Appendix B. ANSYS Fluent File Formats

This appendix provides information about the following:

B.1. Case and Data File Formats

This section describes the contents and formats of ANSYS Fluent case and data files. After discussing the Guidelines (p. 3301) and Formatting Conventions in Binary and Formatted Files (p. 3301), the section descriptions are grouped according to function:

- Grid Sections (p. 3302) : Creating grids for ANSYS Fluent.
- Other (Non-Grid) Case Sections (p. 3313)
- Data Sections (p. 3315) : Importing solutions into another postprocessor.

The case and data files may contain other sections that are intended for internal use only.

B.1.1. Guidelines

The ANSYS Fluent case and data files are broken into several sections according to the following guidelines:

- Each section is enclosed in parentheses and begins with a decimal integer indicating its type. This integer is different for formatted and binary files (Formatting Conventions in Binary and Formatted Files (p. 3301)).
- 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 releases.
- Header information for lists of items is enclosed in separate sets of parentheses preceding the items, and the items are enclosed in their own parentheses.

B.1.2. Formatting Conventions in Binary and Formatted Files

For formatted files, examples of file sections are given in Grid Sections (p. 3302) and Other (Non-Grid) Case Sections (p. 3313). For binary files, the header indices described in this section (for example, `10` for the node section) are preceded by `20` for single-precision binary data, or by `30` for double-precision binary data (for example, `2010` or `3010` instead of `10`). The end of the binary data is indicated by `End of Binary Section 2010` or `End of Binary Section 3010` before the closing parenthesis of the section.

An example with the binary data represented by periods is as follows:

```
(2010 (2 1 2aad 2 3)(
 .
```

```
.
.
)
End of Binary Section 2010)
```

B.1.3. Grid Sections

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. The currently defined grid sections are:

- Comment (See Comment (p. 3302))
- Header (See Header (p. 3303))
- Dimensions (See Dimensions (p. 3303))
- Nodes (See Nodes (p. 3303))
- Periodic Shadow Faces (See Periodic Shadow Faces (p. 3304))
- Cells (See Cells (p. 3305))
- Faces (See Faces (p. 3306))
- Face Tree (See Face Tree (p. 3308))
- Cell Tree (See Cell Tree (p. 3308))
- Interface Face Parents (See Interface Face Parents (p. 3309))

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`), which is available from ANSYS Inc., or as macros in a C header file (`xfile.h`), which is located in your installation area.

B.1.3.1. Comment

Index:	0
Scheme symbol:	`xf-comment`
C macro:	`XF_COMMENT`
Status:	optional

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

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

You should precede each long section, or group of related sections, by a comment section explaining what is to follow.

Example:

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

B.1.3.2. Header

Index:	1
Scheme symbol:	`xf-header`
C macro:	XF_HEADER
Status:	optional

Header sections can appear anywhere in the file (except within other sections). The following is an example:

```
(0 "fluent19.0.0 build-id: 0")
```

The purpose of this section is to identify the program that wrote the file. Although it can appear anywhere, it is one of the first sections in the file. Additional header sections indicate other programs that may have been used in generating the file. It provides a history mechanism showing where the file came from and how it was processed.

B.1.3.3. Dimensions

Index:	2
Scheme symbol:	`xf-dimension`
C macro:	XF_DIMENSION
Status:	optional

The dimensions of the grid appear as:

```
(2 ND)
```

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

B.1.3.4. Nodes

Index:	10
Scheme symbol:	`xf-node`
C macro:	XF_NODE
Status:	required

Format:

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

- If `zone-id` is zero, this provides the total number of nodes in the grid. `first-index` will then be one, `last-index` will be the total number of nodes *in hexadecimal*, `type` is equal to `1`, `ND` is the dimensionality of the grid, and there are no coordinates following (the parentheses for the coordinates are omitted as well).

  For example: `(10 (0 1 2d5 1 2))`

- If `zone-id` is greater than zero, it indicates the zone to which the nodes belong. `first-index` and `last-index` are the indices of the nodes in the zone, *in hexadecimal*. The values of `last-index` in each zone must be less than or equal to the value in the declaration section. **Type** is always equal to `1`.

  `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 by the node header, then only x and y coordinates are present on each line.

