REGIONS Proposal (version III)

As proposed by CGNS Steering Subcommittee:

Chris Rumsey, Marc Poinot, Bob Bush, Mark Fisher

Contact point:
Chris Rumsey, NASA Langley Research Center, Hampton, VA 23681
c.l.rumsey@nasa.gov

The need to be able to handle "regions" has been identified for some time. By "regions," we mean the ability to give flowfield or other information over a *subset* of the entire zone in a CGNS file. This subset may be over a portion of a boundary, or it may be over a portion of the volume field.

This amended proposal is a follow-on to an earlier one submitted by Alan Sayre (Element Regions), which was for unstructured grids only, and was not implemented. The current proposal attempts to cover both structured and unstructured grids.

The original version of this proposal (RegionsII) was patterned after the BC node structure, but after comments from users and internal discussions, the subcommittee has re-worked (and simplified) the proposal. Now, the new Region structure is patterned more after Flowsolution_t. In particular, Region_t now only has 1 layer, rather than 3.

The proposal is the following:

Under Zone_t, allow any number of the optional node:

```
Region_t< int IndexDimension, int CellDimension > :=
   {
   List( Descriptor_t Descriptor1 ... DescriptorN ) ;              (o)

   GridLocation_t GridLocation ;                                   (o/d)

   RegionCellDimension_t RegionCellDimension ;                     (o/d)

   Rind_t<IndexDimension> Rind;                                    (o/d)

   IndexRange_t<IndexDimension> PointRange ;                       (r:o:o:o:o)
   IndexArray_t<IndexDimension, ListLength, int> PointList ;       (o:r:o:o:o)
   IndexRange_t<IndexDimension> ElementRange ;                     (o:o:r:o:o)
   IndexArray_t<IndexDimension, ListLength, int> ElementList ;     (o:o:o:r:o)
   BCPointer_t BCPointer  ;                                        (o:o:o:o:r)
```

List(DataArray_t<DataType, ListLength> DataArray1 ... DataArrayN) ; (o)

FamilyName_t FamilyName ; (o)

DataClass_t DataClass ; (o)

DimensionalUnits_t DimensionalUnits ; (o)

List(UserDefinedData_t UserDefinedData1 ... UserDefinedDataN) ; (o)

Notes

1. Default names for the Descriptor_t, DataArray_t, and UserDefinedData_t lists are as shown; users may choose other legitimate names. Legitimate names must be unique within a given instance of Region_t and shall not include the names (... need to define).
2. The area over which the region is defined is specified by one of PointRange, PointList, ElementRange, ElementList, or BCPointer. Only one of these may be specified. The BCPointer is a node whose only entry is a character string giving the path to an existing BC_t node within the current CGNSBase_t. This method for defining the region range may be useful if the current range exactly coincides with an existing BC region definition. If BCPointer is used, then GridLocation under Region_t is ignored, and the GridLocation from the BC region is used.
3. PointRange and PointList refer to vertices, cell centers, or cell faces, depending on the value of GridLocation. GridLocation may be set to Vertex, CellCenter, IFaceCenter, JFaceCenter, KFaceCenter, or FaceCenter. If GridLocation is absent, then its default value is Vertex. When GridLocation is set to Vertex, then PointList or PointRange refer to node indices, for both structured and unstructured grids. When GridLocation is set to CellCenter, then PointList or PointRange refer to cell centers. Cell centers are indexed using different methods depending if the zone is structured or unstructured. For a structured zone, they are indexed using the minimum of the connecting vertex indices, as described in the section Structured Grid Notation and Indexing Conventions. For an unstructured zone, they are indexed using their element numbering, as defined in the Elements_t data structures. When GridLocation is set to FaceCenter, then PointList or PointRange refer to face elements. Face elements are indexed using different methods depending if the zone is structured or unstructured. For a structured zone, they are indexed using the minimum of the connecting vertex indices, as described in the section Structured Grid Notation and Indexing Conventions. For an unstructured zone, they are indexed using their element numbering, as defined in the Elements_t data structures.
4. ElementRange and ElementList always refer to cell elements, as described in the Elements_t data structures. When ElementRange or ElementList are used, then GridLocation should be set to be either CellCenter or FaceCenter.

