CFD General Notation System
Overview and Entry-Level Document

Document Version 3.1.1

Contents

Foreward

This document describes the structure of CGNS (CFD General Notation System), its software, and its documentation. The Overview, which was written primarily for new and prospective users of CGNS, (1) introduces terminology, (2) identifies the elements of the system and their relationships, and (3) describes the various documents that elaborate the details. Reading the material on the purpose (Section 1) and general description (Section 2) of CGNS should help users determine whether CGNS will meet their needs. Those wanting a bit more detail should read Section 3, which describes the various elements of CGNS. For those interested only in understanding the scope and capabilities of CGNS, or for the end user unconcerned with the internal workings of the system, the Overview may prove sufficient documentation by itself.

The Overview also includes certain information that is current as of the document date but which may change with time. All information on CGNS compatible "applications" software (i.e., external programs such as grid generators, flow codes, or postprocessors in Section 4) is of this type. Also subject to change is the information on the acquisition of the software and documentation in Section 5, and the current status of CGNS in Section 6.

1 Purpose and Scope

The general purpose of CGNS is to provide a standard for recording and recovering computer data associated with the numerical solution of the equations of fluid dynamics.

CGNS consists of a collection of conventions, and software implementing those conventions, for the storage and retrieval of CFD (Computational Fluid Dynamics) data. The system consists of two parts: (1) a standard format for recording the data, and (2) software that reads, writes, and modifies data in that format. The format is a conceptual entity established by the documentation, and is intended to be general, portable, expandable, and durable. The software is a physical product supplied to enable developers to access and produce data recorded in that format. All CGNS software is completely free and open to anyone.

The CGNS standard, applied through the use of the supplied software, is intended to:

- facilitate the exchange of CFD data
  - between sites
  - between applications codes
  - across computing platforms
- stabilize the archiving of CFD data

The principal target of CGNS is data normally associated with compressible viscous flow (i.e., the Navier-Stokes equations), but the standard is also applicable to subclasses such as Euler and potential flows. The CGNS standard addresses the following types of data.

- Structured, unstructured, and hybrid grids
- Flow solution data, which may be nodal, cell-centered, face-centered, or edge-centered
- Multizone interface connectivity, both abutting and overset
- Boundary conditions
- Flow equation descriptions, including the equation of state, viscosity and thermal conductivity models, turbulence models, and multi-species chemistry models
- Time-dependent flow, including moving and deforming grids
- Dimensional units and nondimensionalization information
- Reference states
- Convergence history
- Association to CAD geometry definitions

Much of the standard and the software is applicable to computational field physics in general. Disciplines other than fluid dynamics would need to augment the data definitions and storage conventions, but the fundamental database software, which provides platform independence, is not specific to fluid dynamics.

2 General Description

A CGNS database describes the current state of one or more entire CFD (Computational Fluid Dynamics) problems, including the following:

- grid
- flowfield
- boundary conditions
- topological connection information
- auxiliary data (e.g., nondimensionalization parameters, reference states)

Not all of these data need to be present at any particular time. The overall view is that of a shared database that can be accessed by the various software tools common to CFD, such as solvers, grid generators, field visualizers, and postprocessors. Each of these "applications" serves as an editor of the data, adding to, modifying, or interpreting it according to that application's specific role.

CGNS conventions and software provide for the recording of a complete and flexible problem description. The exact meaning of a subsonic inflow boundary condition, for example, can be described in complete detail if desired. User comments can be included nearly anywhere, affording the opportunity, for instance, for date stamping or history information to be included. Dimension and sizing information is carefully defined. Any number of flow variables may be recorded, with or without standard names, and it is also possible to add user-defined or site-specific data. These features afford the opportunity for applications to perform extensive error checking if desired.

Because of this generality, CGNS provides for the recording of much more descriptive information than current applications normally use. However, the provisions for this data are layered so that much of it is optional. It should be practical to convert most current applications to CGNS with little or no conceptual change, retaining the option to take advantage of more detailed descriptions as that becomes desirable.

CGNS specifications currently cover the bulk of CFD data that one might wish to exchange among sites or applications; for instance, nearly any type of field data can be recorded, and, based on its name, found and understood by any code that needs it. Global data (e.g., freestream Mach number, Reynolds number, angle of attack) and physical modeling instructions (e.g., thin layer assumptions, turbulence model) may be specified. Nevertheless, there are items specific to individual applications for which there is currently no specification within CGNS. Most commonly, these are operational instructions, such as number of sweeps, solution method, multigrid directives, and so on. Owing to the miscellaneous nature of this data, there has been no attempt to codify it within a global standard. It is therefore expected that many applications will continue to require small user-generated input files, presumably in ASCII format.

CGNS itself does not initiate action or undertake any function normally handled by the operating system. The user still performs CFD tasks according to existing processes. This includes selecting the computing platform, maintaining the files, and launching the applications.

However, the ease of communication between applications that CGNS provides should motivate the development of new batch and interactive mechanisms for the convenient application of CFD tools.

3 Elements and Documentation

3.1 Introduction

CGNS concerns itself with the recording and retrieval of data associated with the computation of fluid flows. Included are such structures as grids, flowfields, boundary conditions, and zone connectivity information. CGNS "understands" this data in the sense that it contains conventions for recording it based on its structure and its role in CFD.

The underlying design of CGNS is that of a common database that is accessed (read, written, or modified) by a collection of "applications" programs such as solvers, grid generators, and postprocessors.

CGNS itself does not synthesize, modify, or interpret the data it stores. The applications create, edit, or display the data; CGNS is limited to recording and retrieving it. Each application's program accesses the data directly using CGNS function calls installed in the application by its developer. The applications are not regarded as part of CGNS itself.

CGNS is passive. It does not initiate action and cannot "push" information into the applications codes or "pull" information out. Rather, the codes must request the information they seek and store the information they produce. The applications must be launched by a user who organizes the location and content of the database. The process and sequence of events remain under user control. Thus CGNS facilitates, but does not incorporate, the development of batch or interactive environments designed to control the CFD process.

