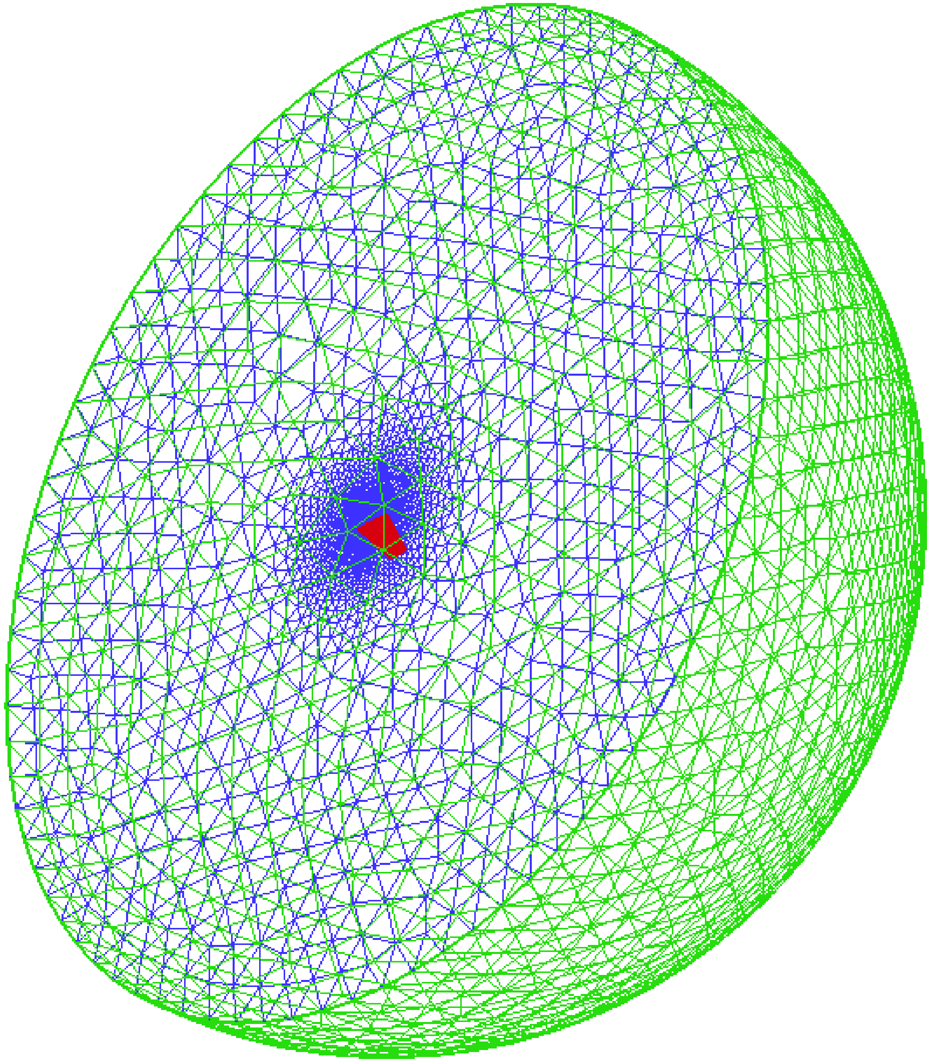
CFD General Notation System (CGNS)

