Armin Wulf

January 13 1999

ICEM CFD Engineering

Overview

- Introduction
- Brief History of the CGNS Project
- CGNS Conceptual Entity: The SIDS
- CGNS Physical Entities:
  - The ADF Core
  - The CGNS Library
- Examples of Implementation
- Current Development
- Future Development
- Conclusion

Introduction

- Objective: To offer the opportunity for seamless communication of CFD analysis data between sites, applications and system architectures.
- Motivation: The disparity of data file format reduces the cost effectiveness of CFD while impairing its development.
- Definition: The CFD General Notation System (CGNS) was conceived to provide a general, portable and extensible standard for the storage and retrieval of CFD analysis data.
- CGNS Elements:
  - Conceptual entity: Collection of conventions for the archiving of CFD data.
  - Physical entity: Software that performs I/O operations in accordance with the defined concepts.

CGNS

Brief History of the CGNS Project

- 1994-1995:
  - Series of meetings between Boeing and NASA addressing means of improving technology transfer from NASA to Industry: The main impediment to technology transfer is the disparity of file formats.
- 1995-1998:
  - Development of the CGNS System (SIDS, ADF) at Boeing Seattle, under NASA Contract with participation from:
    - Boeing Commercial Aircraft Group, Seattle
    - NASA Ames/Langley/Lewis Research Centers
    - Boeing St-Louis (former McDonnell Douglas Corporation)
    - Arnold Engineering Development Center, for the NPARC Alliance
    - Wright-Patterson Air Force Base
    - ICEM CFD Engineering Corporation
- 1997-1998:
  - Development of the CGNS Library.
  - Institution of the CGNS website (http://www.cgns.org) and first official release of the CGNS software and documentation.

