ... v12_nn	nn floats
v13_n1 v13_n2 ... v13_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats

**TENSOR9 FILE:**

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
v11_e1 v11_e2 ... v11_ne	ne floats
v12_e1 v12_e2 ... v12_ne	ne floats
v13_e1 v13_e2 ... v13_ne	ne floats
v21_e1 v21_e2 ... v21_ne	ne floats
v22_e1 v22_e2 ... v22_ne	ne floats
v23_e1 v23_e2 ... v23_ne	ne floats
v31_e1 v31_e2 ... v31_ne	ne floats
v32_e1 v32_e2 ... v32_ne	ne floats
v33_e1 v33_e2 ... v33_ne	ne floats
element type	80 chars
.	
.	

part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = (i-1)*(j-1)*(k-1) 80 chars
v11_n1 v11_n2 ... v11_nn	nn floats
v12_n1 v12_n2 ... v12_nn	nn floats
v13_n1 v13_n2 ... v13_nn	nn floats
v21_n1 v21_n2 ... v21_nn	nn floats
v22_n1 v22_n2 ... v22_nn	nn floats
v23_n1 v23_n2 ... v23_nn	nn floats
v31_n1 v31_n2 ... v31_nn	nn floats
v32_n1 v32_n2 ... v32_nn	nn floats
v33_n1 v33_n2 ... v33_nn	nn floats

**COMPLEX SCALAR FILES (Real and/or Imaginary):**

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
s_e1 s_e2 ... s_ne	ne floats
element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = (i-1)*(j-1)*(k-1) 80 chars
s_n1 s_n2 ... s_nn	nn floats

**COMPLEX VECTOR FILES (Real and/or Imaginary):**

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
vx_e1 vx_e2 ... vx_ne	ne floats
vy_e1 vy_e2 ... vy_ne	ne floats
vz_e1 vz_e2 ... vz_ne	ne floats
element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block	# nn = (i-1)*(j-1)*(k-1) 80 chars
vx_n1 vx_n2 ... vx_nn	nn floats
vy_n1 vy_n2 ... vy_nn	nn floats
vz_n1 vz_n2 ... vz_nn	nn floats

**Fortran Binary form:**

SCALAR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'element type'	80 chars
's_e1 s_e2 ... s_ne'	ne floats
'element type'	80 chars
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block'                      # nn = (i-1)*(j-1)*(k-1)	80 chars
's_n1 s_n2 ... s_nn'	nn floats

### VECTOR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'element type'	80 chars
'vx_e1 vx_e2 ... vx_ne'	ne floats
'vy_e1 vy_e2 ... vy_ne'	ne floats
'vz_e1 vz_e2 ... vz_ne'	ne floats
'element type'	80 chars
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block'                      # nn = (i-1)*(j-1)*(k-1)	80 chars
'vx_n1 vx_n2 ... vx_nn'	nn floats
'vy_n1 vy_n2 ... vy_nn'	nn floats
'vz_n1 vz_n2 ... vz_nn'	nn floats

### TENSOR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'element type'	80 chars
'v11_e1 v11_e2 ... v11_ne'	ne floats
'v22_e1 v22_e2 ... v22_ne'	ne floats
'v33_e1 v33_e2 ... v33_ne'	ne floats
'v12_e1 v12_e2 ... v12_ne'	ne floats
'v13_e1 v13_e2 ... v13_ne'	ne floats
'v23_e1 v23_e2 ... v23_ne'	ne floats
'element type'	80 chars
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int

'block'	# nn = (i-1)*(j-1)*(k-1)	80 chars
'v11_n1 v11_n2 ... v11_nn'		nn floats
'v22_n1 v22_n2 ... v22_nn'		nn floats
'v33_n1 v33_n2 ... v33_nn'		nn floats
'v12_n1 v12_n2 ... v12_nn'		nn floats
'v13_n1 v13_n2 ... v13_nn'		nn floats
'v23_n1 v23_n2 ... v23_nn'		nn floats

**TENSOR9 FILE:**

'description line 1'		80 chars
'part'		80 chars
'#'		1 int
'element type'		80 chars
'v11_e1 v11_e2 ... v11_ne'		ne floats
'v12_e1 v12_e2 ... v12_ne'		ne floats
'v13_e1 v13_e2 ... v13_ne'		ne floats
'v21_e1 v21_e2 ... v21_ne'		ne floats
'v22_e1 v22_e2 ... v22_ne'		ne floats
'v23_e1 v23_e2 ... v23_ne'		ne floats
'v31_e1 v31_e2 ... v31_ne'		ne floats
'v32_e1 v32_e2 ... v32_ne'		ne floats
'v33_e1 v33_e2 ... v33_ne'		ne floats
'element type'		80 chars
.		
.		
'part'		80 chars
.		
.		
'part'		80 chars
'#'		1 int
'block'	# nn = (i-1)*(j-1)*(k-1)	80 chars
'v11_n1 v11_n2 ... v11_nn'		nn floats
'v12_n1 v12_n2 ... v12_nn'		nn floats
'v13_n1 v13_n2 ... v13_nn'		nn floats
'v21_n1 v21_n2 ... v21_nn'		nn floats
'v22_n1 v22_n2 ... v22_nn'		nn floats
'v23_n1 v23_n2 ... v23_nn'		nn floats
'v31_n1 v31_n2 ... v31_nn'		nn floats
'v32_n1 v32_n2 ... v32_nn'		nn floats
'v33_n1 v33_n2 ... v33_nn'		nn floats

**COMPLEX SCALAR FILES (Real and/or Imaginary):**

'description line 1'		80 chars
'part'		80 chars
'#'		1 int
'element type'		80 chars
's_e1 s_e2 ... s_ne'		ne floats
'element type'		80 chars
.		
.		
'part'		80 chars
.		
.		
'part'		80 chars
'#'		1 int
'block'	# nn = (i-1)*(j-1)*(k-1)	80 chars

<code>'s_n1 s_n2 ... s_nn'</code>	nn floats
-----------------------------------	-----------

**COMPLEX VECTOR FILES (Real and/or Imaginary):**

<code>'description line 1'</code>	80 chars
<code>'part'</code>	80 chars
<code>'#'</code>	1 int
<code>'element type'</code>	80 chars
<code>'vx_e1 vx_e2 ... vx_ne'</code>	ne floats
<code>'vy_e1 vy_e2 ... vy_ne'</code>	ne floats
<code>'vz_e1 vz_e2 ... vz_ne'</code>	ne floats
<code>'element type'</code>	80 chars
<code>.</code>	
<code>.</code>	
<code>'part'</code>	80 chars
<code>.</code>	
<code>.</code>	
<code>'part'</code>	80 chars
<code>'#'</code>	1 int
<code>'block'</code>	# nn = (i-1)*(j-1)*(k-1)
<code>'vx_n1 vx_n2 ... vx_nn'</code>	nn floats
<code>'vy_n1 vy_n2 ... vy_nn'</code>	nn floats
<code>'vz_n1 vz_n2 ... vz_nn'</code>	nn floats

**ASCII form:****SCALAR FILE:**

description line 1	A (max of 80 typ)
part	A
#	I10
element type	A
s_e1	12.5 1/line (ne)
s_e2	
.	
.	
s_ne	
element type	A
.	
.	
part	A
.	
.	
part	A
#	I10
block	# nn = (i-1)*(j-1)*(k-1)
s_n1	A
s_n2	E12.5 1/line (nn)
.	
.	
s_nn	

**VECTOR FILE:**

description line 1	A (max of 80 typ)
part	A
#	I10
element type	A

vx_e1		E12.5	1/line (ne)
vx_e2			
.			
.			
vx_ne			
vy_e1		E12.5	1/line (ne)
vy_e2			
.			
.			
vy_ne			
vz_e1		E12.5	1/line (ne)
vz_e2			
.			
.			
vz_ne			
element type		A	
.			
.			
part		A	
.			
.			
part		A	
#		I10	
block	# nn = (i-1)*(j-1)*(k-1)	A	
vx_n1		E12.5	1/line (nn)
vx_n2			
.			
.			
vx_nn			
vy_n1		E12.5	1/line (nn)
vy_n2			
.			
.			
vy_nn			
vz_n1		E12.5	1/line (nn)
vz_n2			
.			
.			
vz_nn			

**TENSOR FILE:**

description line 1		A (max of 80 typ)
part		A
#		I10
element type		A
v11_e1		E12.5
v11_e2		1/line (ne)
.		
.		
v11_ne		
v22_e1		E12.5
v22_e2		1/line (ne)
.		
.		
v22_ne		
v33_e1		E12.5
v33_e2		1/line (ne)
.		
.		
v33_ne		

11.1 EnSight Gold Per\_Element Variable File Format

v12_e1		E12.5	1/line (ne)
v12_e2			
.			
.			
v12_ne			
v13_e1		E12.5	1/line (ne)
v13_e2			
.			
.			
v13_ne			
v23_e1		E12.5	1/line (ne)
v23_e2			
.			
.			
v23_ne			
element type		A	
.			
.			
part		A	
.			
.			
part		A	
#		I10	
block	# nn = (i-1)*(j-1)*(k-1)	A	
v11_n1		E12.5	1/line (nn)
v11_n2			
.			
.			
v11_nn			
v22_n1		E12.5	1/line (nn)
v22_n2			
.			
.			
v22_nn			
v33_n1		E12.5	1/line (nn)
v33_n2			
.			
.			
v33_nn			
v12_n1		E12.5	1/line (nn)
v12_n2			
.			
.			
v12_nn			
v13_n1		E12.5	1/line (nn)
v13_n2			
.			
.			
v13_nn			
v23_n1		E12.5	1/line (nn)
v23_n2			
.			
.			
v23_nn			

TENSOR9 FILE:

description line 1	A (max of 80 typ)
part	A
#	I10
element type	A



v11_e1		E12.5	1/line (ne)
v11_e2			
.			
.			
v11_ne			
v12_e1		E12.5	1/line (ne)
v12_e2			
.			
.			
v12_ne			
v13_e1		E12.5	1/line (ne)
v13_e2			
.			
.			
v13_ne			
v21_e1		E12.5	1/line (ne)
v21_e2			
.			
.			
v21_ne			
v22_e1		E12.5	1/line (ne)
v22_e2			
.			
.			
v22_ne			
v23_e1		E12.5	1/line (ne)
v23_e2			
.			
.			
v23_ne			
v31_e1		E12.5	1/line (ne)
v31_e2			
.			
.			
v31_ne			
v32_e1		E12.5	1/line (ne)
v32_e2			
.			
.			
v32_ne			
v33_e1		E12.5	1/line (ne)
v33_e2			
.			
.			
v33_ne			
element type		A	
.			
.			
part		A	
.			
.			
part		A	
#		I10	
block	# nn = (i-1)*(j-1)*(k-1)	A	
v11_n1		E12.5	1/line (nn)
v11_n2			
.			
.			
v11_nn			
v12_n1		E12.5	1/line (nn)
v12_n2			

11.1 EnSight Gold Per\_Element Variable File Format

```

      .
      .
v12_nn
v13_n1      E12.5  1/line  (nn)
v13_n2
      .
      .
v13_nn
v21_n1      E12.5  1/line  (nn)
v21_n2
      .
      .
v21_nn
v22_n1      E12.5  1/line  (nn)
v22_n2
      .
      .
v22_nn
v23_n1      E12.5  1/line  (nn)
v23_n2
      .
      .
v23_nn
v31_n1      E12.5  1/line  (nn)
v31_n2
      .
      .
v31_nn
v32_n1      E12.5  1/line  (nn)
v32_n2
      .
      .
v32_nn
v33_n1      E12.5  1/line  (nn)
v33_n2
      .
      .
v33_nn
```

COMPLEX SCALAR FILES (Real and/or Imaginary):

```

description line 1      A (max of 80 typ)
part                   A
#                      I10
element type           A
s_e1                   12.5  1/line  (ne)
s_e2
.
.
s_ne
element type           A
.
.
part                   A
.
.
part                   A
#                      I10
block                  # nn = (i-1)*(j-1)*(k-1)  A
```

s_n1	E12.5	1/line (nn)
s_n2		
.		
.		
s_nn		

#### COMPLEX VECTOR FILES (Real and/or Imaginary):

description line 1		A (max of 80 typ)
part		A
#		I10
element type		A
vx_e1	E12.5	1/line (ne)
vx_e2		
.		
.		
vx_ne		
vy_e1	E12.5	1/line (ne)
vy_e2		
.		
.		
vy_ne		
vz_e1	E12.5	1/line (ne)
vz_e2		
.		
.		
vz_ne		
element type		A
.		
.		
part		A
.		
.		
part		A
#		I10
block	# nn = (i-1)*(j-1)*(k-1)	A
vx_n1	E12.5	1/line (nn)
vx_n2		
.		
.		
vx_nn		
vy_n1	E12.5	1/line (nn)
vy_n2		
.		
.		
vy_nn		
vz_n1	E12.5	1/line (nn)
vz_n2		
.		
.		
vz_nn		

The following variable file examples reflect scalar, vector, tensor, and complex variable values *per element* for the previously defined EnSight Gold Geometry File Example with 11 defined unstructured nodes and a 2x3x2 structured Part (Part number 3). The values are summarized in the following table

*Note: These are the same values as listed in the EnSight6 per\_element variable file section. Subsequently, the following example files contain the same data as the example files in the EnSight6 section - only they are listed in gold format. (No asymmetric tensor example data given)*

							Complex Scalar	
		Element	Element	Scalar	Vector	Tensor (2nd order symm.)	Real	Imaginary
		Index	Id	Value	Values	Values	Value	Value
Unstructured								
bar2								
	1	101	(1.)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.3, 1.4, 1.5, 1.6)	(1.1)	(1.2)	
tria3								
	1	102	(2.)	(2.1, 2.2, 2.3)	(2.1, 2.2, 2.3, 2.4, 2.5, 2.6)	(2.1)	(2.2)	
	2	103	(3.)	(3.1, 3.2, 3.3)	(3.1, 3.2, 3.3, 3.4, 3.5, 3.6)	(3.1)	(3.2)	
hexa8								
	1	104	(4.)	(4.1, 4.2, 4.3)	(4.1, 4.2, 4.3, 4.4, 4.5, 4.6)	(4.1)	(4.2)	
Structured								
block	1	1	(5.)	(5.1, 5.2, 5.3)	(5.1, 5.2, 5.3, 5.4, 5.5, 5.6)	(5.1)	(5.2)	

**Per\_element (Scalar) Variable Example 1:** This example shows an ASCII scalar file (`engold.Esca`) for the gold geometry example.

```
Per_elem scalar values for the EnSight Gold geometry example
part
    1
    tria3
    2.00000E+00
    3.00000E+00
    hexa8
    4.00000E+00
part
    2
    bar2
    1.00000E+00
part
    3
    block
    5.00000E+00
    6.00000E+00
```

**Per\_element (Vector) Variable Example 2:** This example shows an ASCII vector file (`engold.Evec`) for the gold geometry example.

```
Per_elem vector values for the EnSight Gold geometry example
part
    1
    tria3
    2.10000E+00
    3.10000E+00
    2.20000E+00
    3.20000E+00
```

```

2.30000E+00
3.30000E+00
hexa8
4.10000E+00
4.20000E+00
4.30000E+00
part
    2
bar2
1.10000E+00
1.20000E+00
1.30000E+00
part
    3
block
5.10000E+00
6.10000E+00
5.20000E+00
6.20000E+00
5.30000E+00
6.30000E+00

```

**Per\_element (Tensor) Variable Example3:** This example shows an ASCII 2nd order symmetric tensor file (`engold.Eten`) for the gold geometry example.

```

Per_elem symmetric tensor values for the EnSight Gold geometry example
part
    1
tria3
2.10000E+00
3.10000E+00
2.20000E+00
3.20000E+00
2.30000E+00
3.30000E+00
2.40000E+00
3.40000E+00
2.50000E+00
3.50000E+00
2.60000E+00
3.60000E+00
hexa8
4.10000E+00
4.20000E+00
4.30000E+00
4.40000E+00
4.50000E+00
4.60000E+00
part
    2
bar2
1.10000E+00
1.20000E+00
1.30000E+00
1.40000E+00
1.50000E+00
1.60000E+00
part
    3
block

```

```

5.10000E+00
6.10000E+00
5.20000E+00
6.20000E+00
5.30000E+00
6.30000E+00
5.40000E+00
6.40000E+00
5.50000E+00
6.50000E+00
5.60000E+00
6.60000E+00

```

**Per\_element (Complex) Variable Example 4:** This example shows ASCII complex real (`engold.Ecmp_r`) and imaginary (`engold.Ecmp_i`) *scalar* files for the gold geometry example. (The same methodology would apply for complex real and imaginary *vector* files.)

Real scalar File:

```

Per_elem complex real scalar values for the EnSight Gold geometry example
part
    1
    tria3
    2.10000E+00
    3.10000E+00
    hexa8
    4.10000E+00
part
    2
    bar2
    1.10000E+00
part
    3
    block
    5.10000E+00
    6.10000E+00

```

Imaginary scalar File:

```

Per_elem complex imaginary scalar values for the EnSight Gold geometry example
part
    1
    tria3
    2.20000E+00
    3.20000E+00
    hexa8
    4.20000E+00
part
    2
    bar2
    1.20000E+00
part
    3
    block
    5.20000E+00
    6.20000E+00

```

## EnSight Gold Undefined Variable Values Format

Undefined variable values are allowed in EnSight Gold scalar, vector, tensor and complex variable file formats. Undefined values are specified on a “per section” basis (i.e. `coordinates`, `element_type`, or `block`) in each EnSight Gold variable file. EnSight first parses any undefined keyword “undef” that may follow the sectional keyword (i.e. `coordinates undef`, `element_type undef`, or `block undef`) on its line. This indicates that the next floating point value is the undefined value used in that section. EnSight reads this undefined value, reads all subsequent variable values for that section; and then converts any undefined (file section) values to an internal undefined value (currently -1.345e-10) recognized computationally by EnSight (Note: the internal, or computational, undefined value can be changed by the user via the “`test: change_undef_value`” command **before any data is read.**)

*Note: EnSight’s undefined capability is for variables only - not for geometry! Also, in determining internally whether a vector or tensor variable is undefined at a node or element, the first component is all that is examined. You cannot have some components defined and others undefined.*

The following per\_node and per\_element ASCII scalar files contain examples of undefined values. For your comparison, these two files are the files `engold.Nsca` and `engold.Esca` written with some undefined values specified. Note that the undefined values per section need not be the same value; rather, it may be any value - usually outside the interval range of the variable. *The same methodology applies to vector, tensor, and complex files.*

### C Binary form: (Per\_node)

SCALAR FILE:

description line 1	80 chars
part	80 chars
#	1 int
coordinates undef	80 chars
undef_value	1 float
s_n1 s_n2 ... s_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block undef # nn = i*j*k	80 chars
undef_value	1 float
s_n1 s_n2 ... s_nn	nn floats

### Fortran Binary form: (Per\_node)

SCALAR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates undef'	80 chars
'undef_value'	1 float
's_n1 s_n2 ... s_nn'	nn floats
'part'	80 chars

.		
.		
'part'		80 chars
'#'		1 int
'block undef'	# nn = i*j*k	80 chars
'undef_value'		1 float
's_n1 s_n2 ... s_nn'		nn floats

**ASCII form: (Per\_node)****SCALAR FILE:**

description line 1	A (max of 79 typ)
part	A
#	I10
coordinates undef	A
undef_value	E12.5
s_n1	E12.5 1/line (nn)
s_n2	
.	
.	
s_nn	
part	A
.	
.	
part	A
#	I10
block undef	# nn = i*j*k
undef_value	A
s_n1	E12.5
s_n2	E12.5 1/line (nn)
.	
.	
s_nn	

Undefined per\_node (Scalar) Variable Example: This example shows undefined data in an ASCII scalar file (engold.Nsca\_u) for the gold geometry example.

Per\_node undefined scalar values for the EnSight Gold geometry example

```

part
  1
coordinates undef
-1.00000E+04
-1.00000E+04
 3.00000E+00
 4.00000E+00
 5.00000E+00
 6.00000E+00
 7.00000E+00
 8.00000E+00
 9.00000E+00
 1.00000E+01
 1.10000E+01
part
  2
coordinates
 1.00000E+00
 2.00000E+00
part
  3

```



```

block undef
-1.23450E-10
 1.00000E+00
 2.00000E+00
 3.00000E+00
 4.00000E+00
 5.00000E+00
-1.23450E-10
 7.00000E+00
 8.00000E+00
 9.00000E+00
 1.00000E+01
 1.10000E+01
 1.20000E+01

```

### C Binary form: (Per\_element)

#### SCALAR FILE:

description line 1	80 chars
part	80 chars
#	1 int
element type undef	80 chars
undef_value	1 float
s_e1 s_e2 ... s_ne	ne floats
element type undef	80 chars
undef_value	1 float
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block undef	# nn = (i-1)*(j-1)*(k-1) 80 chars
undef_value	1 float
s_n1 s_n2 ... s_nn	nn floats

### Fortran Binary form: (Per\_element)

#### SCALAR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'element type undef'	80 chars
'undef_value'	1 float
's_e1 s_e2 ... s_ne'	ne floats
'element type undef'	80 chars
'undef_value'	1 float
.	
.	
'part'	80 chars
.	
.	