The elements of CGNS address all activities associated with the storage of the data on external media and its movement to and from the applications programs. These elements include the following:

- The ADF ("Advanced Data Format") Database Manager, consisting of both a file format specification and its I/O software, which handles the actual reading and writing of data from and to external storage media. In versions 2.4 and later of the CGNS software, the user can choose to use the HDF5 storage file format (supported externally by NCSA) in place of ADF.
- The Standard Interface Data Structures (SIDS), which specify the intellectual content of CFD data and the conventions that govern naming and terminology.
- The SIDS-to-ADF File Mapping and SIDS-to-HDF File Mapping conventions, which specify the exact location where the CFD data defined by the SIDS is to be stored within the ADF or HDF5 file(s).

It should be noted that because of HDF5's parallel and compression capability as well as its support, the CGNS Steering Committee has made the decision to slowly transition (beginning in 2005) to HDF5 as the official data storage mechanism. However, ADF will continue to be available for use, with the CGNS mid-level library capable of (1) using either format and (2) translating back and forth between the two.

The ADF and HDF5 Database Managers provide minimal collections of data access functions. This collection we refer to as the "ADF Core," and the "HDF5 Core." In principle, it is possible to install CGNS I/O into an application using only the ADF Core or HDF5 Core routines. However, such an approach would require the installer to access the data at a very fundamental level and would result in lengthy sequences of core function calls. Therefore, the system also includes a Mid-Level Library, an API (Application Programming Interface) that contains additional routines intended to

facilitate higher-level access to the data. These are CFD-knowledgeable routines suitable for direct installation into applications codes.

The following sections discuss in more detail the roles of the CGNS elements and introduce their documentation.

3.2 ADF Files and the ADF Core

In this section, we refer solely to the ADF Database Manager. However, the HDF5 storage format, which can be invoked as an alternative storage method in place of ADF, possesses similar functionality. Refer to the HDF5 website at http://hdf.ncsa.uiuc.edu/HDF5/ for more information.

ADF (Advanced Data Format), with its software implementation called the ADF Core, constitutes an extremely general database manager that is particularly suited to the storage of large numerical data sets. The Core is composed of a set of routines that perform standard operations on a database consisting of ADF files. Typical operations include the following: open, close, create, delete, read, and write.

An ADF file is, by definition, a file that has been written by the ADF Core. ADF files are written onto the external medium in a binary format that is completely determined by the Core. It is by this means that file portability is ensured. Consequently, the file contents are not accessible except by using the Core routines. The Core routines are internally documented, but they contain no user-serviceable parts and *should not be modified*.

The principal advantages of such an independently designed storage system are portability, data integrity, and the opportunity for self-identification of files. The ADF Core can always open and read an ADF file regardless of its origin or specific content. The universal availability of ADF, resulting from its nonproprietary nature, is also an important feature.

All ADF Core routines are coded in C, but both C and Fortran function calls are provided. The Core routines provide only the minimum functionality required to access the database.

The conceptual form of an ADF file is that of a tree, each node of which has the same internal structure. Each node contains the following:

- identification information
- pointers to its subnodes
- descriptors of any data recorded at the node
- the data itself

This format was chosen with the storage of scientific data in mind, but it is general enough that the ADF system could be used to store a wide variety of types of information. There is nothing in the ADF file specifications that restricts the data types to CFD-related information.

It is possible, through a mechanism of links, for a database implemented in ADF to span more than one ADF file. In the case of CFD, for instance, this feature allows multiple flow solutions to share the same grid without duplicating the data.

ADF contains no mechanisms to ensure the security or availability of ADF files. It is the user's responsibility to keep track of the locations, file permissions, and names of ADF files. ADF does not assume or modify any functions of the operating system. In particular, there is no file "locking"; i.e., there is nothing to prevent two applications from accessing the same file simultaneously. ADF file links are "soft" — if the linked file is moved or deleted, the connection is broken.

A more thorough overview of ADF may be found in the *SIDS-to-ADF File Mapping* document, and complete documentation appears in the *ADF User's Guide*.

3.3 Standard Interface Data Structures (SIDS)

The establishment of a standard for storing CFD-related information requires a detailed specification of the content and meaning of the data to be stored. For example, it is necessary to state the meaning of the words "boundary condition" in a form sufficiently concrete to be recorded precisely, and yet sufficiently flexible to embrace current and future practice. The *Standard Interface Data Structures* (SIDS) document describes this "intellectual content" of CFD-related data in detail.

An exact description of the intellectual content is required not only to define the precise form of the data but also to guarantee that the meaning of the data is consistently interpreted by practitioners. Thus the SIDS include a collection of naming conventions that specify the precise meaning of nomenclature (e.g., the strings `DensityStatic` and `BCWallViscous`).

The SIDS are written in a self-contained C-like descriptive language. SIDS data structures are defined in a hierarchical manner in which more complex entities are built from simpler ones. These structures are closely reflected in CGNS-compliant ADF (or HDF5) files: simple entities are often stored in single nodes, while more complex structures are stored in entire subtrees.

3.4 SIDS-to-ADF File Mapping and SIDS-to-HDF File Mapping

3.4.1 General File Mapping Principles

Because of the generality of the ADF (or HDF5) tree structure, there are many conceivable means of encoding CFD data. But for any application to access, say, the boundary conditions for zone "UnderWing", requires a single convention with regard to where in the file that data has been stored. The *SIDS-to-ADF File Mapping* document, sometimes referred to as the "File Mapping," establishes the precise ADF node, and location within that node, where each piece of CGNS data should be recorded. Similarly, the SIDS-to-HDF File Mapping document does the same thing for HDF5. The CGNS Mid-Level Library relies on the File Mapping to locate CFD-related data within the file.

