

Impact of CGNS on CFD Workflow

M. Poinot*

ONERA, Chatillon, F-92322, France

C. L. Rumsey†

NASA Langley Research Center, Hampton, VA 23681-2199, U.S.A.

M. Mani‡

Boeing, St Louis, MO 63146, U.S.A.

CFD tools are an integral part of industrial and research processes, for which the amount of data is increasing at a high rate. These data are used in a multi-disciplinary fluid dynamics environment, including structural, thermal, chemical or even electrical topics. We show that the data specification is an important challenge that must be tackled to achieve an efficient workflow for use in this environment. We compare the process with other software techniques, such as network or database type, where past experiences showed how difficult it was to bridge the gap between completely general specifications and dedicated specific applications. We show two aspects of the use of CFD General Notation System (CGNS) that impact CFD workflow: as a data specification framework and as a data storage means. Then, we give examples of projects involving CFD workflows where the use of the CGNS standard leads to a useful method either for data specification, exchange, or storage.

I. Introduction

Today's CFD applications are generating a large amount of data. And current software techniques, used with large and heterogeneous information systems, allow users to share data between different applications, thus generating even more new data.^{1,2} The phrase *CFD workflow* implies many data representations, involving different formalisms, tools and translators. Emergent technologies, based on XML or CORBA or any other general software bus are used by the CFD community today.³ These generic systems can handle data descriptions, and most of them are providing the user with a means to define his own types or embed data specifics in a self-described frame. Today's interoperable systems have efficient mechanisms for such definitions. However, too much generalization may lead to a loss of *information*, and peer-to-peer applications have to define their own context to understand the *data*.⁴ Within the CFD workflow, CFD General Notation System (CGNS)⁵⁻⁹ has been developed to overcome these difficulties.

CGNS was conceived in the mid 1990's to improve the transfer of NASA technology to industry. It started as a NASA-Boeing effort, but quickly grew to include other companies and organizations. It is now promoted by a steering committee made up of members from approximately 15 companies and organizations, and is on the way to becoming an international ISO (*International Organisation for Standardization*) standard for the representation of fluid dynamic data.

The most important component of CGNS is the Standard Interface Data Structure (SIDS). It is essentially a high level description of CFD types. In this sense, the CGNS standard should be understood as an applicative model, not a low level storage layer.

We show in this paper that data specification is one of the primary challenges in today's complex CFD workflow, while actual storage of data can be achieved in many ways, as long as one defines an *open system* architecture¹⁰ for the information system. Several examples are given of CGNS usage in CFD workflows.

*Software Engineer, Computational Fluid Dynamics and Aeroacoustics Department, Member AIAA

†Research Scientist, Computational Modeling and Simulation Branch, Associate Fellow AIAA

‡Boeing Associate Technical Fellow (CFD), 344N, Associate Fellow AIAA

II. CGNS as a specification standard

A. Context

The CFD solver is a part of a larger set of tools, in both the industrial context as well as in the research context. More and more, with the availability of large computational resources, researchers are coupling solvers to obtain more accurate solutions for complex physical problems. The range of data used and produced gets broader as researchers make use of more complex systems. CFD related data is now often tightly bound to structural or thermal related data, for example.

Some of these data are obviously the same, from the physical point of view, but each solver or application code usually manages these data their own way. The use of a common basis, for the common subset of data, is becoming one of the critical aspects in the computational process that can lead to greater efficiency. Even if we do not take into account the coupling problem at the physical level, and instead only focus on this common basis of data exchange, it is easy to recognize the need for a public standard for data specification. CGNS is such a standard, and is gradually gaining broader acceptance throughout the CFD community.

The standard should not be seen as an attempt to define the *ultimate* set of data that is allowed to be provided or produced by a CFD tool, but rather as the common subset on which the CFD community agrees.