'part'		80 chars
'#'		1 int
'block undef'	# nn = (i-1)*(j-1)*(k-1)	80 chars
'undef_value'		1 float
's_n1 s_n2 ... s_nn'		nn floats

**ASCII form: (Per\_element)****SCALAR FILE:**

description line 1	A (max of 80 typ)
part	A
#	I10
element type undef	A
undef_value	E12.5
s_e1	E12.5 1/line (ne)
s_e2	
.	
.	
s_ne	
element type undef	A
undef_value	E12.5
.	
.	
part	A
.	
.	
part	A
#	I10
block undef	# nn = (i-1)*(j-1)*(k-1)
undef_value	A
s_n1	E12.5
s_n2	E12.5 1/line (nn)
.	
.	
s_nn	

**Undefined per\_element (Scalar) Variable Example:** This example shows undefined data in an ASCII scalar file (engold.Esca\_u) for the gold geometry example.

```
Per_elem undefined scalar values for the EnSight Gold geometry example
part
    1
    tria3 undef
    -1.00000E+02
    2.00000E+00
    -1.00000E+02
    hexa8
    4.00000E+00
part
    2
    bar2
    1.00000E+00
part
    3
    block undef
    -1.23450E-10
    -1.23450E-10
    6.00000E+00
```

## EnSight Gold Partial Variable Values Format

Partial variable values are allowed in EnSight Gold scalar, vector, tensor and complex variable file formats. Partial values are specified on a “per section” basis (i.e. `coordinates`, `element_type`, or `block`) in each EnSight Gold variable file. EnSight first parses any partial keyword “`partial`” that may follow the sectional keyword (i.e. `coordinates partial`, `element_type partial`, or `block partial`) on its line. This indicates that the next integer value is the number of partial values defined in that section. EnSight reads the number of defined partial values, next reads this number of integer partial indices, and finally reads all corresponding partial variable values for that section. Afterwards, any variable value not specified in the list of partial indices is assigned the internal “undefined” (see previous section) value. Values interpolated between time steps must be defined for both time steps; otherwise, they are undefined.

The following `per_node` and `per_element` ASCII scalar files contain examples of partial values. For your comparison, these two files are the files `engold.Nsca` and `engold.Esca` written with some partial values specified. The same methodology applies to vector, tensor, and complex files.

### C Binary form: (Per\_node)

SCALAR FILE:

description line 1	80 chars
part	80 chars
#	1 int
coordinates partial	80 chars
nn	1 int
i_n1 i_n2 ... i_nn	nn ints
s_n1 s_n2 ... s_nn	nn floats
part	80 chars
.	
.	
part	80 chars
#	1 int
block partial	80 chars
nn	1 int
i_n1 i_n2 ... i_nn	nn ints
s_n1 s_n2 ... s_nn	nn floats

### Fortran Binary form: (Per\_node)

SCALAR FILE:

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'coordinates partial'	80 chars
'nn'	1 int
'i_n1 i_n2 ... i_nn'	nn ints
's_n1 s_n2 ... s_nn'	nn floats
'part'	80 chars
.	

.		
'part'		80 chars
'#'		1 int
'block partial'	# nn = i*j*k	80 chars
'nn'		1 int
'i_n1 i_n2 ... i_nn'		nn ints
's_n1 s_n2 ... s_nn'		nn floats

**ASCII form: (Per\_node)****SCALAR FILE:**

description line 1	A (max of 79 typ)
part	A
#	I10
coordinates partial	A
nn	I10
i_n1	I10 1/line (nn)
i_n2	
.	
.	
i_nn	
s_n1	E12.5 1/line (nn)
s_n2	
.	
.	
s_nn	
part	A
.	
.	
part	A
#	I10
block partial	A
nn	I10
i_n1	I10 1/line (nn)
i_n2	
.	
.	
i_nn	
s_n1	E12.5 1/line (nn)
s_n2	
.	
.	
s_nn	

**Partial per\_node (Scalar) Variable Example:** This example shows partial data in an ASCII scalar file (engold.Nsca\_p) for the gold geometry example.

Per\_node partial scalar values for the EnSight Gold geometry example  
part

```

      1
coordinates partial
      9
      2
      3
      4
      5
      6
      7
      8

```

```

      9
     10
3.00000E+00
4.00000E+00
5.00000E+00
6.00000E+00
7.00000E+00
8.00000E+00
9.00000E+00
1.00000E+01
1.10000E+01
part
      2
coordinates
  1.00000E+00
  2.00000E+00
part
      3
block
  1.00000E+00
  2.00000E+00
  3.00000E+00
  4.00000E+00
  5.00000E+00
  6.00000E+00
  7.00000E+00
  8.00000E+00
  9.00000E+00
  1.00000E+01
  1.10000E+01
  1.20000E+01

```

### C Binary form: (Per\_element)

#### SCALAR FILE:

description line 1	80 chars
part	80 chars
#	1 int
element type partial	80 chars
ne	1 int
i_n1 i_n2 ... i_ne	ne ints
s_e1 s_e2 ... s_ne	ne floats
element type partial	80 chars
ne	1 int
i_n1 i_n2 ... i_ne	ne ints
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block partial	80 chars
me	1 int
i_n1 i_n2 ... i_me	me ints
s_n1 s_n2 ... s_me	me floats

# me= (i-1)\*(j-1)\*(k-1)

**Fortran Binary form: (Per\_element)****SCALAR FILE:**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'element type partial'	80 chars
'ne'	1 int
'i_n1 i_n2 ... i_ne'	ne ints
's_e1 s_e2 ... s_ne'	ne floats
'element type partial'	80 chars
'ne'	1 int
'i_n1 i_n2 ... i_ne'	ne ints
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block partial'	# me = (i-1)*(j-1)*(k-1) 80 chars
'me'	1 int
'i_n1 i_n2 ... i_me'	me ints
's_n1 s_n2 ... s_me'	me floats

**ASCII form: (Per\_element)****SCALAR FILE:**

description line 1	A (max of 80 typ)
part	A
#	I10
element type partial	A
ne	I10
i_n1	I10 1/line (ne)
i_n2	
.	
.	
i_ne	
s_e1	E12.5 1/line (ne)
s_e2	
.	
.	
s_ne	
element type partial	A
ne	I10
i_n1	I10 1/line (ne)
i_n2	
.	
.	
i_ne	
.	
.	
part	A
.	
.	
part	A

```

#                                     I10
block partial                        # me = (i-1)*(j-1)*(k-1)    A
me                                  I10
i_n1                               I10  1/line  (me)
i_n2
.
.
i_me
s_n1                               E12.5  1/line  (me)
s_n2
.
.
s_me

```

Partial per\_element (Scalar) Variable Example: This example shows partial data in an ASCII scalar file (engold.Esca\_p) for the gold geometry example.

```

Per_elem partial scalar values for the EnSight Gold geometry example
part
    1
tria3 partial
    1
    1
    2.00000E+00
hexa8
    4.00000E+00
part
    2
bar2
    1.00000E+00
part
    3
block partial
    1
    2
    6.00000E+00

```

## EnSight Gold Measured/Particle File Format

The format of a Measured/Particle geometry file is as follows:

- Line 1  
This line is a description line.
- Line 2  
Indicates that this file contains particle coordinates. The words “particle coordinates” should be entered on this line without the quotes.
- Line 3  
Specifies the number of Particles.
- Line 4 through the end of the file.  
Each line contains the ID and the X, Y, and Z coordinates of each Particle. The format of this line is “integer real real real” written out in the following format:

From C:                    %8d%12.5e%12.5e%12.5e format

From FORTRAN:        i8, 3e12.5 format

A generic measured/Particle geometry file is as follows:

```
A description line
particle coordinates
#_of_Particles
id xcoord ycoord zcoord
id xcoord ycoord zcoord
id xcoord ycoord zcoord
.
.
.
```

### *Measured Geometry Example*

The following illustrates a measured/Particle file with seven points:

```
This is a simple measured geometry file
particle coordinates
7
101 0.00000E+00 0.00000E+00 0.00000E+00
102 1.00000E+00 0.00000E+00 0.00000E+00
103 1.00000E+00 1.00000E+00 0.00000E+00
104 0.00000E+00 1.00000E+00 0.00000E+00
205 5.00000E-01 0.00000E+00 2.00000E+00
206 5.00000E-01 1.00000E+00 2.00000E+00
307 0.00000E+00 0.00000E+00-1.50000E+00
```

### *Measured Variable Files*

Measured variable files are the same as **EnSight6 case** per\_node variable files. Please note that they are NOT the same as the EnSight gold per\_node variable files. ([see EnSight6 Per\\_Node Variable File Format, in Section 11.2](#))



## *EnSight Gold Material Files Format*

This section contains descriptions of the three EnSight Gold material files; i.e. material id file, mixed-material id file, and mixed-material values file. A simple example dataset is also appended for quick reference.

All three EnSight Gold material files correlate to and follow the same syntax of the other EnSight Gold file formats.

### *Material Id File*

The material id file follows the same syntax as the per\_element variable files, except that its values are integers for each element of designated types of designated parts. First comes a single description line. Second comes a Part line. Third comes a line containing the part number. Fourth comes an element type line (***Note, this is the only material file that has an element type line***). And then comes the corresponding integer value for each element of that type and part (and so on for each part).

The integer value is either positive or negative. A positive integer is the material number/id for the entire element. A negative integer indicates that this element is composed of multiple, or mixed, materials. The absolute value of this negative number is a relative (1-bias) index into the mixed ids file that points to the mixed material data for each element under its part (see example below).

### *Mixed (Material) Ids File*

The mixed-material id file also contains integer values, and follows EnSight Gold syntax with exceptions as noted below. First comes a single description line. Second comes a Part line. Third comes a line containing the part number. Fourth comes a “mixed ids” keyword line. Fifth comes the size of the total mixed id array for all the mixed elements of this part. Next comes the mixed id element data for each of the elements with mixed materials for this part (and so on for each part).

The mixed id data for each of the “mixed elements” has the following order of syntax. First comes the number of mixed materials. Second comes a list of material ids that comprise that element. Next comes a negative number whose absolute value is a relative (1-bias) index into the mixed values file that points to the group of mixed-material fraction values that correspond to each listed material of that element under its part (see example below).

### *Mixed (Material) Values File*

The mixed-material values file contains float values, and also follows EnSight Gold syntax with exceptions as noted below. First comes a single description line. Second comes a Part line. Third comes a line containing the part number. Fourth comes a “mixed values” keyword line. Fifth comes the size of the total mixed values array for all the mixed elements of this part. Next comes the mixed material fraction values whose order corresponds to the order of the material ids listed for that element in the mixed ids file.

**C Binary form:****MATERIAL ID FILE:**

description line 1	80 chars
part	80 chars
#	1 int
element type	80 chars
matid_e1 matid_e2 ... matid_ne	ne ints
element type	80 chars
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
block # nbe = (i-1)*(j-1)*(k-1)	80 chars
matid_e1 matid_e2 ... matid_nbe	nbe ints

**MIXED IDS FILE:**

description line 1	80 chars
part	80 chars
#	1 int
mixed ids	80 chars
ni	1 int
mixid_1 mixid_2 ... mixid_ni	ni ints
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
mixed ids	80 chars
ni	1 int
mixid_1 mixid_2 ... mixid_ni	ni ints

**MIXED VALUES FILE:**

description line 1	80 chars
part	80 chars
#	1 int
mixed values	80 chars
nf	1 int
mixval_1 mixval_2 ... mixval_nf	nf floats
.	
.	
part	80 chars
.	
.	
part	80 chars
#	1 int
mixed values	80 chars
nf	1 int
mixval_1 mixval_2 ... mixval_nf	nf floats

**Fortran Binary form:****MATERIAL ID FILE:**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'element type'	80 chars
'matid_e1 matid_e2 ... matid_ne'	ne ints
'element type'	80 chars
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'block'      # nbe = (i-1)*(j-1)*(k-1)	80 chars
'matid_e1 matid_e2 ... matid_nbe'	nbe ints

**MIXED IDS FILE:**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'mixed ids'	80 chars
'ni'	1 int
'mixid_1 mixid_2 ... mixid_ni'	ni ints
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'mixed ids'	80 chars
'ni'	1 int
'mixid_1 mixid_2 ... mixid_ni'	ni ints

**MIXED VALUES FILE:**

'description line 1'	80 chars
'part'	80 chars
'#'	1 int
'mixed values'	80 chars
'nf'	1 int
'mixval_1 mixval_2 ... mixval_nf'	nf floats
.	
.	
'part'	80 chars
.	
.	
'part'	80 chars
'#'	1 int
'mixed values'	80 chars
'nf'	1 int
'mixval_1 mixval_2 ... mixval_nf'	nf floats

**ASCII form:****MATERIAL ID FILE:**

description line 1	A (max of 79 typ)
part	A
#	I10
element type	A
matid_e1	I10 1/line (ne)
matid_e2	
...	
matid_ne	
element type	A
.	
.	
part	A
.	
.	
part	A
#	I10
block # nbe = (i-1)*(j-1)*(k-1)	A
matid_e1	I10 1/line (nbe)
matid_e2	
...	
matid_nbe	

**MIXED IDS FILE:**

description line 1	A (max of 79 typ)
part	A
#	I10
mixed ids	A
ni	I10
mixid_1	I10 1/line (ni)
mixid_2	
...	
mixid_ni . .	
part	A
.	
.	
part	A
#	I10
mixed ids	A
ni	I10
mixid_1	I10 1/line (ni)
mixid_2	
...	
mixid_ni	

**MIXED VALUES FILE:**

description line 1	A (max of 80 typ)
part	A
#	I10
mixed values	A
nf	I10
mixval_1	E12.5 1/line (nf)
mixval_2	
...	
mixval_nf	
.	
.	
part	A
.	
.	
part	A
#	I10
mixed values	A
nf	I10
mixval_1	E12.5 1/line (nf)
mixval_2	
...	
mixval_nf	

## Example Material Dataset

The following example dataset of ASCII EnSight Gold geometry and material files show the definition of material fractions for an unstructured model.

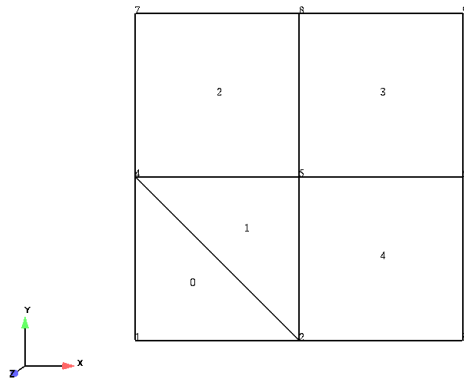
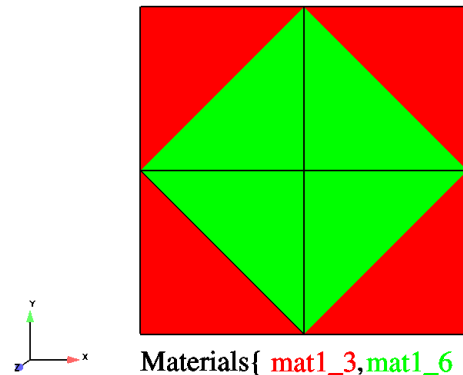


Figure 11-2  
Geometry for Example Material Dataset



Materials{ **mat1\_3**,**mat1\_6** }

Figure 11-2  
Materials for Example Material Dataset

## Case file

```
# Sample Case File for 2D Material Dataset
# Created: 03Apr03:mel
#
FORMAT
type:  ensight gold

GEOMETRY
model:                                zmat2d.geo

VARIABLE
scalar per node:                      zmat2d.sca

MATERIAL
material set number:                  1 Mat1
material id count:                    2
material id numbers:                  3 6
material id names:                    mat1_3 mat1_6
material id per element:              zmat2d.mat1
material mixed ids:                   zmat2d.mixi
material mixed values:                zmat2d.mixv
```

```
#
#      y
#      ^
#      | Case Material ids = {3,6}
#
# 6. 7-----8-----9
#      | \   /  | \   /  |
#      |  e2  |  e3  |
#      |  q0  |  q2  |
#      | { .5, .5 } | { .5, .5 } |
#      |-----|-----|
# 3. 4-----5-----6
#      | \   /  |  \   /  |
#      |  t1  |  e4  |
#      |  e0  |  e1  |
#      |  t0  |  q3  |
#      | { 1., 0. } | { .5, .5 } |
#      |-----|-----|
# 0. 1-----2-----3 --> x
#      | 0.      3.      6.
#
```

## EnSight 8 User Manual

```

Geometry file
Example 2D Material Dataset
node id given
element id given
part
    1
2d-mesh
coordinates
    9
        1
        2
        3
        4
        5
        6
        7
        8
        9
0.00000e+00
3.00000e+00
6.00000e+00
0.00000e+00
3.00000e+00
6.00000e+00
0.00000e+00
3.00000e+00
6.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
3.00000e+00
3.00000e+00
3.00000e+00
6.00000e+00
6.00000e+00
6.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
0.00000e+00
tria3
    2
    0
    1
    1          2
    2          5
quad4
    3
    2
    3
    4
    4          5
    8          5
    2          3

```

Material Number/Id File  
 part  
     1  
 tria3  
     3  
     6  
 quad4  
   -1  
   -5  
   -9

Material Number/ID File  
 (zmat2d.mati)

Mixed Ids File  
 part  
     1  
 mixed ids  
     12  
     2  
     3  
     6  
   -1  
     2  
     3  
     6  
   -3  
     2  
     3  
     6  
   -5

Mixed Material Ids File  
 (zmat2d.mixi)

Mixed Material Values File  
 (zmat2d.mixv)

Mixed Values File  
 part  
     1  
 mixed values  
     6  
     0.50000e+00  
     0.50000e+00  
     0.50000e+00  
     0.50000e+00  
     0.50000e+00  
     0.50000e+00

Scalar File (zmat2d.sca)

Scalar File  
 part  
     1  
 coordinates  
   0.00000e+00  
   1.00000e+00  
   2.00000e+00  
   1.00000e+00  
   2.00000e+00  
   3.00000e+00  
   2.00000e+00  
   3.00000e+00  
   4.00000e+00





## 11.2 EnSight6 Casefile Format

Included in this section:

[EnSight6 General Description](#)

[EnSight6 Geometry File Format](#)

[EnSight6 Case File Format](#)

[EnSight6 Wild Card Name Specification](#)

[EnSight6 Variable File Format](#)

[EnSight6 Per\\_Node Variable File Format](#)

[EnSight6 Per\\_Element Variable File Format](#)

[EnSight6 Measured/Particle File Format](#)

[Writing EnSight6 Binary Files](#)

### EnSight6 General Description

EnSight6 data consists of the following files:

- Case (required) (points to all other needed files including model geometry, variables, and possibly measured geometry and variables)

EnSight6 supports constant result values as well as scalar, vector, 2nd order symmetric tensor, and complex variable fields.

EnSight makes no assumptions regarding the physical significance of the variable values in the files. These files can be from any discipline. For example, the scalar file can include such things as pressure, temperature, and stress. The vector file can be velocity, displacement, or any other vector data. And so on.

All variable results for EnSight6 are contained in disk files—one variable per file. Additionally, if there are multiple time steps, there must either be a set of disk files for each time step (transient multiple-file format), or all time steps of a particular variable or geometry in one disk file (transient single-file format). Thus, all EnSight6 transient geometry and variable files can be expressed in either multiple file format or single file format.

Sources of EnSight6 data include the following:

- Data that can be translated to conform to the EnSight6 data format
- Data that originates from one of the translators supplied with the EnSight application

The EnSight6 format supports an unstructured defined element set as shown in the figure on the following page. Unstructured data must be defined in this element set. Elements that do not conform to this set must either be subdivided or discarded. The EnSight6 format also supports a structured block data format which is very similar to the PLOT3D format. *For the structured format, the standard order of nodes is such that I's advance quickest, followed by J's, and then K's.* A given EnSight6 model may have either unstructured data, structured data, or a mixture of both.

### *ens\_checker*

A program is supplied with EnSight which attempts to verify the integrity of the *format* of EnSight 6 and EnSight Gold files. If you are producing EnSight formatted data, this program can be very helpful, especially in your development stage, in making sure that you are adhering to the published format. It makes no attempt to verify the validity of floating point values, such as coordinates, variable values, etc. This program takes a casefile as input. Thus, it will check the format of the casefile, and all associated geometry and variable files referenced in the casefile. See [How To Use ens\\_checker](#).