The mapping provides ADF (or HDF5) locations for an extensive set of CFD data. Most applications will make use of only a small subset of this data. Further, inasmuch as applications are viewed as editors that are in the process of building the database, most of them are intended for use on incomplete data sets. Therefore, it is not required that all the data elements specified by the CGNS conventions be complete in order for a database to be CGNS compliant. The user must ensure that the current state of the database will support whatever application he may launch. Of course, the application should gracefully handle any absence or deficiency of data.

CGNS conventions do not specify the following:

- the use the applications programs may make of the data
- the means by which the applications programs modify the data
- the form in which the data is stored internal to an application

The validity, accuracy and completeness of the data are determined entirely by the applications software.

The tree structure of ADF (or HDF5) makes it possible for applications to ignore data for which they have no use. (In fact, they cannot even discover the data's existence without a specific inquiry.) Therefore, it is permissible for an ADF (or HDF5) file containing a CGNS database to contain additional nodes not specified by the mapping. Such nodes will be disregarded by software not prepared to use them. However, if data essential to the CFD process is stored in a manner not consistent with CGNS conventions, that data becomes invisible and therefore useless to other applications.

Note that the SIDS serve not only to facilitate the mapping of data onto the ADF (or HDF5) file structure but also to standardize the meaning of the recorded data. Thus there are two kinds of conventions operative within CGNS. Adherence to the File Mapping conventions guarantees that the software will be able to find and read the data. Adherence to the SIDS guarantees uniformity of meaning among users and between applications. The *SIDS-to-ADF File Mapping* document establishes the use of ADF in the context of CGNS (the *SIDS-to-HDF File Mapping* document does the same thing for HDF5); the SIDS define the nomenclature, content, and meaning of the stored data.

The File Mapping generally avoids the storage of redundant data. Sometimes an application may require an alternate (but intellectually equivalent) form of the data; in such cases it is recommended that the alternate form be prepared at the time of use and kept separate from the CGNS data. This avoids habitual reliance on the alternate form, which would invalidate the standard. If the alternate form is appended to the ADF (or HDF5) file, care must be taken to update the primary (CGNS) form whenever permanent changes are made.

3.4.2 Structure of ADF Files and ADF Nodes

In this section, the structure of ADF files and nodes is discussed. The HDF5 storage format, which can be invoked as an alternative storage method in place of ADF, is not referred to at all in this section. Refer to the HDF5 website at http://hdf.ncsa.uiuc.edu/HDF5/ for more information on the structure of HDF5 files and nodes.

Further description of the File Mapping requires some understanding of the internal structure of ADF Files. In this document, we summarize this structure as it is used in CGNS. The *SIDS-to-ADF File Mapping* document contains a complete description of the ADF file structure as used in CGNS. The *ADF User's Guide* contains a complete description of ADF file structure without regard to its use in CGNS.

An ADF file consists of a collection of elements called nodes. These nodes are arranged in a tree structure that is logically similar to a UNIX file system. The nodes are said to be connected in a "child-parent" relationship according to the following simple rules:

1. Each node may have any number of child nodes.
2. Except for one node, called the root, each node is the child of exactly one other node, called its parent.
3. The root node has no parent.

Each node in an ADF file has exactly the same internal structure. The entities associated with each node are the following:

- Node Identifier (ID)

- Name
- Label
- Data Type
- Dimension
- Dimension Values
- Data
- Child Table

Node Identifier. The Node ID is a floating point number assigned by the system when the ADF file is opened or created. Applications may record the ID and use it to return directly to the corresponding node when required. The Node ID is valid only while the file is open; subsequent openings of the same file may be expected to yield different IDs.

Name. The Name field holds a character string chosen by the user or specified by the SIDS to identify the particular instance of the data being recorded.

Label. The Label, also a character string, is specified by the CGNS mapping conventions and identifies the kind of data being recorded. For example, a node with label `Zone_t` may record (at and below it) information on the zone with Name "UnderWing." It is a general (ADF) rule that no node may have more than one child with the same name, but the CGNS mapping conventions commonly specify many children with the same label. For some nodes, the mapping conventions specify that the name field has significance for the meaning of the data (e.g., `EnthalpyStagnation`). Although the user may specify another name, these "paper" conventions serve the transfer of data between users and between applications. These names and their meanings are established by the SIDS.

Data Type, Dimension, Dimension Values, Data. ADF nodes may or may not contain data. For those that do, CGNS specifies a single array whose type (integer, etc.), dimension, and size are recorded in the Data Type, Dimension, and Dimension Value fields, respectively. The mapping conventions specify some nodes that serve to establish the tree structure and point to further data below but contain no data themselves. For these nodes, the Data Type is `MT`, and the other fields are empty.

Child Table. The Child Table contains a list of the node's children. It is maintained by the system as children are created and deleted. Its role in CGNS is the same as in ADF in general.

3.4.3 High-Level Organization of the CGNS Database

For a full specification of the location of CFD data in the ADF (or HDF5) file, the user should see the *SIDS-to-ADF File Mapping* document or the *SIDS-to-HDF File Mapping* document. For convenience, we summarize the high-level structure below.

A CGNS database consists of a tree of nodes implemented as all or part of one or more ADF (or HDF5) files. All information is identified by and accessed through a single node in one of these files. There is thus no notion of a CGNS *file*, only of a CGNS *database* implemented within one or more *ADF (or HDF5) files*.

By definition, the root node of a CGNS database has the Label `CGNSBase_t`. The Name of the database can be specified by the user and is stored in the "Name" field of the `CGNSBase_t` node. Note that this is the root of the CGNS database, not the root node of the ADF (or HDF5) file. Current CGNS conventions require that the `CGNSBase_t` node be located directly below the ADF (or HDF5) root node.

An ADF (or HDF5) file may contain multiple CGNS databases, and thus multiple `CGNSBase_t` nodes. However, each node labeled `CGNSBase_t` in a single ADF (or HDF5) file must have a unique name. The user or application must know the name of the file containing the entry-level node and, if there is more than one node labeled `CGNSBase_t` in that file, the name of the database as well.