B. Technical vs semantic interoperability

First, we aim to show that CGNS is more than a mere file format. As two entities communicate, they are obviously using a common language or context. For example, consider ρU or *MomentumX*. These represent specific values that can be more or less meaningful depending on who uses them. The CFD expert knows their meanings and all that they imply. The general scientist can understand a field of ρU values, without needing to understand the background, and he can still use the values. The code developer can consider it simply as an array of floats; he may not need to have the whole context in mind. All of these levels can be found in the CGNS architecture.

C. Clear layers make clear interfaces

An actual exchange of data would lie on at least two levels: the specification level and the implementation level. It has been shown, with the OSI (*Open System Interconnection*) seven layers reference model,¹¹ that levels of interoperability can be split into layers. Each level N layer can only communicate with the layer N-1 and the layer N+1 through specific Application Program Interfaces (APIs), as well as with its own level N peer layer.

Fig. 1 shows a schematic representation of two codes (a fluid solver and a structure solver) that are interacting and sharing data. The two codes are at the highest level in the hierarchy. Below them lies the Standard Interface Data Structure (SIDS)⁶ of CGNS, which is the standard format defined in the CGNS system for recording the data. SIDS is in effect a detailed description of the intellectual content of the information that is stored. For example, at the SIDS level would be a specification of how the grid points and flow solution data are stored. The SIDS level, say N, has an N+1 interface we call *Mid level interface*. At the level N-1 lies the SIDS-to-ADF layer, which translates between the semantics of the SIDS and its representation in the computer file. Here, ADF stands for “Advanced Data Format.”¹² It is a stand-alone database manager that implements a tree-like structure as a binary file. This is the low-level method for storing data currently employed in the CGNS software.

So far, the only *physical layer* CGNS can handle is the file system. There have been prototypes using memory or shared memory as the *physical layer*, without any change for the *ADF* interface, but these still remain prototypes. The use of the memory physical layer can lead to memory transfer mechanisms, such as MPI, or even RPC. The peer application can use any of the N+1 level, as long as the interfaces are kept the same.

D. CGNS interfaces

Back to the two solvers, we can now have a look at the exchange levels. As shown in Fig. 2 the applications actually are exchanging two arrays of float values. This is done through the ADF interface. The meaning of these arrays is usually known only within the specific context of the particular exchange taking place, which is guaranteed by the standard at the SIDS level. At the ADF level, there are possibilities of adding semantics

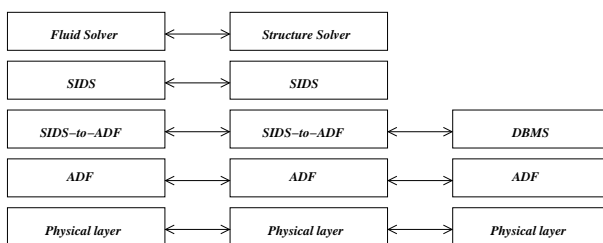


Figure 1. OSI model applied to CGNS

to the data itself. In this case, both parties should know the meaning of the data because of the descriptive semantics included. The semantics can be added in two ways:

1. It is described as private information; no other party can use the data. A new code can only see two arrays of floats.
2. It is described as public information; any party that uses the public specification is able to understand the meaning of the data.

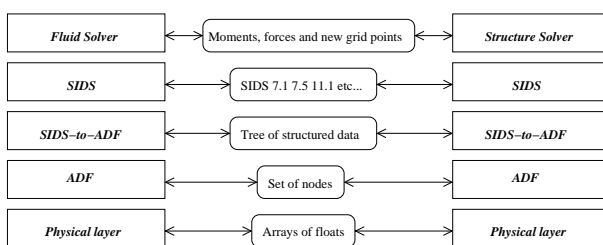


Figure 2. Data exchange levels

One can also add multiple levels of description. With case 2, the data can be completely described in the public specification, or this specification can define a way to add semantics to the data. In the latter case, an interpreter-like system is required. The semantics of the data could be represented with a specific language, such as in a DBMS, XML or any formal specification language with a high level of semantics.¹³ A view of the schema can be specialized for a given application, given that all applications can share the overall view on a standard basis,^{14, 15}

All these cases described are “process” oriented, because this is one of the most valuable aspects of CGNS, compared to other “anonymous” formats such as HDF.¹⁶ However, note that in Fig. 1 the ADF layer can be changed to an HDF layer without impact on the application. What is important for the application is that the meaning of the data (the SIDS layer) is the same. One will always have the ability to translate one data structure into another data structure at the lowest levels of the hierarchy.