The following is an example of a 2D 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
 ))
```

Because the grid connectivity is composed of integers representing pointers (see Cells and Faces), using hexadecimal conserves space in the file and provides for faster file input and output. The header indices are 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.1.3.5. Periodic Shadow Faces

Index:	18
Scheme symbol:	`xf-periodic-face`
C macro:	`XF_PERIODIC_FACE`
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 f11
 f20 f21
 .
 .
 .
 ))
```

where

`first-index` = index of the first periodic face pair in the list

`last-index` = index of the last periodic face pair in the list

`periodic-zone` = zone ID of the periodic face zone

`shadow-zone` = 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.

Note

In this case, `first-index` and `last-index` do *not* refer to face indices. They refer to indices in the list of periodic pairs.

Example:

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

B.1.3.6. Cells

Index:	12
Scheme symbol:	`xf-cell`
C macro:	XF_CELL
Status:	required

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

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

Again, `zone-id` is zero to indicate 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 ANSYS Fluent that it cannot be used. In a declaration section, the `type` has a value of zero and the `element-type` is not present.

For example,

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

It 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
7	polyhedral	NN	NF

where NN and NF will vary, depending on the specific polyhedral cell.

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.

In the earlier versions of ANSYS Fluent, a distinction was made between solid and fluid zones. This is now determined by properties (that is, material type).

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

Example:

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

Here, 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.

B.1.3.7. Faces

Index:	13
Scheme symbol:	`xf-face`
C macro:	XF_FACE
Status:	required

The format for face sections is as follows:

```
(13 (zone-id first-index last-index bc-type face-type))
```

where

`zone-id` = zone ID of the face section
`first-index` = index of the first face in the list
`last-index` = index of the last face in the list
`bc-type` = ID of the boundary condition represented by the face section
`face-type` = ID of the type(s) of face(s) in the section

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

`bc-type`	*description*
2	interior
3	wall
4	pressure-inlet, inlet-vent, intake-fan
5	pressure-outlet, exhaust-fan, outlet-vent

7	symmetry
8	periodic-shadow
9	pressure-far-field
10	velocity-inlet
12	periodic
14	fan, porous-jump, radiator
20	mass-flow-inlet, mass-flow-outlet
24	interface
31	parent (hanging node)
36	outflow
37	axis

The faces resulting from the intersection of non-conformal grids are placed in a separate face zone, where a factor of 1000 is added to the `bc-type` (for example, 1003 is a wall zone).

The current valid face types are defined in the following table:

`face-type`	*description*	*nodes/face*
0	mixed	
2	linear	2
3	triangular	3
4	quadrilateral	4
5	polygonal	NN

where NN will vary, depending on the specific polygonal face.

A `zone-id` of zero indicates a declaration section, which provides a count of the total number of faces in the file. Such a section omits the `bc-type` and is not followed by a body with further information.

A nonzero `zone-id` indicates a regular face section, and will be followed by a body that contains information about the grid connectivity. Each line of the body will describe one face and will have the following format:

```
n0 n1 n2 c0 c1
```

where,

`n*` = defining nodes (vertices) of the face

`c*` = adjacent cells

This is the format for a 3D grid with a triangular face format. The actual number of nodes depends on the `face-type`. The order of the cell indices is important, and is determined by the right-hand rule: if you curl the fingers of your right hand in the order of the nodes, your thumb will point toward `c0`.

For 2D grids, `n2` is omitted. `c1` is determined by the cross product of two vectors, $\hat{r}$ and $\hat{k}$. The $\hat{r}$ vector extends from `n0` to `n1`, whereas the $\hat{k}$ vector has its origin at `n0` and points out of the grid plane toward the viewer. If you extend your right hand along $\hat{r}$ and curl your fingers in the direction of the angle between $\hat{r}$ and $\hat{k}$, your thumb will point along $\hat{r}\times\hat{k}$ toward `c1`.

If the face zone is of mixed type (face-type= 0) or of polygonal type (face-type= 5), each line of the section body will begin with a reference to the number of nodes that make up that particular face, and has the following format:

```
x n0 n1 ... nf c0 c1
```

where,

x = the number of nodes (vertices) of the face

nf = the final node of the face

All cells, faces, and nodes have positive indices. If a face has a cell only on one side, then either c0 or c1 is zero. For files containing only a surface mesh, both these values are zero.

