Module 2

The CGNS Standard Components

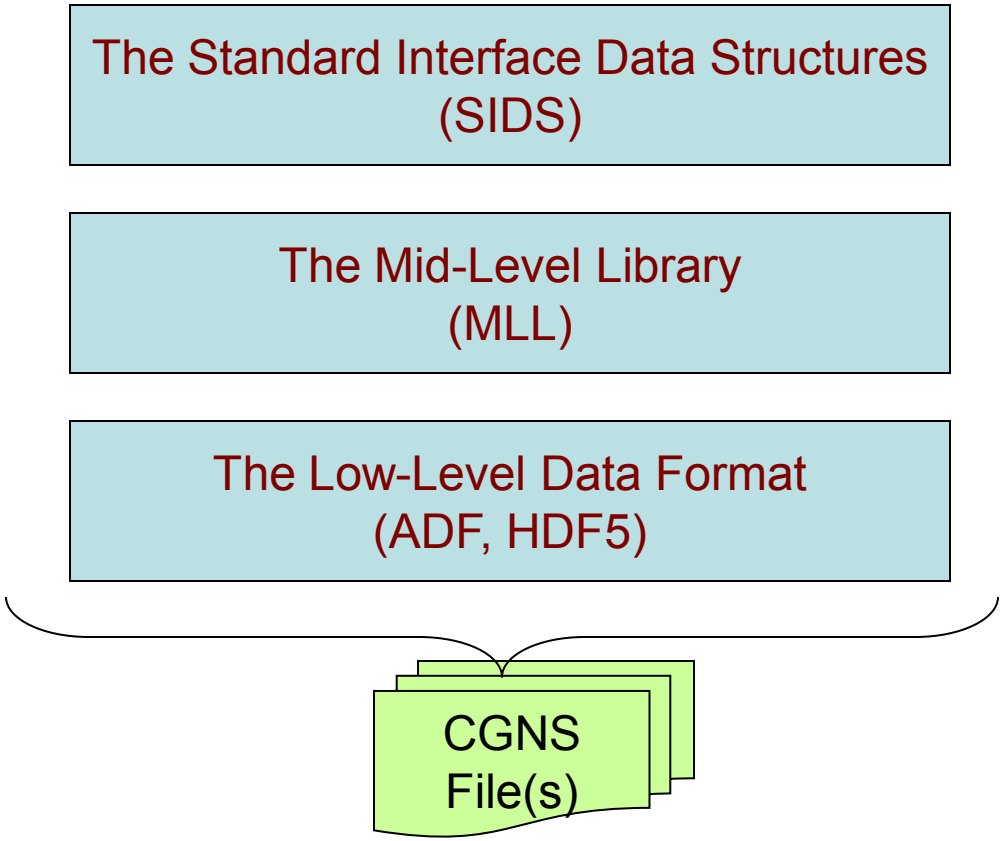
Dr. Ken Alabi
TTC Technologies Inc.
New York.

Outline

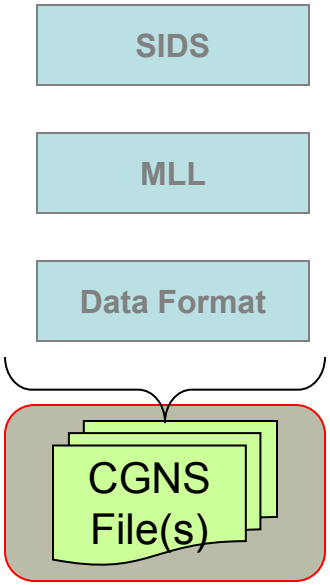
- Standard Interface Data Structures
- Mid-Level Library
- Introduction to the Low-Level Data Format
  - ADF
  - HDF5