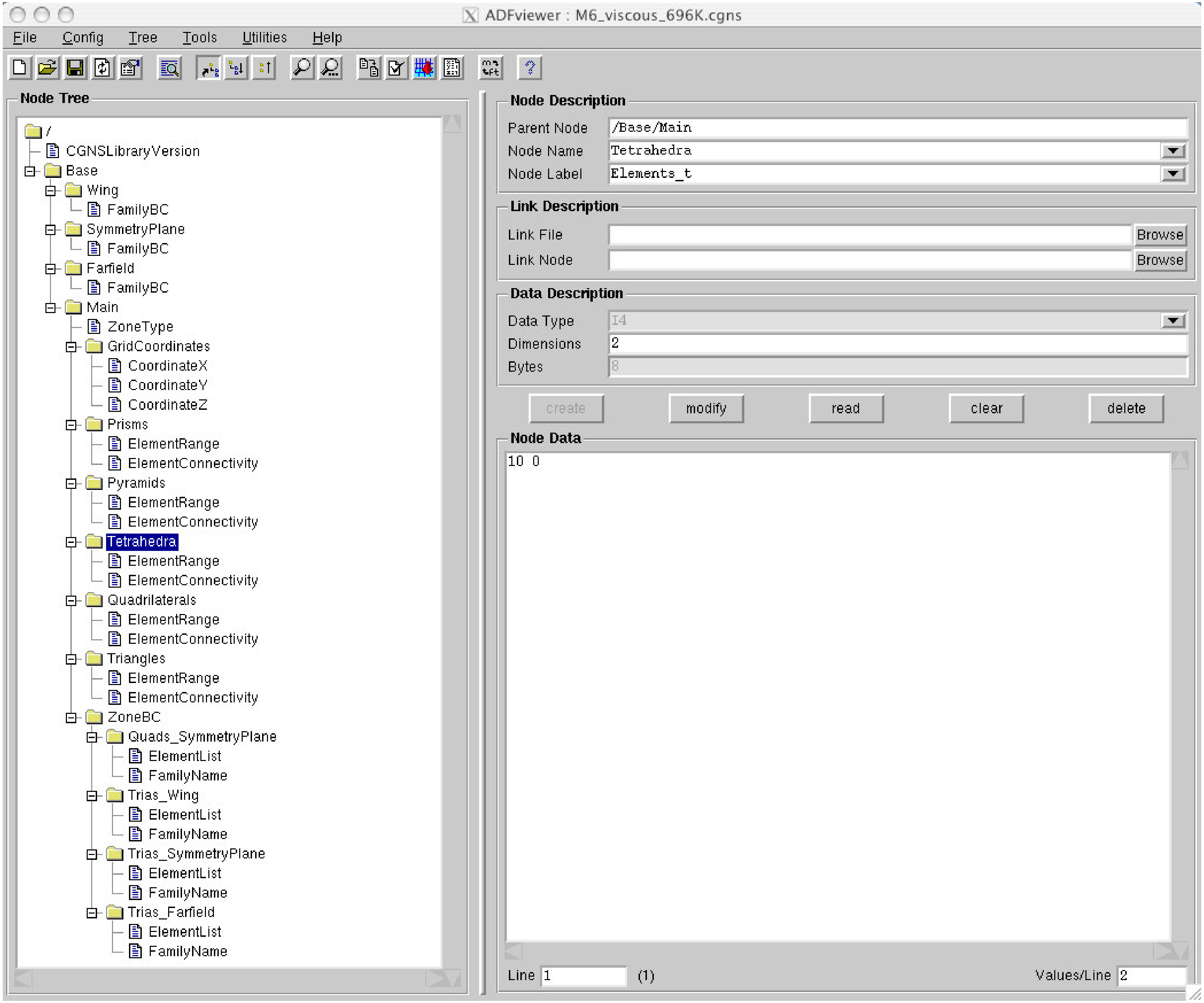
Usage for unstructured grids

Edwin van der Weide

Stanford University

Example Unstructured Grid





Node Tree

```
/
├ CGNSLibraryVersion
└ Base
  ├ Wing
  │ └ FamilyBC
  ├ SymmetryPlane
  │ └ FamilyBC
  ├ Farfield
  │ └ FamilyBC
  └ Main
    ├ ZoneType
    ├ GridCoordinates
    │ ├ CoordinateX
    │ ├ CoordinateY
    │ └ CoordinateZ
    ├ Prisms
    │ ├ ElementRange
    │ └ ElementConnectivity
    ├ Pyramids
    │ ├ ElementRange
    │ └ ElementConnectivity
    ├ Tetrahedra
    │ ├ ElementRange
    │ └ ElementConnectivity
    ├ Quadrilaterals
    │ ├ ElementRange
    │ └ ElementConnectivity
    ├ Triangles
    │ ├ ElementRange
    │ └ ElementConnectivity
    └ ZoneBC
      ├ Quads_SymmetryPlane
      │ ├ ElementList
      │ └ FamilyName
      ├ Trias_Wing
      │ ├ ElementList
      │ └ FamilyName
      ├ Trias_SymmetryPlane
      │ ├ ElementList
      │ └ FamilyName
      └ Trias_Farfield
        ├ ElementList
        └ FamilyName
```

Node Description

Parent Node	/Base/Main
Node Name	Tetrahedra
Node Label	Elements_t

Link Description

Link File	
Link Node	

Data Description

Data Type	I4
Dimensions	2
Bytes	8

Node Data

```
10 0
```

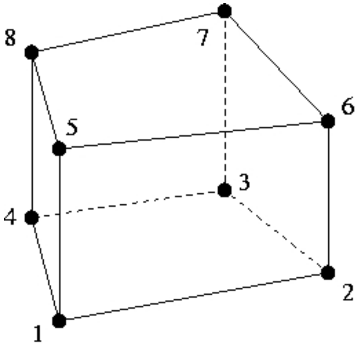
Line 1 (1) Values/Line 2

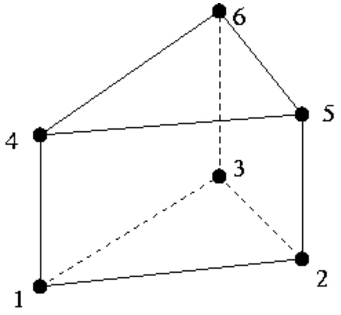
Unstructured grid storage

- Several possibilities to store an unstructured grid.
  - Every element type is stored in a separate Elements_t node. Recommended.
  - One Elements_t node, which stores all elements using the MIXED Element type.
  - Store all elements as arbitrary polygons, NGON_n Element type.
  - Arbitrary combinations of the possibilities above.
  - Pros
    - Flexibility.
  - Cons
    - Reading becomes complicated.

Connectivities (linear elements)