Below the `CGNSBase_t` node, the mapping conventions specify a subnode for each zone. This node has label `Zone_t`. Its Name refers to the particular zone whose characteristics are recorded at and below the node, such as "UnderWing." In general, names can be specified by the user, but defaults are specified for nodes that the user does not choose to name. For the `Zone_t` nodes, the defaults are `Zone1`, `Zone2`, and so forth, in order of creation. A similar convention for default names applies elsewhere. It is impossible to create a node without a name (or with a name of zero length). The CGNS Mid-Level Library conforms to the default convention.

Below each zone node will be found nodes for the grid, flowfield, boundary conditions, and connectivity information; these, in turn, are parents of nodes specifying extent, spatial location, and so on.

The file mapping specifies that one or more "Descriptor" nodes may be inserted anywhere in the file. Descriptor nodes are used to record textual information regarding the file contents. The size of Descriptor nodes is unlimited, so entire documents could be named and stored within the data field if desired. Descriptors are intended to store human-readable text, and they are not processed by any supplied CGNS software (except, of course, that the text may be stored and retrieved).

It is possible, by using the linking capability of CGNS, for a child of any node to be a node in another ADF (or HDF5) file, or elsewhere within the same file. This mechanism enables one database to share a grid, for example, with another database without duplicating the information.

3.5 Mid-Level Library, or API

The CGNS Mid-Level Library, or Applications Programming Interface (API), is one of the most directly visible parts of CGNS, and it is of particular interest to applications code developers. It consists of a set of routines that are designed to allow applications to access CGNS data according to the role of the data in CFD. Unlike the ADF (or HDF5) Core, routines in the CGNS Mid-Level Library "understand" the SIDS-defined CFD data structures and the File Mapping. This enables applications developers to insert CGNS I/O into their applications without having detailed knowledge of the File Mapping. For instance, an application might use CGNS mid-level calls to retrieve all boundary conditions for a given zone.

The *CGNS Mid-Level Library* document contains complete descriptions and usage instructions for all mid-level routines. As with the ADF (or HDF5) Core, all calls are provided in both C and Fortran.

3.6 Documentation

The five CGNS elements described above are documented individually, as follows:

- *ADF User's Guide*
- *Standard Interface Data Structures*
- *SIDS-to-ADF File Mapping Manual*
- *SIDS-to-HDF File Mapping Manual*
- *Mid-Level Library*

In addition, the following documentation is also recommended:

- *CGNS Overview and Entry-Level Document* (this document)
- "The CGNS System", AIAA Paper 98-3007
- "Advances in the CGNS Database Standard for Aerodynamics and CFD", AIAA Paper 2000-0681
- "CFD General Notation System (CGNS): Status and Future Directions", AIAA Paper 2002-0752

The specific documents of interest vary with the level of intended use of CGNS.

3.6.1 Prospective Users

Prospective users are presumably unfamiliar with CGNS. They will probably wish to begin with the current Overview document, or, if they require more detailed information, the AIAA papers listed above. Beyond that, most will find a quick read of the *SIDS-to-ADF File Mapping Manual* (or the *SIDS-to-HDF File Mapping Manual*) enlightening as to the logical form of the contents of CGNS files. Browsing the figures in the File Mapping Manual, as well as the SIDS itself, will provide some feel for the scope of the system. The *User's Guide to CGNS*, and the *CGNS Mid-Level Library* document, should give an indication of what might be required to implement CGNS in a given application. Prospective users should probably not concern themselves with the details of ADF (or HDF5).

3.6.2 End Users

The end user is the practitioner of CFD who generates the grids, runs the flow codes and/or analyzes the results. For this user, a scan of this Overview document will sufficiently explain the overall workings of the system. This includes end user responsibilities for matters not governed by CGNS, such as the maintenance of files and directories. The end user will also find useful the *User's Guide to CGNS*, as well as those portions of the SIDS which deal with standard data names. The AIAA papers listed above may also be useful if more details about the capabilities of CGNS are desired.

3.6.3 Applications Code Developers

The applications code developer builds or maintains code to support the various sub-processes encountered in CFD, e.g., grid generation, flow solution, post-processing, or flow visualization. The code developer must be able to install CGNS compliant I/O. The most convenient method for doing so is to utilize the CGNS Mid-Level Library. The *User's Guide to CGNS* is the starting point for learning to use the Mid-Level Library to create and use CGNS files. The *CGNS Mid-Level Library* document itself should also be considered essential. This library of routines will perform the most common I/O operations in a CGNS-compliant manner. However, even when the Mid-Level Library

suffices to implement all necessary I/O, an understanding of the file mapping and SIDS will be useful. It will likely be necessary to consult the SIDS to determine the precise meaning of the nomenclature.

Applications code developers wishing to read or write data in ADF (or HDF5) files, that isn't supported by the Mid-Level Library, will have to use the ADF (or HDF5) Core routines directly, and will need the *ADF User's Guide* for ADF, and separate HDF5 documentation (not available here) for HDF5.

3.6.4 CGNS System Developers

CGNS System development can be kept somewhat compartmentalized. Developers responsible for maintenance of the ADF Core, or the building of supplements to the ADF Core, need not concern themselves with documentation other than the *ADF User Guide*. (Development and maintenance of HDF5 is under the purview of NCSA, so has no relevance here.) System developers wishing to add to the CGNS Mid-Level Library will need all the documents. Theoretical developments, such as extensions to the SIDS, may possibly be undertaken with a knowledge of the SIDS alone, but such contributions must also be added to the File Mapping before they can be implemented.

4 Applications Software

The development of CGNS-compliant applications, e.g., grid generators, postprocessors, and the like, has not been a direct undertaking of the CGNS team. Rather, it has been the intent to make the attractiveness of interoperable CFD applications, together with general acceptance of the CGNS standard by Boeing, NASA, and others, sufficient to induce applications developers to incorporate CGNS I/O into their offerings.