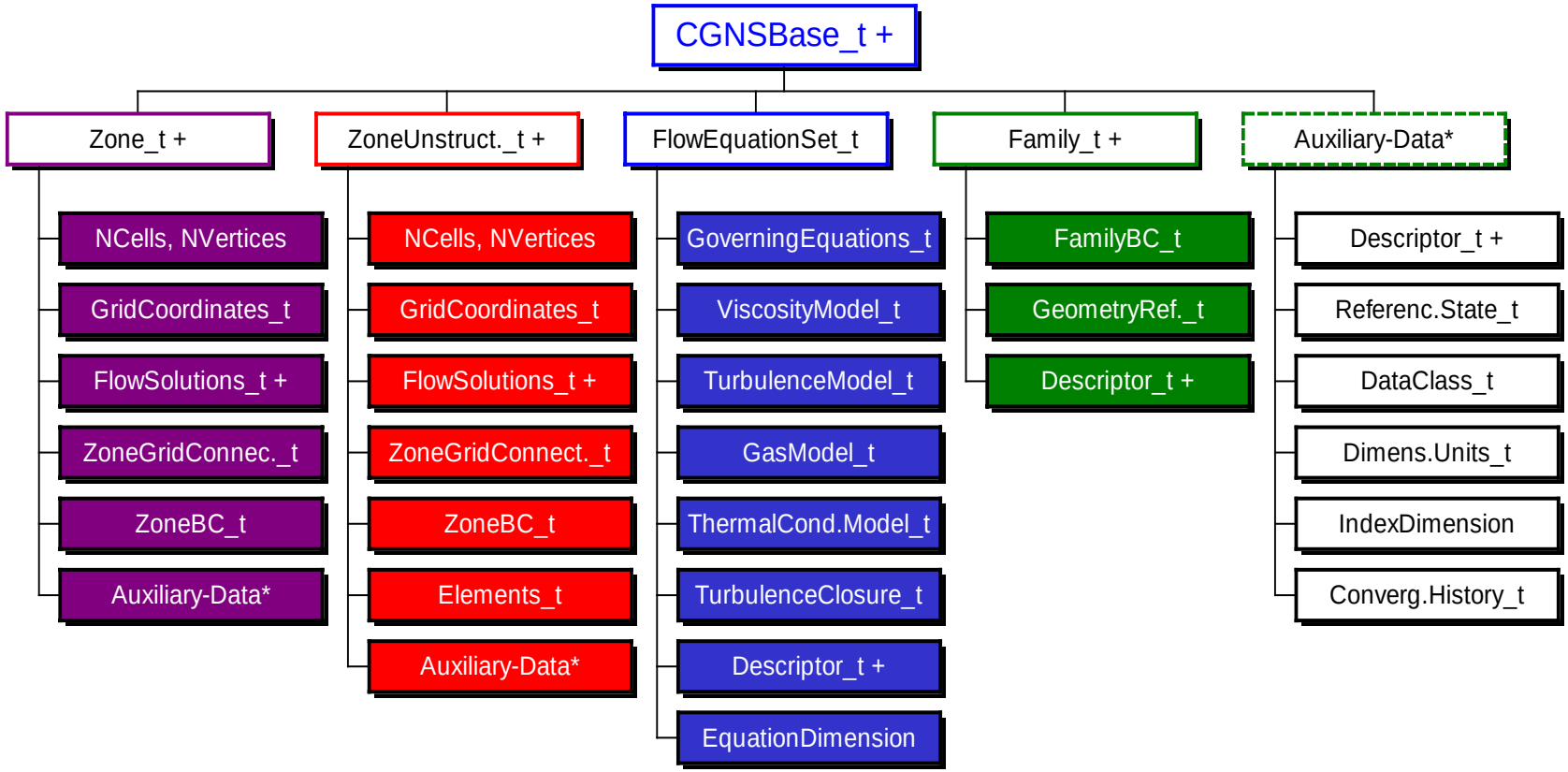
CGNS

CGNS Conceptual Entity : The Standard Interface Data Structures (SIDS)

- The SIDS constitutes the essence of CGNS. It defines:
  - The intellectual content of CFD-related data
  - The organizational data structure
  - The naming conventions

- Principal Characteristics of the SIDS:
  - Hierarchical data structure
  - Highly descriptive way of recording the data
  - Ability to include unlimited documentation
  - Complete and explicit problem description
  - Layered so that much of the data structures are optional