For information on face-node connectivity for various cell types in ANSYS Fluent, refer to Face-Node Connectivity in ANSYS Fluent (p. 503).

B.1.3.8. Face Tree

Index:	59
Scheme symbol:	xf-face-tree
C macro:	XF_FACE_TREE
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 = index of the first parent face in the section

face-id1 = index of the last parent face in the section

parent-zone-id = ID of the zone containing parent faces

child-zone-id = ID of the zone containing children faces

number-of-kids = the number of children of the parent face

kid-id-n = the face IDs of the children

These are in hexadecimal format.

B.1.3.9. Cell Tree

Index:	58
Scheme symbol:	xf-cell-tree
C macro:	XF_CELL_TREE
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` = index of the first parent cell in the section

`cell-id1` = index of the last parent cell in the section

`parent-zone-id` = ID of the zone containing parent cells

`child-zone-id` = ID of the zone containing children cells

`number-of-kids` = the number of children of the parent cell

`kid-id-n` = the cell IDs of the children

These are in hexadecimal format.

B.1.3.10. Interface Face Parents

Index:	61
Scheme symbol:	`xf-face-parents`
C macro:	XF_FACE_PARENTS
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` = index of the first child face in the section

`face-id1` = index of the last child face in the section

`parent-id-*` = index of parent faces

These are in hexadecimal format.

If you set up and save a non-conformal mesh in the solution mode of Fluent and then read it using the meshing mode of Fluent, this section will be skipped; consequently, all the information necessary to

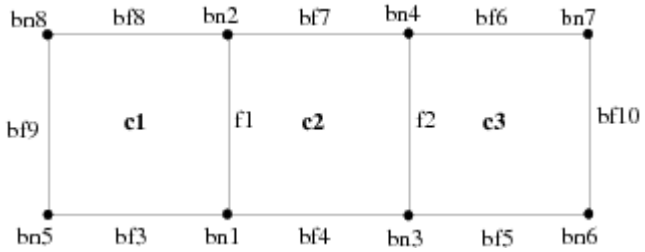
preserve the non-conformal interface will not be maintained. When you switch to or read the mesh back into the solution mode, you will need to recreate the interface.

B.1.3.11. Example Files

B.1.3.11.1. Example 1

Figure 1: Quadrilateral Mesh (p. 3310) illustrates a simple quadrilateral mesh with no periodic boundaries or hanging nodes.

Figure 1: Quadrilateral Mesh



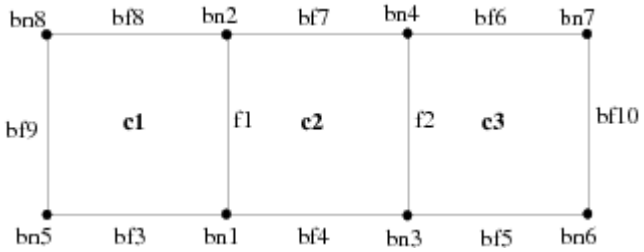
The following describes this mesh:

```
(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))
```

B.1.3.11.2. Example 2

Figure 2: Quadrilateral Mesh with Periodic Boundaries (p. 3311) 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 2: Quadrilateral Mesh with Periodic Boundaries



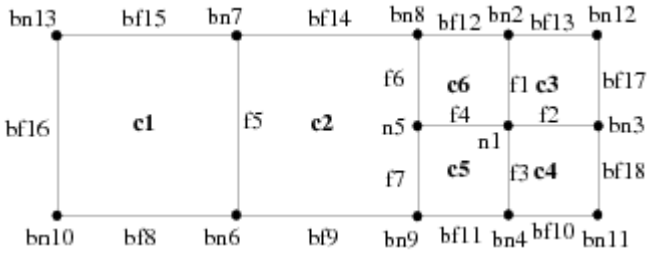
The following describes this mesh:

```
(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))
```

B.1.3.11.3. Example 3

Figure 3: Quadrilateral Mesh with Hanging Nodes (p. 3312) illustrates a simple quadrilateral mesh with hanging nodes.