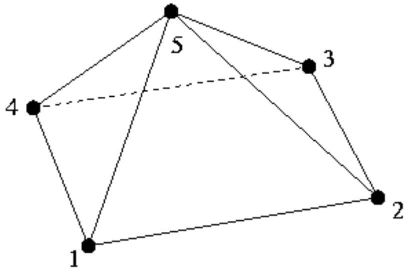
HEXA_8

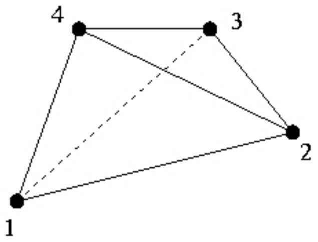




PENTA_6

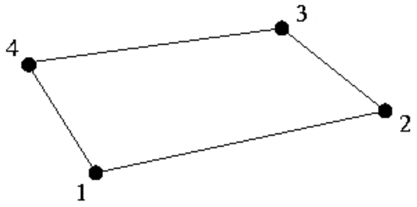
PYRA_5

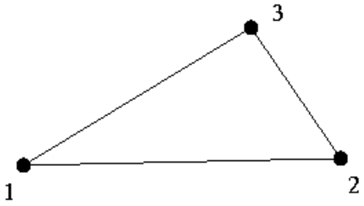




TETRA_4

QUAD_4



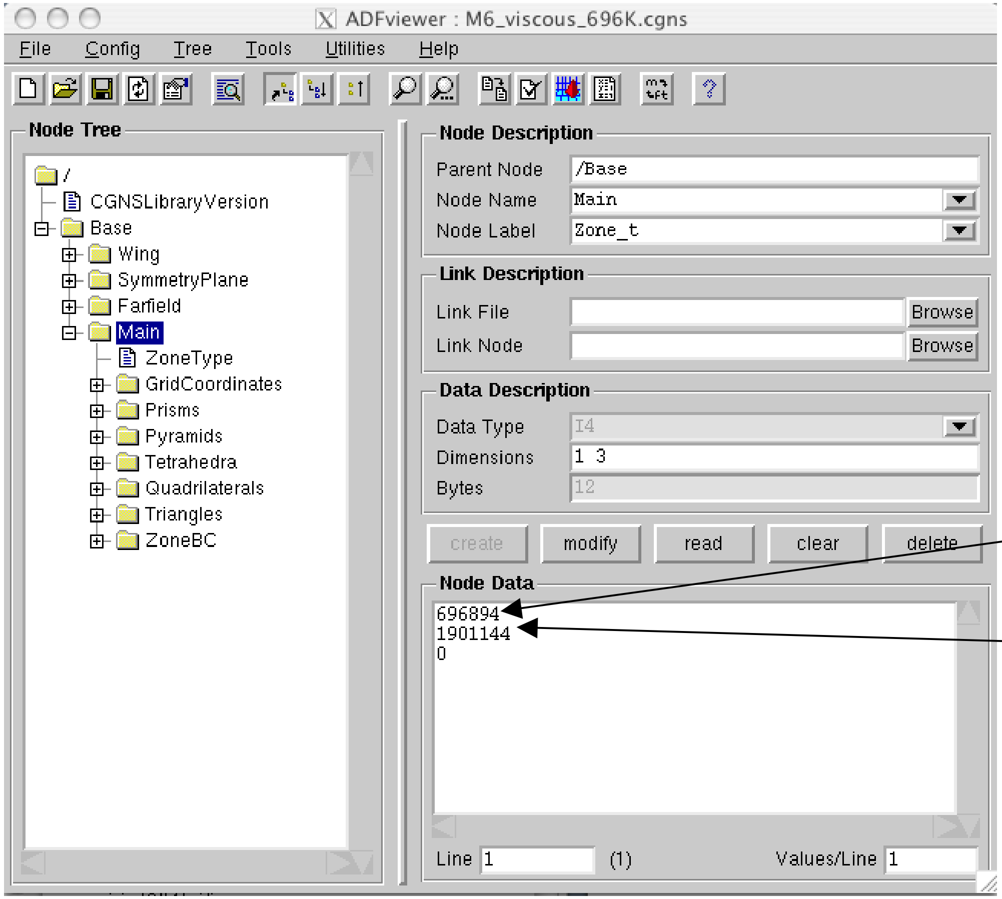


TRI_3

See http://www.grc.nasa.gov/WWW/cgns/sids/conv.html#unstructgrid for all supported elements.

Info in the zone

- # elements = # elements of highest dimension.
  - E.g. for a 3D problem the number elements of the surface grid should NOT be stored in the zone.



```
/
  CGNSLibraryVersion
  Base
    Wing
    SymmetryPlane
    Farfield
    Main
      ZoneType
      GridCoordinates
      Prisms
      Pyramids
      Tetrahedra
      Quadrilaterals
      Triangles
      ZoneBC
```

Parent Node: /Base
Node Name: Main
Node Label: Zone_t
Link File:
Link Node:
Data Type: I4
Dimensions: 1 3
Bytes: 12

Node Data:

```
696894
1901144
0
```

Line 1 (1) Values/Line 1

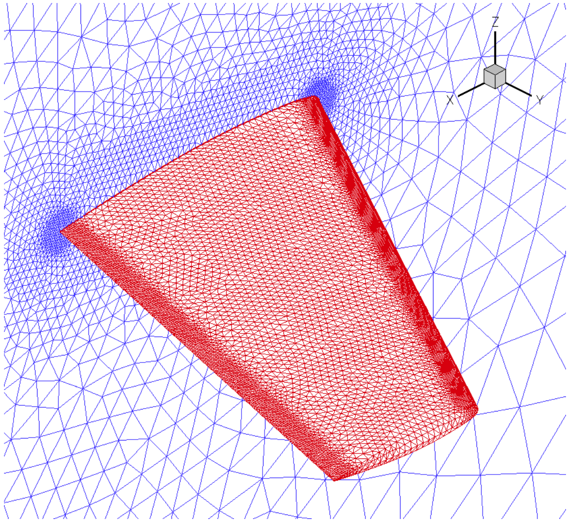
Number of grid points

Number of volume elements

Single Zone vs. Multiple Zones

Single Zone

No relative motion



Multiple Zones

Relative motion or non-matching grids

QuickTime™ and a
decompressor
are needed to see this picture.