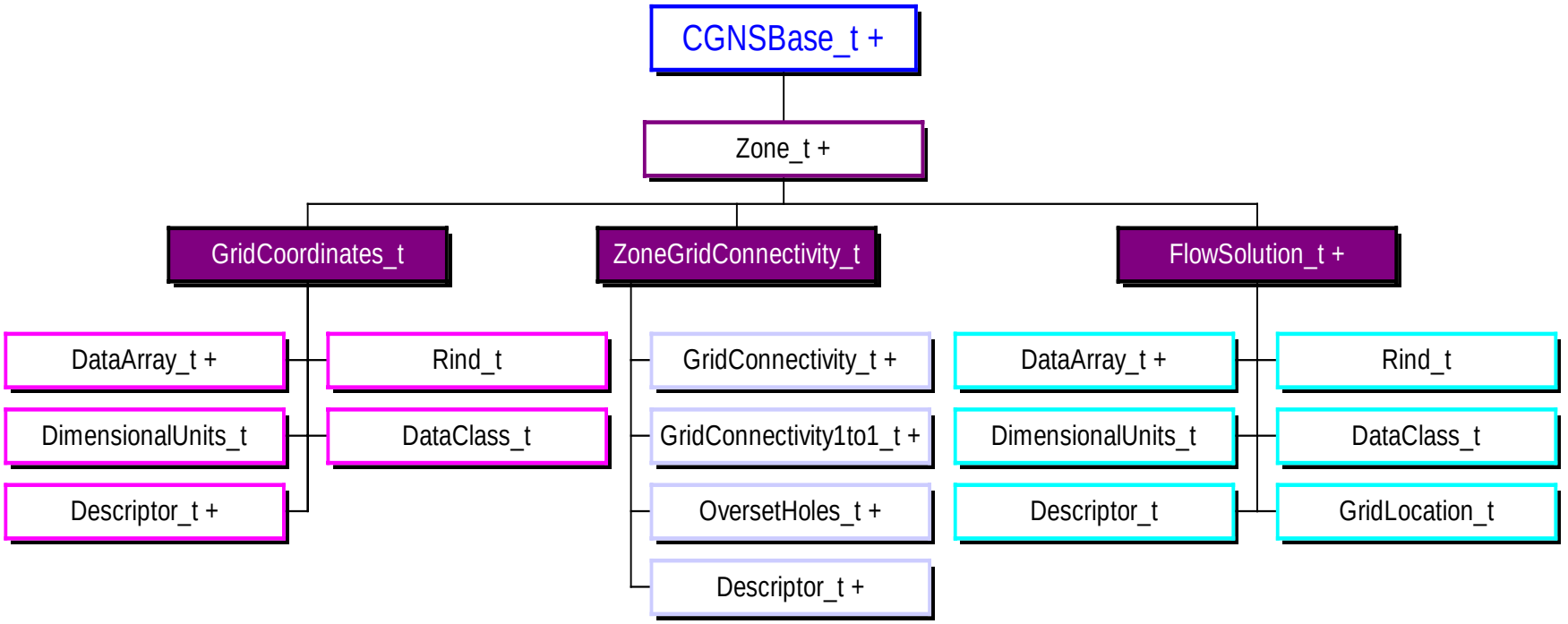
CGNS High Levels Chart



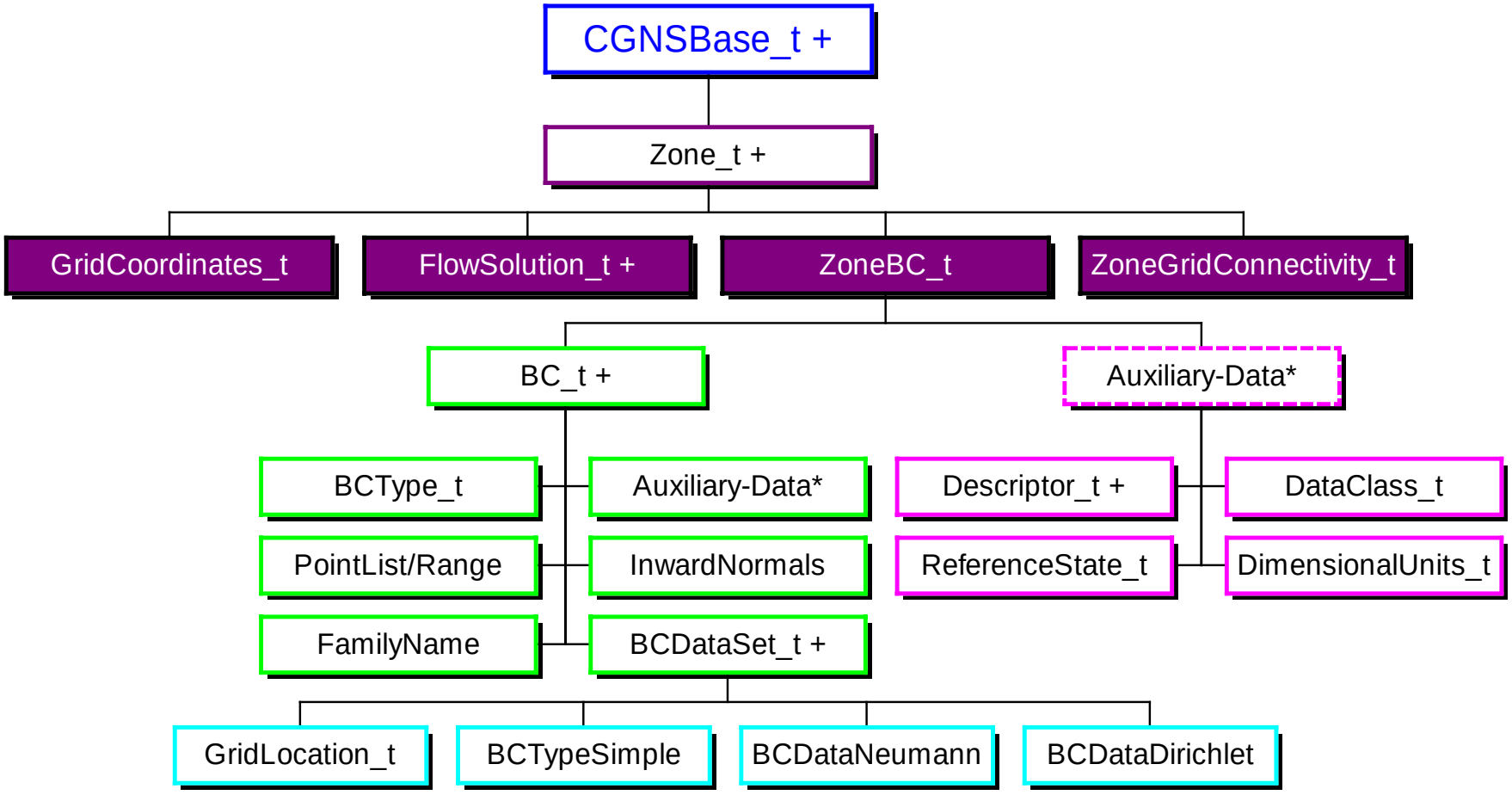
Note: Auxiliary-Data is used here to group the data structures containing auxiliary data. It is not a data structure of CGNS.