Figure 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))
```

B.1.4. Other (Non-Grid) Case Sections

The following sections store boundary conditions, material properties, and solver control settings.

B.1.4.1. Zone

Index:	39 or 45
Scheme symbol:	`xf-rp-tv`
C macro:	XF_RP_TV
Status:	required

There is typically one zone section for each zone referenced by the grid. Although some grid zones may not have corresponding zone sections, there cannot be more than one zone section for each zone.

A zone section has the following form:

```
(39 (zone-id zone-type zone-name domain-id)(
(condition1 . value1)
(condition2 . value2)
(condition3 . value3)
.
.
.
 ))
```

Grid generators and other preprocessors need only provide the section header and leave the list of conditions empty, as in

```
(39 (zone-id zone-type zone-name domain-id)())
```

The empty parentheses at the end are required. The solver adds conditions as appropriate, depending on the zone type. When only `zone-id`, `zone-type`, `zone-name`, and `domain-id` are specified, the index 45 is preferred for a zone section. However, the index 39 must be used if boundary conditions

are present, because any and all remaining information in a section of index 45 after zone-id, zone-type, zone-name, and domain-id will be ignored.

Here the zone-id is in *decimal* format. This is in contrast to the use of hexadecimal in the grid sections.

The zone-type is one of the following:

```
axis
exhaust fan
fan
fluid
inlet vent
intake fan
interface
interior
mass-flow-inlet
mass-flow-outlet
outlet vent
outflow
periodic
porous-jump
pressure-far-field
pressure-inlet
pressure-outlet
radiator
shadow
solid
symmetry
velocity-inlet
wall
```

The interior, fan, porous-jump, and radiator types can be assigned only to zones of faces inside the domain. The interior type is used for the faces within a cell zone; the others are for interior faces that form infinitely thin surfaces within the domain. ANSYS Fluent allows the wall type to be assigned to face zones both on the inside and on the boundaries of the domain. Some zone types are valid only for certain types of grid components. For example, cell (element) zones can be assigned only one of the following types:

```
fluid
 solid
```

All of the other types listed above can be used only for boundary (face) zones.

The zone-name is a user-specified label for the zone. It must be a valid Scheme symbol [1] and is written without quotes. The rules for a valid zone-name (Scheme symbol) are as follows:

- The first character must be a lowercase letter [2] or a special-initial.
- Each subsequent character must be a lowercase letter, a special-initial, a digit, or a special-subsequent.

where a special-initial character is one of the following:

```
! $ % & * / : < = > ? ~ _ ^
```

and a special-subsequent is one of the following:

```
. + -
```

[1] See Revised (4) Report on the Algorithmic Language Scheme, William Clinger and Jonathan Rees (Editors), 2 November 1991, Section 7.1.1.

[2] The Standard actually only requires that case be insignificant; the ANSYS Fluent implementation accomplishes this by converting all uppercase input to lowercase.

Examples of valid zone names are `inlet-port/cold!—, eggs/easy`, and `e=m*c^2`.

Some examples of zone sections produced by grid generators and preprocessors are as follows:

```
(39 (1 fluid fuel 1)())
 (39 (8 pressure-inlet pressure-inlet-8 2)())
 (39 (2 wall wing-skin 3)())
 (39 (3 symmetry mid-plane 1)())
```

The `domain-id` is an integer that appears after the zone name, associating the boundary condition with a particular phase or mixture (sometimes referred to as phase-domains and mixture-domains).

B.1.4.2. Partitions

Index:	40
Scheme symbol:	`xf-partition`
C macro:	`XF_PARTITION`
Status:	only for partitioned grids

This section indicates each cell's partition. The format of the section is as follows:

```
(40 (zone-id first-index last-index partition-count)(
p1
p2
p3
.
.
.
pn
))
```

where,

`p1` = the partition of the cell whose ID is `first-index`

`p2` = the partition of the cell whose ID is `first-index` +1, and so on

`pn` = the partition of the cell whose ID is `last-index`

`partition-count` = the total number of partitions

Partition IDs must be between 0 and one less than `partition-count`.

B.1.5. Data Sections

The following sections store iterations, residuals, and data field values.

B.1.5.1. Grid Size

Index:	33
Scheme symbol:	`xf-grid-size`
C macro:	`XF_GRID_SIZE`
Status:	optional

This section indicates the number of cells, faces, and nodes in the grid that corresponds to the data in the file. This information is used to check that the data and grid match. The format is

