Appendix B. Mesh File Format

This document describes the content and format of TGrid and FLUENT mesh files. These files are broken into several sections depending on the following guidelines:

- Each section is enclosed in parentheses and begins with an integer (in decimal) indicating its type.
- All groups of items are enclosed in parentheses. This makes skipping to ends of (sub)sections and parsing them very easy. It also allows for easy and compatible addition of new items in future eleases.
- Header information for lists of items is enclosed in separate sets of parentheses, preceding the items, which are in their own parentheses.

Grid sections are stored in the case file. A *grid* file is a subset of a case file, containing only those sections pertaining to the grid. Currently defined grid sections are explained in the following sections.

The section ID numbers are indicated in both symbolic and numeric forms. The symbolic representations are available as symbols in a Scheme source file (`xfile.scm`), or as macros in a C header file (`xfile.h`). Both these files are available from Fluent Inc.

B.1 Comment

Index:	0
Scheme symbol:	`xf-comment`
C macro:	`XF_COMMENT`
Codes:	FLUENT, TGrid
Status:	optional

Comment sections can appear anywhere in the file (except within other sections). It appear as:

```
(0 "comment text")
```

It is recommended that each long section, or group of related sections, be preceded by a comment section explaining what is to follow.

For example,

```
(0 "Variables:")
(37 (
(relax-mass-flow 1)
(default-coefficient ())
(default-method 0)
))
```

B.2 Header

Index: 1
Scheme symbol: `xf-header`
C macro: `XF_HEADER`
Codes: FLUENT, TGrid
Status: optional

Header sections appear anywhere in the file (except within other sections). Although it can appear anywhere, it is typically one of the first sections in the file. It appear as:

```
(1 "TGrid ND 3.5.1")
```

where `N` is 2 or 3. The purpose of this section is to identify the program that wrote the file. Additional header sections indicate other programs that may have been used in generating the file and thus provide a history mechanism showing where the file came from and how it was processed.

B.3 Dimensions

Index: 2
Scheme symbol: `xf-dimension`
C macro: `XF_DIMENSION`
Codes: FLUENT, TGrid
Status: optional

The dimensions of the grid is expressed as:

```
(2 ND)
```

where `ND` is 2 or 3. This section is currently supported as a check that the grid has the appropriate dimensions.

B.4 Nodes

Index: 10
Scheme symbol: `xf-node`
C macro: `XF_NODE`
Codes: FLUENT, TGrid
Status: required

```
(10 (zone-id first-index last-index type ND)(
x1 y1 z1
x2 y2 z2
.
.
.
))
```

If `zone-id` is zero, it provides the total number of nodes in the grid. In such a case:

- `first-index` will be one.
- `last-index` will be the.
- Total number of nodes *in hexadecimal*, `type` is meaningless.
- `ND` is the dimensionality of the grid.
- There are no coordinates following.

The parentheses for the coordinates are not there either. For example,

```
(10 (0 1 2d5 0 2))
```

If `zone-id` is greater than zero, it indicates the zone to which the nodes belong. In such a case:

- `first-index` and `last-index` are the indices of the nodes in the zone *in hexadecimal*.
- `last-index` in each zone must be less than or equal to the value in the declaration section.
- `type` indicates the type of nodes in the zone. TGrid uses the value of types to indicate the following types:
  - Zero for "virtual" nodes.

  - One for no (any) type.
  - Two for boundary nodes.

    TGrid ignores nodes of type zero and read all others, but these codes write only type one.

- `ND` is an optional argument that indicates the dimensionality of the node data, where `ND` is 2 or 3.

  If the number of dimensions in the grid is two, as specified in the Section B.3: Dimensions or in the node header, then only x and y coordinates are present on each line.

Example of a two-dimensional grid:

```
(10 (1 1 2d5 1 2)(
1.500000e-01 2.500000e-02
1.625000e-01 1.250000e-02
          .
          .
          .
1.750000e-01 0.000000e+00
2.000000e-01 2.500000e-02
1.875000e-01 1.250000e-02
))
```

Grid connectivity is composed of integers representing pointers (see Sections B.7 and B.6), hence using hexadecimal conserves space in the file and provides faster file input and output. The header indices are also in hexadecimal so that they match the indices in the bodies of the grid connectivity sections. The `zone-id` and `type` are also in hexadecimal for consistency.