CGNS

Grid Coordinates, Flow Solutions and Zone Connectivity Data Structures



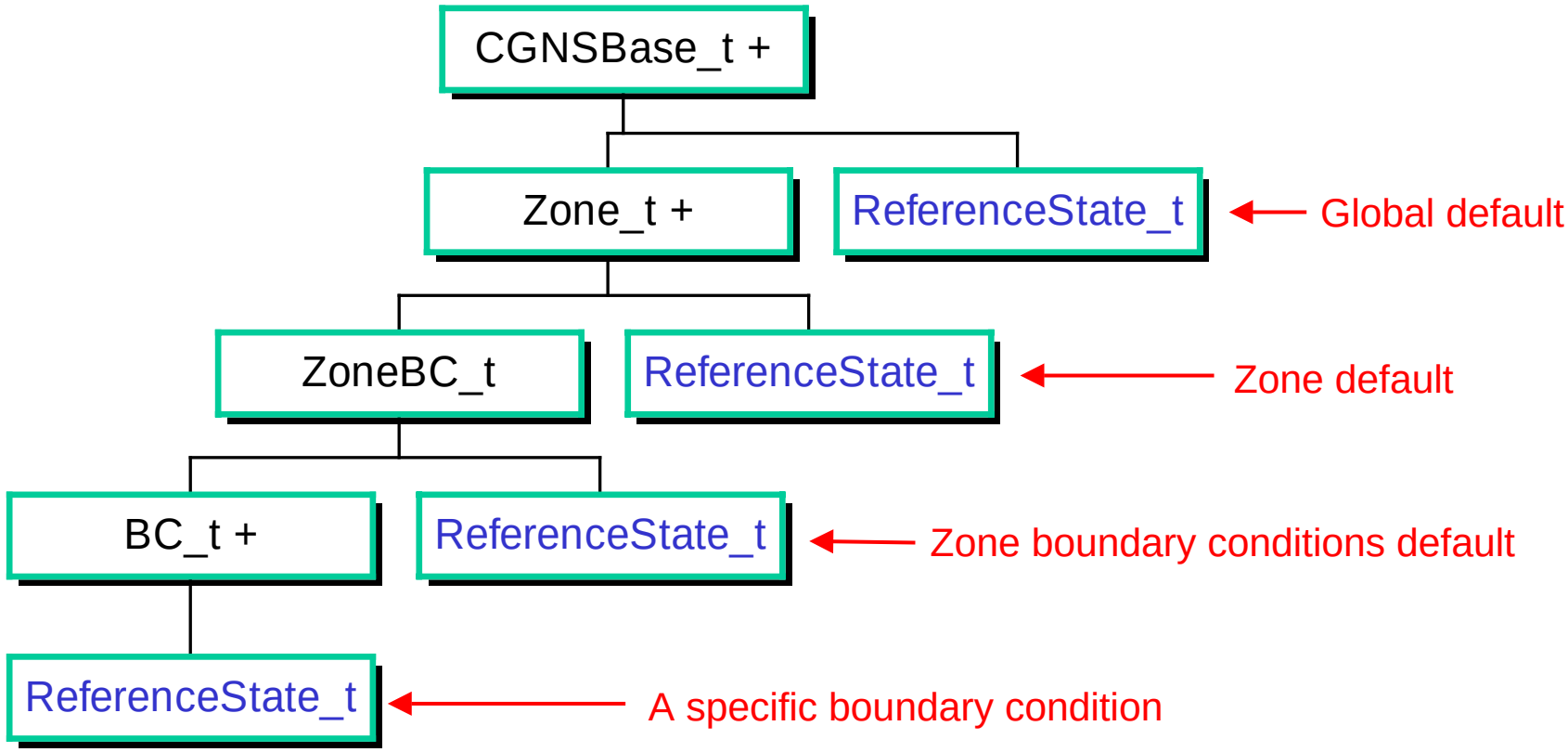
Zone Boundary Conditions Data Structure



Note: Auxiliary-Data is used here to group the data structures containing auxiliary data. It is not a data structure of CGNS.