```
(33 (n-elements n-faces n-nodes))
```

where the integers are written in decimal.

B.1.5.2. Data Field

Index:	300
Scheme symbol:	`xf-rf-seg-data`
C macro:	XF_RF_SEG_DATA
Status:	required

This section lists a flow field solution variable for a cell or face zone. The data are stored in the same order as the cells or faces in the case file. Separate sections are written out for each variable for each face or cell zone on which the variable is stored. The format is

```
(300 (sub-section-id zone-id size n-time-levels
 n-phases first-id last-id)
 ( data for cell or face with id = first-id
 data-for-cell-or-face with id = first-id+1
 ..
 data-for-cell-or-face with id = last-id
 ))
```

where `sub-section-id` is a (decimal) integer that identifies the variable field (for example, 1 for pressure, 2 for velocity). The complete list of these is available in the header file (`xfile.h`), which is located in your installation area.

where,

`zone-id` = the ID number of the cell or face zone

`size` = the length of the variable vector

`zone-id` matches the ID used in case file. `size` is 1 for a scalar, 2 or 3 for a vector, equal to the number of species for variables defined for each species). `n-time-levels` currently are not used.

A sample data file section for the velocity field in a cell zone for a steady-state, single-phase, 2D problem is shown below:

```
(300 (2 16 2 0 0 17 100)
(8.08462024e-01 8.11823010e-02
 8.78750622e-01 3.15509699e-02
 1.06139672e+00 -3.74040119e-02
...
 1.33301604e+00 -5.04243895e-02
 6.21703446e-01 -2.46118382e-02
 4.41687912e-01 -1.27046436e-01
 1.03528820e-01 -1.01711005e-01
 ))
```

The variables that are listed in the data file depend on the models active at the time the file is written. Variables that are required by the solver based on the current model settings but are missing from the data file are set to their default values when the data file is read. Any extra variables that are present in the data file but are not relevant according to current model settings are ignored.

B.1.5.3. Residuals

Index:	302
Scheme symbol:	`xf-rf-scaled-residuals`
C macro:	XF_RF_SCALED_RESIDUALS
Status:	optional

This section lists the values of the residuals for a particular data field variable at each iteration:

```
(302 (n residual-section-id size domain-id)
(
iteration_number unscaled_residual scaling_factor
 .
 .
 .
 ))
```

where,

n = the number of residuals

`size` = the length of the variable vector

`residual-section-id` = an integer (decimal) indicating the equation

`domain-id` = domain ID

`size` is 1 for a scalar, 2 or 3 for a vector, equal to the number of species for variables defined for each species. The `residual-section-id` indicates the equation for which the residual is stored in the section, according to the C constants defined in a header file (`xfile.h`) available in your installation area, as noted in Grid Sections (p. 3302).

The equations for which residuals are listed in the data file depend on the models active at the time the file is written. If the residual history is missing from the data file for a currently active equation, it is initialized with zeros.

B.2. Mesh Morpher/Optimizer File Formats

This section describes the format of the ASCII text files that can be used as part of the setup for the mesh morpher/optimizer, as described in Using the Mesh Morpher/Optimizer (p. 2571).

The following is an example of a file that can be read or written from the Define Control Points Dialog Box (p. 3134):

```
Control Point Positions
CP No.        X            Y             Z
1             0.73398191   -0.048181638  5.0344162
2             0.51003051   0.52833155    5.100265
3             0.10806149   0.72560567    5.1306067
4             -0.37284759  0.62890923    5.1368165
5             -0.69803506  0.21491928    5.144598
```

The following is an example of a file that can be read or written from the Motion Settings Dialog Box (p. 3200) when **Regular** is selected from the **Control Point Distribution** list in the **Regions** tab of the Mesh Morpher/Optimizer Dialog Box (p. 3185):

```
Reg     CP      Par      X       Y       Z
def-0   2       par1     -1      0       0
def-0   4       par1     2       -1      3
```

```
def-0   2       par2     0       1       0
def-1   2       par3     0       0       1
def-1   3       par1     2       0       2
```

The following is an example of a file that can be read or written from the Motion Settings Dialog Box (p. 3200) when **Unstructured** is selected from the **Control Point Distribution** list in the **Regions** tab of the Mesh Morpher/Optimizer Dialog Box (p. 3185):