The CGNS Standard Components

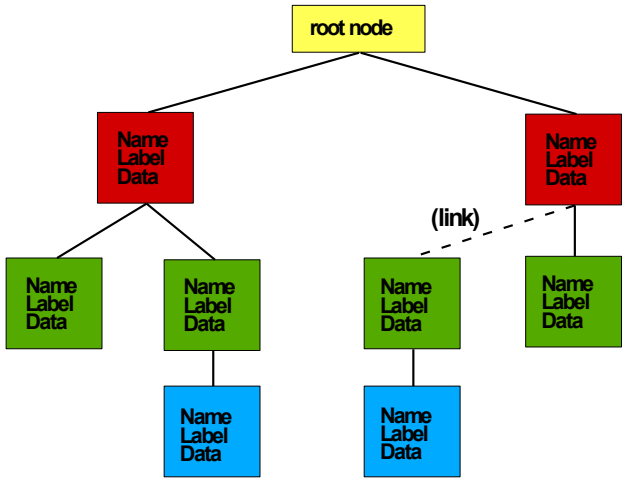
- The CGNS is comprised of both specification and software that implement the specification. The parts of CGNS include:



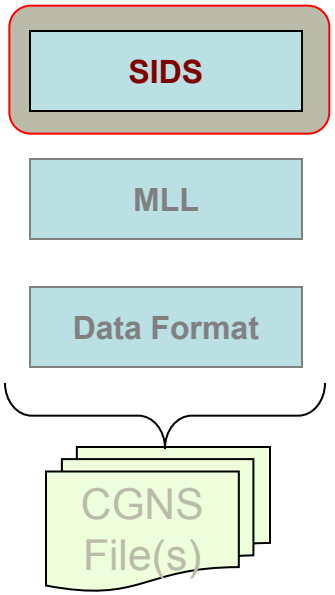
The CGNS File(s)



- The physical CGNS file is a compact C binary file arranged in a hierarchical database structure
- The file has the following properties:
  - Platform independent (portable across different systems)
  - Compact
  - Quickly traversed and sorted (due to the hierarchical structure)
  - Can be linked. A database may be composed of several files.

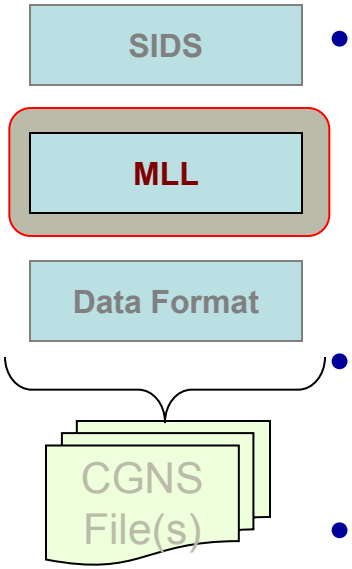


Standard Interface Data Structures (SIDS)



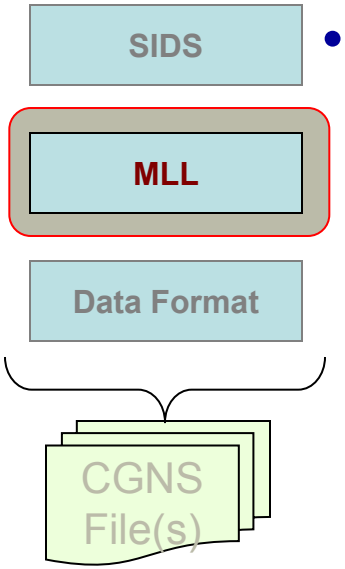
- Collection of conventions and definitions that defines the intellectual content of CFD-related data.
- Defines how data is written in the files in such a way that it can be uniformly interpreted independent of its source
- The specification is independent of the physical file format. Details of the actual format is handled in the low-level data format

The Mid-Level Library (MLL)



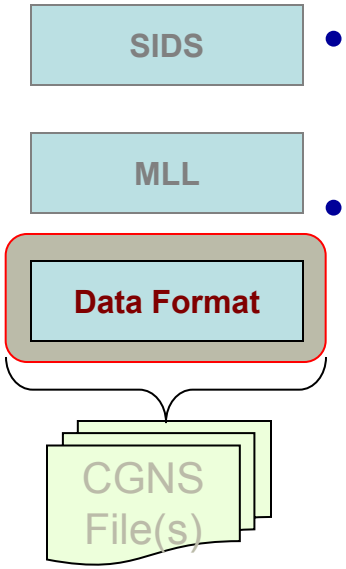
- The mid-level library consists of software application programming interface (API) developed to implement the CGNS specification (SIDS)
- Built on top of the low-level data format (does not perform any direct I/O operation)
- CGNS implementations can use the mid-level library and avoid the low level details of implementing the specifications themselves
- Using the MLL ensures uniformity of interpretation and implementation of the specs.