Several CGNS-compatible applications have indeed been developed, and more continue to appear. This section lists the known applications at the time of publication. Please contact Tony Iannetti (*Anthony.C.Iannetti@nasa.gov*) to modify or add to the lists.

4.1 CGNS-Compatible Applications

This section lists major software packages that read and/or write CGNS files. The software listed here is taken from Table 1 of AIAA Paper 2002-0752, and is for informational purposes only. The CGNS Project Group cannot endorse specific software packages.

Table 1: CGNS-Compatible Grid Generators and Pre-Processors

Organization	Application	Contact
Commercial		
Ansys, Inc.	ANSYS ICEM CFD	*ansysinfo@ansys.com*
NUMECA	IGG	*http://www.numeca.be/index.php?id=offices*
Pointwise, Inc.	GRIDGEN	*http://www.pointwise.com/pointwise/contact.shtml*
Industry		
Boeing Rocketdyne	APPT	

Table 2: CGNS-Compatible Flow Solvers

Organization	Application	Contact
Commercial		
Ansys, Inc.	CFX5	*ansysinfo@ansys.com*
	CFX-TASCFlow	*ansysinfo@ansys.com*
Aerosoft, Inc.	GASP	*http://www.aerosft.com/Misc/contact.php*
CD ADAPCO	STAR-CD	*http://www.cd-adapco.com/about/global_locations.html*
Fluent, Inc.	Fluent	*http://www.fluent.com/contact/*
Newmerical Tech.	FENSAP	*http://www.newmerical.com/public/eng/index.php?pageId=about&language=en*
NUMECA	EURANUS	*http://www.numeca.be/index.php?id=offices*

Continued on next page

Table 2: CGNS-Compatible Flow Solvers (*Continued*)

Organization	Application	Contact
Government		
DLR/MTU	TRACE	*Frank.Eulitz@dlr.de*
NASA Langley	PEGSUS	*Pieter.G.Buning@nasa.gov*
	OVERFLOW	*Pieter.G.Buning@nasa.gov*
	Cart3D	*http://people.nas.nasa.gov/~aftosmis/*
NASA Glenn	NCC	*Nan-Suey.Liu-1@nasa.gov*
NASA Langley	CFL3D	*http://cfl3d.larc.nasa.gov/Cfl3dv6/cfl3dv6.html#contact*
	UPS	
NPARC Alliance	WIND	*npsupport@info.arnold.af.mil*
ONERA	elsA	*http://elsa.onera.fr/elsA/contacts.html*
Industry		
Boeing Seattle	TLNS3D	
Rolls-Royce Allison	ADPAC	
Rolls-Royce Oxford	HYDRA	*pierre.moinier@comlab.ox.ac.uk*

Table 3: CGNS-Compatible Post Processors

Organization	Application	Contact
Commercial		
Advanced Visual Sys.	AVS/Express	*http://www.avs.com/contact/*
Aerosoft, Inc.	GASP	*http://www.aerosft.com/Misc/contact.php*
Tecplot, Inc.	Tecplot	*http://www.tecplot.com/contact_us/*
CEI	Ensight	*http://www.ensight.com/contact-us-3.html*
	Ensight Gold	*http://www.ensight.com/contact-us-3.html*
Fluent, Inc.	Fluent	*http://www.fluent.com/contact/*
Ansys, Inc.	ANSYS ICEM CFD	*ansysinfo@ansys.com*
Intelligent Light	Fieldview	*fieldview@ilight.com*
Newmerical Tech.	DROP3D	*http://www.newmerical.com/public/eng/index.php?pageId=about&language=en*
	ICE3D	*http://www.newmerical.com/public/eng/index.php?pageId=about&language=en*
NUMECA	CFView	*http://www.numeca.be/index.php?id=offices*
Government		
NASA Ames	PLOT3D	*Cetin.C.Kiris@nasa.gov*

4.2 CGNS Utilities

A variety of CGNS utility packages have been developed, both by CGNS team members and by CGNS users. While not formally a part of CGNS itself, these utilities are available from SourceForge, at *http://sourceforge.net/projects/cgns*, and/or from the CGNS web site, at *http://cgns.sourceforge.net/*.

Plot3dg_to_CGNS	Converts a "standard" Plot3D grid file to a CGNS grid file. Contact: Chris Rumsey (*Christopher.L.Rumsey@nasa.gov*)
CGNS_to_Plot3d	Converts a CGNS file with a cell-centered solution to "standard" Plot3D grid and solution files. Contact: Chris Rumsey (*Christopher.L.Rumsey@nasa.gov*)
CGNS_readhist	Reads a CGNS file and writes history data to a formatted file. Contact: Chris Rumsey (*Christopher.L.Rumsey@nasa.gov*)
FTU	Converts CGNS files to and from Plot3D format. Has a text-based menu allowing the manipulation of a CGNS base. Contact: Bob Bush (*bushrh@pweh.com*)
ADFviewer	An ADF/CGNS file viewer and editor, written in *Tcl/Tk*. The GUI allows complete editing of ADF/CGNS files, and handles links and data conversions. It includes built-in CGNS node and label information, and documentation support through HTML (including access to the ADF and CGNS docs). Contact: Bruce Wedan (*brucewedan@hotmail.com*)
CGNS Tools	A variety of tools for viewing, editing, and manipulating the contents of CGNS files; requires *Tcl/Tk* 8.3. Included are: • *CGNSplot*, for plotting grids • *CGNScalc*, a calculator using data in CGNS files • Utilities for converting from Patran, and from/to PLOT3D and Tecplot • Utilities for modifying the grid location (vertex or cell-center), solution variables (primitive or conservative), and data class (`Dimensional`, `NormalizedByDimensional`, or `NormalizedByUnknownDimensional`) • Utilities for creating a subset or interpolated version of a CGNS file These may be used as stand-alone utilities, or initiated from *ADFviewer*, which is included. Contact: Bruce Wedan (*brucewedan@hotmail.com*)
CGNS Viewer	An alpha release of another ADF/CGNS file viewer and editor. Requires the GTK+ library version 1.2 or newer (installed by default on newer Linux systems using Gnome; otherwise found at *http://www.gtk.org/*). Contact: Christian Lundh (*xistan@yahoo.se*)
PyCGNS	A Python binding to the ADF Core and Mid-Level Library. For more information, see the pyCGNS home page at *http://elsa.onera.fr/CGNS/releases/*. Contact: Marc Poinot (*Marc.Poinot@onera.fr*)