CGNS

Globally Applicable Data and Precedence



CGNS Physical Entities

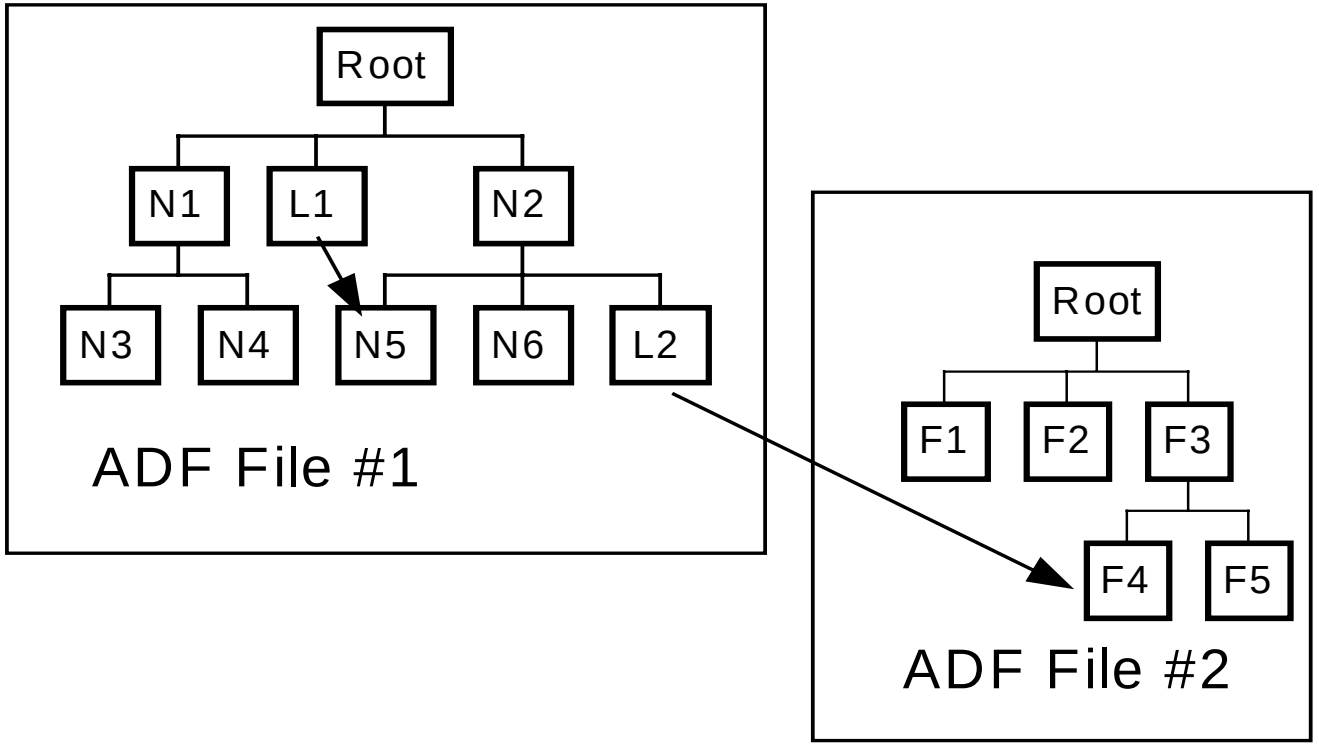
- The (ADF) Core:
  - The Advanced Data Format (ADF) Core is a set of software routines performing I/O operations to/from an extremely general database system consisting of ADF files.
- The CGNS Library:
  - The CGNS Library is an Application Programming Interface facilitating higher level access of the data contained in an ADF database. The CGNS Library is built on top of the ADF Core and does not perform any direct I/O operation.

ADF Core Characteristics

- Hierarchical data structure
  - Quickly traversed and sorted
  - No need to process irrelevant data
- Based on a single data structure called an ADF node
- Directed graph
- May encompass several ADF files through the use of links
- ADF database is self describing
- Supports any type of data (integer, real, character, complex, byte, link …)
- Software written in ANSI C to insure its portability
- Complete Fortran and C interface
- Stored in compact C binary format
- Independent of system architectures (Cray/Unicos, Sun/Solaris, SGI/IRIX, IBM/AIX, DEC-Alpha/OSF, Intel=Paragon)
- Universal database (not specific to CFD only)

CGNS

ADF Core


Root
N1
L1
N2
N3
N4
N5
N6
L2
ADF File #1
Root
F1
F2
F3
F4
F5
ADF File #2

ADF Node Content

- **ID**: A unique identifier to access a node within a file.
- **Name**: A character field used to name the node. It must be unique for a given parent.
- **Label**: A character field used to described the type of information contained in the node.
- **Data type**: A character field specifying the type of data (e.g. real, complex) associated with the node.
- **Number of dimensions**: The dimensionality of the data.
- **Dimensions**: An integer vector containing the number of elements within each dimension.
- **Data**: The data associated with the node.

- **Number of sub-nodes**: The number of children directly attached to a node.
- **Name of sub-nodes**: The list of children names.