## Supported EnSight Elements

The elements that are supported by the EnSight6 format are:

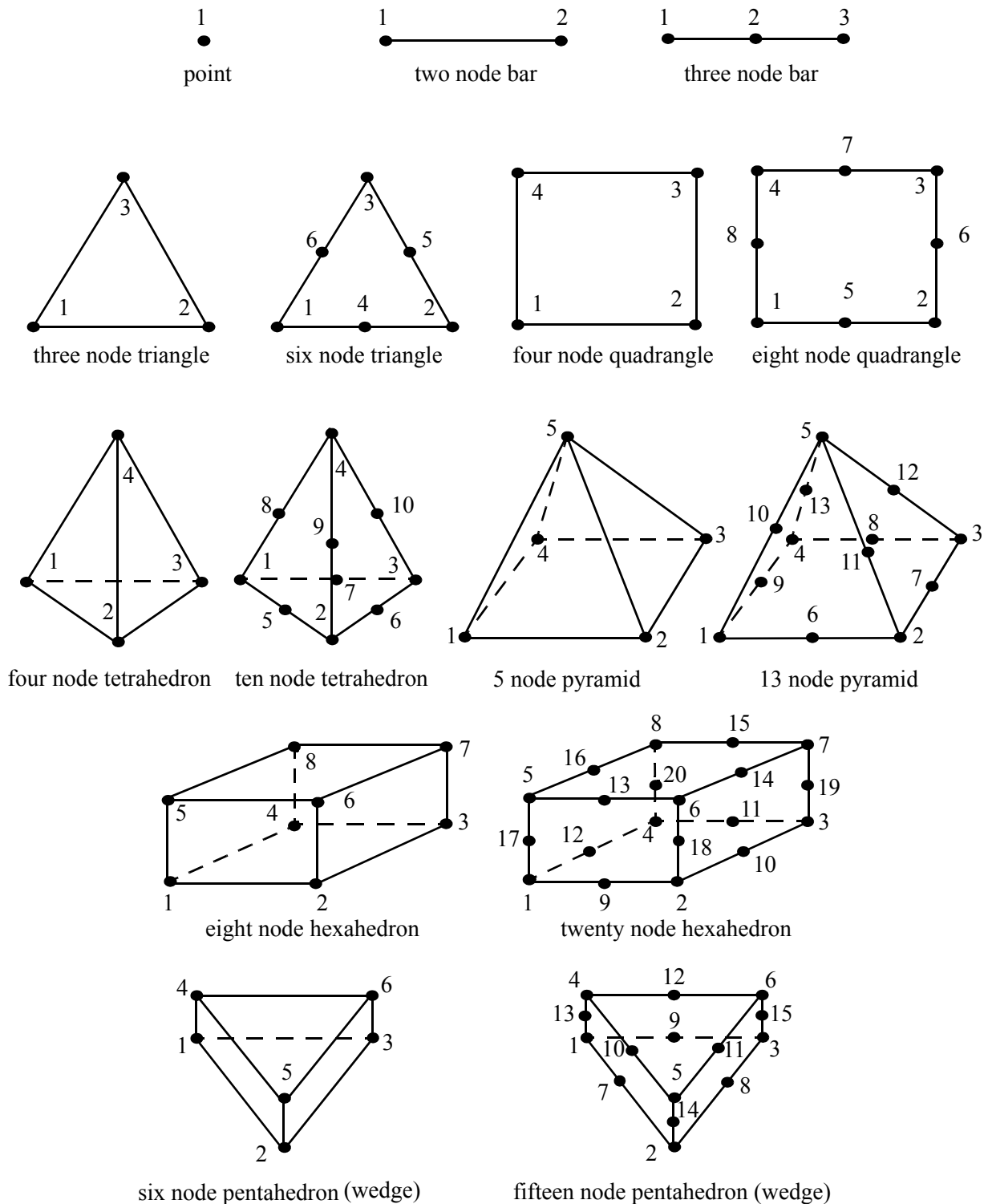


Figure 11-3  
Supported EnSight6 Elements

## EnSight6 Geometry File Format

The EnSight6 format consists of keywords followed by information. The following seven items are important when working with EnSight6 geometry files:

1. You do not have to assign node IDs. If you do, the element connectivities are based on the node numbers. If you let EnSight assign the node IDs, the nodes are considered to be sequential starting at node 1, and element connectivity is done accordingly. If node IDs are set to off, they are numbered internally; however, you will not be able to display or query on them. If you have node IDs in your data, you can have EnSight ignore them by specifying “node id ignore.” Using this option may reduce some of the memory taken up by the Client and Server, but display and query on the nodes will not be available.
2. You do not need to specify element IDs. If you specify element IDs, or you let EnSight assign them, you can show them on the screen. If they are set to off, you will not be able to show or query on them. If you have element IDs in your data you can have EnSight ignore them by specifying “element id ignore.” Using this option will reduce some of the memory taken up by the Client and Server. This may or may not be a significant amount, and remember that display and query on the elements will not be available.
3. The format of integers and real numbers **must be followed** (See the Geometry Example below).

4. Integers are written out using the following integer format:

From C: 8d format

From FORTRAN: i8 format

Real numbers are written out using the following floating-point format:

From C: 12.5e format

From FORTRAN: e12.5 format

**The number of integers or reals per line must also be followed!**

5. By default, a Part is processed to show the outside boundaries. This representation is loaded to the Client host system when the geometry file is read (unless other attributes have been set on the workstation, such as feature angle).
6. Coordinates for unstructured data must be defined before any Parts can be defined. The different elements can be defined in any order (that is, you can define a hexa8 before a bar2).
7. A Part containing structured data cannot contain any unstructured element types or more than one block. **Each structured Part is limited to a single block.** A structured block is indicated by following the Part description line with either the “block” line or the “block iblanked” line. An “iblanked” block must contain an additional integer array of values at each node, traditionally called the iblank array. Valid iblank values for the EnSight format are:
  - 0 for nodes which are exterior to the model, sometimes called blanked-out nodes
  - 1 for nodes which are interior to the model, thus in the free stream and to be used
  - <0 or >1 for any kind of boundary nodes

In EnSight's structured Part building dialog, the iblank option selected will control which portion of the structured block is "created". Thus, from the same structured block, the interior flow field part as well as a symmetry boundary part could be "created".

*Note: By default EnSight does not do any "partial" cell iblank processing. Namely, only complete cells containing no "exterior" nodes are created. It is possible to obtain partial cell processing by issuing the "test:partial\_cells\_on" command in the Command Dialog before reading the file.*

*Note also that for the structured format, the standard order of nodes is such that I's advance quickest, followed by J's, and then K's.*

#### Generic Format

Not all of the lines included in the following generic example file are necessary:

```
description line 1 |
description line 2 |
node id <off/given/assign/ignore> | All geometry files must
element id <off/given/assign/ignore> | contain these first six lines
coordinates |
# of unstructured nodes |
id x y z
id x y z
id x y z
.
.
.
part #
description line
point
number of points
id nd
id nd
id nd
.
.
.
bar2
number of bar2's
id nd nd
id nd nd
id nd nd
.
.
.
bar3
number of bar3's
id nd nd nd
id nd nd nd
id nd nd nd
.
.
.
tria3
number of three node triangles
id nd nd nd
id nd nd nd
id nd nd nd
.
.
.
```

```

tria6
number of six node triangles
id nd nd nd nd nd nd
.
.
.
quad4
number of quad 4's
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
.
.
.
quad8
number of quad 8's
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
.
.
.
tetra4
number of 4 node tetrahedrons
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
.
.
.
tetra10
number of 10 node tetrahedrons
id nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd
.
.
.
pyramid5
number of 5 node pyramids
id nd nd nd nd nd
id nd nd nd nd nd
id nd nd nd nd nd
id nd nd nd nd nd
.
.
.
pyramid13
number of 13 node pyramids
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
.
.
.

```

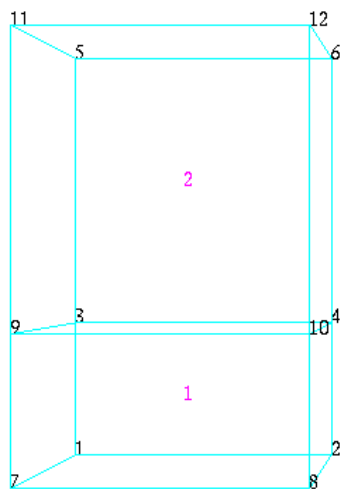
```

hexa8
number of 8 node hexahedrons
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
.
.
.
hexa20
number of 20 node hexahedrons
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
.
.
.
penta6
number of 6 node pentahedrons
id nd nd nd nd nd nd
id nd nd nd nd nd nd
id nd nd nd nd nd nd
id nd nd nd nd nd nd
id nd nd nd nd nd nd
id nd nd nd nd nd nd
.
.
.
penta15
number of 15 node pentahedrons
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
.
.
.
part #
description line
block                                     #nn=i*j*k
      i           j           k
x_n1 x_n2 x_n3 ..... x_nn             (6/line)
y_n1 y_n2 y_n3 ..... y_nn             "
z_n1 z_n2 z_n3 ..... z_nn             "

part #
description line
block      iblanked                     #nn=i*j*k
      i           j           k
x_n1 x_n2 x_n3 ..... x_nn             (6/line)
y_n1 y_n2 y_n3 ..... y_nn             "
z_n1 z_n2 z_n3 ..... z_nn             "
ib_n1 ib_n2 ib_n3 ..... ib_nn         (10/line)

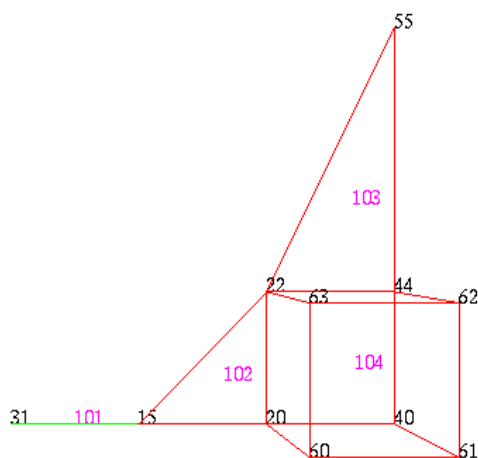
```

## Structured Part



part 3

## Unstructured Parts



part 1



part 2

### EnSight6 Geometry File Example

The following is an example of an ASCII EnSight6 geometry file with 11 defined unstructured nodes from which 2 unstructured parts are defined, and a 2x3x2 structured part as depicted in the above diagram. (See Case File Example 1 for reference to this file.)

```

This is the 1st description line of the EnSight6 geometry example
This is the 2nd description line of the EnSight6 geometry example
node id given
element id given
coordinates
  11
  15 4.00000e+00 0.00000e+00 0.00000e+00
  31 3.00000e+00 0.00000e+00 0.00000e+00
  20 5.00000e+00 0.00000e+00 0.00000e+00
  40 6.00000e+00 0.00000e+00 0.00000e+00
  22 5.00000e+00 1.00000e+00 0.00000e+00
  44 6.00000e+00 1.00000e+00 0.00000e+00
  55 6.00000e+00 3.00000e+00 0.00000e+00
  60 5.00000e+00 0.00000e+00 2.00000e+00
  61 6.00000e+00 0.00000e+00 2.00000e+00
  62 6.00000e+00 1.00000e+00 2.00000e+00
  63 5.00000e+00 1.00000e+00 2.00000e+00
part 1
2D uns-elements (description line for part 1)
tria3
  2
  102 15 20 22
  103 22 44 55
hexa8
  1
  104 20 40 44 22 60 61 62 63
part 2
1D uns-elements (description line for part 2)
bar2
  1
  101 31 15
part 3

```



```

3D struct-part (description line for part 3)
block iblanked
      2      3      2
0.00000e+00 2.00000e+00 0.00000e+00 2.00000e+00 0.00000e+00 2.00000e+00
0.00000e+00 2.00000e+00 0.00000e+00 2.00000e+00 0.00000e+00 2.00000e+00
0.00000e+00 0.00000e+00 1.00000e+00 1.00000e+00 3.00000e+00 3.00000e+00
0.00000e+00 0.00000e+00 1.00000e+00 1.00000e+00 3.00000e+00 3.00000e+00
0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00 0.00000e+00
2.00000e+00 2.00000e+00 2.00000e+00 2.00000e+00 2.00000e+00 2.00000e+00
      1      1      1      1      1      1      1      1      1
      1      1

```

## EnSight6 Case File Format

The Case file is an ASCII free format file that contains all the file and name information for accessing model (and measured) geometry, variable, and time information. It is comprised of five sections (FORMAT, GEOMETRY, VARIABLE, TIME, FILE) as described below:

*Notes:* All lines in the Case file are limited to 79 characters.  
The titles of each section must be in all capital letters.  
Anything preceded by a “#” denotes a comment and is ignored. Comments may append information lines or be placed on their own lines.  
Information following “:” may be separated by white spaces or tabs.  
Specifications encased in “[ ]” are optional, as indicated.

### Format Section

This is a required section which specifies the type of data to be read.

Usage:

```

FORMAT
type:      ensight

```

### Geometry Section

This is a required section which specifies the geometry information for the model (as well as measured geometry if present, and periodic match file ([see Section 11.9, Periodic Matchfile Format](#)) if present).

Usage:

```

GEOMETRY
model:      [ts] [fs]      filename      [change_coords_only]
measured:   [ts] [fs]      filename      [change_coords_only]
match:      filename
boundary:   filename

```

where: *ts* = time set number as specified in TIME section. This is optional.

*fs* = corresponding file set number as specified in FILE section below.

*filename* = The filename of the appropriate file.

-> Model or measured filenames for a static geometry case, as well as match and boundary filenames will not contain “\*” wildcards.

-> Model or measured filenames for a changing geometry case will contain “\*” wildcards.

*change\_coords\_only* = The option to indicate that the changing geometry (as indicated by wildcards in the filename) is coords only. Otherwise, changing geometry connectivity will be assumed.

*Variable Section*

This is an optional section which specifies the files and names of the variables. Constant variable values can also be set in this section.

**Usage:**

```
VARIABLE
constant per case:           [ts]           description  const_value(s)
scalar per node:             [ts] [fs]       description  filename
vector per node:             [ts] [fs]       description  filename
tensor symm per node:        [ts] [fs]       description  filename
scalar per element:          [ts] [fs]       description  filename
vector per element:          [ts] [fs]       description  filename
tensor symm per element:     [ts] [fs]       description  filename
scalar per measured node:    [ts] [fs]       description  filename
vector per measured node:    [ts] [fs]       description  filename
complex scalar per node:     [ts] [fs]       description  Re_fn      Im_fn      freq
complex vector per node:     [ts] [fs]       description  Re_fn      Im_fn      freq

complex scalar per element:   [ts] [fs]       description  Re_fn      Im_fn      freq
complex vector per element:   [ts] [fs]       description  Re_fn      Im_fn      freq
```

where:

ts = The corresponding time set number (or index) as specified in TIME section below. This is only required for transient constants and variables.

fs = The corresponding file set number (or index) as specified in FILE section below.

description = The variable (GUI) name (ex. Pressure, Velocity, etc.)

const\_value(s) = The constant value. If constants change over time, then ns (see TIME section below) constant values of ts.

filename = The filename of the variable file. Note: only transient filenames contain “\*” wildcards.

Re\_fn = The filename for the file containing the real values of the complex variable.

Im\_fn = The filename for the file containing the imaginary values of the complex variable.

freq = The corresponding harmonic frequency of the complex variable. For complex variables where harmonic frequency is undefined, simply use the text string: UNDEFINED.

*Note:* As many variable description lines as needed may be used.

*Note:* Variable descriptions have the following restrictions:

*The variable description is limited to 19 characters in the current release.*

*Duplicate variable descriptions are not allowed.*

*Leading and trailing white space will be eliminated.*

*Variable descriptions must not start with a numeric digit.*

*Variable descriptions must not contain any of the following reserved characters:*

```
(  [  +  @  !  *  $
)  ]  -  space  #  ^  /
```

*Time Section*

This is an optional section for steady state cases, but is required for transient cases. It contains time set information. Shown below is information for one time set. Multiple time sets (up to 16) may be specified for measured data as shown in Case File Example 3 below.

**Usage:**

```
TIME
```

```

time set:          ts [description]
number of steps:   ns
filename start number: fs
filename increment: fi
time values:       time_1 time_2 .... time_ns

```

or

```

TIME
time set:          ts [description]
number of steps:   ns
filename numbers:  fn
time values:       time_1 time_2 .... time_ns

```

where: *ts* = timeset number. This is the number referenced in the GEOMETRY and VARIABLE sections.

*description* = optional timeset description which will be shown in user interface.

*ns* = number of transient steps

*fs* = the number to replace the “\*” wildcards in the filenames, for the first step

*fi* = the increment to *fs* for subsequent steps

*time* = the actual time values for each step, each of which must be separated by a white space and which may continue on the next line if needed

*fn* = a list of numbers or indices, to replace the “\*” wildcards in the filenames.

#### File Section

This section is optional for expressing a transient case with single-file formats. This section contains single-file set information. This information specifies the number of time steps in each file of each *data entity*, i.e. each geometry and each variable (model and/or measured). Each data entity’s corresponding file set might have multiple *continuation* files due to system file size limit, i.e. ~2 GB for 32-bit and ~4 TB for 64-bit architectures. Each file set corresponds to one and only one time set, but a time set may be referenced by many file sets. The following information may be specified in each file set. For file sets where all of the time set data exceeds the maximum file size limit of the system, both *filename index* and *number of steps* are repeated within the file set definition for each continuation file required. Otherwise *filename index* may be omitted if there is only one file. File set information is shown in Case File Example 4 below.

#### Usage:

```

FILE
file set:          fs
filename index:    fi # Note: only used when data continues in other files
number of steps:   ns

```

where: *fs* = file set number. This is the number referenced in the GEOMETRY and VARIABLE sections above.

*ns* = number of transient steps

*fi* = file index number in the file name (replaces “\*” in the filenames)

Case File Example 1 The following is a minimal EnSight6 case file for a steady state model with some results.

*Note: this (en6.case) file, as well as all of its referenced geometry and variable files (along with a couple of command files) can be found under your installation directory (path: \$CEI\_HOME/ensight80/data/user\_manual). The EnSight6 Geometry File Example and the Variable File Examples are the contents of these files.*

```

FORMAT
type: ensight

```

```

GEOMETRY
model: en6.geo

VARIABLE
constant per case:          Cden    .8
scalar per element:         Esca    en6.Esca
scalar per node:            Nsca    en6.Nsca
vector per element:         Evec    en6.Evec
vector per node:            Nvec    en6.Nvec

tensor symm per element:    Eten    en6.Eten
tensor symm per node:       Nten    en6.Nten

complex scalar per element:  Ecmp    en6.Ecmp_r en6.Ecmp_i  2.
complex scalar per node:    Ncmp    en6.Ncmp_r en6.Ncmp_i  4.

```

**Case File Example 2** The following is a Case file for a transient model. The connectivity of the geometry is also changing.

```

FORMAT
type: ensight

GEOMETRY
model:          1          example2.geo**

VARIABLE
scalar per node: 1      Stress      example2.scl**
vector per node: 1      Displacement example2.dis**

TIME
time set:        1
number of steps: 3
filename start number: 0
filename increment: 1
time values:     1.0  2.0  3.0

```

The following files would be needed for Example 2:

example2.geo00	example2.scl00	example2.dis00
example2.geo01	example2.scl01	example2.dis01
example2.geo02	example2.scl02	example2.dis02

**Case File Example 3** The following is a Case file for a transient model with measured data.

*This example has pressure given per element.*

```

FORMAT
type: ensight

GEOMETRY
model:          1          example3.geo*
measured:       2          example3.mgeo**

VARIABLE
constant per case:          Gamma    1.4
constant per case:          1      Density    .9 .9 .7 .6 .6
scalar per element          1      Pressure      example3.pre*
vector per node:            1      Velocity      example3.vel*
scalar per measured node:    2      Temperature  example3.mtem**
vector per measured node:    2      Velocity      example3.mvel**

```

```

TIME
time set:                1
number of steps:         5
filename start number:   1
filename increment:      2
time values:             .1 .2 .3          # This example shows that time
                        .4 .5          # values can be on multiple lines
time set:                2
number of steps:         6
filename start number:   0
filename increment:      2
time values:             .05 .15 .25 .34 .45 .55

```

The following files would be needed for Example 3:

example3.geo1	example3.pre1	example3.vel1
example3.geo3	example3.pre3	example3.vel3
example3.geo5	example3.pre5	example3.vel5
example3.geo7	example3.pre7	example3.vel7
example3.geo9	example3.pre9	example3.vel9
example3.mgeo00	example3.mtem00	example3.mvel00
example3.mgeo02	example3.mtem02	example3.mvel02
example3.mgeo04	example3.mtem04	example3.mvel04
example3.mgeo06	example3.mtem06	example3.mvel06
example3.mgeo08	example3.mtem08	example3.mvel08
example3.mgeo10	example3.mtem10	example3.mvel10

Case File Example 4    The following is Case File Example 3 expressed in transient single-file formats.