The Mid-Level Library (cont'd)



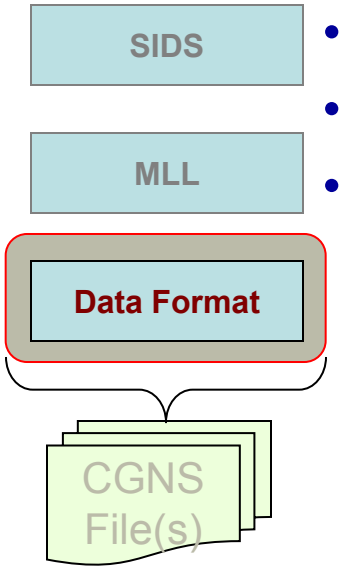
- The mid-level library is currently available for C and FORTRAN programs
  - C++ and Python extensions are also available

Introduction to the Low-Level Data Format



- The actual data format is implemented in ADF or HDF5
- Advanced Data Format (ADF)
  - Software that performs the I/O operations
  - Directed graph based on a single data structure (the ADF node)
  - Defines how data is organized in the storage media.
- HDF5

Introduction to the Low-Level Data Format



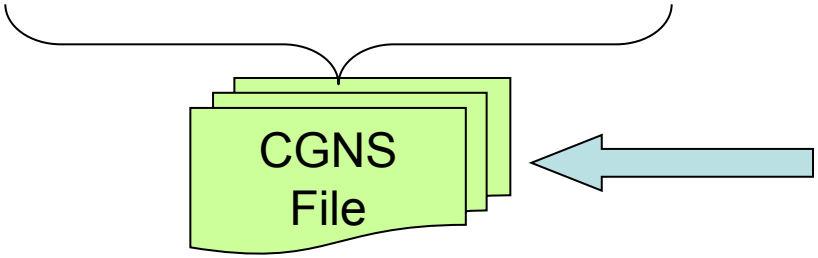
- The actual data format is implemented in ADF and HDF5
- Advanced Data Format (ADF)
- HDF5
  - Alternative data format to the ADF
  - Implementation in CGNS released in 2004
  - Advantages
    - Used in many applications
    - Parallel I/O using MPI
    - Faster access through linked files
  - Disadvantages
    - File sizes are 2 to 3 times larger
    - I/O times are generally 2 to 3 times slower, but may be up to an order of magnitude for a large number of nodes
- Both formats can currently be used in CGNS

CGNS Standard Components:
An Example

- A structured grid with one block:

$$x(i,j,k)$$
$$y(i,j,k)$$
$$z(i,j,k)$$
$$i = \,\rfloor,N\lceil, j = \,\rfloor,M\lceil, k = \,\rfloor,L\lceil$$

  - The SIDS specification
  - MLL Implementation of the spec.
  - Low Level Data Structure

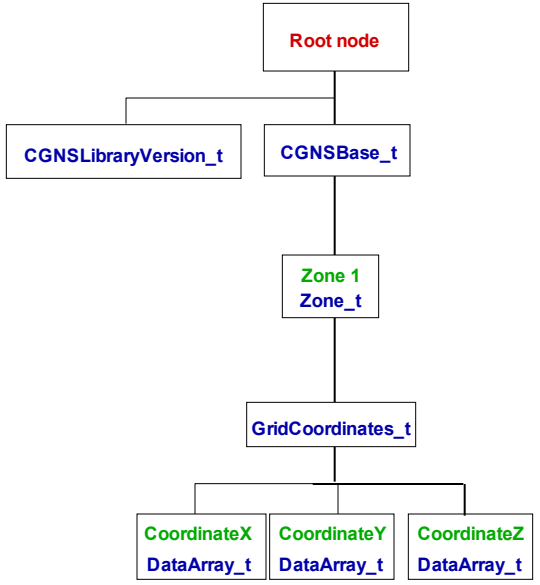


Resulting portable, compact,
CGNS file with fast data access

CGNS Standard Components: An Example (cont'd)