Multiple zones can be used to store a domain decomposition

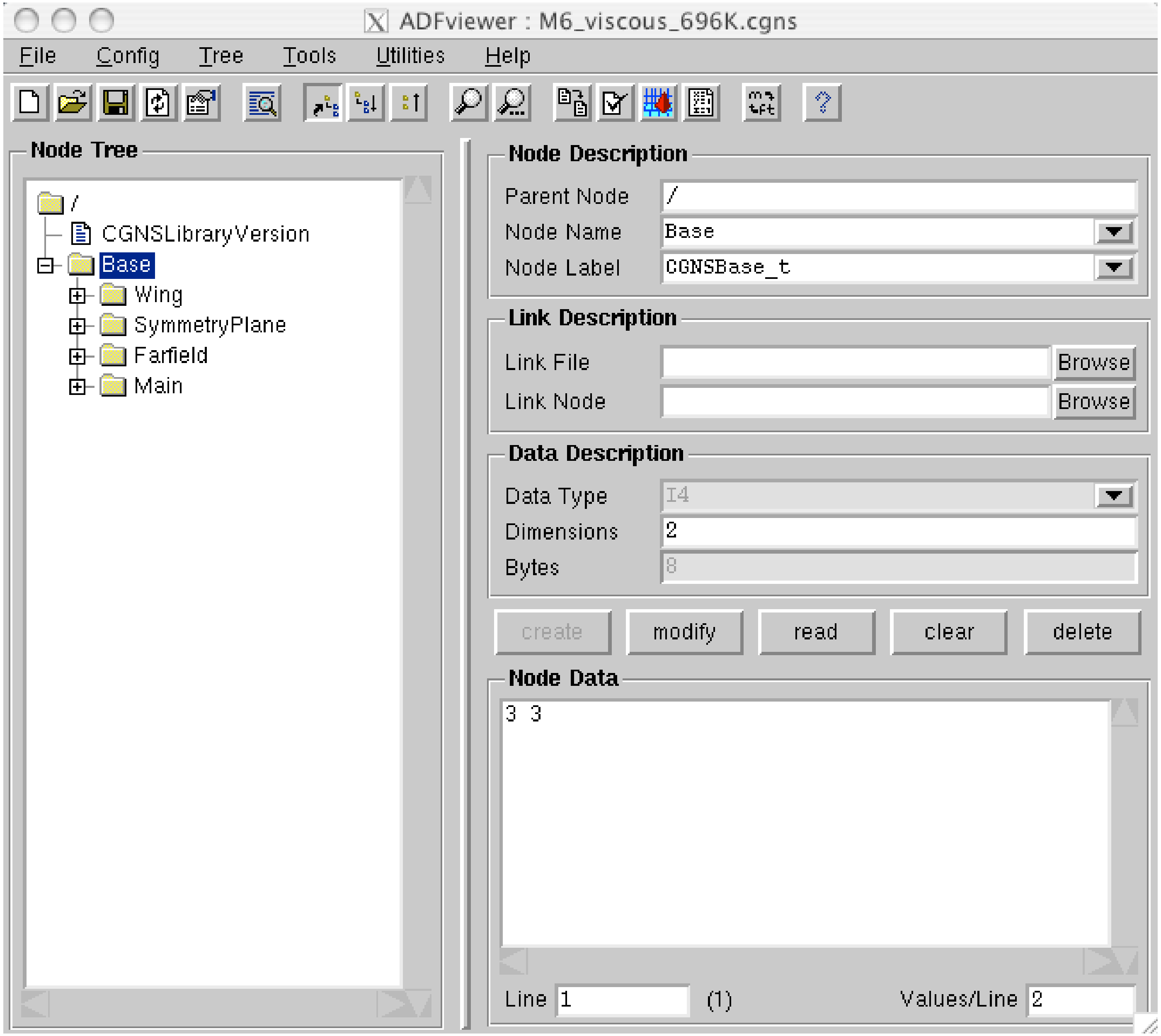
Drawback: not very flexible

Better: use the partial read/write functions

CGNS
cfd data standard

Example – CGNS Code (1)

```
#include “cgnslib.h”

  /* Open the CGNS for reading and check if the file was found. */

  if(cg_open(gridFile, MODE_READ, &fileInd) != CG_OK)
    Terminate(“readGridCGNS”, cg_get_error());

  /* Determine the of bases in the grid. This example assumes */
  /* one base. However it is allowed to have multiple bases.  */

  if(cg_nbases(fileInd, &nBases)!= CG_OK)
    Terminate(“readGridCGNS”, cg_get_error());
  if(nBases != 1)
    Terminate(“readGridCGNS”, “This example assumes one base”);
  base = 1;

  /* Check the cell and physical dimensions of the bases. */
  /* Both should be 3.                                    */

  if(cg_base_read(fileInd, base, cgnsName, &cellDim,
                  &physDim) != CG_OK)
    Terminate(“readGridCGNS”, cg_get_error());
```



/
- CGNSLibraryVersion
- Base
  - Wing
  - SymmetryPlane
  - Farfield
  - Main

Parent Node: /

Node Name: Base

Node Label: CGNSBase_t

Link File:

Link Node:

Data Type: I4

Dimensions: 2

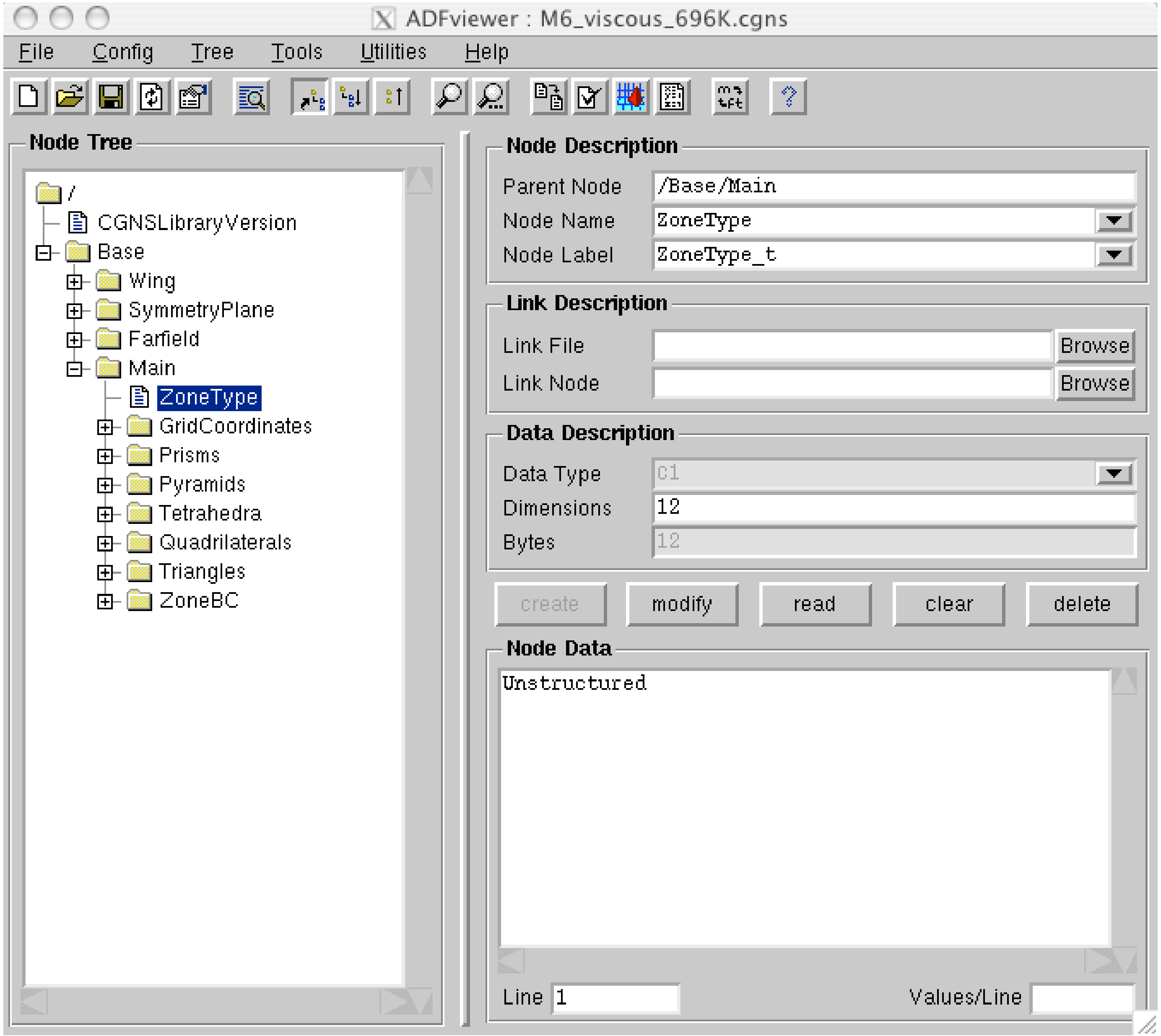
Bytes: 8

```
3 3
```

Line 1 (1) Values/Line 2

Example – CGNS Code (2)

```
/* Read the number of zones in the grid. */
/* This example assumes one zone.        */

if(cg_nzones(fileInd, base, &nZones) != CG_OK)
  Terminate(“readGridCGNS”, cg_get_error());
if(nZones != 1)
  Terminate(“readGridCGNS”, “This example assumes one zone”);
zone = 1;

/* Check the zone type. This should be Unstructured. */

if(cg_zone_type(fileInd, base, zone, &zoneType) != CG_OK)
  Terminate(“readGridCGNS”, cg_get_error());
if(zoneType != Unstructured)
  Terminate(“readGridCGNS”, “Unstructured zone expected”);

/* Determine the number of vertices and volume elements in this */
/* zone (and thus in the grid, because one zone is assumed).    */

if(cg_zone_read(fileInd, base, zone, zoneName, sizes) != CG_OK)
  Terminate(“readGridCGNS”, cg_get_error());
nVertices    = sizes[0];
nVolElements = sizes[1];
```



ADFviewer : M6_viscous_696K.cgns

Node Tree

```
/
├─ CGNSLibraryVersion
└─ Base
   ├─ Wing
   ├─ SymmetryPlane
   ├─ Farfield
   └─ Main
      ├─ ZoneType
      ├─ GridCoordinates
      ├─ Prisms
      ├─ Pyramids
      ├─ Tetrahedra
      ├─ Quadrilaterals
      ├─ Triangles
      └─ ZoneBC
```

Node Description

Parent Node: /Base/Main

Node Name: ZoneType

Node Label: ZoneType_t

Link Description

Link File:

Link Node:

Data Description

Data Type: C1

Dimensions: 12

Bytes: 12

Node Data

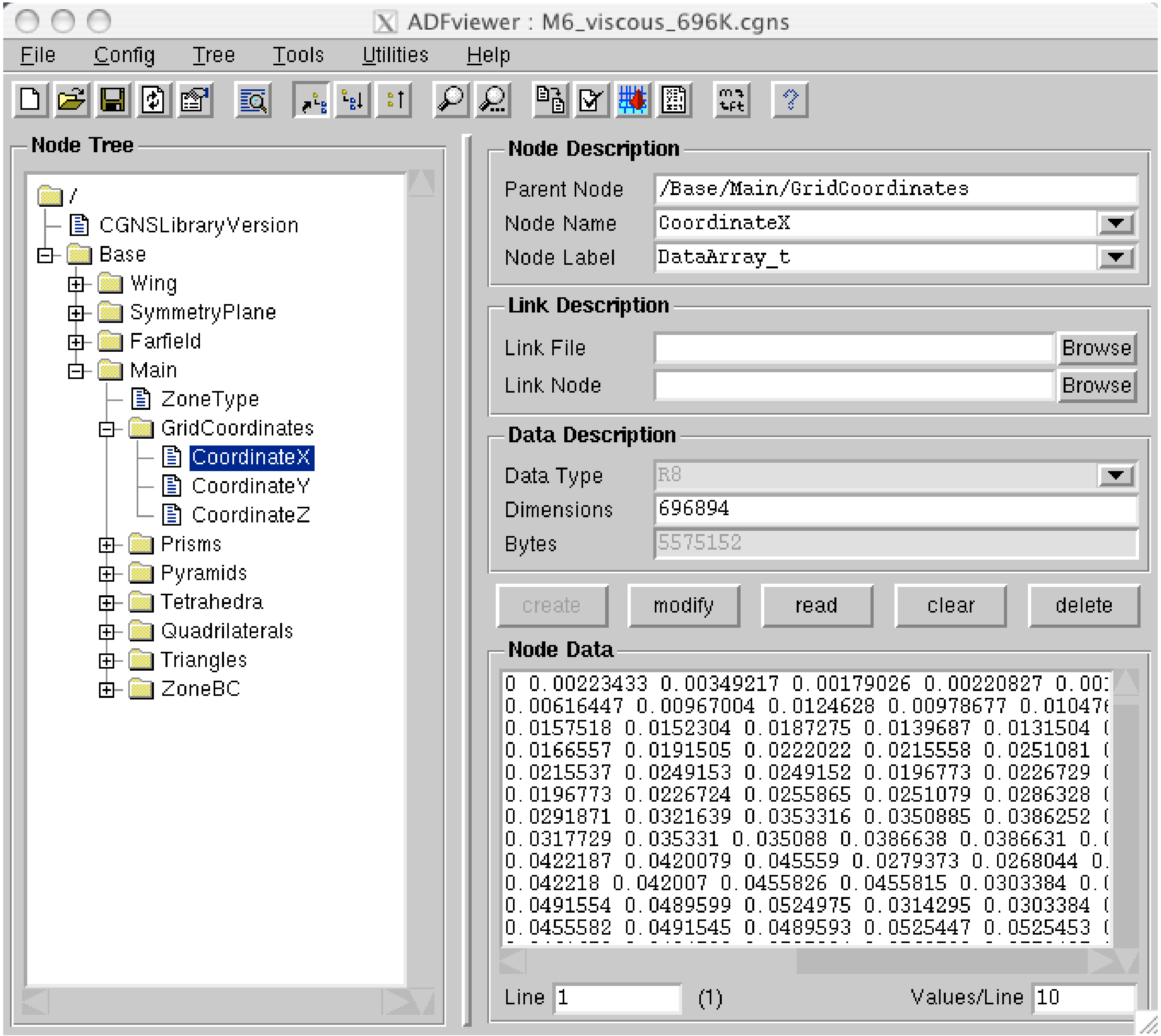
```
Unstructured
```

Line: 1

Values/Line:

Example – CGNS Code (3)

```
/* Determine the number and names of the coordinates. */

if(cg_ncoords(fileInd, base, zone, &nCoords) != CG_OK)
  Terminate(“readGridCGNS”, cg_get_error());

if(cg_coord_info(fileInd, base, zone, 1, &dataType, name) != CG_OK)
  Terminate(“readCGNS”, cg_get_error());

/* Read the x-coordinates. The y and z-coordinates can be read  */
/* similarly. Just replace CoordinateX by CoordinateY and       */
/* CoordinateZ respectively. This assumes Cartesian coordinates */
/* in double precision. Note that CGNS starts the numbering at  */
/* 1 even if C is used.                                         */

one = 1;
if(cg_coord_read(fileInd, base, zone, “CoordinateX”, realDouble,
                 &one, &nVertices, coorX) != CG_OK)
  Terminate(“readGridCGNS”, cg_get_error());

/* Determine the number of sections for this zone. Note that */
/* surface elements can be stored in a volume zone, but they */
/* are NOT taken into account in the number obtained from    */
/* cg_zone_read.                                             */

if(cg_nsections(fileInd, base, zone, &nSections) != CG_OK)
  Terminate(“readGridCGNS”, cg_get_error());
```



Node Tree

```
/
├ CGNSLibraryVersion
⊟ Base
   ⊞ Wing
   ⊞ SymmetryPlane
   ⊞ Farfield
   ⊟ Main
      ├ ZoneType
      ⊟ GridCoordinates
         ├ CoordinateX
         ├ CoordinateY
         └ CoordinateZ
      ⊞ Prisms
      ⊞ Pyramids
      ⊞ Tetrahedra
      ⊞ Quadrilaterals
      ⊞ Triangles
      ⊞ ZoneBC
```

Node Description

Parent Node: /Base/Main/GridCoordinates

Node Name: CoordinateX

Node Label: DataArray_t

Link Description

Link File:

Link Node:

Data Description

Data Type: R8

Dimensions: 696894

Bytes: 5575152

Node Data

```
0 0.00223433 0.00349217 0.00179026 0.00220827 0.00:
0.00616447 0.00967004 0.0124628 0.00978677 0.01047
0.0157518 0.0152304 0.0187275 0.0139687 0.0131504 (
0.0166557 0.0191505 0.0222022 0.0215558 0.0251081 (
0.0215537 0.0249153 0.0249152 0.0196773 0.0226729 (
0.0196773 0.0226724 0.0255865 0.0251079 0.0286328 (
0.0291871 0.0321639 0.0353316 0.0350885 0.0386252 (
0.0317729 0.035331 0.035088 0.0386638 0.0386631 0.(
0.0422187 0.0420079 0.045559 0.0279373 0.0268044 0.
0.042218 0.042007 0.0455826 0.0455815 0.0303384 0.(
0.0491554 0.0489599 0.0524975 0.0314295 0.0303384 (
0.0455582 0.0491545 0.0489593 0.0525447 0.0525453 (
```