*In this example, the transient data for the measured velocity data entity happens to be greater than the maximum file size limit. Therefore, the first four time steps fit and are contained in the first file, and the last two time steps are 'continued' in a second file.*

```

FORMAT
type: ensight

GEOMETRY
model:                1      example4.geo
measured:             2      example4.mgeo

VARIABLE
constant per case:           Density      .5
scalar per element:         1      1      Pressure      example4.pre
vector per node:            1      1      Velocity       example4.vel
scalar per measured node:    2      2      Temperature    example4.mtem
vector per measured node:    2      3      Velocity       example4.mvel*

TIME
time set:                1      Model
number of steps:         5
time values:             .1 .2 .3 .4 .5

time set:                2      Measured
number of steps:         6

```

```
time values:          .05 .15 .25 .34 .45 .55

FILE
file set:             1
number of steps:      5

file set:             2
number of steps:      6

file set:             3
filename index:       1
number of steps:      4
filename index:       2
number of steps:      2
```

The following files would be needed for Example 4:

```
example4.geo      example4.pre      example4.vel
example4.mgeo     example4.mtem     example4.mvel1
                  example4.mvel2
```

Contents of  
Transient  
Single Files

Each file contains transient data that corresponds to the specified number of time steps. The data for each time step sequentially corresponds to the simulation time values (time values) found listed in the TIME section. In transient single-file format, the data for each time step essentially corresponds to a standard EnSight6 geometry or variable file (model or measured) as expressed in multiple file format. The data for each time step is enclosed between two *wrapper* records, i.e. preceded by a BEGIN TIME STEP record and followed by an END TIME STEP record. Time step data is not split between files. If there is not enough room to append the data from a time step to the file without exceeding the maximum file limit of a particular system, then a continuation file must be created for the time step data and any subsequent time step. Any type of user comments may be included before and/or after each transient step wrapper.

*Note 1: If transient single file format is used, EnSight expects all files of a dataset to be specified in transient single file format. Thus, even static files must be enclosed between a BEGIN TIME STEP and an END TIME STEP wrapper.*

*Note 2: For binary geometry files, the first BEGIN TIME STEP wrapper must follow the <C Binary/Fortran Binary> line. Both BEGIN TIME STEP and END TIME STEP wrappers are written according to type (1) in binary. (see Writing EnSight6 Binary Files, in Section 11.2)*

*Note 3: Efficient reading of each file (especially binary) is facilitated by appending each file with a **file index**. A file index contains appropriate information to access the file byte positions of each time step in the file. (EnSight automatically appends a file index to each file when exporting in transient single file format.) If used, the file index must follow the last END TIME STEP wrapper in each file.*

**File Index Usage:**

ASCII	Binary	Item	Description
"%20d\n"	sizeof(int)	n	Total number of data time steps in the file.
"%20d\n"	sizeof(long)	fb <sub>1</sub>	File byte loc for contents of 1 <sup>st</sup> time step*
"%20d\n"	sizeof(long)	fb <sub>2</sub>	File byte loc for contents of 2 <sup>nd</sup> time step*
...	...	...	...
"%20d\n"	sizeof(long)	fb <sub>n</sub>	File byte loc for contents of n <sup>th</sup> time step*
"%20d\n"	sizeof(int)	flag	Miscellaneous flag (= 0 for now)

ASCII	Binary	Item	Description
"%20d\n"	sizeof(long)	fb of item n	File byte loc for Item n above
"%s\n"	sizeof(char)*80	"FILE_INDEX"	File index keyword

*\* Each file byte location is the first byte that follows the BEGIN TIME STEP record.*

Shown below are the contents of each of the above files, using the data files from Case File Example 3 for reference (without FILE\_INDEX for simplicity).

Contents of file example4.geo\_1:

```
BEGIN TIME STEP
Contents of file example3.geo1
END TIME STEP
BEGIN TIME STEP
Contents of file example3.geo3
END TIME STEP
BEGIN TIME STEP
Contents of file example3.geo5
END TIME STEP
BEGIN TIME STEP
Contents of file example3.geo7
END TIME STEP
BEGIN TIME STEP
Contents of file example3.geo9
END TIME STEP
```

Contents of file example4.pre\_1:

```
BEGIN TIME STEP
Contents of file example3.pre1
END TIME STEP
BEGIN TIME STEP
Contents of file example3.pre3
END TIME STEP
BEGIN TIME STEP
Contents of file example3.pre5
END TIME STEP
BEGIN TIME STEP
Contents of file example3.pre7
END TIME STEP
BEGIN TIME STEP
Contents of file example3.pre9
END TIME STEP
```

Contents of file example4.vel\_1:

```
BEGIN TIME STEP
Contents of file example3.vel1
END TIME STEP
BEGIN TIME STEP
Contents of file example3.vel3
END TIME STEP
BEGIN TIME STEP
Contents of file example3.vel5
END TIME STEP
BEGIN TIME STEP
Contents of file example3.vel7
END TIME STEP
BEGIN TIME STEP
Contents of file example3.vel9
END TIME STEP
```

Contents of file example4.mgeo\_1:

```
BEGIN TIME STEP
Contents of file example3.mgeo00
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mgeo02
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mgeo04
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mgeo06
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mgeo08
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mgeo10
END TIME STEP
```

Contents of file example4.mtem\_1:

```

BEGIN TIME STEP
Contents of file example3.mtem00
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mtem02
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mtem04
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mtem06
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mtem08
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mtem10
END TIME STEP

Contents of file example4.mvel1_1:
BEGIN TIME STEP
Contents of file example3.mvel100
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mvel102
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mvel104
END TIME STEP
BEGIN TIME STEP
Contents of file example3.mvel106
END TIME STEP

Contents of file example4.mvel2_1:
Comments can precede the beginning wrapper here.

BEGIN TIME STEP
Contents of file example3.mvel108
END TIME STEP

Comments can go between time step wrappers here.

BEGIN TIME STEP
Contents of file example3.mvel110
END TIME STEP

Comments can follow the ending time step wrapper.

```

## EnSight6 Wild Card Name Specification

For transient data, if multiple time files are involved, the file names must conform to the EnSight wild-card specification. This specification is as follows:

- File names must include numbers that are in ascending order from beginning to end.
- Numbers in the files names must be zero filled if there is more than one significant digit.
- Numbers can be anywhere in the file name.
- When the file name is specified in the EnSight result file, you must replace the numbers in the file with an asterisk(\*). The number of asterisks specified is the number of significant digits. The asterisk must occupy the same place as the numbers in the file names.

## EnSight6 Variable File Format

EnSight6 variable files can either be per\_node or per\_element. They cannot be both. However, an EnSight model can have some variables which are per\_node and other variables which are per\_element.



## EnSight6 Per\_Node Variable File Format

EnSight6 variable files for per\_node variables contain any values for each unstructured node followed by any values for each structured node.

First comes a single description line.

Second comes any unstructured node value. The number of values per node depends on the type of field. An unstructured scalar field has one, a vector field has three (order: x,y,z), a 2nd order symmetric tensor field has 6 (order: 11, 22, 33, 12, 13, 23), and a 2nd order asymmetric tensor field has 9 values per node (order: 11, 12, 13, 21, 22, 23, 31, 32, 33). An unstructured complex variable in EnSight6 consists of two scalar or vector fields (one real and one imaginary), with scalar and vector values written to their separate files respectively.

Third comes any structured data information, starting with a part # line, followed by a line containing the “block”, and then lines containing the values for each structured node which are output in the same IJK component order as the coordinates. Briefly, a structured scalar is the same as an unstructured scalar, one value per node. A structured vector is written one value per node per component, thus three sequential scalar field blocks. Likewise for a structured 2nd order symmetric tensor, written as six sequential scalar field blocks, and a 2nd order tensor, written as nine sequential scalar field blocks. The same methodology applies for a complex variable only with the real and imaginary fields written to separate structured scalar or vector files.

The values **must be written** in the following floating point format (**6 per line** as shown in the examples below):

From C: 12.5e format

From FORTRAN: e12.5 format

The format of a per\_node variable file is as follows:

- Line 1  
This line is a description line.
- Line 2 through the end of the file contains the values at each node in the model.

A generic example for *per\_node* variables:

```
One description line for the entire file
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+**
part #
block
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+**
```

*Note that there is a format difference between the unstructured and structured*

(block) portions of the vector and tensor data. For example a multiple component unstructured vector appears as  $x\ y\ z$  triplets, while the structured counterpart lists all  $x$  then all  $y$  and finally all  $z$ .

The following variable file examples reflect scalar, vector, tensor, and complex variable values *per node* for the previously defined EnSight6 Geometry File Example with 11 defined unstructured nodes and a 2x3x2 structured Part (Part number 3). The values are summarized in the following table.

	ComplexScalar						
	Node	Node	Scalar	Vector	Tensor (2nd order symm.)	Real	Imaginary
	Index	Id	Value	Values	Values	Value	Value
Unstructured							
	1	15	(1.)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.3, 1.4, 1.5, 1.6)	(1.1)	(1.2)
	2	31	(2.)	(2.1, 2.2, 2.3)	(2.1, 2.2, 2.3, 2.4, 2.5, 2.6)	(2.1)	(2.2)
	3	20	(3.)	(3.1, 3.2, 3.3)	(3.1, 3.2, 3.3, 3.4, 3.5, 3.6)	(3.1)	(3.2)
	4	40	(4.)	(4.1, 4.2, 4.3)	(4.1, 4.2, 4.3, 4.4, 4.5, 4.6)	(4.1)	(4.2)
	5	22	(5.)	(5.1, 5.2, 5.3)	(5.1, 5.2, 5.3, 5.4, 5.5, 5.6)	(5.1)	(5.2)
	6	44	(6.)	(6.1, 6.2, 6.3)	(6.1, 6.2, 6.3, 6.4, 6.5, 6.6)	(6.1)	(6.2)
	7	55	(7.)	(7.1, 7.2, 7.3)	(7.1, 7.2, 7.3, 7.4, 7.5, 7.6)	(7.1)	(7.2)
	8	60	(8.)	(8.1, 8.2, 8.3)	(8.1, 8.2, 8.3, 8.4, 8.5, 8.6)	(8.1)	(8.2)
	9	61	(9.)	(9.1, 9.2, 9.3)	(9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	(9.1)	(9.2)
	10	62	(10.)	(10.1, 10.2, 10.3)	(10.1, 10.2, 10.3, 10.4, 10.5, 10.6)	(10.1)	(10.2)
	11	63	(11.)	(11.1, 11.2, 11.3)	(11.1, 11.2, 11.3, 11.4, 11.5, 11.6)	(11.1)	(11.2)
Structured							
	1	1	(1.)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.3, 1.4, 1.5, 1.6)	(1.1)	(1.2)
	2	2	(2.)	(2.1, 2.2, 2.3)	(2.1, 2.2, 2.3, 2.4, 2.5, 2.6)	(2.1)	(2.2)
	3	3	(3.)	(3.1, 3.2, 3.3)	(3.1, 3.2, 3.3, 3.4, 3.5, 3.6)	(3.1)	(3.2)
	4	4	(4.)	(4.1, 4.2, 4.3)	(4.1, 4.2, 4.3, 4.4, 4.5, 4.6)	(4.1)	(4.2)
	5	5	(5.)	(5.1, 5.2, 5.3)	(5.1, 5.2, 5.3, 5.4, 5.5, 5.6)	(5.1)	(5.2)
	6	6	(6.)	(6.1, 6.2, 6.3)	(6.1, 6.2, 6.3, 6.4, 6.5, 6.6)	(6.1)	(6.2)
	7	7	(7.)	(7.1, 7.2, 7.3)	(7.1, 7.2, 7.3, 7.4, 7.5, 7.6)	(7.1)	(7.2)
	8	8	(8.)	(8.1, 8.2, 8.3)	(8.1, 8.2, 8.3, 8.4, 8.5, 8.6)	(8.1)	(8.2)
	9	9	(9.)	(9.1, 9.2, 9.3)	(9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	(9.1)	(9.2)
	10	10	(10.)	(10.1, 10.2, 10.3)	(10.1, 10.2, 10.3, 10.4, 10.5, 10.6)	(10.1)	(10.2)
	11	11	(11.)	(11.1, 11.2, 11.3)	(11.1, 11.2, 11.3, 11.4, 11.5, 11.6)	(11.1)	(11.2)

**Per\_node (Scalar) Variable Example 1** This example shows ASCII scalar file (en6.Nsca) for the geometry example.

```
Per_node scalar values for the EnSight6 geometry example
1.00000E+00 2.00000E+00 3.00000E+00 4.00000E+00 5.00000E+00 6.00000E+00
7.00000E+00 8.00000E+00 9.00000E+00 1.00000E+01 1.10000E+01
part 3
block
1.00000E+00 2.00000E+00 3.00000E+00 4.00000E+00 5.00000E+00 6.00000E+00
7.00000E+00 8.00000E+00 9.00000E+00 1.00000E+01 1.10000E+01
```

**Per\_node (Vector) Variable Example 2** This example shows ASCII vector file (en6.Nvec) for the geometry example.

```
Per_node vector values for the EnSight6 geometry example
1.10000E+00 1.20000E+00 1.30000E+00 2.10000E+00 2.20000E+00 2.30000E+00
3.10000E+00 3.20000E+00 3.30000E+00 4.10000E+00 4.20000E+00 4.30000E+00
5.10000E+00 5.20000E+00 5.30000E+00 6.10000E+00 6.20000E+00 6.30000E+00
7.10000E+00 7.20000E+00 7.30000E+00 8.10000E+00 8.20000E+00 8.30000E+00
9.10000E+00 9.20000E+00 9.30000E+00 1.01000E+01 1.02000E+01 1.03000E+01
1.11000E+01 1.12000E+01 1.13000E+01
part 3
```

```

block
1.10000E+00 2.10000E+00 3.10000E+00 4.10000E+00 5.10000E+00 6.10000E+00
7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01 1.11000E+01
1.20000E+00 2.20000E+00 3.20000E+00 4.20000E+00 5.20000E+00 6.20000E+00
7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01 1.12000E+01
1.30000E+00 2.30000E+00 3.30000E+00 4.30000E+00 5.30000E+00 6.30000E+00
7.30000E+00 8.30000E+00 9.30000E+00 1.03000E+01 1.13000E+01

```

**Per\_node (Tensor) Variable Example 3** This example shows an ASCII 2nd order symmetric tensor file (`en6.Nten`) for the geometry example.

```

Per_node symmetric tensor values for the EnSight6 geometry example
1.10000E+00 1.20000E+00 1.30000E+00 1.40000E+00 1.50000E+00 1.60000E+00
2.10000E+00 2.20000E+00 2.30000E+00 2.40000E+00 2.50000E+00 2.60000E+00
3.10000E+00 3.20000E+00 3.30000E+00 3.40000E+00 3.50000E+00 3.60000E+00
4.10000E+00 4.20000E+00 4.30000E+00 4.40000E+00 4.50000E+00 4.60000E+00
5.10000E+00 5.20000E+00 5.30000E+00 5.40000E+00 5.50000E+00 5.60000E+00
6.10000E+00 6.20000E+00 6.30000E+00 6.40000E+00 6.50000E+00 6.60000E+00
7.10000E+00 7.20000E+00 7.30000E+00 7.40000E+00 7.50000E+00 7.60000E+00
8.10000E+00 8.20000E+00 8.30000E+00 8.40000E+00 8.50000E+00 8.60000E+00
9.10000E+00 9.20000E+00 9.30000E+00 9.40000E+00 9.50000E+00 9.60000E+00
1.01000E+01 1.02000E+01 1.03000E+01 1.04000E+01 1.05000E+01 1.06000E+01
1.11000E+01 1.12000E+01 1.13000E+01 1.14000E+01 1.15000E+01 1.16000E+01
part 3
block
1.10000E+00 2.10000E+00 3.10000E+00 4.10000E+00 5.10000E+00 6.10000E+00
7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01 1.11000E+01
1.20000E+00 2.20000E+00 3.20000E+00 4.20000E+00 5.20000E+00 6.20000E+00
7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01 1.12000E+01
1.30000E+00 2.30000E+00 3.30000E+00 4.30000E+00 5.30000E+00 6.30000E+00
7.30000E+00 8.30000E+00 9.30000E+00 1.03000E+01 1.13000E+01
1.40000E+00 2.40000E+00 3.40000E+00 4.40000E+00 5.40000E+00 6.40000E+00
7.40000E+00 8.40000E+00 9.40000E+00 1.04000E+01 1.14000E+01
1.50000E+00 2.50000E+00 3.50000E+00 4.50000E+00 5.50000E+00 6.50000E+00
7.50000E+00 8.50000E+00 9.50000E+00 1.05000E+01 1.15000E+01
1.60000E+00 2.60000E+00 3.60000E+00 4.60000E+00 5.60000E+00 6.60000E+00
7.60000E+00 8.60000E+00 9.60000E+00 1.06000E+01 1.16000E+01

```

**Per\_node (Complex) Variable Example 4** This example shows the ASCII complex real (`en6.Ncmp_r`) and imaginary (`en6.Ncmp_i`) *scalar* files for the geometry example. (The same methodology would apply for complex real and imaginary *vector* files.)

Real scalar File:

```

Per_node complex real scalar values for the EnSight6 geometry example
1.10000E+00 2.10000E+00 3.10000E+00 4.10000E+00 5.10000E+00 6.10000E+00
7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01 1.11000E+01
part 3
block
1.10000E+00 2.10000E+00 3.10000E+00 4.10000E+00 5.10000E+00 6.10000E+00
7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01 1.11000E+01 1.21000E+00

```

Imaginary scalar File:

```

Per_node complex imaginary scalar values for the EnSight6 geometry example
1.20000E+00 2.20000E+00 3.20000E+00 4.20000E+00 5.20000E+00 6.20000E+00
7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01 1.12000E+01
part 3
block
1.20000E+00 2.20000E+00 3.20000E+00 4.20000E+00 5.20000E+00 6.20000E+00
7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01 1.12000E+01 1.22000E+00

```

## EnSight6 Per\_Element Variable File Format

EnSight variable files for `per_element` variables contain values for each element of designated types of designated Parts. First comes a single description line. Second comes a Part line. Third comes an element type line and fourth comes the value for each element of that type and part. If it is a scalar variable, there is one value per element, while for vector variables there are three values per element. (The number of elements of the given type are obtained from the corresponding EnSight6 geometry file.)

The values must be written in the following floating point format (6 per line as shown in the examples below):

From C: 12.5e format

From FORTRAN: e12.5 format

The format of a `per_element` variable file is as follows:

- Line 1 This line is a description line.
- Line 2 Part line, with part number corresponding to the geometry file.
- Line 3 Element type line ( example: `tria3`, `hexa8`, ... )
- Line 4 Repeats until next element type line, part line, or end of file is reached. Lists values for each element of this part and type.

A generic example for *per\_element* variables:

```
One description line for the entire file
part #
element type
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+**
part #
block
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+**
```

*Note that there is a format difference between the unstructured and structured (block) portions of the vector and tensor data. For example a multiple component unstructured vector appears as  $x\ y\ z$  triplets, while the structured counterpart lists all  $x$  then all  $y$  and finally all  $z$ .*

The following variable file examples reflect scalar, vector, tensor, and complex variable values *per element* for the previously defined EnSight6 Geometry File Example with 11 defined unstructured nodes and a 2x3x2 structured Part (Part number 3). The values are summarized in the following table.