B.5 Periodic Shadow Faces

Index:	18
Scheme symbol:	`xf-periodic-face`
C macro:	`XF_PERIODIC_FACE`
Codes:	FLUENT, TGrid
Status:	required only for grids with periodic boundaries

This section indicates the pairings of periodic faces on periodic boundaries. Grids without periodic boundaries do not have sections of this type.

The format of the section is as follows:

```
(18 (first-index last-index periodic-zone shadow-zone)(
f00 f01
f10 f21
f20 f21
.
.
.
))
```

where,

`first-index` is the index of the first periodic face pair in the list.

`last-index` is the last periodic face pair in the list.

`periodic-zone` is the zone ID of the periodic face zone.

`shadow-zone` is the zone ID of the corresponding shadow face zone.

These are in hexadecimal format.

The indices in the section body (`f*`) refer to the faces on each of the periodic boundaries (in hexadecimal), the indices being offsets into the list of faces for the grid. The `first-index` and `last-index` do *not* refer to face indices. They refer to indices in the list of periodic pairs.

A partial example of such a section is as follows:

```
(18 (1 2b a c) (
12 1f
13 21
ad 1c2
.
.
.
))
```

B.6 Cells

Index: 12
Scheme symbol: `xf-cell`
C macro: `XF_CELL`
Codes: FLUENT, TGrid
Status: required

The declaration section for cells is similar to that for nodes.

```
(12 (zone-id first-index last-index type element-type))
```

When `zone-id` is zero it indicates that it is a declaration of the total number of cells. If `last-index` is zero, then there are no cells in the grid. This is useful when the file contains only a surface mesh to alert the solvers that it cannot be used.

The `type` is ignored in a declaration section and is usually given as zero, while the `element-type` is ignored completely.

For example,

```
(12 (0 1 3e3 0))
```

states that there are 3e3 (hexadecimal) = 995 cells in the grid. This declaration section is required and must precede the regular cell sections.

The `element-type` in a regular cell section header indicates the type of cells in the section, as follows:

`element-type`	*description*	*nodes/cell*	*faces/cell*
0	mixed		
1	triangular	3	3
2	tetrahedral	4	4
3	quadrilateral	4	4
4	hexahedral	8	6
5	pyramid	5	5
6	wedge	6	5

Regular cell sections have no body, but they have a header of the same format where `first-index` and `last-index` indicate the range for the particular zone, `type` indicates whether the cell zone is an active zone (solid or fluid), or inactive zone (currently only parent cells resulting from hanging node adaption). Active zones are represented with `type`=1, while inactive zones are represented with `type`=32.

A `type` of zero indicates a dead zone and will be skipped by TGrid. If a zone is of mixed type (`element-type`=0), it will have a body that lists the `element type` of each cell.

In the following example, there are 3d (hexadecimal) = 61 cells in cell zone 9, of which the first 3 are triangles, the next 2 are quadrilaterals, and so on.

```
(12 (9 1 3d 0 0)(
1 1 1 3 3 1 1 3 1
.
.
.
))
```

The cell section is not required for TGrid when the file contains only a surface mesh.

B.7 Faces

Index:	13
Scheme symbol:	`xf-face`
C macro:	`XF_FACE`
Codes:	FLUENT, TGrid
Status:	required

The face section has a header with the same format as that for cells (but with a section index of 13).

```
(13 (zone-id first-index last-index type element-type))
```

A `zone-id` of zero indicates a declaration section with no body, and `element-type` indicates the type of faces in that zone.

The body of a regular face section contains the grid connectivity, and each line appears as follows:

```
n0 n1 n2 cr cl
```

where `n*` are the defining nodes (vertices) of the face, and `c*` are the adjacent cells.

This is an example of the triangular face format; the actual number of nodes depends on the `element type`. The ordering of the cell indices is important. The first cell, `cr`, is the cell on the right side of the face and `cl` is the cell on the left side.