```
Motion Definition
Name        Type        Par         Dir-x       Dir-y       Dir-z       Orig-x      Orig-y      Orig-z
rad-par1-1  Rad         par-1       0           0           1           0           0           0
tra-par2-1  Trans       par-2       1           1           1
rot-par3-1  Rot         par-3       1           0           0           2           1           1

Assignment to Control Points
Name        Control Points...
rad-par1-1  1           2           3           4           5           6           7
```

When creating or editing a text file used to set up the mesh morpher/optimizer, note the following:

- The file can be created from nothing using a text editor or spreadsheet program, as long as it adheres to the formats described in this section. An easier alternative is to apply some sample definitions using the appropriate dialog box, write a file, and then edit the file using a text editor or spreadsheet program.
- The file must be a tab-delimited ASCII text file.
- The names of the deformation regions (for example, `def-0`), deformation parameters (for example, `par1`), and motions (for example, `tra-par2-1`) must correspond to those that exist in the intended case file.
- When defining control points, the coordinates must lie within a deformation region in the intended case file.
- For the motion settings, the control point numbers must range between 1 and the maximum number of available control points in the intended case file.
- The direction components and coordinates can be any real numbers.

B.3. Shell Conduction Settings File Format

This section describes the format of the CSV files that can define shell conduction settings, as described in Managing Shell Conduction Walls (p. 1233). The following is an example of such a file:

```
Zone ID,Zone Name,Layer Number,Material,Layer Thickness (m),Heat Generation Rate (w/m3)
15,wall-1,1,substrate,5.00E-08,0
15,wall-1,2,substrate,5.00E-08,0
15,wall-1,3,substrate,5.00E-08,0
15,wall-1,4,substrate,5.00E-08,0
15,wall-1,5,gold,2.50E-08,0
15,wall-1,6,gold,2.50E-08,0
16,wall-2,1,silicon,5.00E-08,6.25E+11
16,wall-2,2,silicon,5.00E-08,6.25E+11
16,wall-2,3,substrate,5.00E-08,0
```

When creating or editing a file that defines the shell conduction settings, note the following:

- The file can be created from nothing using a text editor or spreadsheet program, as long as it adheres to the format described in this section. An easier alternative is to apply some sample shell conduction definitions using the Shell Conduction Manager Dialog Box (p. 3257), generate a file using the **Write...** button, and then edit the file using a text editor or spreadsheet program.
- The file must be a comma-delimited ASCII file.

- The first line contains the column headings shown in the previous example, and then each of the subsequent lines define the settings for a layer.
- The zone IDs, zone names, layer numbers, and materials (for example, `wall-1`, `substrate`) must correspond to those that exist in the intended case file.
- The layer thicknesses must be positive, nonzero values.
- Information will not be written for any wall that defines the heat generation rate using a UDF.

B.4. 3D Fan Curve File Format

This section describes the format of the ASCII text files that can define the relationship between pressure and flow rate for all of the 3D fan cell zones in a case file, as described in 3D Fan Zones (p. 679). The following is an example of such a file:

```
cell zone id
17 !CELL_ID
Pressure(Pa) Flowrate(m3/s)
-324.0 1.255
-215.0 1.162
-109.0 1.032
0.0 0.997
63.0 0.913
105.0 0.862
302.0 0.584
394.0 0.492
406.0 0.432
443.0 0.365
cell zone id
18 !CELL_ID
Pressure(Pa) Flowrate(m3/s)
-333.0 1.130
-215.0 1.122
-165.0 1.064
0.0 0.981
42.0 0.912
90.0 0.855
112.0 0.786
303.0 0.546
344.0 0.492
395.0 0.451
411.0 0.399
442.0 0.324
```

When creating or editing a file that defines the 3D fan zone curve, note the following:

- The file can be created from nothing using a text editor or spreadsheet program, as long as it adheres to the format described in this section.
- The file must be a tab-delimited ASCII file.
- A single file can define multiple cell zones, but it must be read in each of the applicable **Fluid** dialog boxes.
- For each section that defines the curve for a cell zone, the first three lines must follow the convention shown in the previous example, and then each of the subsequent lines must list the experimental results.
- The zone IDs (for example, 12) must correspond to those that exist in the intended case file.
- The units must match those in the case file.