		Element	Element	Scalar	Vector	Tensor (2nd order symm.)	Complex Scalar	
		Index	Id	Value	Values	Values	Real	Imaginary
							Value	Value
Unstructured								
	bar2	1	101	(1.)	(1.1, 1.2, 1.3)	(1.1, 1.2, 1.3, 1.4, 1.5, 1.6)	(1.1)	(1.2)
	tria3	1	102	(2.)	(2.1, 2.2, 2.3)	(2.1, 2.2, 2.3, 2.4, 2.5, 2.6)	(2.1)	(2.2)
		2	103	(3.)	(3.1, 3.2, 3.3)	(3.1, 3.2, 3.3, 3.4, 3.5, 3.6)	(3.1)	(3.2)
	hexa8	1	104	(4.)	(4.1, 4.2, 4.3)	(4.1, 4.2, 4.3, 4.4, 4.5, 4.6)	(4.1)	(4.2)
Structured								
	block	1	1	(5.)	(5.1, 5.2, 5.3)	(5.1, 5.2, 5.3, 5.4, 5.5, 5.6)	(5.1)	(5.2)
		2	2	(6.)	(6.1, 6.2, 6.3)	(6.1, 6.2, 6.3, 6.4, 6.5, 6.6)	(6.1)	(6.2)
		3	3	(7.)	(7.1, 7.2, 7.3)	(7.1, 7.2, 7.3, 7.4, 7.5, 7.6)	(7.1)	(7.2)
		4	4	(8.)	(8.1, 8.2, 8.3)	(8.1, 8.2, 8.3, 8.4, 8.5, 8.6)	(8.1)	(8.2)
		5	5	(9.)	(9.1, 9.2, 9.3)	(9.1, 9.2, 9.3, 9.4, 9.5, 9.6)	(9.1)	(9.2)
		6	6	(10.)	(10.1, 10.2, 10.3)	(10.1, 10.2, 10.3, 10.4, 10.5, 10.6)	(10.1)	(10.2)
		7	7	(11.)	(11.1, 11.2, 11.3)	(11.1, 11.2, 11.3, 11.4, 11.5, 11.6)	(11.1)	(11.2)
		8	8	(12.)	(12.1, 12.2, 12.3)	(12.1, 12.2, 12.3, 12.4, 12.5, 12.6)	(12.1)	(12.2)

**Per\_element (Scalar) Variable Example 1** This example shows an ASCII scalar file (`en6.Esca`) for the geometry example.

```
Per_elem scalar values for the EnSight6 geometry example
part 1
tria3
  2.00000E+00 3.00000E+00
hexa8
  4.00000E+00
part 2
bar2
  1.00000E+00
part 3
block
  5.00000E+00 6.00000E+00 7.00000E+00 8.00000E+00 9.00000E+00 1.00000E+01
  1.10000E+01 1.20000E+01
```

**Per\_element (Vector) Variable Example 2** This example shows an ASCII vector file (`en6.Evec`) for the geometry example.

```
Per_elem vector values for the EnSight6 geometry example
part 1
tria3
  2.10000E+00 2.20000E+00 2.30000E+00 3.10000E+00 3.20000E+00 3.30000E+00
hexa8
  4.10000E+00 4.20000E+00 4.30000E+00
part 2
bar2
  1.10000E+00 1.20000E+00 1.30000E+00
part 3
block
  5.10000E+00 6.10000E+00 7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01
  1.11000E+01 1.21000E+01
```

```

5.20000E+00 6.20000E+00 7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01
1.12000E+01 1.22000E+01
5.30000E+00 6.30000E+00 7.30000E+00 8.30000E+00 9.30000E+00 1.03000E+01
1.13000E+01 1.23000E+01

```

**Per\_element (Tensor) Variable Example 3** This example shows the ASCII 2nd order symmetric tensor file (`en6.Eten`) for the geometry example.

```

Per_elem symmetric tensor values for the EnSight6 geometry example
part 1
tria3
2.10000E+00 2.20000E+00 2.30000E+00 2.40000E+00 2.50000E+00 2.60000E+00
3.10000E+00 3.20000E+00 3.30000E+00 3.40000E+00 3.50000E+00 3.60000E+00
hexa8
4.10000E+00 4.20000E+00 4.30000E+00 4.40000E+00 4.50000E+00 4.60000E+00
part 2
bar2
1.10000E+00 1.20000E+00 1.30000E+00 1.40000E+00 1.50000E+00 1.60000E+00
part 3
block
5.10000E+00 6.10000E+00 7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01
1.11000E+01 1.21000E+01
5.20000E+00 6.20000E+00 7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01
1.12000E+01 1.22000E+01
5.30000E+00 6.30000E+00 7.30000E+00 8.30000E+00 9.30000E+00 1.03000E+01
1.13000E+01 1.23000E+01
5.40000E+00 6.40000E+00 7.40000E+00 8.40000E+00 9.40000E+00 1.04000E+01
1.14000E+01 1.24000E+01
5.50000E+00 6.50000E+00 7.50000E+00 8.50000E+00 9.50000E+00 1.05000E+01
1.15000E+01 1.25000E+01
5.60000E+00 6.60000E+00 7.60000E+00 8.60000E+00 9.60000E+00 1.06000E+01
1.16000E+01 1.26000E+01

```

**Per\_element (Complex) Variable Example 4** This example shows the ASCII complex real (`en6.Ecmp_r`) and imaginary (`en6.Ecmp_i`) *scalar* files for the geometry example. (The same methodology would apply for complex real and imaginary *vector* files).

Real scalar File:

```

Per_elem complex real scalar values for the EnSight6 geometry example
part 1
tria3
2.10000E+00 3.10000E+00
hexa8
4.10000E+00
part 2
bar2
1.10000E+00
part 3
block
5.10000E+00 6.10000E+00 7.10000E+00 8.10000E+00 9.10000E+00 1.01000E+01
1.11000E+01 1.21000E+01

```

Imaginary scalar File:

```

Per_elem complex imaginary scalar values for the EnSight6 geometry example
part 1
tria3
2.20000E+00 3.20000E+00
hexa8
4.20000E+00
part 2
bar2
1.20000E+00
part 3
block
5.20000E+00 6.20000E+00 7.20000E+00 8.20000E+00 9.20000E+00 1.02000E+01
1.12000E+01 1.22000E+01

```



## EnSight6 Measured/Particle File Format

The format of a Measured/Particle geometry file is as follows:

- Line 1  
This line is a description line.
- Line 2  
Indicates that this file contains particle coordinates. The words “particle coordinates” should be entered on this line without the quotes.
- Line 3  
Specifies the number of Particles.
- Line 4 through the end of the file.  
Each line contains the ID and the X, Y, and Z coordinates of each Particle. The format of this line is “integer real real real” written out in the following format:

From C:                    %8d%12.5e%12.5e%12.5e format

From FORTRAN:        i8, 3e12.5 format

A generic measured/Particle geometry file is as follows:

```
A description line
particle coordinates
#_of_Particles
id xcoord ycoord zcoord
id xcoord ycoord zcoord
id xcoord ycoord zcoord
.
.
.
```

### *Measured Geometry Example*

The following illustrates a measured/Particle file with seven points:

```
This is a simple measured geometry file
particle coordinates
7
101 0.00000E+00 0.00000E+00 0.00000E+00
102 1.00000E+00 0.00000E+00 0.00000E+00
103 1.00000E+00 1.00000E+00 0.00000E+00
104 0.00000E+00 1.00000E+00 0.00000E+00
205 5.00000E-01 0.00000E+00 2.00000E+00
206 5.00000E-01 1.00000E+00 2.00000E+00
307 0.00000E+00 0.00000E+00 -1.50000E+00
```

### *Measured Variable Files*

Measured variable files use the same format as EnSight6 per\_node variable files.

## Writing EnSight6 Binary Files

This section describes the EnSight6 binary files. This format is used to increase the speed of reading data into EnSight.



For binary files, there is a header that designates the type of binary file. This header is: “C Binary” or “Fortran Binary.” This must be the first thing in the geometry file only. The format for the file is then essentially the same format as the ASCII format, with the following exceptions:

The ASCII format puts the node and element ids on the same “line” as the corresponding coordinates. The BINARY format writes all node id’s then all coordinates.

The ASCII format puts all element id’s of a type within a Part on the same “line” as the corresponding connectivity. The BINARY format writes all the element ids for that type, then all the corresponding connectivities of the elements.

FORTRAN binary files should be created as sequential access unformatted files.

Float arrays (such as coordinates and variable values) must be single precision. Double precision is not supported.

In all the descriptions of binary files that follow, the number on the left end of the line corresponds to the type of write of that line, according to the following code:

1. This is a write of 80 characters to the file:

C example: 

```
char buffer[80];
strcpy(buffer, "C Binary");
fwrite(buffer, sizeof(char), 80, file_ptr);
```

FORTRAN: 

```
character*80 buffer
buffer = "Fortran Binary"
write(10) buffer
```

2. This is a write of a single integer:

C example: 

```
fwrite(&num_nodes, sizeof(int), 1, file_ptr);
```

FORTRAN: 

```
write(10) num_nodes
```

3. This is a write of an integer array:

C example: 

```
fwrite(node_ids, sizeof(int), num_nodes, file_ptr);
```

FORTRAN: 

```
write(10) (node_ids(i), i=1, num_nodes)
```

4. This is a write of a float array:

C example: 

```
fwrite(coords, sizeof(float), 3*num_nodes, file_ptr);
```

FORTRAN: 

```
write(10) ((coords(i,j), i=1,3), j=1, num_nodes)
```

*Note: For EnSight6 format, when using **Fortran binary**, and writing arrays composed of components, such as coordinates, or vector values, it is **very important** that they all be written with a **single Fortran write statement**. If you instead were to write the coords in the statement above with a loop per component, such that the write statement is executed three times, like the following, **EnSight will not be able to read it!***

```
FORTRAN: do 200 i=1,3
           write(10) (coords(i,j), j=1, num_nodes)
200 continue
```

*EnSight6 Binary Geometry*

An EnSight binary geometry file contains information in the following order:

- (1) <C Binary/Fortran Binary>
- (1) description line 1
- (1) description line 2
- (1) node id <given/off/assign/ignore>
- (1) element id <given/off/assign/ignore>
- (1) coordinates
- (2) #\_of\_points
- (3) [point\_ids]
- (4) coordinate\_array *(For FORTRAN make sure only one write statement is used)*
- (1) part #
- (1) description line
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- :
- (1) part #
- (1) description line
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- (1) part #
- (1) description line
- (1) block [iblanke]
- (3) i j k
- (4) all i coords, all j coords, all k coords *(For FORTRAN make sure only one write statement is used)*
- (3) [iblanke]
- :

*Per\_node Binary Scalar* An EnSight6 binary scalar file contains information in the following order:

- (1) description line
- (4) scalar\_array for unstructured nodes
- (1) part #
- (1) block
- (4) scalar\_array for part's structured nodes

*Per\_node Binary Vector* An EnSight6 binary vector file contains information in the following order:

- (1) description line
- (4) vector\_array for unstructured nodes *(For FORTRAN make sure only one write statement is used)*

- (1) part #
- (1) block
- (4) vector\_array for part's structured nodes *(For FORTRAN make sure only one write statement is used)*

*Per\_node Binary Tensor* An EnSight6 binary tensor file contains information in the following order:

- (1) description line
- (4) tensor\_array for unstructured nodes *(For FORTRAN make sure only one write statement is used)*
- (1) part #
- (1) block
- (4) tensor\_array for part's structured nodes *(For FORTRAN make sure only one write statement is used)*

*Per\_node Binary Complex* An EnSight6 binary complex real and imaginary *scalar* files contain information in the following order: (The same methodology applies for the complex real and imaginary vector files.)

Real scalar file:

- (1) description line
- (4) real scalar\_array for unstructured nodes
- (1) part #
- (1) block
- (4) real scalar\_array for part's structured nodes

Imaginary scalar file:

- (1) description line
- (4) imaginary scalar\_array for unstructured nodes
- (1) part #
- (1) block
- (4) imaginary scalar\_array for part's structured nodes

*Per\_element Binary Scalar* An EnSight6 binary scalar file contains information in the following order:

- (1) description line
- (1) part #
- (1) element type (tria3, quad4, ...)
- (4) scalar\_array for elements of part and type
- (1) part #
- (1) block
- (4) scalar\_array for structured elements of part

*Per\_element Binary Vector* An EnSight6 binary vector file contains information in the following order:

- (1) description line
- (1) part #
- (1) element type (tria3, quad4, ...)
- (4) vector\_array for elements of part and type *(For FORTRAN make sure only one write statement is used)*
- (1) part #
- (1) block
- (4) vector\_array for structured elements of part *(For FORTRAN make sure only one write statement is used)*

*Per\_element Binary Tensor* An EnSight6 binary tensor file contains information in the following order:

- (1) description line
- (1) part #
- (1) element type (tria3, quad4, ...)
- (4) tensor\_array for unstructured elements of part and type *(For FORTRAN make sure only one write statement is used)*
- (1) part #
- (1) block
- (4) tensor\_array for structured elements of part and type *(For FORTRAN make sure only one write statement is used)*

*Per\_element Binary Complex* EnSight6 binary complex real and imaginary *scalar* files contain information in the following order: (The same methodology applies for the complex real and imaginary vector files.)

Real scalar file:

- (1) description line
- (1) part #
- (1) element type (tria3, quad4, ...)
- (4) real scalar\_array for unstructured elements of part and type
- (1) part #
- (1) block
- (4) real scalar\_array for structured elements of part and type

Imaginary scalar file:

- (1) description line
- (1) part #
- (1) element type (tria3, quad4, ...)
- (4) imaginary scalar\_array for unstructured elements of part and type
- (1) part #
- (1) block
- (4) imaginary scalar\_array for structured elements of part and type

*Binary Measured Geometry*

An EnSight6 binary measured/particle geometry file contains information in the following order:

- (1) <C Binary/Fortran Binary>
- (1) description line 1
- (1) particle coordinates
- (2) #\_of\_points
- (3) point\_ids
- (4) coordinate\_array *(For FORTRAN make sure only one write statement is used)*

*Binary Measured Variable Files*

EnSight6 binary measured/discrete particle scalar and vector files follow the same binary formats as EnSight6 model per-node scalar and vector files.

## 11.3 EnSight5 Format

Included in this section:

[EnSight5 General Description](#)

[EnSight5 Geometry File Format](#)

[EnSight5 Result File Format](#)

[EnSight5 Wild Card Name Specification](#)

[EnSight5 Variable File Format](#)

[EnSight5 Measured/Particle File Format](#)

[Writing EnSight5 Binary Files](#)

### EnSight5 General Description

*Note: The EnSight6 format replaces and includes all aspects of the older EnSight5 format. This description is included for completeness but use of the EnSight6 format with EnSight 6.x and later versions is encouraged!*

EnSight5 data consists of the following files:

- Geometry (required)
- Results (optional) (points to other variable files and possibly to changing geometry files)
- Measured (optional) (points to measured geometry and variable files)

The results file contains information concerning scalar and vector variables. EnSight makes no assumptions regarding the physical significance of the scalar and vector variables. These files can be from any discipline. For example, the scalar file can include such things as pressure, temperature, and stress. The vector file can be velocity, displacement, or any other vector data.

All variable results for EnSight5 are contained in disk files—one variable per file. Additionally, if there are multiple time steps, there must be a set of disk files for each time step.

Sources of EnSight5 data include the following:

- Data that can be translated to conform to the EnSight5 data format
- Data that originates from one of the translators supplied with the EnSight application

The EnSight5 format supports a defined element set as shown below. The data must be defined in this element set. Elements that do not conform to this set must either be subdivided or discarded.

### Supported EnSight5 Elements

The elements that are supported by the EnSight5 format are:

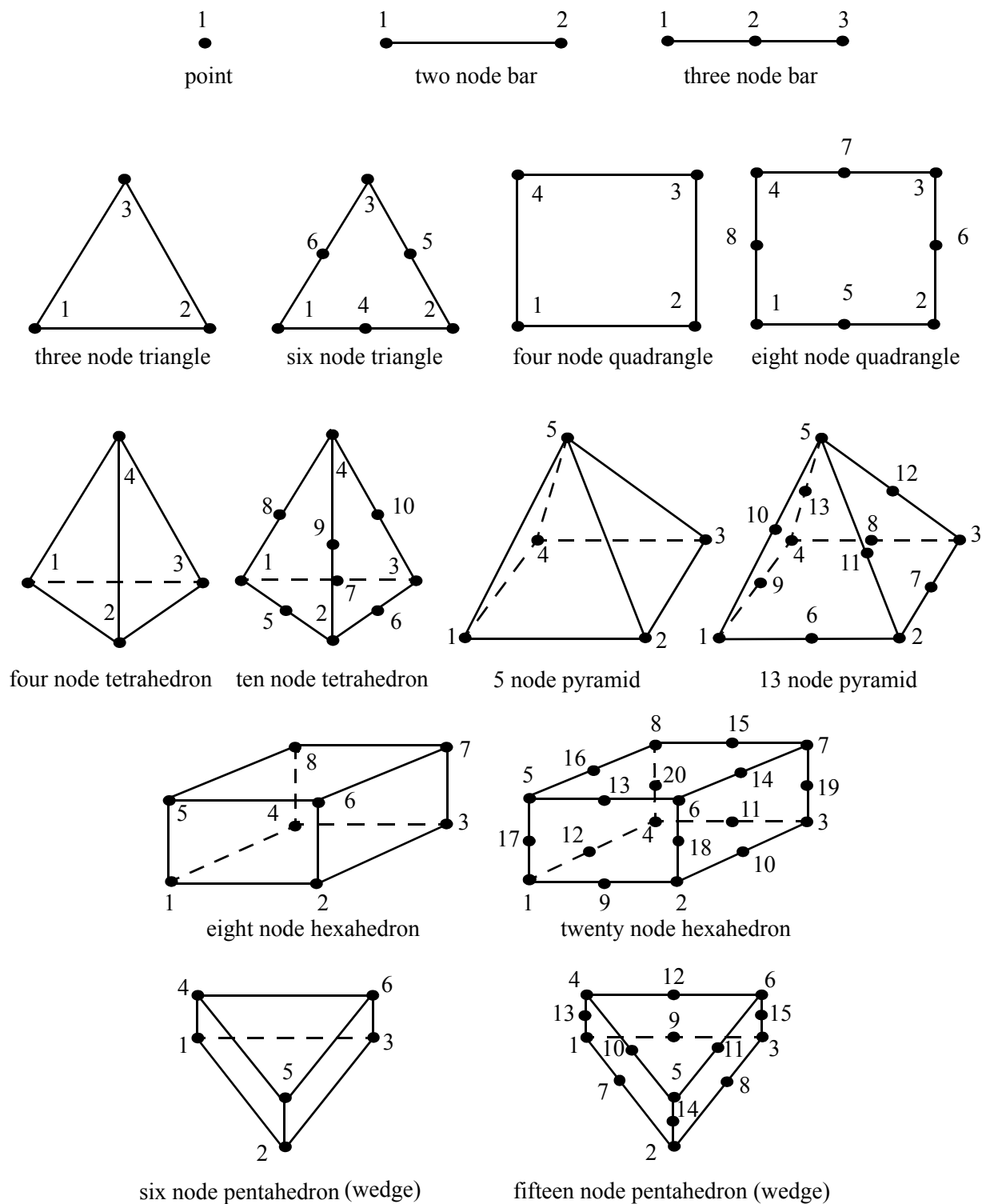


Figure 11-4  
Supported EnSight5 Elements

## EnSight5 Geometry File Format

The EnSight5 format consists of keywords followed by information. The following items are important to remember when working with EnSight5 geometry files:

1. You do not have to assign node IDs. If you do, the element connectivities are based on the node numbers. If you let EnSight assign the node IDs, the nodes are considered to be sequential starting at node 1, and element connectivity is done accordingly. If node IDs are set to off, they are numbered internally; however, you will not be able to display or query on them. If you have node IDs in your data, you can have EnSight ignore them by specifying “node id ignore.” Using this option may reduce some of the memory taken up by the Client and Server, but remember that display and query on the nodes will not be available.
2. You do not need to specify element IDs. If you specify element IDs, or you let EnSight assign them, you can show them on the screen. If they are set to off, you will not be able to show or query on them. If you have element IDs in your data you can have EnSight ignore them by specifying “element id ignore.” Using this option will reduce some of the memory taken up by the Client and Server. This may or may not be a significant amount, and remember that display and query on the elements will not be available.
3. The format of integers and real numbers **must be followed** (See the Geometry Example below).
4. Integers are written out using the following integer format:

From C: 8d format

From FORTRAN: i8 format

Real numbers are written out using the following floating-point format:

From C: 12.5e format

From FORTRAN: e12.5 format

**The number of integers or reals per line must also be followed!**

5. By default, a Part is processed to show the outside boundaries. This representation is loaded to the Client host system when the geometry file is read (unless other attributes have been set on the workstation, such as feature angle).
6. Coordinates must be defined before any Parts can be defined. The different elements can be defined in any order (that is, you can define a hexa8 before a bar2).

### *Generic Format*

Not all of the lines included in the following generic example file are necessary:

```
description line 1
description line 2
node id <off/given/assign/ignore>
element id <off/given/assign/ignore>
coordinates
# of points
id x y z
id x y z
```

## 11.3 EnSight5 Geometry File Format

```
id x y z
.
.
.
part #
description line
point
number of points
id nd
id nd
id nd
.
.
.
bar2
number of bar2's
id nd nd
id nd nd
id nd nd
.
.
.
bar3
number of bar3's
id nd nd nd
id nd nd nd
id nd nd nd
.
.
.
tria3
number of three node triangles
id nd nd nd
id nd nd nd
id nd nd nd
.
.
.
tria6
number of six node triangles
id nd nd nd nd nd nd
.
.
.
quad4
number of quad 4's
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
.
.
.
quad8
number of quad 8's
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
.
.
```



```

.
tetra4
number of 4 node tetrahedrons
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
id nd nd nd nd
.
.
.
tetra10
number of 10 node tetrahedrons
id nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd
.
.
.
pyramid5
number of 5 node pyramids
id nd nd nd nd nd
id nd nd nd nd nd
id nd nd nd nd nd
id nd nd nd nd nd
.
.
.
pyramid13
number of 13 node pyramids
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd
.
.
.
hexa8
number of 8 node hexahedrons
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd
.
.
.
hexa20
number of 20 node hexahedrons
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
.
.
.
penta6
number of 6 node pentahedrons
id nd nd nd nd nd nd
id nd nd nd nd nd nd
id nd nd nd nd nd nd

```

```
id nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd
.
.
.
penta15
number of 15 node pentahedrons
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
id nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd nd
.
.
.
```

### *EnSight5 Geometry Example*

The following is an example of an EnSight geometry file:

```
this is an example problem
this is the second description line
node id given
element id given
coordinates
  10
    5 1.00000e+00 0.00000e+00 0.00000e+00
  100 0.00000e+00 1.00000e+00 0.00000e+00
  200 0.00000e+00 0.00000e+00 1.00000e+00
  40 1.00000e+00 1.00000e+00 0.00000e+00
  22 1.00000e+00 0.00000e+00 1.00000e+00
  1000 2.00000e+00 0.00000e+00 0.00000e+00
  55 0.00000e+00 2.00000e+00 0.00000e+00
  44 0.00000e+00 0.00000e+00 2.00000e+00
  202 2.00000e+00 2.00000e+00 0.00000e+00
  101 2.00000e+00 0.00000e+00 2.00000e+00
part 1
This is Part 1, a pretty strange Part
tria3
  2
    101    100    200    40
    201    101     5   1000
tetra4
  1
    102    100    202    101    1000
part 2
This is Part 2, it's pretty strange also
bar2
  1
    103    101    1000
```

## EnSight5 Result File Format

The Result file is an ASCII free format file that contains variable and time step information that pertains to a Particular geometry file. The following information is included in this file:

- Number of scalar variables
- Number of vector variables
- Number of time steps
- Starting file number extension and skip-by value
- Flag that specifies whether there is changing geometry
- Names of the files that contain the values of scalar and vector variables
- The names of the geometry files that will be used for the changing geometry.