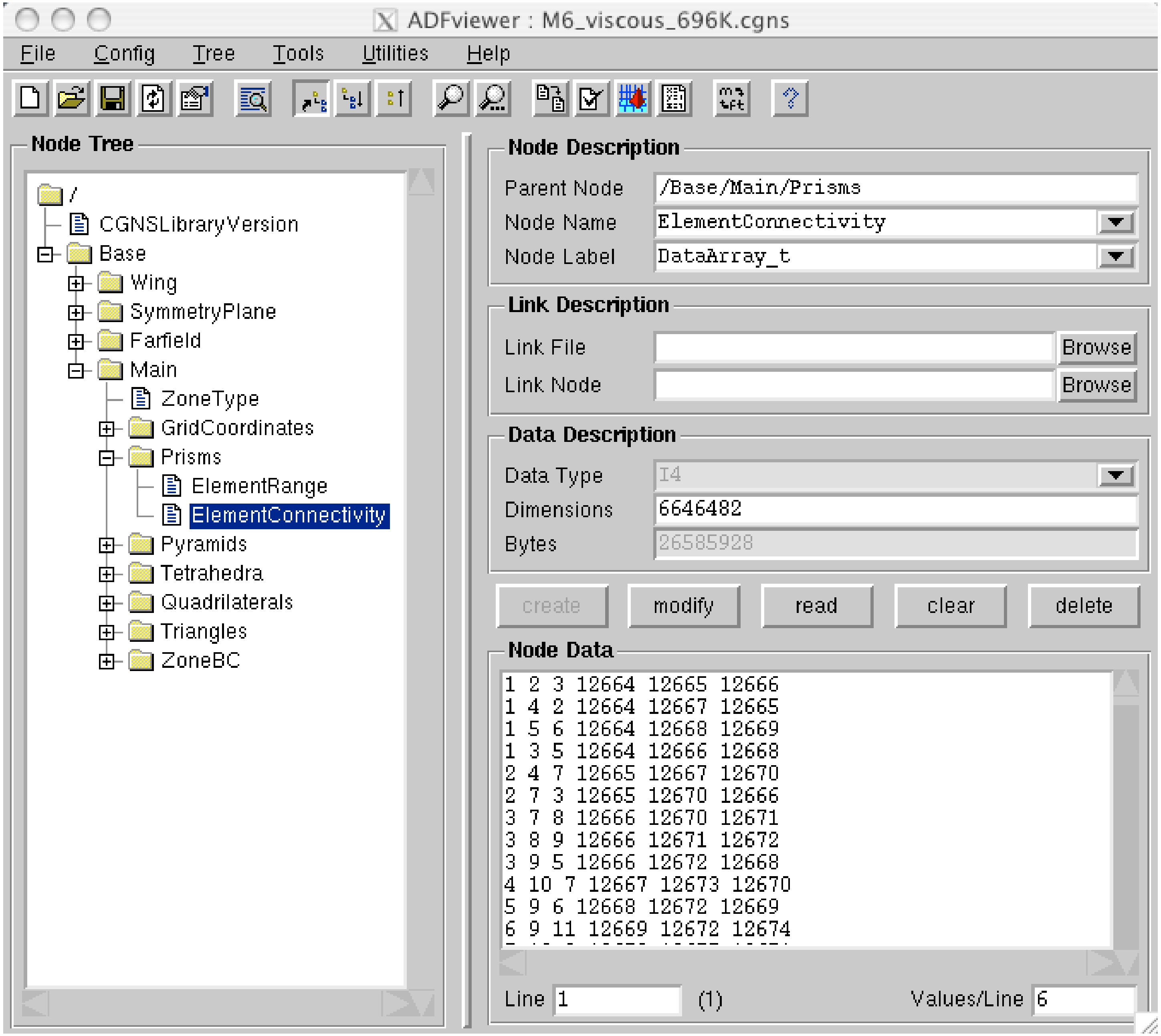
Line 1 (1) Values/Line 10

Example – CGNS Code (4)

```
/* Loop over the number of sections and read the element */
/* connectivities. As CGNS starts the numbering at 1 the */
/* for-loop starts at 1 as well.                         */

for(sec=1; sec<=nSections; sec++)
{
  /* Determine the element type and set the pointer for the */
  /* connectivity accordingly.                              */

  if(cg_section_read(fileInd, base, zone, sec, secName, &type,
                     &eBeg, &eEnd, &nBdry, &parentFlag) != CG_OK)
    Terminate(“readGridCGNS”, cg_get_error());

  switch (type)
  {
    case TETRA_4:
      conn = connTetra;  break;
    case PYRA_5:
      conn = connPyra;   break;
    case PENTA_6:
      conn = connPrisms; break;
    case HEXA_8:
      conn = connHexa;   break;
```

Example – CGNS Code (5)

```
     case TRI_3:
      conn = connTri;    break;
    case QUAD_4:
      conn = connQuad;   break;
    default:
      Terminate(“readGridCGNS”, “Unsupported element encountered.”);
      break;
  }

  /* Read the connectivity. Again, the node numbering of the   */
  /* connectivities start at 1. If internally a starting index */
  /* of 0 is used (typical for C-codes) 1 must be substracted  */
  /* from the connectivities read.                             */

  if(cg_elements_read(fileInd, base, zone, sec, conn, NULL) != CG_OK)
    Terminate(“readGridCGNS”, cg_get_error());
}
```



Node Tree

```
/
├ CGNSLibraryVersion
⊟ Base
   ⊞ Wing
   ⊞ SymmetryPlane
   ⊞ Farfield
   ⊟ Main
      ├ ZoneType
      ⊞ GridCoordinates
      ⊟ Prisms
         ├ ElementRange
         └ ElementConnectivity
      ⊞ Pyramids
      ⊞ Tetrahedra
      ⊞ Quadrilaterals
      ⊞ Triangles
      ⊞ ZoneBC
```

Node Description

Parent Node: /Base/Main/Prisms

Node Name: ElementConnectivity

Node Label: DataArray_t

Link Description

Link File:

Link Node:

Data Description

Data Type: I4

Dimensions: 6646482

Bytes: 26585928

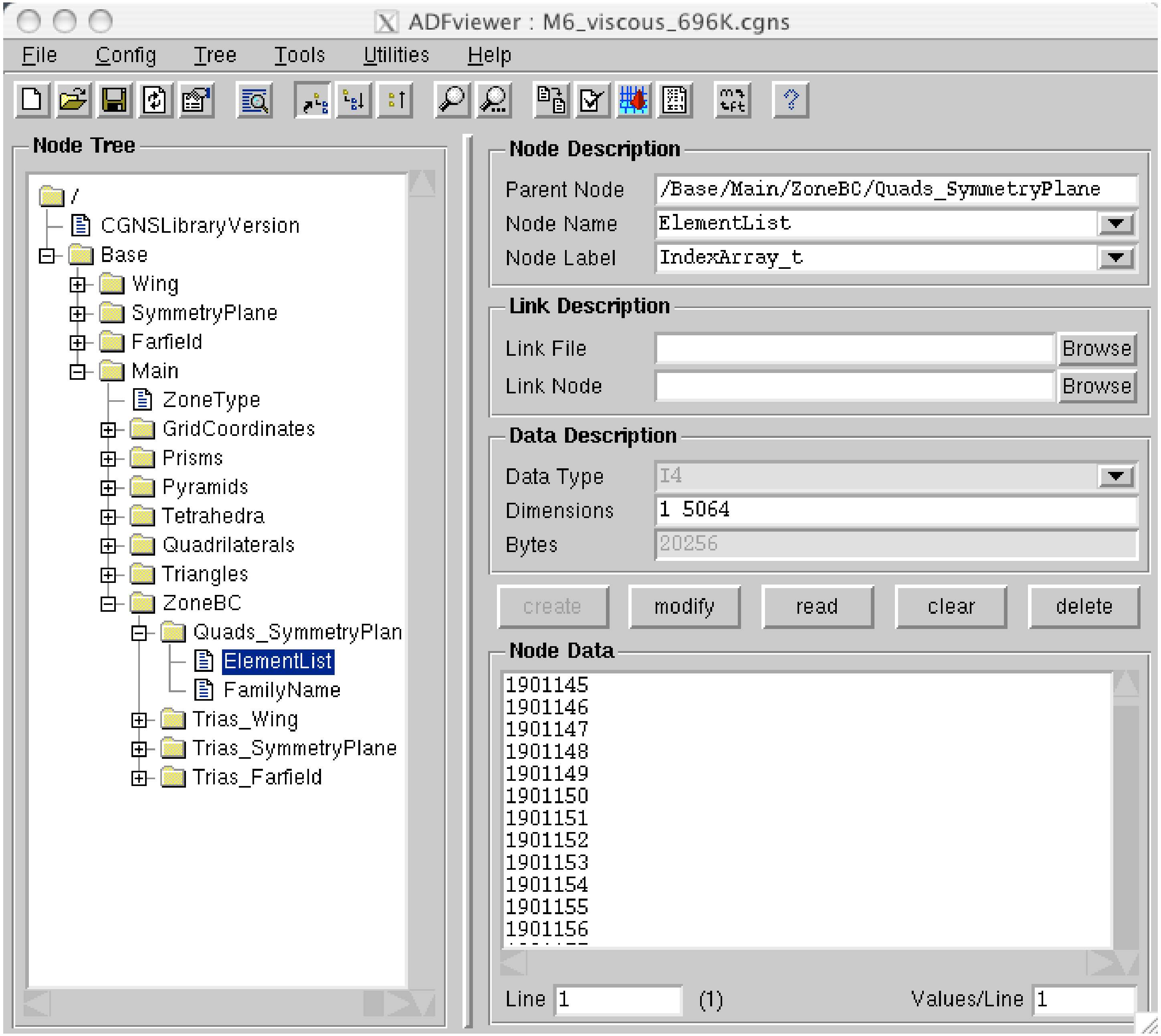
Node Data

```
1  2  3  12664  12665  12666
1  4  2  12664  12667  12665
1  5  6  12664  12668  12669
1  3  5  12664  12666  12668
2  4  7  12665  12667  12670
2  7  3  12665  12670  12666
3  7  8  12666  12670  12671
3  8  9  12666  12671  12672
3  9  5  12666  12672  12668
4  10  7  12667  12673  12670
5  9  6  12668  12672  12669
6  9  11  12669  12672  12674
```

Line 1 (1) Values/Line 6

Example – CGNS Code (6)

```
  /* Determine the number of boundary conditions for this zone. */

  if(cg_nbocos(fileInd, base, zone, &nBocos) != CG_OK)
    Terminate(“readGridCGNS”, cg_get_error());

  /* Loop over the number of boundary conditions. */

  for(boco=1; boco<=nBocos; boco++)
  {
    /* Read the info for this boundary condition. */

    if(cg_boco_info(fileInd, base, zone, boco, bocoName, &bocoType,
                    &ptsetType, &nBCElem, &normalIndex,
                    &normListFlag, &normDataType, &nDataSet) != CG_OK)
      Terminate(“readGridCGNS”, cg_get_error());

    /* Read the element ID’s. */

    if(cg_boco_read(fileInd, base, zone, boco, BCElemRead,
                    NULL) != CG_OK)
      Terminate(“readGridCGNS”, cg_get_error());

    /* And much more to make it fit into the */
    /* internal datastructures.              */
}
```



Node Tree

- /
  - CGNSLibraryVersion
  - Base
    - Wing
    - SymmetryPlane
    - Farfield
    - Main
      - ZoneType
      - GridCoordinates
      - Prisms
      - Pyramids
      - Tetrahedra
      - Quadrilaterals
      - Triangles
      - ZoneBC
        - Quads_SymmetryPlan
          - ElementList
          - FamilyName
        - Trias_Wing
        - Trias_SymmetryPlane
        - Trias_Farfield

Node Description

Parent Node	/Base/Main/ZoneBC/Quads_SymmetryPlane
Node Name	ElementList
Node Label	IndexArray_t

Link Description

Link File	
Link Node	

Data Description

Data Type	I4
Dimensions	1 5064
Bytes	20256

Node Data

```
1901145
1901146
1901147
1901148
1901149
1901150
1901151
1901152
1901153
1901154
1901155
1901156
```

Line 1 (1) Values/Line 1

Conclusions

- CGNS can store a wide variety of unstructured mesh types.
- Midlevel API offers many functions to read/write CGNS files, see http://www.grc.nasa.gov/WWW/cgns/midlevel/index.html
- Simple example to read a grid has been given.
- In a real code more API-functions will be used for checking the available data, etc.