Direction is determined by the right-hand rule. It states that, if you curl the fingers of your right hand in the order of the nodes, your thumb will point to the right side of the face. In 2D grids, the $\hat{k}$ vector pointing outside the grid plane is used to determine the right-hand-side cell (`cr`) from $\hat{k} \times \hat{r}$.

If there is no adjacent cell, then either `cr` or `cl` is zero. All cells, faces, and nodes have positive indices. For files containing only a boundary mesh, both these values are zero. If it is a two-dimensional grid, `n2` is omitted.

If the face zone is of mixed type (`element-type` = 0), the body of the section will include the face type and will appear as follows:

```
type v0 v1 v2 c0 c1
```

where `type` is the type of face, as defined in the following table:

`element-type`	*face type*	*nodes/face*
0	mixed	
2	linear	2
3	triangular	3
4	quadrilateral	4

The current valid boundary conditions types are defined in the following table:

`bc name`	*bc id*
interior	2
wall	3
pressure-inlet, inlet-vent, intake-fan	4
pressure-outlet, exhaust-fan, outlet-vent	5
symmetry	7
periodic-shadow	8
pressure-far-field	9
velocity-inlet	10
periodic	12
fan, porous-jump, radiator	14
mass-flow-inlet	20
interface	24
parent (hanging node)	31
outflow	36
axis	37

For non-conformal grid interfaces, the faces resulting from the intersection of the non-conformal grids are placed in a separate face zone. A factor of 1000 is added to the `type` of these sections, e.g., 1003 is a wall zone.

B.8 Face Tree

Index:	59
Scheme symbol:	`xf-face-tree`
C macro:	`XF_FACE_TREE`
Codes:	FLUENT
Status:	only for grids with hanging-node adaption

This section indicates the face hierarchy of the grid containing hanging nodes. The format of the section is as follows:

```
(59 (face-id0 face-id1 parent-zone-id child-zone-id)
(
 number-of-kids kid-id-0 kid-id-1 ... kid-id-n
 .
 .
 .
))
```

where,

`face-id0` is the index of the first parent face in section.

`face-id1` is the index of the last parent face in section.

`parent-zone-id` is the ID of the zone containing the parent faces

`child-zone-id` is the ID of the zone containing the children faces.

`number-of-kids` is the number of children of the parent face.

`kid-id-n` are the face IDs of the children.

These are in hexadecimal format. TGrid can read files that contain this section.

B.9 Cell Tree

Index:	58
Scheme symbol:	`xf-cell-tree`
C macro:	`XF_CELL_TREE`
Codes:	FLUENT
Status:	only for grids with hanging-node adaption

This section indicates the cell hierarchy of the grid containing hanging nodes.

The format of the section is as follows:

```
(58 (cell-id0 cell-id1 parent-zone-id child-zone-id)
(
 number-of-kids kid-id-0 kid-id-1 ... kid-id-n
 .
 .
 .
))
```

where,

`cell-id0` is the index of the first parent cell in section.

`cell-id1` is the index of the last parent cell in section.

`parent-zone-id` is the ID of the zone containing the parent cells.

`child-zone-id` is the ID of the zone containing the children cells.

`number-of-kids` is the number of children of the parent cell.

`kid-id-n` are the cell IDs of the children.

These are in hexadecimal format. TGrid cannot read files that contain this section.

B.10 Interface Face Parents

Index:	61
Scheme symbol:	`xf-face-parents`
C macro:	`XF_FACE_PARENTS`
Codes:	FLUENT
Status:	only for grids with non-conformal interfaces

This section indicates the relationship between the intersection faces and original faces. The intersection faces (children) are produced from intersecting two non-conformal surfaces (parents) and are some fraction of the original face. Each child will refer to at least one parent.

The format of the section is as follows:

```
(61 (face-id0 face-id1)
(
 parent-id-0 parent-id-1
 .
 .
 .
))
```

where,

`face-id0` is the index of the first child face in the section.

`face-id1` is the index of the last child face in the section.

`parent-id-0` is the index of the right-side parent face.

`parent-id-1` is the index of the left-side parent face.

These are in hexadecimal format.