The format of the EnSight5 result file is as follows:

- Line 1  
Contains the number of scalar variables, the number of vector variables and a geometry-changing flag. (If the geometry-changing flag is 0, the geometry of the model does not change over time. If it is 1, then there is connectivity changing geometry. If it is 2, then there is coordinate only changing geometry.)
- Line 2  
Indicates the number of time steps that are available.
- Line 3  
Lists the time that is associated with each time step. There must be the same number of values as are indicated in Line 2. This “line” can actually span several lines in the file. You do not have to have one very long line.
- Line 4  
Specified only if more than one time step is indicated in Line 2. The two values on this line indicate the file extension value for the first time step and the offset between files. If the values on this line are 0 5, the first time step available has a subscript of 0, the second time step available has a subscript of 5, the third time step has a subscript of 10, and so on.
- Line 5  
Contains the names of the geometry files that will be used for changing geometry. This line exists only if the flag on Line 1 is set to 1 or 2. The geometry file name must follow the EnSight5 wild card specification.
- Line 6 through Line [5+N] where N is the number of scalar variables specified in Line 1.  
  
List **BOTH** the file names **AND** variable description that correspond to each scalar variable. There must be a file name for each scalar variable that is specified in Line 1.

If there is more than one time step, the file name must follow the EnSight5 wild card specification. See Note below.

- Lines that follow the scalar variable files.

List the file names that correspond to each vector variable. There must be a file name for each vector variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight5 wild card specification. See Note below.

*Note: Variable descriptions have the following restrictions:*  
*The variable description is limited to 19 characters in the current release.*  
*Duplicate variable descriptions are not allowed.*  
*Leading and trailing white space will be eliminated.*  
*Variable descriptions must not start with a numeric digit.*  
*Variable descriptions must not contain any of the following reserved characters:*

```
(      [      +      @      !      *      $
)      ]      -      space    #      ^      /
```

The generic format of a result file is as follows:

```
#_of_scalars #_of_vectors geom_chang_flag
#_of_timesteps
time1 time2 time3 .....
start_file_# skip_by_value
geometry_file_name.geo**
scalar0_file_name**  description (19 characters max)
scalar1_file_name**  description
.
.
.
vector0_file_name**  description (19 characters max)
vector1_file_name**  description
.
```

#### EnSight5 Result File Example 1

The following example illustrates a result file specified for a non-changing geometry file with only one time step:

```
2 1 0
1
0.0
exone.scl0 pressure
exone.scl1 temperature
exone.dis0 velocity
```

#### EnSight5 Result File Example 2

This example illustrates a result file that specifies a connectivity changing geometry that has multiple time steps.

```
1 2 1
4
1.0 2.0 2.5 5.0
0 1
extwo.geom**
pres.scl** pressure
vel.dis** velocity
grad.dis** gradient
```

The following files would be needed for example 2:

```
extwo.geom00 pres.scl00 vel.dis00 grad.dis00
extwo.geom01 pres.scl01 vel.dis01 grad.dis01
extwo.geom02 pres.scl02 vel.dis02 grad.dis02
extwo.geom03 pres.scl03 vel.dis03 grad.dis03
```

### EnSight5 Wild Card Name Specification

If multiple time steps are involved, the file names must conform to the EnSight5 wild-card specification. This specification is as follows:

- File names must include numbers that are in ascending order from beginning to end.
- Numbers in the files names must be zero filled if there is more than one significant digit.
- Numbers can be anywhere in the file name.
- When the file name is specified in the EnSight5 result file, you must replace the numbers in the file with an asterisk(\*). The number of asterisks specified is the number of significant digits. The asterisk must occupy the same place as the numbers in the file names.

### EnSight5 Variable File Format

Variables files have one description line followed by a value for each node. For a scalar file there is one value per node, while for vector files there are three values per node.

The values **must be written** in the following floating point format (**6 per line** as shown in the examples below):

From C: 12.5e format

From FORTRAN: e12.5 format

The format of a variables file is as follows:

- Line 1  
This line is a description line.
- Line 2 through the end of the file contains the values at each node in the model. A generic example:

```
A description line
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
*.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+** *.*****E+**
```

#### EnSight5 Variable File Example 1

This example shows a scalar file for a geometry with seven defined nodes.

```
These are the pressure values for a 7 node geometry
1.00000E+00 2.00000E+00 3.00000E+00 4.00000E+00 5.00000E+00 6.00000E+00
7.00000E+00
```

#### EnSight5 Variable File Example 2

This example shows the vector file for a geometry with seven defined nodes.

```
These are the velocity values for a 7 node geometry
1.00000E+00 1.00000E+00 1.00000E+00 2.00000E+00 2.00000E+00 2.00000E+00
3.00000E+00 3.00000E+00 3.00000E+00 4.00000E+00 4.00000E+00 4.00000E+00
```

```
5.00000E+00 5.00000E+00 5.00000E+00 6.00000E+00 6.00000E+00 6.00000E+00
7.00000E+00 7.00000E+00 7.00000E+00
```

## EnSight5 Measured/Particle File Format

This file allows you to define Particle locations, sizes, etc. to display with the geometry. Typical uses are fuel droplets for combustion analysis or data derived from experiments on prototypes.

The measured/Particle files consist of the following:

- Measured/Particle geometry file (referenced by the measured results file)
- Measured/Particle results file (the filename which is put into the Data Reader's "(Set) Measured" field)
- Measured/Particle variables file (referenced by the measured results file)

The format of the EnSight5 Measured/Particle geometry file is described below.

Note that there is only one description line and there *must* be an ID for each measured point.

Note also that the number of Particles can be different in each of the geometry file (if you have transient data), however, the number of values in each of the corresponding variable files must coincide, and the IDs of the Particles must be consistent in order to track the Particles at intermediate times or locations.

The format of an EnSight5 Measured/Particle geometry file is as follows:

- Line 1  
This line is a description line.
- Line 2  
Indicates that this file contains Particle coordinates. The words "particle coordinates" should be entered on this line without the quotes.
- Line 3  
Specifies the number of Particles.
- Line 4 through the end of the file.  
Each line contains the ID and the X, Y, and Z coordinates of each Particle. The format of this line is "integer real real real" written out in the following format:

From C:                    %8d%12.5e%12.5e%12.5e format

From FORTRAN:    i8, 3e12.5 format

A generic measured/Particle geometry file is as follows:

```
A description line
particle coordinates
#_of_Particles
id xcoord ycoord zcoord
id xcoord ycoord zcoord
id xcoord ycoord zcoord
```

.

.

.

### *EnSight5 Measured Geometry/Particle File Example*

The following illustrates an EnSight5 Measured Geometry/Particle file with seven points:

This is a simple ensight5 measured geometry/particle file  
particle coordinates

```

7
101 0.00000E+00 0.00000E+00 0.00000E+00
102 1.00000E+00 0.00000E+00 0.00000E+00
103 1.00000E+00 1.00000E+00 0.00000E+00
104 0.00000E+00 1.00000E+00 0.00000E+00
205 5.00000E-01 0.00000E+00 2.00000E+00
206 5.00000E-01 1.00000E+00 2.00000E+00
307 0.00000E+00 0.00000E+00 -1.50000E+00

```

### *EnSight5 Measured/ Particle File Format*

The format of the EnSight5 Measured/Particle results file is as follows:

- Line 1  
Contains the number of scalar variables, the number of vector variables, and a measured geometry changing flag. If the measured geometry changing flag is 0, only one time step is indicated.
- Line 2  
Indicates the number of available time steps.
- Line 3  
Lists the time that is associated with each time step. The time step information does not have to coincide with the model time step information. This “line” can actually span several lines in the file. You do not have to have one very long line.
- Line 4  
Specified only if Line 2 specifies more than one time step. The line contains two values; the first value indicates the file extension value for the first time step, and the second value indicates the offset between files. If this line contains the values 0 and 5, the first time step has a subscript of 0, the second of 5, the third of 10, and so on.
- Line 5  
Contains the name of the measured geometry file. If there is more than one time step, the file name must follow the EnSight wild card specification.
- Line 6 through Line [5+N] where N is the number of scalar variables specified in Line 1.  
List the file names that correspond to each scalar variable. There must be a file name for each scalar variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight wild card

specification.

- Lines that follow the scalar variable files.

List the names of the files that correspond to each vector variable. There must be a file name for each vector variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight wild card specification.

A generic EnSight5 Measured/Particle results file is as follows:

```
#_of_scalars #_of_vectors geom_chang_flag
#_of_timesteps
time1 time2 time3 .....
start_file_# skip_by_value
measured_geom_file_name**
scalar0_file_name**  description
scalar1_file_name**  description
.
.
.
vector0_file_name**  description
vector1_file_name**  description
.
.
.
```

#### *Measured/Particle Results File Example 1*

This example illustrates an EnSight5 Measured/Particle result file that specifies a non-changing geometry with only one time step:

```
2 1 0
1
0.0
exone.geom
exone.scl0 pressure
exone.scl1 temperature
exone.dis0 velocity
```

#### *Measured/Particle Results File Example 2*

This example illustrates an EnSight5 Measured/Particle result file that specifies a changing geometry with multiple time steps:

```
1 2 1
4
1.0 2.0 2.5 5.0
0 1
extwo.geom**
pres.scl** pressure
vel.dis** velocity
grad.dis** gradient
```

The following files are needed for Example 2:

extwo.geom00pres.scl00vel.dis00	grad.dis00
extwo.geom01pres.scl01vel.dis01	grad.dis01
extwo.geom02pres.scl02vel.dis02	grad.dis02
extwo.geom03pres.scl03vel.dis03	grad.dis03

#### *Measured /Particle Results Variable files*

The EnSight5 Measured/Particle variable files referred to in the measured Results file follow the same format as EnSight5 Variable files. The number of values in



each of these variable files must correspond properly to the number of Particles in the corresponding measured geometry files.

## Writing EnSight5 Binary Files

This section describes the EnSight5 binary files. This format is used to increase the speed of reading data into EnSight. A utility exists for converting EnSight5 ASCII files to EnSight5 binary files—it is called `asciitobin5` and is found on the release tape under `ensight/server/utilities/asciitobin5`.

For binary files, there is a header that designates the type of binary file. This header is: “C Binary” or “Fortran Binary.” This must be the first thing in the file. The format for the file is then essentially the same format as the ASCII format, with the following exceptions:

The ASCII format puts the node and element ids on the same “line” as the corresponding coordinates. The BINARY format writes all node id’s then all coordinates.

The ASCII format puts all element id’s of a type within a Part on the same “line” as the corresponding connectivity. The BINARY format writes all the element ids for that type, then all the corresponding connectivities of the elements.

In all the descriptions of binary files that follow, the number on the left end of the line corresponds to the type of write of that line, according to the following code:

1. This is a write of 80 characters to the file:

C example: 

```
char buffer[80];
strcpy(buffer,"C Binary");
fwrite(buffer,sizeof(char),80,file_ptr);
```

FORTTRAN: 

```
character*80  buffer
buffer = "Fortran Binary"
write(10) buffer
```

2. This is a write of a single integer:

C example: 

```
fwrite(&num_nodes,sizeof(int),1,file_ptr);
```

FORTTRAN: 

```
write(10) num_nodes
```

3. This is a write of an integer array:

C example: 

```
fwrite(node_ids,sizeof(int),num_nodes,file_ptr);
```

FORTTRAN: 

```
write(10) (node_ids(i),i=1,num_nodes)
```

4. This is a write of a float array:

C example: 

```
fwrite(coords,sizeof(float),3*num_nodes,file_ptr);
```

FORTTRAN: 

```
write(10) ((coords(i,j),i=1,3),j=1,num_nodes)
```

(Note: *Coords* is a single precision array, double precision will not work!)

### *EnSight5 Binary*

*Geometry File Format* An EnSight5 binary geometry file contains information in the following order:

- (1) <C Binary/Fortran Binary>
- (1) description line 1
- (1) description line 2
- (1) node id <given/off/assign/ignore>
- (1) element id <given/off/assign/ignore>
- (1) coordinates
- (2) #\_of\_points
- (3) [point\_ids]
- (4) coordinate\_array
- (1) part #
- (1) description line
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- .
- .
- .
- (1) part #
- (1) description line
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- (1) element\_type
- (2) #\_of\_element\_type
- (3) [element\_ids] for the element\_type
- (3) connectivities for the element\_type
- .
- .
- .

### *Binary Scalar*

An EnSight5 binary scalar file contains information in the following order:

- (1) description line
- (4) scalar\_array

### *Binary Vector*

An EnSight5 binary vector file contains information in the following order:

- (1) description line
- (4) vector\_array

*Binary Measured*

An EnSight5 binary measured/Particle geometry file contains information in the following order:

- (1) <C Binary/Fortran Binary>
- (1) description line 1
- (1) particle coordinates
- (2) #\_of\_points
- (3) point\_ids
- (4) coordinate\_array

## 11.4 FAST UNSTRUCTURED Results File Format

FAST UNSTRUCTURED input data consists of the following:

- Geometry file (required) (GRID file).
- Results file (optional).
- [EnSight5 Measured/Particle Files](#) (optional). The measured .res file references the measured geometry and variable files.

FAST UNSTRUCTURED data files can be read as:

Workstation: ASCII, C Binary, or FORTRAN binary

Cray: ASCII, C Binary, or COS-Blocked FORTRAN binary

Due to the different number of representations on a Cray Research vector system and workstations, binary files created on a Cray Research vector system can *not* be read on the workstation, and visa versa.

EnSight reads the geometry (grid files) directly. However, an EnSight-like results file is needed in order to read the results unless a “standard” Q-file is provided in its place. See FAST UNSTRUCTURED Result File below.

### *FAST UNSTRUCTURED Geometry file notes*

Only the single zone format can be read into EnSight. Any tetrahedral elements will be placed into the first “domain” Part. Triangular elements are placed into Parts based on their “tag” value.

The FAST UNSTRUCTURED solution file or function file formats can be used for variable results. The I J K values need to be I=Number of points and J=K=1. This does require the use of a modified EnSight results file as explained below.

Node and element numbers are assigned sequentially allowing for queries to be made within EnSight. Tetrahedron elements will be assigned before triangular elements.

### *FAST UNSTRUCTURED Result file format*

The FAST UNSTRUCTURED result file was defined by CEI and is very similar to the EnSight results file and contains information needed to relate variable names to variable files, step information, etc. There is a slight variation from the normal EnSight results file because of the differences between the solution (Q file) and function files. The difference lies on the lines which relate variable filenames to a description. These lines have the following format:

```
<filename> <type> <number(s)> <description>
```

See FAST UNSTRUCTURED Result File below for the definition of each.

The following information is included in a FAST UNSTRUCTURED result file:

- Number of scalar variables
- Number of vector variables
- Number of time steps

- Starting file number extension and skip-by value
- Flag that specifies whether there is changing geometry.
- Names of the files that contain the values of scalar and vector variables. An indication as to the type of the file being used for the variable, which variable in the file and the name given to that variable.
- The names of the geometry files that will be used for the changing geometry.

*Generic FAST UNSTRUCTURED Result File Format*

The format of the Result file is as follows:

- Line 1  
Contains the number of scalar variables, the number of vector variables and a geometry changing flag. If the geometry changing flag is 0, the geometry of the model does not change over time. If the flag is 1, the geometry can change connectivity. If the flag is 2, only coordinates can change.
- Line 2  
Indicates the number of time steps that are available. If this number is positive, then line 3 information must be present. If this number is negative, then Line 3 information must not be present and the times will be read from the solution file. Thus, one must have a solution file in one of the lines from Line 6 on.
- Line 3  
Lists the time that is associated with each time step. There must be the same number of values as are indicated in Line 2. This “line” can actually span several lines in the file. Specify only if Line 2 value is positive.
- Line 4  
Specified only if more than one time step is indicated in Line 2. The two values on this line indicate the file extension value for the first time step and the offset between files. If the values on this line are 0 5, the first time step available has a subscript of 0, the second time step available has a subscript of 5, the third time step has a subscript of 10, and so on.
- Line 5  
This line exists only if the changing geometry flag on Line 1 has been set to 1 or 2. Line contains name of the FAST UNSTRUCTURED grid file. The file name must follow the EnSight wild card specification.
- Line 6 through Line [5+N] where N is the number of scalar variables specified in Line 1.  
List the file names that correspond to each scalar variable. There must be a file name for each scalar variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight wild card specification.  
  
These lines also contain the type of file being used, solution or function, and the location of the variable value in the file. The contents are:

```
<filename> <type> <number> <description>
```

where filename is the name of solution file or function file containing the variable; type is “S” for solution file, or “F” for function file; number is which variable in the file to use (specify just one number); and description is the Description of the variable.

The solution file (“s”) is the traditional .q file in which normally the first variable is density, the second through fourth variables are the components of momentum, and the fifth variable is total energy.

- Lines that follow the scalar variable files.

List the file names that correspond to each vector variable. There must be a file name for each vector variable that is specified in Line 0. If there is more than one time step, the file name must follow the EnSight wild card specification.

These lines also contain the type of file being used, solution or function, and the location(s) of the variable values in the file. The contents are:

```
<filename> <type> <numbers> <description>
```

where filename is the name of solution file or function file containing the variable; type is “S” for solution file, or “F” for function file; numbers are which variables in the file to use (specify just three numbers); and description is the Description of the variable.

The generic format of the result file is as follows:

```
#_of_scalars #_of_vectors geom_chng_flag
#_of_timesteps
time1 time2 time3 .....
start_file # skip_by_value
geometry_file_name.geo**
scalar0_file_name** type # description
scalar1_file_name** type # description
.
.
.
vector0_file_name** type # # # description
vector1_file_name** type # # # description
.
.
.
```

**FAST UNSTRUCTURED** This example illustrates a result file that specifies a non-changing geometry with one time step.

*Example*

```
3 2 0
1
0.0
block.sol S 1 Density
block.sol S 5 Total_Energy
block.scl F 1 Temperature
block.var F 1 2 3 Displacement
block.sol S 2 3 4 Momentum
```

Thus, this model will get two scalars from the solution file (block.sol). The first is Density in the first location in the file and the next is Total energy in the fifth

location in the solution file. It will also get a Temperature scalar from the first location in the function file (block.scl).

It will get a Displacement vector from the function file called block.var. The three components of this vector are in the 1st, 2nd, and 3rd locations in the file. Finally, a Momentum vector will be obtained from the 2nd, 3rd, and 4th locations of the solution file.

Example 2 is somewhat similar, except that it is transient, with coordinate changing geometry. Note also that the times will come from the solution file.

```
3 2 2
-10
0 1
block***.grid
block***.sol S 1 Density
block***.sol S 5 Total_Energy
block***.scl F 1 Temperature
block***.var F 1 2 3 Displacement
block***.sol S 2 3 4 Momentum
```

## 11.5 FLUENT UNIVERSAL Results File Format

This section describes the FLUENT results file format and provides an example of this file. For transient cases, you *must* supply this result file. For static models this file is not required. The FLUENT result file is a slightly modified EnSight5 results file and provides a way to describe multiple time-step FLUENT Universal files to EnSight.

When using multiple FLUENT files with this result file definition, you *must* make sure that the files contain the same defined variables. In other words, any variable that exists in one must exist in all.

The Result file is an ASCII free format file that contains time step and universal file information for each available time step. The following information is included in this file:

- Number of time steps
- Simulation Time Values
- Starting file number extension and skip-by value
- Name of the universal file with EnSight wild card specification.