ADFM	A development version of software for managing an in-memory representation of ADF trees. For more information, see "Proposal and Prototype for an in-memory representation of ADF trees" at *http://cgns.sourceforge.net/utilities/ADFM/ADFM.pdf*. Contact: Marc Poinot (*Marc.Poinot@onera.fr*)
CGNS++	A C++ binding to the ADF Core and Mid-Level Library. For more information, see the CGNS++ home page at *http://cgnspp.sourceforge.net/*. Contact: Manuel Kessler (*kessler@iag.uni-stuttgart.de*)

5 Acquiring CGNS

5.1 Software

The CGNS software is available free of charge, under the terms of the CGNS License, from SourceForge, at *http://sourceforge.net/projects/cgns*. Also available there are the *cgnstools* utilities, the source code examples from *A User's Guide to CGNS*, and additional Fortran source code examples.

The CGNS Library contains source code for both the Mid-Level Library and the ADF Core, plus makefiles and a configure script for building the library for a variety of platforms.

5.2 Documentation

The CGNS documentation may be accessed via the CGNS Documentation home page at *http://www.grc.nasa.gov/WWW/cgns/*. Documentation may also be available for the current beta version of CGNS. All the CGNS documentation is available in both PDF and HTML forms.

In addition to the CGNS documentation itself, several conference papers and slide presentations are available, as well as minutes from the CGNS meetings and telecons.

6 History and Current Status

6.1 Initial Development

The initial development of CGNS took place from March 1995 through May 1998, when Version 1.0 of the CGNS software was released. The project originated through a series of meetings in 1994-1995 between NASA, Boeing, and McDonnell Douglas, who were then working under the Integrated Wing Design element of NASA's Advanced Subsonic Technology Program. The work being planned involved extensive use of CFD, and the possibility of collaborative analyses by many organizations. It was thus necessary to establish a common data format suitable to meet the needs of production CFD tools in the mid- to late-1990s. This format would be used to enable interchange of data among different CFD-related tools and different computing platforms, and to provide a mechanism for archive and retrieval of CFD data

At that time, the *de facto* standard for CFD data was the file format used by Plot3D, a flow visualization program developed at NASA Ames. However, the Plot3D format was developed to expedite post-processing, and was never intended as a general-purpose standard for the storage and exchange of CFD data. By the early 1990s, as CFD became more sophisticated, the limitations of the Plot3D format as a general-purpose standard had become apparent. It had no provisions for storing data associated with modern CFD technology, such as unstructured grids, turbulence models based on solutions of partial differential equations, and flows with multiple chemical species.

Individual organizations were overcoming these limitations by defining extensions to the Plot3D format to meet their needs. These extensions were not coordinated among different organizations, and therefore data stored in these extended formats generally could not be utilized outside the originating organization. Further, the Plot3D format was not self-documenting, and it was necessary to rely on file-naming conventions or external notes to maintain awareness of the flow conditions and analyzed geometry of each PLOT3D data file.

To overcome these limitations, several database options were considered by the NASA / Boeing / McDonnell Douglas team from December 1994 to March 1995. In March 1995, the decision was made to build a new data standard called CGNS (Complex Geometry Navier Stokes). This standard was a "clean sheet" development, but it was heavily influenced by the McDonnell Douglas Common File Format (CFF), which had been established and deployed in 1989 and heavily revised in 1992.

Agreement was reached to develop CGNS at Boeing, under NASA Contract NAS1-20267, with active participation by a team of CFD researchers from

- NASA Langley Research Center
- NASA Glenn Research Center
- NASA Ames Research Center
- McDonnell Douglas Corporation (now Boeing St. Louis)
- Boeing Commercial Airplane Group (Seattle) Aerodynamics
- Boeing Commercial Airplane Group (Seattle) Propulsion
- ICEM CFD Engineering Corporation

The original development work itself occurred from 1995–1998, and consisted of essentially four activities:

- Development of the standalone database manager (the ADF Core)
- Identification and layout of the CFD data (the SIDS and File Mapping)
- Development of an API for use in applications codes (the Mid-Level Library)
- Demonstration of system capability

6.1.1 Development of ADF

As noted earlier, the decision to develop CGNS was made after an examination of several existing formats. Because of its hierarchical structure, the CFF (Common File Format) that was then in use at McDonnell Douglas was judged the most compatible with the SIDS. However, CFF had known limitations, and it was decided to adopt the CFF structure but recast it in a more general form. This gave the developers control over the software and facilitated the development of a robust, portable, reliable, and flexible database manager, which could be distributed freely within the system.

These low-level routines were developed during 1995, and make up the ADF (Advanced Data Format) Core. ADF was written and extensively tested at Boeing by a small subteam familiar with standard software development practices, and it may now be considered a stable product.

6.1.2 Identification and layout of the CFD data

Another task, undertaken in parallel with the development of ADF, was to identify the data associated with CFD and establish a means of encoding it within the ADF format. This task was the primary concern of the national team, with Boeing proposing the standards and the remainder of the team serving to critique, test, and improve them. This began with the development of the SIDS and its language. The hierarchical nature of the SIDS was used to suggest the SIDS-to-ADF File Mapping conventions, which were developed slightly after the SIDS but along the same track.

6.1.3 Development of an API

At a review in June 1997, the CGNS team (NASA, Boeing, and McDonnell Douglas) determined that additional support would be required to produce an adequate Mid-Level Library. Subcontracts were issued to the ICEM CFD Engineering Company, in Berkeley, CA, following this decision. ICEM CFD in effect became the lead organization for the development of the Mid-Level Library.