5. FamilyName can identify the family to which the region belongs. Family names can link the region to the CAD surfaces. (See the section on Family Data Structure Definition for more details.)
6. The UserDefinedData_t data structure allows arbitrary user-defined data to be stored in Descriptor_t and DataArray_t children without the restrictions or implicit meanings imposed on these node types at other node locations.
7. Rind is an optional field that indicates the number of rind planes (for structured grids) or rind points (for unstructured grids). If Rind is absent, then the DataArray_t structure entities contain only core data of length ListLength, as defined for this region. If Rind is present, it will provide information on the number of rind elements, in addition to the ListLength, that are contained in the DataArray_t structures. The bottom line is that Rind simply adds a specified number to ListLength, as used by the DataArray_t structures. Care must be exercised when using Rind. For example, if the range is defined over only an interior portion of a face, then Rind does not make sense and should not be used.
8. The region's flow solution data is stored in the list of DataArray_t entities; each DataArray_t structure entity contains a single quantity. See, for example, the quantities described in Appendix A. The GridLocation specifies the location of the solution data with respect to the grid; if absent, the data is assumed to coincide with grid vertices (i.e., GridLocation = Vertex). All data within a given instance of Region_t must reside at the same grid location.
9. DataClass defines the default class for data contained in the DataArray_t entities. For dimensional flow solution data, DimensionalUnits may be used to describe the system of units employed. If present, these two entities take precedence over the corresponding entities at higher levels of the CGNS hierarchy, following the standard precedence rules.
10. There may be multiple Region_t nodes in a given zone. These may be associated with different times / different solutions in a time-dependent simulation (in which case ZoneIterativeData should be used to associate them), or these may simply be multiple regions defined for a single solution.
11. The RegionCellDimension_t node is included as an aid to defining what type of information is included in the Region_t node. If RegionCellDimension is not specified, it is assumed to be the same as CellDimension for the Zone. Only one cell dimension type should be used for a given Region_t. It is anticipated that one of the widest uses for Region_t will be to store specific boundary-only information. For example, in a 3-D simulation, one may wish to store data at walls. In this case, the RegionCellDimension would be set to 2. For a structured grid, then, the IndexRange must lie on a coordinate plane (either the i-, j-, or k-dimension must be constant). For an unstructured grid, the element types must all be 2-D types such as TRI_3's or QUAD_4's.

Furthermore, under ZoneIterativeData_t, the following optional DataArray_t node must be added:

DataArray_t<char, 2, [32, NumberOfSteps]> RegionPointers ; (o)

The ZoneIterativeData_t node is used to associate multiple FlowSolution_t and/or multiple Region_t nodes for time-dependent flow.

Example X.1:
For this example, it is assumed that a 1-zone 3-D structured grid exists of size (197x97x33). Inside of this zone, the user wishes to output a special subset region of interior data (say, temperature and kinematic viscosity) at the specific cell-center locations i = 121-149, j = 17-45, k = 21-23. Even though this same data may possibly exist under FlowSolution_t (which holds the flowfield data for the entire zone), this particular location may represent a special region of interest where the user wants to focus his attention, or perhaps where he wants to output different types of flowfield variables or UserDefined data.

Under Zone_t:

```
Region_t<3,3>  Region1 =
 {{
    GridLocation_t  GridLocation = CellCenter ;
    RegionCellDimension_t RegionCellDimension = 3;
    IndexRange_t<3>  PointRange =
     {{
        int[3] Begin = [121,17,21];
        int[3] End = [149,45,21];
     }};
       !  ListLength = (149-121+1)*(45-17+1)*(23-21+1) = 29*29*3 = 2523
     DataArray_t<real,1,2523>  Temperature =
     {{
       Data(real,1,2523) = temperature at the specific cell centers specified
      }} ;
     DataArray_t<real,1,2523>  ViscosityKinematic =
     {{
       Data(real,1,2523) = kinematic viscosity at the specific cell centers specified
      }} ;
 }} ;     ! end Region1
```

Example X.2:
This example is like the previous one, except it is for an unstructured zone. Inside of this zone, the user wishes to output a special subset region of data (say, temperature and kinematic viscosity) at a specific list of 2523 cell-center locations, located somewhere within the (larger) field of elements.

Under Zone_t:

```
Region_t<1,3>  Region1 =
 {{
```

```
GridLocation_t  GridLocation = CellCenter ;
RegionCellDimension_t RegionCellDimension = 3;
IndexArray_t<1,2523,int>  PointList =
 {{
    int[1] ElementList = list of 3-D element numbers where region data given
 }} ;
!  ListLength = length of the element list = 2523
 DataArray_t<real,1,2523>  Temperature =
 {{
   Data(real,1,2523) = temperature at the specific cell centers specified
 }} ;
 DataArray_t<real,1,2523>  ViscosityKinematic =
 {{
   Data(real,1,2523) = kinematic viscosity at the specific cell centers specified
 }} ;
}} ;     ! end Region1
```

Example X.3:
In this example, boundary data is output at the same locations where the BCs are specified in a particular BC_t node (in this case the ListLength is 25). Note that because this is data on a topologically-2-D boundary (in a 3-D simulation), RegionCellDimension is set to 2. GridLocation is not specified, because it is inherited from the BC_t node along with the ListLength.

Under Zone_t:

```
Region_t<1,3>  Region1 =
 {{
    RegionCellDimension_t RegionCellDimension = 2;
    BCPointer_t<>  BCPointer = {"Path to a ZoneBC/BC_t node"} ;
    !  ListLength = length of the element list from BC_t = 25
     DataArray_t<real,1,25>  Temperature =
     {{
       Data(real,1,25) = temperature at the specific BC locations specified
     }} ;
     DataArray_t<real,1,25>  ViscosityKinematic =
     {{
       Data(real,1,25) = kinematic viscosity at the specific cell centers specified
     }} ;
}} ;     ! end Region1
```