The format of the Result file is as follows:

- Line 1  
Indicates the number of time steps that are available.
- Line 2  
Lists the time that is associated with each time step. There must be the same number of values as are indicated in Line 1. This “line” can actually span several lines in the file. You do not have to have one very long line.
- Line 3  
Specified only if more than one time step is indicated in Line 1. The two values on this line indicate the file extension value for the first time step and the offset between files. If the values on this line are 0 5, the first time step available has a subscript of 0, the second time step available has a subscript of 5, the third time step has a subscript of 10, and so on.
- Line 4  
Contains the names of the universal file that will be used for the changing time step information. The universal file name must follow the EnSight5 wild card specification.

The generic format of the result file is as follows:

```
#_of_timesteps
time1 time2 time3 .....
start_file_# skip_by_value
universal_file_name***
```



*FLUENT Example*

This example illustrates a FLUENT result file

```
4
1.0 2.0 3.0 4.0
0 1
extwo**.uni
```

The following FLUENT universal files will need to exist for the result file:

```
extwo00.uni
extwo01.uni
extwo02.uni
extwo03.uni
```

## 11.6 Movie.BYU Results File Format

For transient cases, you must supply an EnSight result file. The result file for the Movie.BYU case is exactly the same as for EnSight5 (it is repeated below for your ease).

The Result file is an ASCII free format file that contains variable and time step information that pertains to a Particular geometry file. The following information is included in this file:

- Number of scalar variables
- Number of vector variables
- Number of time steps
- Starting file number extension and skip-by value
- Flag that specifies whether there is changing geometry
- Names of the files that contain the values of scalar and vector variables
- The names of the geometry files that will be used for the changing geometry.

The format of the Movie.BYU (EnSight5) result file is as follows:

- Line 1  
Contains the number of scalar variables, the number of vector variables and a geometry-changing flag. (If the geometry-changing flag is 0, the geometry of the model does not change over time. If it is 1, then there is connectivity changing geometry. If it is 2, then there is coordinate only changing geometry.)
- Line 2  
Indicates the number of time steps that are available.
- Line 3  
Lists the time that is associated with each time step. There must be the same number of values as are indicated in Line 2. This “line” can actually span several lines in the file. You do not have to have one very long line.
- Line 4  
Specified only if more than one time step is indicated in Line 2. The two values on this line indicate the file extension value for the first time step and the offset between files. If the values on this line are 0 5, the first time step available has a subscript of 0, the second time step available has a subscript of 5, the third time step has a subscript of 10, and so on.
- Line 5  
Contains the names of the geometry files that will be used for changing geometry. This line exists only if the flag on Line 1 is set to 1 or 2. The geometry file name must follow the EnSight5 wild card specification.
- Line 6 through Line [5+N] where N is the number of scalar variables specified in Line 1.

List **BOTH** the file names **AND** variable description that correspond to each scalar variable. There must be a file name for each scalar variable that is specified in Line 1.

If there is more than one time step, the file name must follow the EnSight5 wild card specification. See Note below.

- Lines that follow the scalar variable files.

List the file names that correspond to each vector variable. There must be a file name for each vector variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight5 wild card specification. See Note below.

*Note: Variable descriptions have the following restrictions:*  
*The variable description is limited to 19 characters in the current release.*  
*Duplicate variable descriptions are not allowed.*  
*Leading and trailing white space will be eliminated.*  
*Variable descriptions must not start with a numeric digit.*  
*Variable descriptions must not contain any of the following reserved characters:*

(	[	+	@	!	*	\$
)	]	-	space	#	^	/

The generic format of a result file is as follows:

```
#_of_scalars #_of_vectors geom_chang_flag
#_of_timesteps
time1 time2 time3 .....
start_file_# skip_by_value
geometry_file_name.geo**
scalar0_file_name**  description (19 characters max)
scalar1_file_name**  description
.
.
.
vector0_file_name**  description (19 characters max)
vector1_file_name**  description
.
```

#### Movie.BYU Result File Example 1

The following example illustrates a result file specified for a non-changing geometry file with only one time step:

```
2 1 0
1
0.0
exone.scl0 pressure
exone.scl1 temperature
exone.dis0 velocity
```

#### Movie.BYU Result File Example 2

This example illustrates a result file that specifies a connectivity changing geometry that has multiple time steps.

```
1 2 1
4
1.0 2.0 2.5 5.0
0 1
extwo.geom**
pres.scl** pressure
```

```
vel.dis** velocity  
grad.dis** gradient
```

The following files would be needed for example 2:

```
extwo.geom00 pres.scl00 vel.dis00 grad.dis00  
extwo.geom01 pres.scl01 vel.dis01 grad.dis01  
extwo.geom02 pres.scl02 vel.dis02 grad.dis02  
extwo.geom03 pres.scl03 vel.dis03 grad.dis03
```

## 11.7 PLOT3D Results File Format

PLOT3D input data consists of the following:

- Geometry file (required) (GRID file).
- Results file (optional).
- [EnSight5 Measured/Particle Files](#) (optional). The measured .res file references the measured geometry and variable files.

PLOT3D data files can be read as:

Workstation: ASCII, C Binary, or FORTRAN binary

Cray: ASCII, C Binary, or COS-Blocked FORTRAN binary

(see [PLOT3D Reader](#), in [Section 2.3](#))

Due to the different number of representations on a Cray Research vector system and workstations, binary files created on a Cray Research vector system can *not* be read on the workstation, and visa versa.

EnSight attempts to ensure that the format of the file being read matches the format you have selected in the Data Reader dialog. However, if you specify that the file is C binary, and it is really FORTRAN binary, this will not be detected and erroneous values will be loaded.

EnSight reads the geometry (xyz files) directly. However, an EnSight-like results file (described below) is needed in order to read the results, unless a “standard” Q-file is provided in its place.

### *PLOT3D Geometry file notes*

The following information is required in order to read PLOT3D files correctly:

1. whether there is Iblanking information in the file
2. whether files are in ASCII, C Binary, or FORTRAN binary
3. whether the file is “Single Zone” or Multi-Zoned”
4. whether the model is 1D, 2D, or 3D in nature.

Iblanking can be one of the following:

0 = Outside (Blanked Out)

1 = Inside

2 = Interior boundaries

<0 = zone that neighbors

If single zone with Iblanking, you can build EnSight Parts from the inside portions, blanked-out portions, or internal boundary portions. If single zone, you can also specify I, J, K limiting ranges for Parts to be created.

If Multi-zoned with Iblanking, you can additionally build Parts that are the boundary between two zones. (For boundary you must specify exactly two zones.)

If Multi-zoned and not using the “between boundary” option, a Part can span several zones.

If Multi-zoned, the dimension of the problem is forced to be 3D.

There can be nodes in different zones which have the same coordinates. No attempt has been made to merge these. Thus, on shared zone boundaries, there will likely be nodes on top of nodes. One negative effect of this is that node labels will be on top of each other.

Currently EnSight only prints out the global conditions in the solution file, fsmach, alpha, re, and time. It does not do anything else with them.

Node and element numbers are assigned in a sequential manner. Queries can be made on these node and element numbers or on nodes by I, J, and K.

### ***PLOT3D Result file format***

The PLOT3D result file was defined by CEI and is very similar to the EnSight results file and contains information needed to relate variable names to variable files, step information, etc. There is a slight variation from the normal EnSight results file because of the differences between the solution (Q file) and function files. The difference lies on the lines which relate variable filenames to a description. These lines have the following format:

```
<filename> <type> <number(s)> <description>
```

See PLOT3D Result File below for the definition of each.

The following information is included in a PLOT3D result file:

- Number of scalar variables
- Number of vector variables
- Number of time steps
- Starting file number extension and skip-by value
- Flag that specifies whether there is changing geometry.
- Names of the files that contain the values of scalar and vector variables. An indication as to the type of the file being used for the variable, which variable in the file and the name given to that variable.
- The names of the geometry files that will be used for the changing geometry.

### ***Generic PLOT3D Result File Format***

The format of the Result file is as follows:

- Line 1  
Contains the number of scalar variables, the number of vector variables and a geometry changing flag. If the geometry changing flag is 0, the geometry of the model does not change over time. Only the coordinates can change for a PLOT3D file at present time.
- Line 2  
Indicates the number of time steps that are available.

- Line 3

Lists the time that is associated with each time step. There must be the same number of values as are indicated in Line 2. This “line” can actually span several lines in the file.

- Line 4

Specified only if more than one time step is indicated in Line 2. The two values on this line indicate the file extension value for the first time step and the offset between files. If the values on this line are 0 5, the first time step available has a subscript of 0, the second time step available has a subscript of 5, the third time step has a subscript of 10, and so on.

- Line 5

This line exists only if the changing geometry flag on Line 1 has been set to 1. Line contains name of the PLOT3D xyz file. The file name must follow the EnSight wild card specification.

- Line 6 through Line [5+N] where N is the number of scalar variables specified in Line 1.

List the file names that correspond to each scalar variable. There must be a file name for each scalar variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight wild card specification.

These lines also contain the type of file being used, solution or function, and the location of the variable value in the file. The contents are:

```
<filename> <type> <number> <description>
```

where filename is the name of solution file or function file containing the variable; type is “S” for solution file, or “F” for function file; number is which variable in the file to use (specify just one number); and description is the Description of the variable.

The solution file (“s”) is the traditional .q file in which normally the first variable is density, the second through fourth variables are the components of momentum, and the fifth variable is total energy.

- Lines that follow the scalar variable files.

List the file names that correspond to each vector variable. There must be a file name for each vector variable that is specified in Line 1. If there is more than one time step, the file name must follow the EnSight wild card specification.

These lines also contain the type of file being used, solution or function, and the location(s) of the variable values in the file. The contents are:

```
<filename> <type> <numbers> <description>
```

where filename is the name of solution file or function file containing the variable; type is “S” for solution file, or “F” for function file; numbers are which variables in the file to use (specify just three numbers); and description is the Description of the variable.

The generic format of the result file is as follows:

```
#_of_scalars #_of_vectors geom_chng_flag
#_of_timesteps
time1 time2 time3 .....
start_file_# skip_by_value
geometry_file_name.geo**
scalar0_file_name** type # description
scalar1_file_name** type # description
.
.
.
vector0_file_name** type # # # description
vector1_file_name** type # # # description
.
.
.
```

### *PLOT3D Example*

This example illustrates a result file that specifies a non-changing geometry with only one time step.

```
3 2 0
1
0.0
block.sol S 1 Density
block.sol S 5 Total_Energy
block.scl F 1 Temperature
block.var F 1 2 3 Displacement
block.sol S 2 3 4 Momentum
```

Thus, this model will get two scalars from the solution file (block.sol). The first is Density in the first location in the file and the next is Total energy in the fifth location in the solution file. It will also get a Temperature scalar from the first location in the function file (block.scl).

It will get a Displacement vector from the function file called block.var. The three components of this vector are in the 1st, 2nd, and 3rd locations in the file. Finally, a Momentum vector will be obtained from the 2nd, 3rd, and 4th locations of the solution file.

Vectors can be 1D, 2D, or 3D. For a vector, always provide three numbers, but a zero will indicate that a component is empty, thus:

```
block.var F 1 0 3 XZ_Displacement
```

would be a 2D vector variable with components only in the X–Z plane.

If the above example had transient variables (but not geometry), with 3 time steps, it would appear as:

```
3 2 0
3
0.0 1.5 4.0
1 1
block.sol** S 1 Density
block.sol** S 5 Total_Energy
block.scl** F 1 Temperature
block.var** F 1 2 3 Displacement
block.sol** S 2 3 4 Momentum
```



The files needed would then be:

block.sol01	block.scl01	block.var01
block.sol02	block.scl02	block.var02
block.sol03	block.scl03	block.var03

And if the geometry changed as well as the variables, it would appear as:

```

3 2 1
3
0.0 1.5 4.0
1 1
block.geo**
block.sol** S 1 Density
block.sol** S 5 Total_Energy
block.scl** F 1 Temperature
block.var** F 1 2 3 Displacement
block.sol** S 2 3 4 Momentum

```

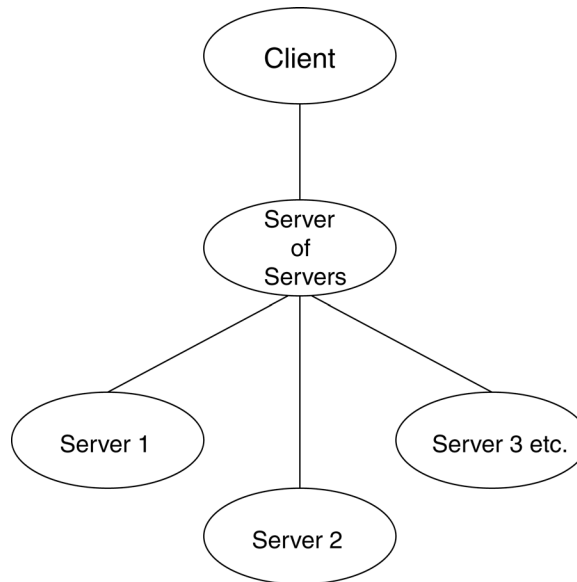
The files needed would then be:

block.sol01	block.scl01	block.var01	block.geo01
block.sol02	block.scl02	block.var02	block.geo02
block.sol03	block.scl03	block.var03	block.geo03

*Note: A “standard” Q-file can be substituted for PLOT3D result file format if desired. A “standard” Q-file has 5 variable components (First is density, then the three components of momentum, and last is energy).*

## 11.8 Server-of-Server Casefile Format

ensight8 (with gold license key) has the capability of dealing with partitioned data in an efficient distributed manner by utilizing what we call a server-of-servers (SOS for short). An SOS server resides between a normal client and a number of normal servers. Thus, it appears as a normal server to the client, and as a normal client to the various normal servers.



This arrangement allows for distributed parallel processing of the various portions of a model, and has been shown to scale quite well.

Currently, EnSight SOS capability is only available for EnSight5, EnSight6, EnSight Gold, Plot3d, and any EnSight User-Defined Reader data. (It is not directly available for Fidap Neutral, Fluent Universal, N3S, Estet, MPGS4, Movie, Ansys, Abaqus, or FAST Unstructured data.)

Please recognize that your data must be partitioned in some manner (hopefully in a way that will be reasonably load balanced) in order for this approach to be useful. (The one exception to this is the use of the `auto_distribute` capability for structured data. This option can be used if the data is structured and is available to all servers defined. It will automatically distribute each structured block over the defined servers - without the user having to partition the data.)

*(Included in the EnSight distribution is an unsupported utility that will take most EnSight Gold binary unstructured datasets and partition it for you. The source for this utility (called “chopper”) can be found in the `$CEI_HOME/ensight80/unsupported/partitioner` directory.)*

Note: If you do your own partitioning of data into EnSight6 or EnSight Gold format, please be aware that each part must be in each partition - but, any given part can be “empty” in any given partition. (All that is required for an empty part is the “part” line, the part number, and the “description” line.)

You should place each partitioned portion of the model on the machine that will compute that portion. Each partitioned portion is actually a self contained set of

EnSight data files, which could typically be read by a normal client - server session of EnSight. For example, if it were EnSight gold format, there will be a casefile and associated gold geometry and variable results file(s). On the machine where the EnSight SOS will be run, you will need to place the sos casefile. This is a simple ascii file which informs the SOS about pertinent information need to run a server on each of the machines that will compute the various portions.

The format for the SOS casefile is as follows: (Note that [ ] indicates optional information, and a blank line or a line with # in the first column are comments.)

FORMAT (Required)

type: master\_server *datatype* (Required)

where: *datatype* is required and is one of the formats of EnSight's internal readers (which use the Part builder), namely:

gold      ensight6    ensight5    plot3d

or it can be the string used to name any of the user-defined readers.

Note: For user-defined readers, the string must be exactly that which is defined in the USERD\_get\_name\_of\_reader routine of the reader (which is what is presented in the Format pulldown of the Data Reader dialog).

If *datatype* is blank, it will default to EnSight6 data type.

[auto\_distribute: *on/off*] (Optional for structured, Ignored for unstructured)

For structured data only, EnSight will automatically distribute data to the servers specified below if this option is present and set to "on". This will require that each of the servers have access to the same data (or identical copies of it).

[plot3d\_iblanked: *true/false*] (Required only if *datatype* is plot3d)

[plot3d\_multi\_zone: *true/false*] (Required only if *datatype* is plot3d)

[plot3d\_dimension: *1d/2d/3d*] (Required only if *datatype* is plot3d)

[plot3d\_source: *ascii/cbin/fortranbin*] (Required only if *datatype* is plot3d)

[plot3d\_grid\_double: *true/false*] (Required only if *datatype* is plot3d)

[plot3d\_results\_double: *true/false*] (Required only if *datatype* is plot3d)

where: iblanking, multi\_zone, dimension, source type, grid file double precision, and results file double precision information should be provided. If it is not provided, it will default to the following (which is likely not to be correct):

plot3d\_iblanked: false

plot3d\_multi\_zone: false

plot3d\_dimension: 3d

plot3d\_source: cbin

plot3d\_grid\_double: false

plot3d\_results\_double: false

NETWORK\_INTERFACES (Note: this whole section is optional. It is needed only when more than one network interface to the sos host is available and it is desired to use them. Thus distributing the servers to sos communication over more than one network interface)

number of network interfaces: *num* (Required - if section used)

where: *num* is the number of network interfaces to be used for the sos host.

network interface: *sos\_network\_interface\_name1*(Required - if section is used)

network interface: *sos\_network\_interface\_name2*(Required - if section is used)

.

.

network interface: *sos\_network\_interface\_namenum*(Required - if section used)

SERVERS (Required)

number of servers: *num* (Required)

where: *num* is the number of servers that will be started and run concurrently.

#Server 1 (Comment only)

machine id: *mid* (Required)

where: *mid* is the machine id of the server.

executable: *../ensight8.server* (Required, must use full path)

[directory: *wd*] (Optional)

where: *wd* is the working directory from which ensight8.server will be run

[login id: *id*] (Optional)

where: *id* is the login id. Only needed if it is different on this machine.

[data\_path: *../dd*] (Optional)

where: *dd* is the data where the data resides. Full path must be provided if you use this line.

casefile: *yourfile.case* (Required, but depending on format, may vary as to whether it is a casefile, geometry file, neutral file, universal file, etc. Relates to the first data field of the Data Reader Dialog.)

[resfile: *yourfile.res*] (Depends on format as to whether required or not. Relates to the second data field of the Data Reader Dialog.)

[measfile: *yourfile.mea*] (Depends on format as to whether required or not. Relates to the third data field of the Data Reader Dialog.)

[bndfile: *yourfile.bnd*] (Depends on format as to whether required or not. Relates to the fourth data field of the Data Reader Dialog.)

#Server 2 (Comment only)

--- Repeat pertinent lines for as many servers as declared to be in this file ---

#### Example

This example deals with a EnSight Gold dataset that has been partitioned into 3 portions, each running on a different machine. The machines are named joe, sally, and bill. The executables for all machines are located in similar locations, but the data is not. Note that the optional data\_path line is used on two of the servers, but not the third.

```

FORMAT
type: master_server gold

SERVERS
number of servers: 3

#Server 1
machine id: joe
executable: /usr/local/bin/ensight80/bin/ensight8.server
data_path: /usr/people/john/data
casefile: portion_1.case

#Server 2
machine id: sally
executable: /usr/local/bin/ensight80/bin/ensight8.server
data_path: /scratch/sally/john/data
casefile: portion_2.case

#Server 3
machine id: bill
executable: /usr/local/bin/ensight80/bin/ensight8.server
casefile: /scratch/temp/john/portion_3.case

```

If we name this example sos casefile - “all.sos”, and we run it on yet another machine - one named george, you would want the data distributed as follows:

On george: all.sos

On joe (in /usr/people/john/data): portion\_1.case, and all files referenced by it.

On sally (in /scratch/sally/john/data): portion\_2.case, and all files referenced by it.

On bill (in /scratch/temp/john): portion\_3.case, and all file referenced by it.

By starting EnSight with the -sos command line option (which will autoconnect using ensight8.sos instead of ensight8.server), or by manually running ensight8.sos in place of ensight8.server, and providing all.sos as the casefile to read in the Data Reader dialog - EnSight will actually start three servers and compute the respective portions on them in parallel.

#### Optional NETWORK\_INTERFACES section notes

If the machine named george had more than one network interface (say it had its main one named george, but also had one named george2), we could add the section shown below to our casefile example:

```

NETWORK_INTERFACES
number of network interfaces: 2
network interface: george
network interface: george2

```

This would cause machine joe to connect back to george, machine sally to connect back to george2, and machine bill to connect back to george. This is because the sos is cycling through its available interfaces as it connects the servers. Remember that this is an optional section, and most users will probably not use it. Also, the contents of this section will be ignored if the -soshostname command line option is used.

*Additional Note: The EnSight SOS provided with release 8.0 supports most EnSight features. Some query operations with constants are not yet supported, but should be in a future release.*

## 11.9 Periodic Matchfile Format

This is an optional file which can be used in conjunction with models which have rotational or translational computational symmetry (or periodic boundary conditions). It is invoked in the GEOMETRY section of the EnSight casefile, using the “match: filename” line. (see Section , EnSight6 Case File Format).

