9.1 MSH ASCII file format

The MSH ASCII file format contains one mandatory section giving information about the file (`$MeshFormat`), followed by several optional sections defining the nodes (`$Nodes`), elements (`$Elements`), region names (`$PhysicalName`) and post-processing datasets (`$NodeData`, `$ElementData`, `$ElementNodeData`). Sections can be repeated in the same file, and post-processing sections can be put into separate files (e.g. one file per time step).

The format is defined as follows:

```
$MeshFormat
version-number file-type data-size
$EndMeshFormat
$Nodes
number-of-nodes
node-number x-coord y-coord z-coord
…
$EndNodes
$Elements
number-of-elements
elm-number elm-type number-of-tags < tag > … node-number-list
…
$EndElements
$PhysicalNames
number-of-names
phyical-number "physical-name"
…
$EndPhysicalNames
$NodeData
number-of-string-tags
< "string-tag" >
…
number-of-real-tags
< real-tag >
…
number-of-integer-tags
< integer-tag >
…
node-number value …
…
$EndNodeData
$ElementData
number-of-string-tags
< "string-tag" >
…
number-of-real-tags
< real-tag >
…
number-of-integer-tags
< integer-tag >
…
elm-number value …
…
$EndElementData
$ElementNodeData
number-of-string-tags
< "string-tag" >
…
number-of-real-tags
< real-tag >
…
number-of-integer-tags
```

```
< integer-tag >
…
elm-number number-of-nodes-per-element value …
…
$ElementEndNodeData
```

where

`version-number`

is a real number equal to 2.0

`file-type`

is an integer equal to 0 in the ASCII file format.

`data-size`

is an integer equal to the size of the floating point numbers used in the file (currently only *data-size* = sizeof(double) is supported).

`number-of-nodes`

is the number of nodes in the mesh.

`node-number`

is the number (index) of the *n*-th node in the mesh; *node-number* must be a postive (non-zero) integer. Note that the *node-number*s do not necessarily have to form a dense nor an ordered sequence.

`x-coord y-coord z-coord`

are the floating point values giving the X, Y and Z coordinates of the *n*-th node.

`number-of-elements`

is the number of elements in the mesh.

`elm-number`

is the number (index) of the *n*-th element in the mesh; *elm-number* must be a postive (non-zero) integer. Note that the *elm-number*s do not necessarily have to form a dense nor an ordered sequence.

`elm-type`

defines the geometrical type of the *n*-th element:

`1`

2-node line.

`2`

3-node triangle.

`3`

4-node quadrangle.

`4`

4-node tetrahedron.

`5`

8-node hexahedron.

6

6-node prism.

7

5-node pyramid.

8

3-node second order line (2 nodes associated with the vertices and 1 with the edge).

9

6-node second order triangle (3 nodes associated with the vertices and 3 with the edges).

10

9-node second order quadrangle (4 nodes associated with the vertices, 4 with the edges and 1 with the face).

11

10-node second order tetrahedron (4 nodes associated with the vertices and 6 with the edges).

12

27-node second order hexahedron (8 nodes associated with the vertices, 12 with the edges, 6 with the faces and 1 with the volume).

13

18-node second order prism (6 nodes associated with the vertices, 9 with the edges and 3 with the quadrangular faces).

14

14-node second order pyramid (5 nodes associated with the vertices, 8 with the edges and 1 with the quadrangular face).

15

1-node point.

16

8-node second order quadrangle (4 nodes associated with the vertices and 4 with the edges).

17

20-node second order hexahedron (8 nodes associated with the vertices and 12 with the edges).

18

15-node second order prism (6 nodes associated with the vertices and 9 with the edges).

19

13-node second order pyramid (5 nodes associated with the vertices and 8 with the edges).