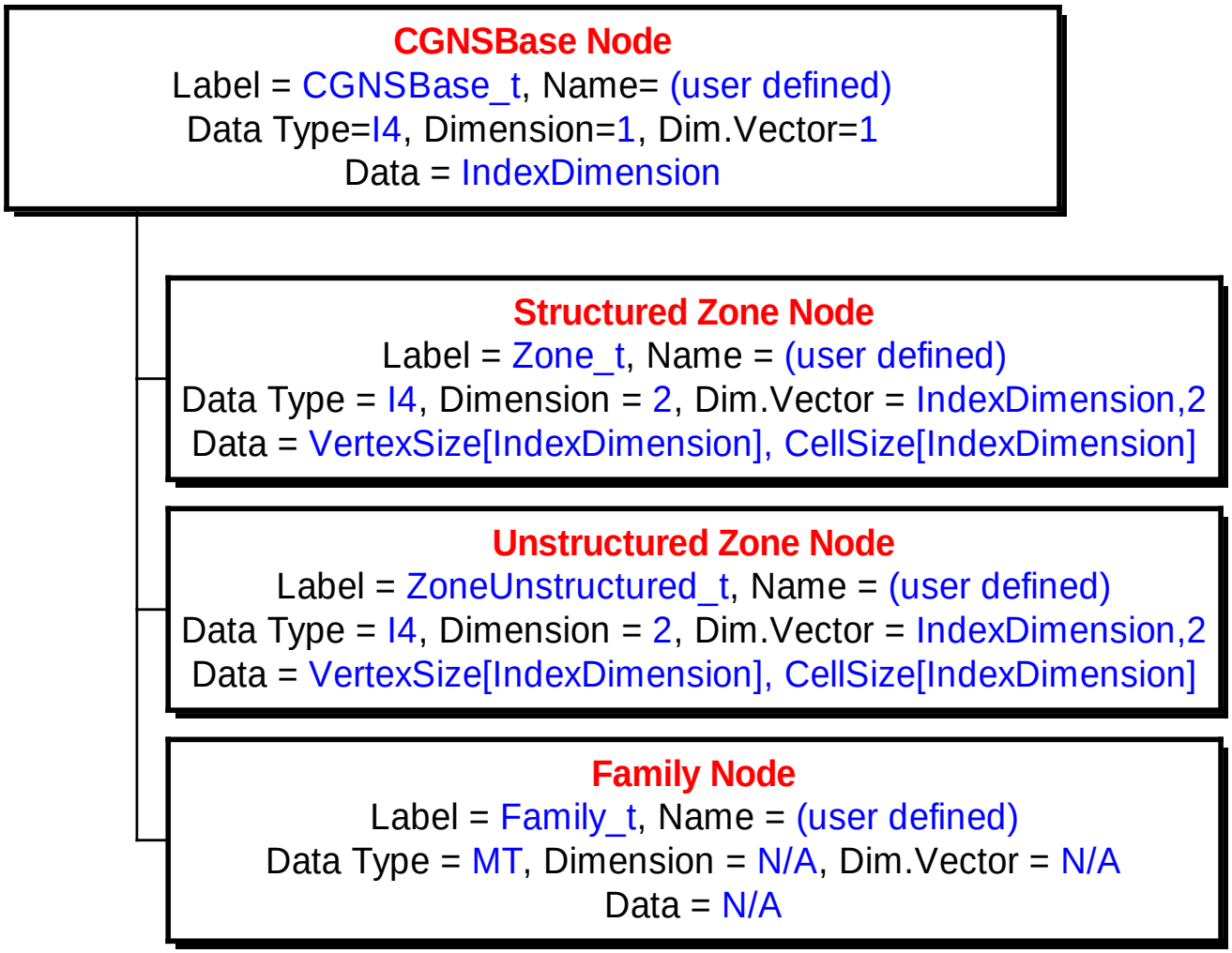
CGNS

Functions of the ADF Core

- The ADF Core is composed of 34 low level functions performing the following operations:
  - open or close ADF file
  - read or set the data binary format
  - get the root-id or a node-id
  - create, delete or move a node
  - create, read or test a node link
  - get the children of a node
  - read or write the constituents of a node: name, label, data type, dimension, dimension vector and data
  - perform version and error control

CGNS

SIDS-to-ADF Mapping of Upper Levels



The Library Reflects Precisely the SIDS and Mapping

1) SIDS

```
CGNSBase_t : = {
   int IndexDimension;
   List (Zone_t<IndexDimension>Zone1,…ZoneN);
   …}

   Zone_t<IndexDimension> : = {
      int[IndexDimension] VertexSize, CellSize;
      …}
```

2) Mapping SIDS to ADF

Name=BaseName
Label=CGNSBase_t
Data type=I4, Dimension=1
Data = IndexDimension

Name=ZoneName
Label=Zone_t
Data type=I4, Dimensions=IndexDim x 2
Data=ZoneSize[]=VertexSize[], Cell Size[]

3) Mid-Level Library Functions

```
int cg_nbases(int FileNo, int *nbases);

int cg_base_read(int FileNo, int BaseNo,
                 char *BaseName, int *IndexDimension);

int cg_nzones(int FileNo, int BaseNo, int *nzones);

int cg_zone_read(int FileNo, int BaseNo, int ZoneNo,
                 char *ZoneName, int *ZoneSize);
```

CGNS

CGNS Library Design Philosophy

- Communication Storage Media <=> Internal Database

```
int cg_open(char *filename, int mode, int *fn);
int cg_close(int fn);
```

- Internal Representation

  C-Structures Organized Hierarchically (SIDS)