E. Conformance to standard

An application which uses and/or produces CGNS data by following the SIDS protocol is said to be *compliant* to the standard. Although it does not yet exist, ultimately one would guarantee compliance to the standard by using some sort of *certification* process that consists of a suite of tests and controls. Being conformant to the standard does not necessarily mean that the application needs to understand 100% of all CGNS specifications, and it also does not mean that all of the application’s provided data must be 100% CGNS. Instead, it means that *if the application provides or uses CGNS data, then these particular data are compliant to the standard*. We give an example to illustrate this concept.

Say that a user has a code that generates CFD data, including gridpoints and flowfield solution. These data are included in the CGNS standard, and must be written to the CGNS file in a particular way in order to remain compliant. But if the application also generates particular data not covered by the standard (for

example, some code-specific information such as Runge-Kutta coefficients or multigrid sequencing instructions), then these data can also be written to the CGNS file (under `UserDefined` or `Descriptor` nodes), and the file will still remain compliant. And if this particular application never uses (for example) convergence history information, then it does not need to be able to read or understand that part of a CGNS file. In other words, the application will still be able to read a compliant CGNS file generated by someone else, even if the file contains convergence history information.

So far, the only way to check compliance to the CGNS standard is to actually produce a physical file by building a file containing the data, using the CGNS libraries. We plan to provide a textual representation of a CGNS specification, for instance using XML. Such a textual representation could be checked by a compiler-like tool, which would act like a Fortran compiler acts when the user asks for a compilation. Any Fortran compiler can check the Fortran compliance of a program text. It is not necessary to produce a binary file in order to determine if the program text is correct or not. The user can use a large set of tools, which are aware of the standard, and which can check the text compliance to the standard.

Here is an example of an XML representation of a CGNS tree. This representation is generated using a Python tool which has its own grammar, described using *RelaxNG*¹⁷ syntax. Thus such an XML output should not be considered as a standard, but rather as proprietary until the CGNS Steering Committee agrees on a given grammar. In this example, the XML validator, i.e. the tools that checks the conformance, detects a problem on the last line. The `name` of the node cannot be `ORPHAN`. It should be `FamilyName` while the `cgnsdata` attribute should be `ORPHAN`.

```
<CGNS-Tree-t CGNSLibraryVersion="1.1">
<CGNSBase-t name="BASE\#1" physicaldimension="3" celldimension="3">
<Zone-t name="domain.1">[[4, 4, 10], [3, 3, 9], [0, 0, 0]]
<FamilyName-t name="ORPHAN" cgnsdata="EMPTY" />
...

```

This check is performed on a textual representation of the CGNS tree. That means that the user can write down its tree skeleton, or rather generate it, and check the conformance of the skeleton without using the CGNS libraries.

The CGNS community has set up a standard extension process. Many users want to see their own CFD-related data specified as a public extension to the base standard.

F. Profiles

The CGNS standard can be specialized for a given application scope, or a project, in order to make rules more strict. This is a standard *profile*. For instance, an obvious profile can set a number of fixed names or name patterns. In the *elsA*¹⁸ solver, we decided that solutions would have the pattern `FlowSolution#<suffix>` where `<suffix>` can be one of:

- `Init` for the initialization (this can be a *moving* link to an actual solution).
- `EndOfRun` for the last computed solution (again, this can be a *moving* link).
- `NNNN` the iteration number, or any other ordinal number represented as integer.