A structured grid with one block:

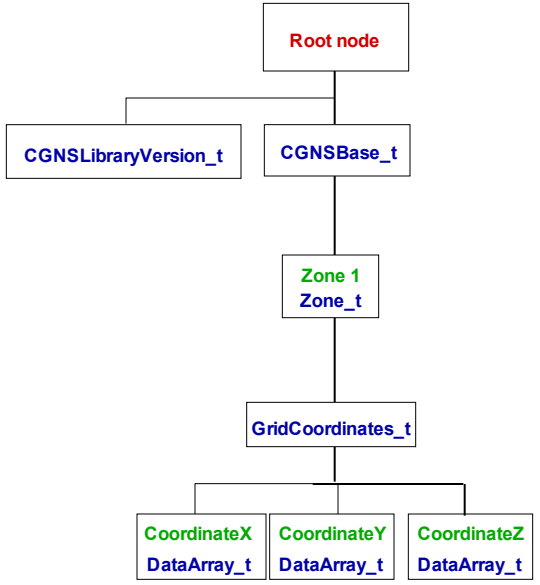
- The SIDS specification
  - The specification consists of descriptions (recommendations) in form of data structures
    - The physical coordinates of the grid vertices are described by the GridCoordinates_t structure.
    - The coordinates are stored as a data array in DataArray_t structure
    - The specification also describes the path to the physical coordinates as shown:
      - CGNS_Base_t
      - Zone_t
      - GridCoordinates_t
      - DataArray_t
    - The root node and CGNSLibraryVersion_t nodes are default nodes
    - The names of the nodes are also shown (in green above the structure type).



CGNS Standard Components: An Example (cont'd)

A structured grid with one block:

- The SIDS specification (cont'd)
  - The SIDS specifications also includes valid (universal or mandatory) names for some structures
    - For example the name for the zone is up to the user while the names for DataArray for the grid coordinates must be: **`CoordinateX`**, **`CoordinateY`**, and **`CoordinateZ`**, respectively

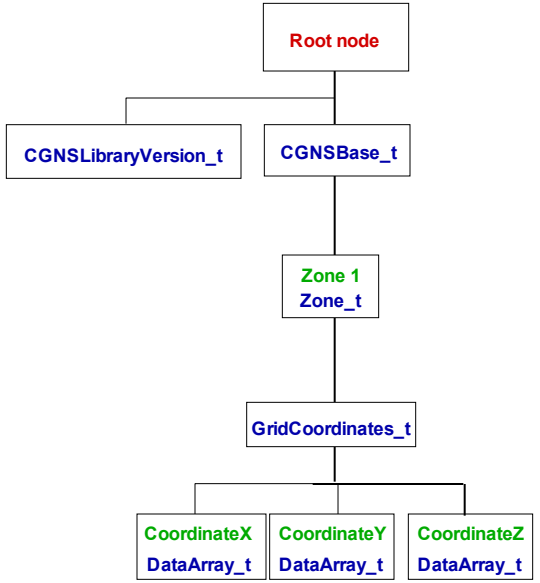


CGNS Standard Components: An Example (cont'd)

A structured grid with one block:

- The SIDS specification (cont'd)
  - Specification for GridCoordinates_t is shown below:

```
GridCoordinates_t< int IndexDimension, int VertexSize[IndexDimension] > :=
    {
    List( Descriptor_t Descriptor1 ... DescriptorN ) ;                  (o)

    Rind_t<IndexDimension> Rind ;                                       (o/d)

    List( DataArray_t<DataType, IndexDimension, DataSize[]>
          DataArray1 ... DataArrayN ) ;                                 (o)

    DataClass_t DataClass ;                                             (o)

    DimensionalUnits_t DimensionalUnits ;                               (o)

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



  - GridCoordinates_t requires two structure parameters: IndexDimension identifies the dimensionality of the grid-size arrays, and VertexSize is the number of vertices in each index direction excluding rind points (rinds are like ghost nodes or planes)
  - GridCoordinates_t may also be used for structured grids.

CGNS Standard Components: An Example (cont'd)

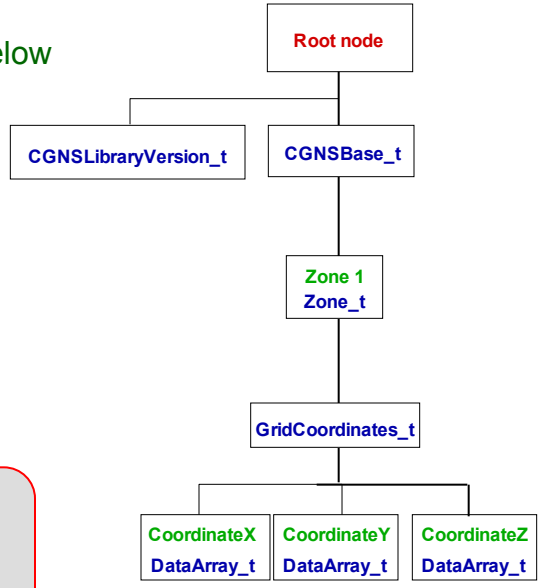
- A structured grid with one block:
  - MLL Implementation of the spec.
    - This requires actual MLL library software and programs calling the library to write the desired data in accordance with the SIDS specification.
    - The MLL API calls in FORTRAN for the current example are shown below

```
call cg_open_f(sfilename,MODE_WRITE,index_file,ier)

call cg_base_write_f(index_file,basename,icelldim, &
            & iphysdim, index_base,ier)

!Grid size
      isize(1,1)=N
      isize(2,1)=M
      isize(3,1)=L
!cell size
      isize(1,2)=isize(1,1)-1
      isize(2,2)=isize(2,1)-1
      isize(3,2)=isize(3,1)-1
!boundary vertex size (always zero for structured grids)
      isize(1,3)=0
      isize(2,3)=0
      isize(3,3)=0
call cg_zone_write_f(index_file,index_base,'Zone 1',isize, &
      & Structured,index_zone,ier)

call cg_coord_write_f(index_file,index_base,index_zone, &
     & RealDouble, 'CoordinateX',X,index_coord,ier)
!Write also for Y and Z...
```

Creates the CGNSBase_t node

Creates the Zone_t node

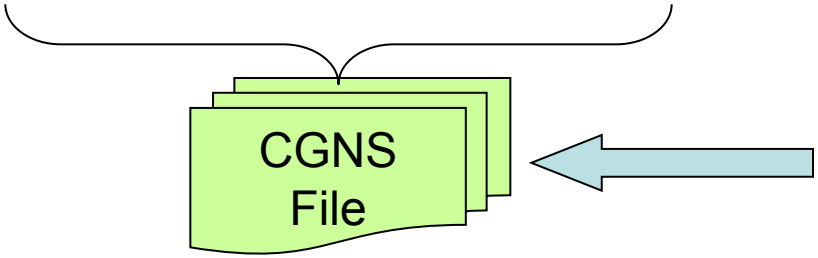
Creates the GridCoordinate_t and DataArray_t nodes



CGNS Standard Components: An Example (cont'd)

- A structured grid with one block:
  - Low Level Data Structure
    - For the current example no low level data calls need to be made
    - The MLL API calls made previously makes the calls the low level format (ADF or HDF5) to write the data
    - This is how most CGNS data is written

$$x(i,j,k)$$
$$y(i,j,k)$$
$$z(i,j,k)$$
$$i = \lfloor,N],j = \lfloor,M],k = \lfloor,L]$$



sfilename - Resulting portable, compact, CGNS file with fast data access

Summary

- The CGNS is comprised of both specification and software that implement the specification. The parts are as follows
  - The Standard Interface Data Structures (SIDS)
    - The CGNS specification is contained in the SIDS
    - This ensures that all software adhering to the specification will be able to understand and inter-exchange data
  - The Mid-Level Library
    - The mid-level library consists of software developed to implement the specification
    - CGNS implementations can use the mid-level library and avoid the low level details of implementing the specifications themselves
    - This also ensures uniformity of interpretation and implementation of the specs.
  - Introduction to the Low-Level Data Format
    - The actual data format is implemented in ADF and HDF5
    - ADF
    - HDF5