`20`

9-node third order incomplete triangle (3 nodes associated with the vertices, 6 with the edges)

`21`

10-node third order triangle (3 nodes associated with the vertices, 6 with the edges, 1 with the face)

`22`

12-node fourth order incomplete triangle (3 nodes associated with the vertices, 9 with the edges)

`23`

15-node fourth order triangle (3 nodes associated with the vertices, 9 with the edges, 3 with the face)

`24`

15-node fifth order incomplete triangle (3 nodes associated with the vertices, 12 with the edges)

`25`

21-node fifth order complete triangle (3 nodes associated with the vertices, 12 with the edges, 6 with the face)

`26`

4-node third order edge (2 nodes associated with the vertices, 2 internal to the edge)

`27`

5-node fourth order edge (2 nodes associated with the vertices, 3 internal to the edge)

`28`

6-node fifth order edge (2 nodes associated with the vertices, 4 internal to the edge)

`29`

20-node third order tetrahedron (4 nodes associated with the vertices, 12 with the edges, 4 with the faces)

`30`

35-node fourth order tetrahedron (4 nodes associated with the vertices, 18 with the edges, 12 with the faces, 1 in the volume)

`31`

56-node fifth order tetrahedron (4 nodes associated with the vertices, 24 with the edges, 24 with the faces, 4 in the volume)

See below for the ordering of the nodes.

`number-of-tags`

gives the number of integer tags that follow for the *n*-th element. By default, the first *tag* is the number of the physical entity to which the element belongs; the second is the number of the elementary geometrical entity to which the element belongs; the third is the number of a mesh partition to which the element belongs. All tags must be postive integers, or zero. A zero tag is equivalent to no tag.

`node-number-list`

is the list of the node numbers of the *n*-th element. The ordering of the nodes is given in Node ordering.

`number-of-string-tags`

gives the number of string tags that follow. By default the first *string-tag* is interpreted as the name of the post-processing view.

`number-of-real-tags`

gives the number of real number tags that follow. By default the first *real-tag* is interpreted as a time value associated with the dataset.

`number-of-integer-tags`

gives the number of integer tags that follow. By default the first *integer-tag* is interpreted as a time step index (starting at 0), the second as the number of field components of the data in the view (1, 3 or 9), the third as the number of entities (nodes or elements) in the view, and the fourth as the partition index for the view data (0 for no partition).

`number-of-nodes-per-elements`

gives the number of node values for an element in an element-based view.

`value`

is a real number giving the value associated with a node or an element. For `NodeData` (respectively `ElementData`) views, there are *ncomp* values per node (resp. per element), where *ncomp* is the number of field components. For `ElementNodeData` views, there are *ncomp* times *number-of-nodes-per-elements* values per element.

Below is a small example (a mesh consisting of two quadrangles with an associated nodal scalar dataset; the comments are not part of the actual file!):

```
$MeshFormat
2.0 0 8
$EndMeshFormat
$Nodes
6                      six mesh nodes:
1 0.0 0.0 0.0            node #1: coordinates (0.0, 0.0, 0.0)
2 1.0 0.0 0.0            node #2: coordinates (1.0, 0.0, 0.0)
3 1.0 1.0 0.0            etc.
4 0.0 1.0 0.0
5 2.0 0.0 0.0
6 2.0 1.0 0.0
$EndNodes
$Elements
2                      two elements:
1 3 2 99 2 1 2 3 4       quad #1: type 3, physical 99, elementary 2, nodes 1 2 3 4
2 3 2 99 2 2 5 6 3       quad #2: type 3, physical 99, elementary 2, nodes 2 5 6 3
$EndElements
$NodeData
1                      one string tag:
"A scalar view"          the name of the view ("A scalar view")
1                      one real tag:
0.0                      the time value (0.0)
3                      three integer tags:
0                        the time step (0; time steps always start at 0)
1                        1-component (scalar) field
6                        six associated nodal values
1 0.0                  value associated with node #1 (0.0)
```

```
2 0.1                value associated with node #2 (0.1)
3 0.2                etc.
4 0.0
5 0.2
6 0.4
$EndNodeData
```