TGrid will skip this section, if you read a non-conformal grid from the solver into TGrid. So it will not maintain all the information necessary to preserve the non-conformal interface. When you read the grid back into the solver, you will need to recreate the interface.

B.11 Example Files

Example 1

Figure B.11.1 illustrates a simple quadrilateral mesh with no periodic boundaries or hanging nodes.

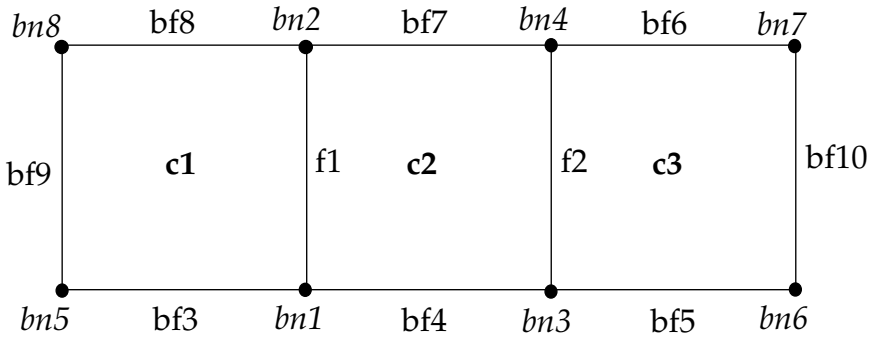

Figure B.11.1: Quadrilateral Mesh

The following describes this mesh:

```
(0 "Grid:")

(0 "Dimensions:")
(2 2)

(12 (0 1 3 0))
(13 (0 1 a 0))
(10 (0 1 8 0 2))

(12 (7 1 3 1 3))

(13 (2 1 2 2 2)(
1 2 1 2
3 4 2 3))

(13 (3 3 5 3 2)(
5 1 1 0
1 3 2 0
3 6 3 0))

(13 (4 6 8 3 2)(
7 4 3 0
4 2 2 0
2 8 1 0))

(13 (5 9 9 a 2)(
8 5 1 0))

(13 (6 a a 24 2)(
6 7 3 0))

(10 (1 1 8 1 2)
(
 1.00000000e+00  0.00000000e+00
 1.00000000e+00  1.00000000e+00
 2.00000000e+00  0.00000000e+00
 2.00000000e+00  1.00000000e+00
 0.00000000e+00  0.00000000e+00
 3.00000000e+00  0.00000000e+00
 3.00000000e+00  1.00000000e+00
 0.00000000e+00  1.00000000e+00))
```

Example 2

Figure B.11.2 illustrates a simple quadrilateral mesh with periodic boundaries but no hanging nodes. In this example, `bf9` and `bf10` are faces on the periodic zones.