Also at this time, the acronym "CGNS" was re-defined to mean "CFD Generalized Notation System", which was more in keeping with the evolved goals of this project.

Version 1.0 of the Mid-Level Library, which met the original goal of supporting structured multi-block analysis codes, was released in May 1998. This set of higher-level routines enables a user to implement the CFD standards defined by the SIDS, without specific knowledge of the File Mapping or the ADF Core. This activity was crucial to the acceptance and implementation of CGNS among the CFD community.

NASA and the informal CGNS committee determined that there was no need for export authority so the CGNS standard, the ADF Core and the Mid-Level Library, and all supporting documentation could be distributed worldwide as freeware. Appropriate legal reviews and approvals were obtained at both NASA and Boeing to validate this decision.

6.1.4 Demonstration of system capability

Prototypes. To test the CGNS data recording and retrieval mechanisms and to gain experience with the installation of the software into common applications codes, the developers modified two CFD solvers (namely, NPARC and TLNS3D) to operate in the CGNS environment. In addition, NASA Langley Research Center modified CFL3D similarly. These modifications were made early in the project and predate the Mid-Level Library and some later portions of the SIDS.

Separate data cases were prepared for each of the prototypes. Each prototype proved capable of starting, exiting, and restarting its data case as expected. In addition, in all three cases, transfer of the CGNS data between workstation (SGI) and mainframe (CRAY) platforms was successfully demonstrated.

Dissimilarity of the *content* of the data required by NPARC and the other two codes (i.e., nodal vs. cell-centered data) prevented restarting NPARC from TLNS3D/CFL3D data and vice versa. However, this type of restart was demonstrated between TLNS3D and CFL3D.

These prototypes were limited in certain ways and, therefore, were not suitable as a final implementation of their respective capabilities in CGNS. The principal limitations are:

- The prototypes implement only the grid coordinates, flow solution, boundary condition, and 1-to-1 interface connectivity portions of the CGNS data specification.
- No attempt was made to modify the internal structure of any of the codes in order to improve compatibility with CGNS data organization.
- The prototypes accessed the ADF files primarily using routines at the ADF Core level. There were some higher-level routines included, and these, in many cases, suggested the content of the CGNS Mid-Level Libraries. But the high-level prototype routines often intermingled ADF functions with CGNS functions and, in some cases, were code-specific or dependent on internal directives to make them so. They were thus less broadly applicable than the current Mid-Level Library routines.
- These codes exercised only a portion of the CGNS boundary condition specifications.
- The prototypes sometimes incorporated extra nodes into their ADF files to carry code-specific data. This practice arose partially from the lack of complete CGNS data specifications at the time the prototypes were written, but it resulted also, in part, because the current code input structure required a different (but equivalent) form of the CGNS data, and the developers opted to duplicate it within the CGNS database.

The general lesson learned from the construction of the prototypes was that professional programmers had no conceptual difficulty in implementing CGNS at the ADF Core level. But the resulting code was cumbersome, and the development of the Mid-Level Library was needed to facilitate dissemination of CGNS among those disinclined to work with the ADF Core.

System Demonstrator. As the CGNS effort progressed, an additional activity was undertaken to demonstrate the capability of the mature system. Known as the "system demonstrator", this tested the ability of CGNS to transfer data seamlessly between applications that had never operated together before. The system demonstrator used most of that portion of the currently existing SIDS that has been included in the Mid-Level Library.

The geometry chosen for the test was a high-lift configuration known as the trapezoidal wing. This is a multielement airfoil with a full-span slat and flap, and a generic fuselage.

Three separate CGNS-compatible application codes were involved in the system demonstrator.

GMAN	A pre-processor and grid generation tool developed by Boeing St. Louis
OVERFLOW	A CFD flow solver developed by NASA Ames
Visual3	A flow visualization tool from ICEM CFD, originally developed at MIT

The system demonstrator consisted of the following tasks:

1. The grid was generated using NASA Ames grid tools and written as a Plot3D file.
2. The grid file was sent to Boeing St. Louis, where it was processed by a locally-modified version of GMAN. GMAN calculated the grid connectivity information, and wrote the grid and connectivity data into a CGNS database. Boundary conditions were also added to the CGNS database at this time.
3. The CGNS file was returned to NASA Ames, where it was read by the newly-modified OVERFLOW code. (Some iteration was necessary here, because the definition of overset holes used by GMAN differs from that normally expected by OVERFLOW. The CGNS file was intact, and served, as intended, to highlight the discrepancy.) OVERFLOW computed the flow field, writing the results into the CGNS database.
4. The CGNS file was next sent to ICEM CFD. There, the CGNS database was read and displayed using Visual3.

The system demonstrator involved significant cross-platform transfer among various workstation and mainframe computing facilities. The results indicated that universal data exchange was well supported by CGNS.

6.2 Subsequent Development

6.2.1 Software

Since the release of Version 1.0 of the CGNS software in May 1998, several improvements and extensions have been made to the SIDS, the File Mapping, and the Mid-Level Library.

- Version 1.1, released in June 1999, added support for unstructured grids, and for associating CAD geometric entities with grid surfaces.
- Version 2.0, released in December 2000, added support for moving and/or deforming grids, and for iterative/time-dependent data.
- Version 2.1, released in May 2002, added support for user-defined data arrays, chemistry, and linked nodes.
- Version 2.2, released in May 2003, added support for axisymmetry information, rotating coordinates, special properties associated with particular grid connectivity patches, such as periodicity or averaging, special properties associated with particular boundary condition patches, such as wall functions and bleed, and gravity.
- Version 2.3, released in Jan 2004, restored the capability to specify a boundary condition patch using `ElementRange` or `ElementList`.
- Version 2.4, released in Aug 2005, added support for describing electric field, magnetic field, and conductivity models used in electromagnetic flows, specification of units for electric current, substance amount, and luminous intensity, more flexible specification of boundary condition locations, and additional user-defined data. Also new in Version 2.4 is the ability to choose, at build time, either ADF or HDF5 as the underlying database manager.