When a model piece is created with periodic boundary conditions, there is usually a built-in correspondence between two faces of the model piece. If you transform a copy of the model piece properly, face 1 of the copy will be at the same location as face 2 of the original piece. It is desirable to know the corresponding nodes between face 1 and face 2 so border elements will not be produced at the matching faces. This correspondence of nodes can be provided in a periodic match file as indicated below. (Please note that if a periodic match file is not provided, by default EnSight will attempt to determine this correspondence using a float hashing scheme. This scheme has been shown to work quite well, but may not catch all duplicates. The user has some control over the “capture” accuracy of the hashing through the use of the command: “test: float\_hash\_digits”. If this command is issued from the command dialog, the user can change the number of digits, in a normalized scheme, to consider in the float hashing. The lower the number of digits, the larger the “capture” distance, and thus the higher the number of digits, the smaller the capture distance. The default is 4, with practical limits between 2 and 7 or 8 in most cases.)

The transformation type and delta value are contained in the file. The periodic match file is an ASCII free format file. For unstructured data, it can be thought of as a series of node pairs, where a node pair is the node number of face 1 and its corresponding node number on face 2. For structured blocks, all that is needed is an indication of whether the i, j, or k planes contain the periodic face. The min plane of this “direction” will be treated as face 1, and the max plane will be treated as face 2.

The file format is as follows:

rotate_x/y/z / translate	The first line is either rotate_x, rotate_y, rotate_z or translate
theta / dx dy dz	The second line contains rotation angle in degrees or the three translational delta values.
np n <sub>11</sub> n <sub>21</sub> n <sub>12</sub> n <sub>22</sub> . . . n <sub>1np</sub> n <sub>2np</sub>	If any unstructured pairs, the third line contains the number of these pairs (np).  And the node ids of each pair follow. (The first subscript indicates face, the second is pair.)
blocks b <sub>min</sub> b <sub>max</sub> i/j/k	Last in the file comes as many of these “blocks” lines as needed. b <sub>min</sub> and b <sub>max</sub> are a range of block numbers. For a single block, b <sub>min</sub> and b <sub>max</sub> would be the same. Only one of i, j, or k can be specified for a given block.

Simple unstructured rotational example:

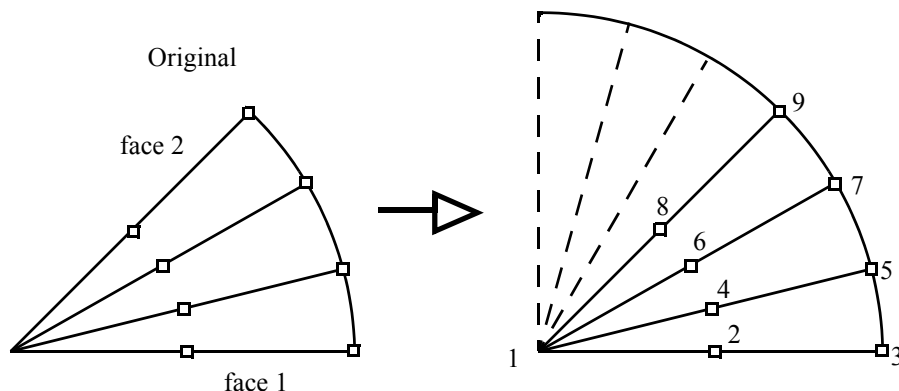


Figure 11-5  
Model Duplication by rotational symmetry

The periodic match file for a rotation of this model about point 1 would be:

```
rotate_z
45.0
3
1 1
2 8
3 9
```

Thus, face 1 of this model is made up of nodes 1, 2, and 3 and face 2 of this model is made up of nodes 1, 8, and 9. So there are 3 node pairs to define, with node 1 corresponding to node 1 after a copy is rotated, node 2 corresponding to node 8, and node 3 corresponding to node 9.

Simple structured translational model:

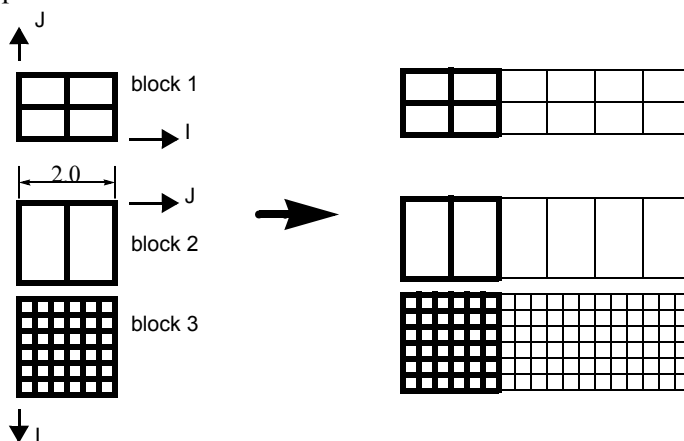


Figure 11-6  
Model Duplication by translational symmetry  
of structured blocks (3 instances)

```
translate
2.0 0.0 0.0
blocks 1 1 i
blocks 2 3 j
```

Since block 1 is oriented differently than blocks 2 and 3 in terms of ijk space, two “blocks” lines were needed in the match file.

**Special Notes / Limitations:**

1. This match file format requires that the unstructured node ids of the model be unique. This is only an issue with EnSight Gold unstructured format, since it is possible with that format to have non-unique node ids.
2. The model instance (which will be duplicated periodically) must have more than one element of thickness in the direction of the duplication. If it has only one element of thickness, intermediate instances will have all faces removed. If you have this unusual circumstance, you will need to turn off the shared border removal process, as explained in note 3.
3. The shared border removal process can be turned off, thereby saving some memory and processing time, by issuing the “test: rem\_shared\_bord\_off” command in the command dialog. The effect of turning it off will be that border elements will be left between each periodic instance on future periodic updates.
4. The matching and hashing processes actually occur together. Thus, matching information does not have to be specified for all portions of a model. If no matching information is specified for a given node, the hashing process is used. By the same token, if matching information is provided, it is used explicitly as specified - even if it has been specified incorrectly.



## 11.10XY Plot Data Format

This file is saved using the Save section of the Query Entity dialog. The file can contain one or more curves. The following is an example XY Data file:

Line	Contents of Line
1	2
2	Distance vs. Temperature for Line Tool
3	Distance
4	Temperature
5	1
6	5
7	0.0 4.4
8	1.0 5.8
9	2.0 3.6
10	3.0 4.6
11	4.0 4.8
12	Distance vs. Pressure for Line Tool
13	Distance
14	Pressure
15	2
16	4
17	0.00 1.2
18	0.02 1.1
19	0.04 1.15
20	0.06 1.22
21	3
22	1.10 1.30
23	1.12 1.28
24	1.14 1.25

Line 1 contains the (integer) number of curves in the file.

Line 2 contains the name of the curve.

Line 3 contains the name of the X-Axis.

Line 4 contains the name of the Y-Axis.

Line 5 contains the number of curve segments in this curve.

Line 6 contains the number of points in the curve segment.

Lines 7-11 contain the X-Y information.

Line 12 contains the name of the second curve.

Line 13 contains the name of the X-Axis

Line 14 contains the name of the Y-Axis

Line 15 contains the number of curve segments in this curve. (For the second curve, the first segment contains 4 points, the second 3 points.)

## 11.11 EnSight Boundary File Format

This file format can be used to represent boundary surfaces of structured data as unstructured parts. The boundaries defined in this file can come from sections of several different structured blocks. Thus, inherent in the file format is a grouping and naming of boundaries across multiple structured blocks.

Additionally, a delta can be applied to any boundary section to achieve the creation of repeating surfaces (such as blade rows in a jet engine).

Note: There is no requirement that the boundaries actually be on the surface of blocks, but they must define either 2D surfaces or 1D lines. You may not use this file to define 3D portions of the block.

The boundary file is read if referenced in the casefile of EnSight data models, or in the boundary field of the Data Reader Dialog for other data formats. Any boundaries successfully read will be listed in the Unstructured Data Part List of the Data Part Loader Dialog.

The format of the EnSight Boundary File is as follows:

Line (1): Required header keyword and version number. ENSBND is required exactly, but the version number could change in the future.

**ENSBND 1.00**

Line (2) through Line (NumBoundaries+1): The names of the boundaries to be defined. (Each name must be no greater than 79 characters.)

For example:

```
inflow
wall
Chimera Boundary
outflow
```

Line (NumBoundaries+2): Required keyword indicating the end of the boundary names and the beginning of the boundary section definitions.

**BOUNDARIES**

Line (NumBoundaries+3) through the end of the file: The boundary section definitions. Each line will have the following information:

```
bnd_num blk_num imin imax jmin jmax kmin kmax [di dj dk n_ins]
```

where:

**bnd\_num** is the number of the boundary in the list of names above.  
(For example: inflow is 1, wall is 2, Chimera Boundary is 3, etc.)

**blk\_num** is the parent structured block for this boundary section.

**imin,imax, jmin,jmax, kmin,kmax** are the ijk ranges for the boundary section. At least one of the min,max pairs must refer to the same plane. A wildcard (“\$”) can be used to indicate the maximum i, j, or k value of that block (the far plane). Additionally, negative numbers can be used to indicate plane values from the far side toward the near side. (-1 = far plane, -2 = one less than the far plane, etc.)

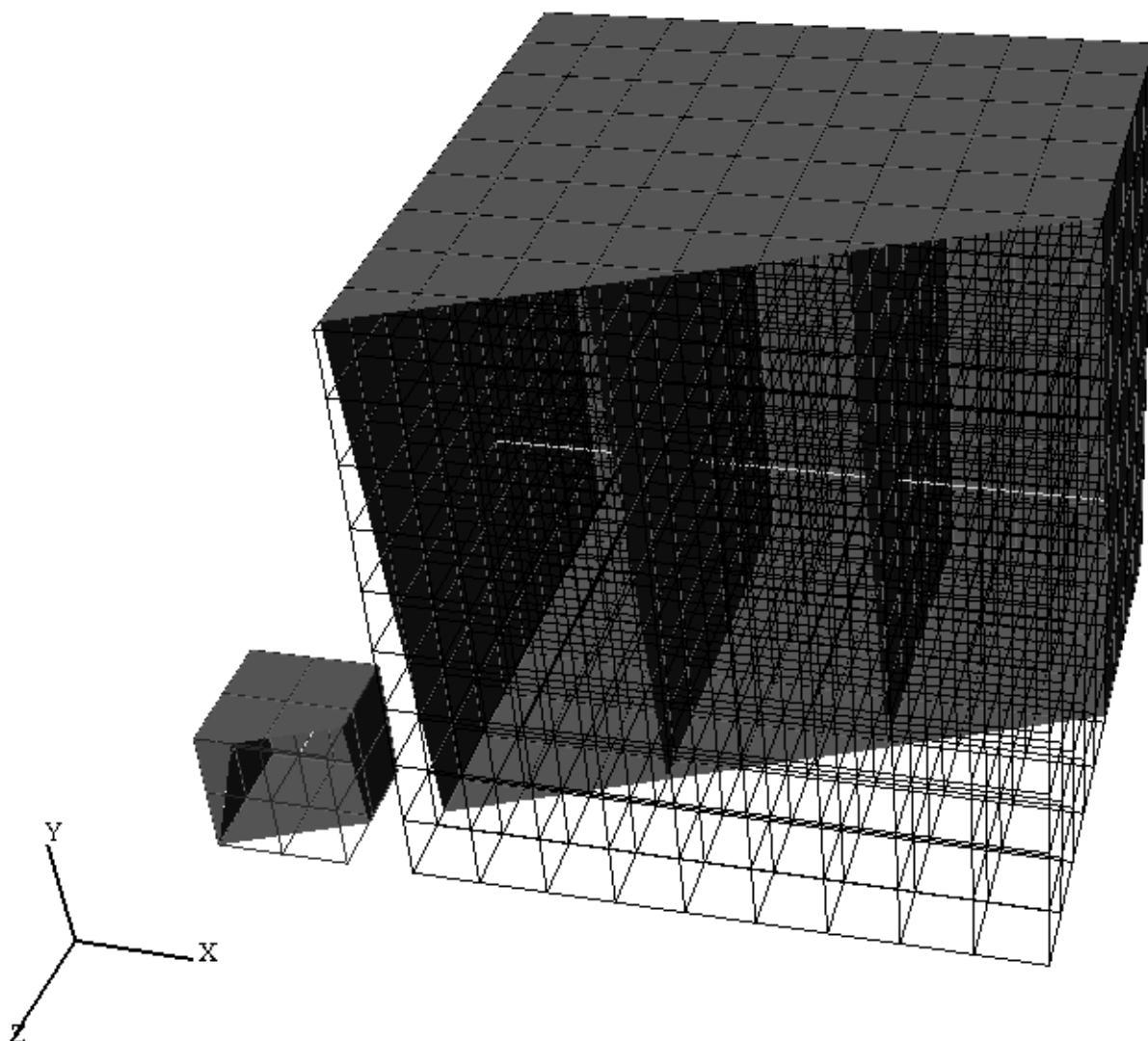
**[di,dj,dk and n\_ins]** are optional delta information which can be used to extract repeating planes. The appropriate di, dj, or dk delta value should be set to the

repeating plane offset value, and the other two delta values must be zero. The non-zero delta must correspond to a min,max pair that are equal. The  $n\_ins$  value is used to indicate the number of repeating instances desired. A (“\$”) wildcard can be used here to indicate that the maximum number of instances that fit in the block should be extracted.

All numbers on the line must be separated by spaces.

Finally, comment lines are allowed in the file by placing a “#” in the first column.

Below is a simple example of a boundary file for two structured blocks, the first of which is a  $3 \times 3 \times 3$  block, and the second is a  $10 \times 10 \times 10$  block. We will define a boundary which is the front and back planes of each block(K planes) and one that is the top and bottom planes of each block(J planes). We will also define some repeating x planes, namely, planes at  $i=1$  and 3 for block one and at  $i=1, 4, 7,$  and 10 for block two. The image below shows the blocks in wire frame and the boundaries shaded and slightly cut away, so you can see the interior x-planes.



The file to accomplish this looks like:

```

ENSBND 1.00
front_back
top_bottom
x-planes
middle lines
BOUNDARIES
#bnd blk imin imax jmin jmax kmin kmax di dj dk n_ins
#--- --- ---- -
1 1 1 3 1 3 $ $
1 2 1 $ 1 $ 10 10
2 1 1 $ $ $ 1 $
2 2 1 10 10 10 1 10
1 1 1 3 1 3 1 1
1 2 1 $ 1 $ 1 1
2 1 1 3 1 1 1 $
2 2 1 $ 1 1 1 $
3 1 1 1 1 $ 1 $ 2 0 0 2
3 2 1 1 1 $ 1 $ 3 0 0 $
4 1 2 2 2 2 1 3
4 2 1 10 5 5 5 5

```

Interpreting the 12 boundary definition lines:

1 1 1 3 1 3 \$ \$  
defines a part of the boundary called front\_back, on block 1, where I=1 to 3, J=1 to 3, and K=3. Thus, the far K plane of block 1.

1 2 1 \$ 1 \$ 10 10  
defines another part of the front\_back boundary, on block 2, where I=1 to 10, J=1 to 10, and K = 10. Thus, the far K plane of block 2.

2 1 1 \$ \$ \$ 1 \$  
defines a part of the boundary called top\_bottom, on block 1, where I=1 to 3, J=3, and K=1 to 3. Thus, the far J plane of block 1.

2 2 1 10 10 10 1 10  
defines another part of the top\_bottom boundary, on block 2, where I=1 to 10, J=10, and K=1 to 10. Thus, the far J plane of block 2.

1 1 1 3 1 3 1 1  
defines another part of the front\_back boundary, on block 1, where I=1 to 3, J=1 to 3, and K=1. Thus, the near K plane of block 1.

1 2 1 \$ 1 \$ 1 1  
defines another part of the front\_back boundary, on block 2, where I=1 to 10, J=1 to 10, and K=1. Thus the near K plane of block 2.

2 1 1 3 1 1 1 \$  
defines another part of the top\_bottom boundary, on block 1, where I=1 to 3, J=1, and K=1 to 3. Thus, the near J plane of block 1.

2 2 1 \$ 1 1 1 \$  
defines another part of the top\_bottom boundary, on block 2, where I=1 to 10, J=1, and K=1 to 10. Thus, the near J plane of block 2.

3 1 1 1 1 \$ 1 \$ 2 0 0 2  
 defines a part of the boundary called x-planes, on block 1, where I=1, J=1 to 3, K=1 to 3, then again where I=3, J=1 to 3, and K=1 to 3. Thus, both the near and far I planes of block 1.

3 2 1 1 1 \$ 1 \$ 3 0 0 \$  
 defines another part of the x-planes boundary, on block 2, where I=1, J=1 to 10, and K=1 to 10, then again where I=4, J=1 to 10, and K=1 to 10, then again where I=7, J=1 to 10, and K=1 to 10, then again where I=10, J=1 to 10, and K=1 to 10. Thus, the I = 1, 4, 7, and 10 planes of block 2.

4 1 2 2 2 2 1 3  
 defines a part of the boundary called middle lines, on block 1, where I=2, J=2, and K=1 to 3. Thus, line through the middle of block 1 in the K direction.

4 2 1 10 5 5 5 5  
 defines another part of the middle lines boundary, on block 2, where I=1 to 10, J=5, and K=5. Thus a line through the middle of the block2 in the I direction.

Please note that the “\$” wildcard was used rather randomly in the example, simply to illustrate how and where it can be used.

The use of negative numbers for ijk planes is indicated below - again rather randomly for demonstration purposes. This file will actually produce the same result as the file above.

```

ENSBND 1.00
front_back
top_bottom
x-planes
middle lines
BOUNDARIES
#bnd blk imin imax jmin jmax kmin kmax di dj dk n_ins
#---
1 1 1 3 1 3 -1 -1
1 2 -3 -1 1 -1 10 10
2 1 1 -1 -1 -1 1 -1
2 2 1 10 10 10 1 10
1 1 1 3 1 3 1 1
1 2 1 -1 1 -1 1 1
2 1 1 3 1 1 1 -1
2 2 1 -1 1 1 1 -1
3 1 1 1 1 -1 1 -1 2 0 0 2
3 2 1 1 1 -1 1 -1 3 0 0 $
4 1 2 -2 -2 2 1 3
4 2 1 10 5 5 -6 -6

```

## 11.12 EnSight Particle Emitter File Format

This file can be used to specify the location of emitter points. It is most useful when the user has specific (and many) emit points to use for particle traces. Rather than type them all in one at a time as a cursor emitter, this file can be used.

Note the following:

1. The first two lines need to be exactly as shown.

```
Version 1.0
EnSight Particle Emitter File
```

2. #’s as the first character on a line, are comment lines. Comment lines can be used anywhere in the file after the first two mandatory lines.
3. Time lines contain the emitter release time (simulation time, NOT time step).
4. Emit lines contain the coordinates of an emitter. The emitter will be associated with the previously specified Time.

Sample File:

```
Version 1.0
EnSight Particle Emitter File
#
Time 0.249063
Emit 0.0669737 0.0134195 -0.013926
Emit 0.0669737 0.0131277 -0.0141079
Emit 0.0669737 0.0128642 -0.0143178
Emit 0.0669737 0.0155969 -0.0113572
Emit 0.0669737 0.0150344 -0.0122012
Emit 0.0669737 0.0146967 -0.0123587
#
Time 0.249113
Emit 0.0669737 0.0137097 -0.0136404
Emit 0.0669737 0.0134218 -0.0138284
Emit 0.0669737 0.0131628 -0.0140438
Emit 0.0669737 0.0158325 -0.0110264
Emit 0.0669737 0.0152879 -0.011882
Emit 0.0669737 0.0149536 -0.0120466
#
Time 0.249163
Emit 0.0669737 0.0139938 -0.0133488
Emit 0.0669737 0.01371 -0.0135428
Emit 0.0669737 0.0134555 -0.0137636
Emit 0.0669737 0.0160612 -0.0106906
Emit 0.0669737 0.0155347 -0.0115575
Emit 0.0669737 0.0152039 -0.0117291
#
Time 0.249213
Emit 0.0669737 0.0142718 -0.0130512
Emit 0.0669737 0.013992 -0.0132511
Emit 0.0669737 0.0137423 -0.0134773
Emit 0.0669737 0.0162827 -0.0103501
Emit 0.0669737 0.0157745 -0.0112279
Emit 0.0669737 0.0154475 -0.0114064
#
```

## 11.12 EnSight Particle Emitter File Format

```
Time 0.249263
Emit 0.0669737 0.0145433 -0.0127479
Emit 0.0669737 0.0142679 -0.0129536
Emit 0.0669737 0.014023 -0.013185
Emit 0.0669737 0.0164969 -0.0100051
Emit 0.0669737 0.0160074 -0.0108933
Emit 0.0669737 0.0156842 -0.0110787
```