- Communication Internal Database <=> User-Application

  46 Functions

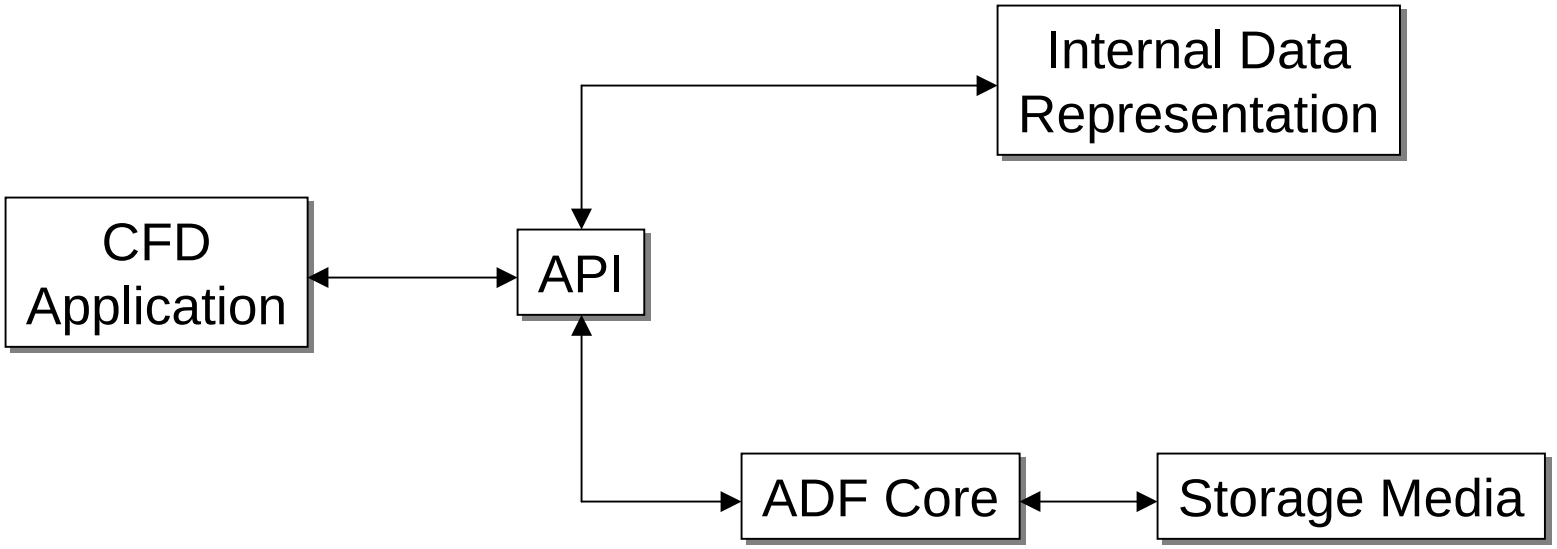
- Exception: Large Arrays

  Keep ADF-ID in Internal Database Instead of Entire Array

  - Reduce memory usage
  - Improve execution speed

CGNS

Data Flow


Internal Data Representation
CFD Application
API
ADF Core
Storage Media

CGNS Library: General Remarks

- C & Fortran functions

  C: **`cg_name`**

  Fortran: **`cg_name_f`**

- Error Status = **`ier.`** If **`ier!=0`**, error detected.

  C: return value

  Fortran: extra argument

- Platform Independent

  CRAY, SGI-IRIX, DEC-Alpha, SunOS, HP, IBM

CGNS

Examples of Implementation:

1. Read bases, zones & coordinates

```
cg_open(filename, MODE_READ, &fn);

cg_nbases(fn, &nbases);

for (B=1; B<=nbases; B++) {

  cg_base_read(fn, B, BaseName, &IndexDim);

  cg_nzones(fn, B, &nzones);

  for (Z=1; Z<=nzones; Z++) {

    cg_zone_read(fn, B, Z, ZoneName, ZoneSize)

    cg_coord_read(fn,B,Z,"CoordinateX",RealSingle,
                  RangeMin, RangeMax, X);
  }
}
cg_close(fn);
```

CGNS

Examples of Implementation:

2. Read Solutions

Given **fn, B, Z**:

```
cg_nsols(fn, B, Z, nsolutions);

for (S=1, S<=nsolutions, S++) {

  cg_sol_info(fn, B, Z, S, SolutionName, GridLocation);

  cg_nfields(fn, B, Z, S, nfields);

  for (F=1; F<=nfields; F++) {
    cg_field_info(fn, B, Z, S, F, DataType, FieldName);

    cg_field_read(fn, B, Z, S, FieldName, RealDouble,
                  RangeMin, RangeMax, DataArray);
  }
}
```

CGNS

Status

- Completed (Version 1.0)
  - Specification & software for ADF
  - SIDS for multi-block structured
  - API for multi-block structured
  - ADF-Edit file browser
  - Website at www.cgns.org: centralizes distribution of software and information, and tracks users (54 as of October 18 ‘98)
  - CGNS-online exchange to/from CGNS-Support@CGNS.org for online help, gathering of suggestions, bug reports, etc.
- In progress (Version 1.1)
  - SIDS for unstructured and hybrid meshes
  - SIDS for geometry links
  - API for unstructured and geometry data