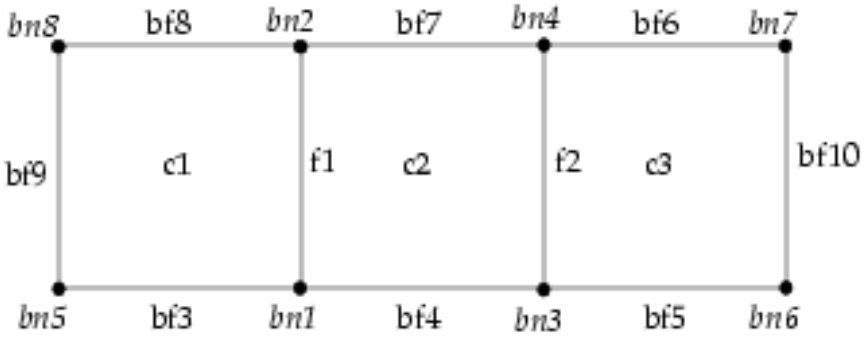

Figure B.11.2: Quadrilateral Mesh with Periodic Boundaries

The following describes this mesh:

```
(0 "Dimensions:")
(2 2)

(0 "Grid:")

(12 (0 1 3 0))
(13 (0 1 a 0))
(10 (0 1 8 0 2))

(12 (7 1 3 1 3))

(13 (2 1 2 2 2)(
1 2 1 2
3 4 2 3))

(13 (3 3 5 3 2)(
5 1 1 0
1 3 2 0
3 6 3 0))

(13 (4 6 8 3 2)(
7 4 3 0
4 2 2 0
2 8 1 0))
```

```
(13 (5 9 9 c 2)(
8 5 1 0))

(13 (1 a a 8 2)(
6 7 3 0))

(18 (1 1 5 1)(
 9 a))

(10 (1 1 8 1 2)(
 1.00000000e+00  0.00000000e+00
 1.00000000e+00  1.00000000e+00
 2.00000000e+00  0.00000000e+00
 2.00000000e+00  1.00000000e+00
 0.00000000e+00  0.00000000e+00
 3.00000000e+00  0.00000000e+00
 3.00000000e+00  1.00000000e+00
 0.00000000e+00  1.00000000e+00))
```

Example 3

Figure B.11.3 illustrates a simple quadrilateral mesh with hanging nodes.

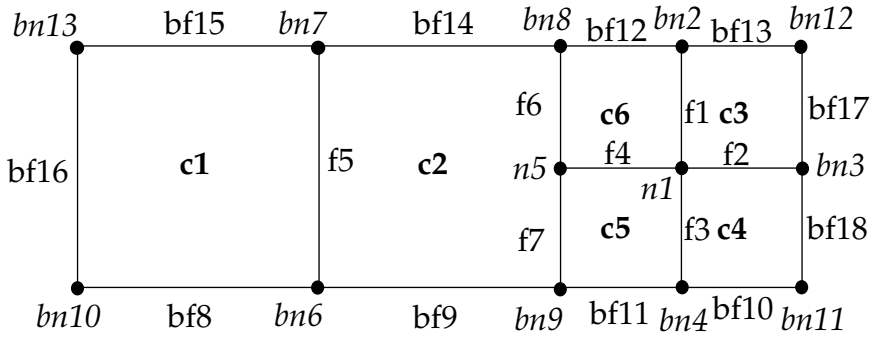

Figure B.11.3: Quadrilateral Mesh with Hanging Nodes

The following describes this mesh:

```
(0 "Grid:")

(0 "Dimensions:")
(2 2)
```

```
(12 (0 1 7 0))
(13 (0 1 16 0))
(10 (0 1 d 0 2))

(12 (7 1 6 1 3))
(12 (1 7 7 20 3))

(58 (7 7 1 7)(
 4 6 5 4 3))

(13 (2 1 7 2 2)(
1 2 6 3
1 3 3 4
1 4 4 5
1 5 5 6
6 7 1 2
5 8 2 6
9 5 2 5))

(13 (3 8 b 3 2)(
a 6 1 0
6 9 2 0
4 b 4 0
9 4 5 0))

(13 (4 c f 3 2)(
2 8 6 0
c 2 3 0
8 7 2 0
7 d 1 0))

(13 (5 10 10 a 2)(
d a 1 0))

(13 (6 11 12 24 2)(
3 c 3 0
b 3 4 0))

(13 (b 13 13 1f 2)(
c 8 7 0))

(13 (a 14 14 1f 2)(
```

```
b c 7 0))

(13 (9 15 15 1f 2)(
9 b 7 0))

(13 (8 16 16 1f 2)(
9 8 2 7))

(59 (13 13 b 4)(
 2 d c))

(59 (14 14 a 6)(
 2 12 11))

(59 (15 15 9 3)(
 2 b a))

(59 (16 16 8 2)(
 2 7 6))

(10 (1 1 d 1 2)
(
 2.50000000e+00  5.00000000e-01
 2.50000000e+00  1.00000000e+00
 3.00000000e+00  5.00000000e-01
 2.50000000e+00  0.00000000e+00
 2.00000000e+00  5.00000000e-01
 1.00000000e+00  0.00000000e+00
 1.00000000e+00  1.00000000e+00
 2.00000000e+00  1.00000000e+00
 2.00000000e+00  0.00000000e+00
 0.00000000e+00  0.00000000e+00
 3.00000000e+00  0.00000000e+00
 3.00000000e+00  1.00000000e+00
 0.00000000e+00  1.00000000e+00))
```