- Version 2.5, released in Sep 2007, added Mid-Level Library functions to check file validity, configure some internal CGNS library options, and provide alternate ways to access a node. Some changes were also made in the use of functions for partial writes of coordinate, element, and solution data. Support was also added for building the CGNS library as a DLL under Windows.

A more detailed list of revisions to the CGNS software is available at the CGNS web site.

ICEM CFD served as the focal point for CGNS software development through the release of Version 2.0, plus the changes in Version 2.5, using internal company resources. New features in Version 2.1 and 2.4 were added by Intelligent Light, with funding from NASA Langley. Eagle Aeronautics, with additional participation by ICEM CFD, added the new features in Version 2.2, again with funding from NASA Langley. Support for HDF as the underlying database in Version 2.4 was added by personnel from ONERA, ICEM CFD, and the U. S. Air Force Arnold Engineering Development Center.

6.2.2 Management and Support

In 1998 NASA announced that the Advanced Subsonic Technology program, which had funded the initial CGNS development, would end in September 1999. Several organizations interested in the continued development of CGNS met in May 1999 to discuss options. The decision was made to create the CGNS Steering Committee, a voluntary public forum made up of international representatives from government and private industry.

The Steering Committee is responsible for coordinating the further development and dissemination of the CGNS standard and its supporting software and documentation. In Jan 2000, the CGNS Steering Committee became a Sub-committee of the AIAA CFD Committee on Standards. Additional details about the mission and responsibilities of the Steering Committee, and its organization, are in the *CGNS Steering Committee Charter*.

The basic CGNS documentation has of course been updated to reflect the software changes described above. In addition, all the documentation has been converted to LaTeX (used to create PDF versions for printing), and to HTML (for interactive use). A new document, *A User's Guide to CGNS*, was made available in October 2001, and is very useful for new and prospective users of the Mid-Level Library.

To encourage communication between CGNS users, a mailing list called CGNStalk was created in Oct 2000. Instructions for subscribing are available at the CGNS web site.

6.3 Current Status

Since the initial release of the Mid-Level Library in May 1998, interest in CGNS has continued to grow throughout the global CFD community. The system is being used by engineers and scientists in academia, industry, and government. By January 2002, 591 users from more than 25 countries had formally registered at the CGNS web site. As of September 2007, the CGNStalk mailing list had 224 participants from 20 different countries and at least 79 different organizations.

6.3.1 Software

The current "production" release of CGNS is Version 2.5, released in September 2007.

Several extensions to CGNS have been formally proposed. Documentation supporting these extensions, and information on their current status, is available via the "Proposals for Extensions" link at the CGNS web site.

6.3.2 Standards

Between 1999 and 2002, an effort was spearheaded by Boeing to establish an ISO-STEP standard for the representation, storage, and exchange of digital data in fluid dynamics based on the CGNS standard. Unfortunately, the effort had to be curtailed because of budget problems. It was subsequently decided that an existing ISO standard on finite element solid mechanics would be rewritten and submitted to include CGNS as well as an integrated engineering analysis framework.

As part of its role as a sub-committee of the AIAA CFD Committee on Standards, the CGNS Steering Committee is also involved in the development of the AIAA Recommended Practice for the storage of CFD data. The Recommended Practice consists of the CGNS Standard Interface Data Structures (SIDS) document, reformatted to conform to AIAA's requirements. The current AIAA Recommended Practice (corresponding to CGNS Version 2.4) is available at the AIAA Online Store, and as a PDF file (1.01M, 200 pages) at the CGNS Documentation web site.

6.4 Noteworthy Contributions

The entire CGNS team deserves credit for its accomplishments. The project might easily be cited as a prime example of effective voluntary cooperation between government and industry. Every member has contributed ideas, enthusiasm, and improvements, and enabled us to represent a cross-section of knowledge and practice from the CFD community.

It would not be practical to outline each member's contributions to a project of this complexity. Nevertheless, it is appropriate to note the contributions of a few individuals who dedicated special effort during the crucial stages leading to the initial release of CGNS in May 1998.

- Steve Allmaras conceived, pursued, and completed the SIDS and developed the language in which it is written.
- Tom Dickens, Matt Smith, and Wayne Jones designed the ADF Core.
- Tom Dickens wrote the ADF Core.
- Dan Owen maintained and tuned the ADF Core after Tom left the group.
- Chuck Keagle designed and executed stringent testing procedures for the ADF Core; that is to say, he did everything he could think of to break it.
- Diane Poirier designed and wrote the CGNS Midlevel Library and drafted the original proposals for unstructured grid. She and Alan Magnuson drafted the proposals for CAD-to-geometry specifications.
- Gary Shurtleff wrote the TLNS3D and NPARC prototypes and provided us with what was, for a long time, the only examples of working CGNS software.
- Chris Rumsey served as liaison with NASA Langley, responding in detail to requests for criticism and improvements and testing the system by writing the CFL3D prototype.
- Cetin Kiris converted the OVERFLOW code to CGNS, twice.

- Wayne Jones tested our abstractions with real code, which found its way into the ADF Midlevel Library and the CGNS Toolkit.
- Matt Smith wrote much of the File Mapping and ADF Core documents and, with knowledge of both CFD and software design, brought good sense to bear on proposals that needed it.
- Ray Cosner's retrospective vantage point made him a reliable supporter who was often able to move things forward when progress slowed.
- Susan Jacob supplied initial guidance as a Productivity+ coach and got us all moving in (more or less) the same direction.
- Doug McCarthy led the project to completion.
- Shay Gould made the documentation intelligible and presentable.

And, a particularly special mention:

- Ben Paul secured the initial funding and shepherded the project through the early phases of the contract. We will never forget the many meetings Ben held to make sure that he thought that he knew that we thought that we knew what we needed to do, and that we were doing it. His patience and good-natured perseverance could not have been replaced.