CGNS

Current Development

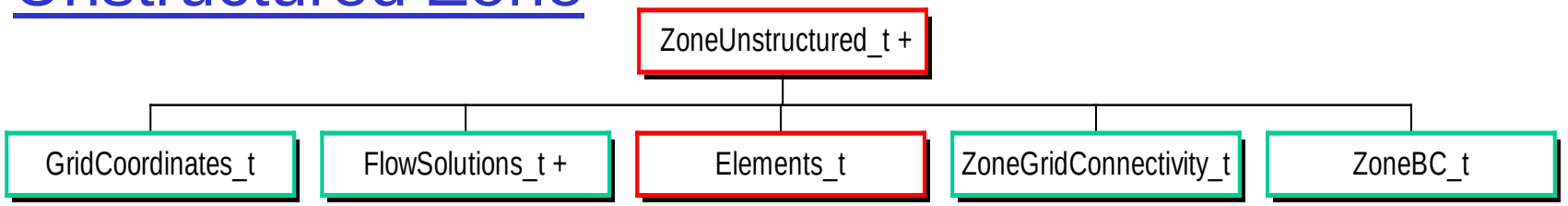
1. Extension to Unstructured Meshes

   – Use existing CGNS hierarchy as much as possible. Incorporate changes only where needed to support unstructured grids.

   – Define new unstructured-grid zones
     - same as structured-grid zones except for connectivity information

   – Define unstructured-grid connectivity information
     - element based connectivity
     - arranged by element type

Unstructured Zone



Unstructured Zone:

- Vertex size: number of nodes
- Cell size: number of elements of highest dimension

- Element Sections:
  - Section name: unique for a zone
  - Element type: Hexa_8, Tetra_4, Penta_6, Quad_4, Tri_3, etc…

    `NPE = NodePerElement(ElementType)`
  - Start and end index: element number of first and last elements

    `nelem = end - start + 1`
  - Elements array:

    `node(1,1), node(1,2) … node(1,NPE)`

    `node(nelem,1), node(nelem,2), … node(nelem,NPE)`

CGNS

Current Development

2. Add Geometry-to-Mesh Association

   – Necessary for:
     - Quick response to design changes
     - Mesh adaptation
     - Analysis and Display of Results

   – Objectives:
     - Compatible to the CGNS System
     - Minimum Changes to the Existing SIDS
     - Reference data in CAD Files
     - Associate CAD Data to Mesh, Solution Data and Boundary Conditions

Concept: Layer of Indirection

- Rarely a 1-to-1 connection between mesh regions & geometric entities.
- Association independent of changes to mesh & geometry.
- Boundary conditions & material properties can be defined on families:
  - Independent of mesh and geometry.
  - Defined only once.

CGNS

Future Actions

1. Build a User Base

- Advertise CGNS more
  - Give presentations at different organizations
  - Send a newsletter or leaflet to CFD groups
  - Advertise in Aerospace America
  - Submit to web search engines
  - Get e-mail lists from various government and industrial organizations,and send CGNS information
- Help implementing CGNS in key solvers, at NASA, Boeing, Gov't Labs…
  - Work together with solver developers to implement CGNS in their code
  - Provide training
  - Make available examples of applications on the web

Future Actions continued

2. Future Development Issues

- Extend supported data
  - Chemistry, Electromagnetic (Maxwell)
  - Experimental data e.g. pressure paint, flight test recording…
  - Add boundary conditions: Special topologies (C-grid), periodic…
  - Specialized data structures for Cartesian Grid support

- Give access to geometry data
  - Link to CAPRI
  - Link to geometry evaluator & repair kit

- Unstructured and hybrid mesh tools
  - Provide alternate connectivity representations, via quick sorting routines
  - Cell, face, edge, vertices neighboring information
  - Compression techniques

CGNS

Future Actions continued

2. Future Development Issues continued

- Data tools
  - Data set interrogator: inventory of cell types, assess compatibility with specific codes, etc.
  - Data consistency check: element quality, b.c., coupling,…
  - Graphical User Interface to view the file content (similar to Windows-Explorer)
- Computer related issues:
  - Run time data management (as in PV3)
  - Parallel handling (MPI or shared memory)
- ISO Standard
  - Documentation and code development to meet further standardization requirements

CGNS

Conclusion

- defines conventions for an extremely complete problem description
- hierarchical data structure optimizes the performance of the data exchange
- machine independent compact binary data files
- most data structures are supported by the CGNS library
- version 1.0 available at www.cgns.org.

- implemented successfully in CFL3D, OVERFLOW, ICEM-CFD Visual3, PEGASUS, ICEM-CFD Output-Interfaces, as well as translators for Plot3D, NPARC, TLNS3D, WIND

- version 1.1 planned for end of December 1998 includes:
  - CGNS Library coverage of the entire SIDS
  - Geometry-to-Mesh Association and Unstructured Mesh

- supports seamless communication of analysis data between user sites system architectures, and CFD applications.
- should lead to the development of shared, reusable software selected on technical merit without concern for I/O compatibility.

CGNS