The specialization can be more complicated, for example, when dealing with unsteady data. The CGNS standard has a set of rules for the unsteady data specification. Because the end-user wants an efficient parse and understanding of the CGNS data, he wants to reduce the effort of looking for the actual data he requires for his specific CFD workflow. The example of an unsteady CGNS tree, specialized to a rigid grid motion, is given in Fig. 3. This figure is a very simplified view of the actual profile, in which the specification implies the semantics of the CGNS SIDS and the constraints described for the target application. In this example, the `IterationValues` node of `BaseIterativeData` node is mandatory. Moreover, it is required as an added constraint to this tree that every iteration step should have a rigid grid motion.

A profile is used to reduce the effort to extract the actual data. Without a profile, the application would have to search through all CGNS conformant structures. The use of profiles also can be understood as a *best practice*, or *common practice*. A CFD workflow, or an end-user application would eventually include such a profile specification as a public reference document. We are presenting here simplified graphs of CGNS trees.

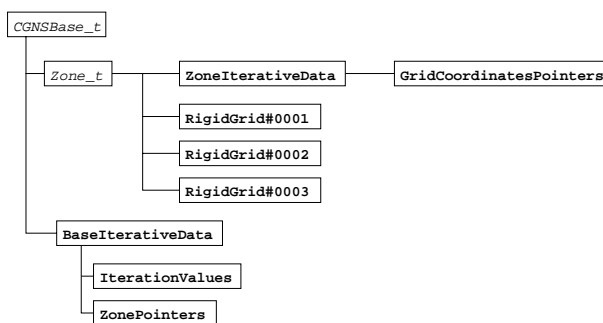


Figure 3. A profile tree for unsteady grid changes

However, the profile definition has to be specified using the precise SIDS syntax. It is a difficult syntax for which semantics of data has to be extended with textual comments, but now many examples are available in the CGNS community. Here is a short example of a SIDS definition relative to the `BaseIterativeData` node:

```

BaseIterativeData_t :=
{
  int NumberOfSteps                                     (r)
  DataArray_t<int,1,NbOfSteps>                         IterationValues ; (r)
  DataArray_t<char,3,[32,MaxZones,NbOfSteps]>          ZonePointers ;   (r)

-- All other options/constraints
-- are allowed w.r.t. SIDS
}

```

G. Prototyping

Without tools for actually defining a data structure, such as a compiler, the specifications consistency can only be insured by the writer and the reviewers. Thus, prototyping is valuable (prototyping is done using the *pyCGNS* package,¹⁹ which is a Python binding of the Mid-level library of CGNS). Prototyping instances of the target data structure can help the user to understand and define a good data structure.

The following Python code tries to create a structure under a BC (boundary condition) node. This action will fail, because the CGNS standard does not allow the storage of anonymous arrays of data at this level, but rather in the sub (child) level.

```

from CGNS import *
range=[(1,1,1),(1,2,1)]
a=pyCGNS("MULBCBASETEST.cgns",MODE_WRITE)
a.basewrite('base',3,3)
a.zonewrite(1,'zone',(3,5,7,2,4,6,0,0,0),Structured)
a.bcwrite(1,1,'BC-01',UserDefined,PointRange,range)
a.bcwrite(1,1,'BC-02',UserDefined,PointRange,range)
a.bcwrite(1,1,'BC-03',UserDefined,PointRange,range)
a.bcwrite(1,1,'BC-04',UserDefined,PointRange,range)
a.close()

```

The CGNS library calls have their own minimal set of standard conformance checks. Again, this emphasizes the point that CGNS is at a higher level than the data structure. For example, it knows the semantics of the application patch for a boundary condition, and enforces this. During a specification process, the exchanged data schema can be tested with these little Python scripts. Such prototyping is used at the specification stage for scenario implementation, and also at the software release test stage.

III. CGNS as a data storage means

The first section described CGNS as the specification means for CFD related codes' interoperability. Now, getting back to the OSI layer model, we also have to insure the actual data interoperability. This also is one of the main goals of the CGNS initiative.

A. Basic storage layer

The ADF layer has the minimal and efficient features required for archiving data. Its key points are compactness, portability, and access time efficiency.

Storage is based on a file system, and it can handle one or many files. In the case of many files, using the link mechanism, it is up to the end-user to handle the link names and their consistency. This actually is a current problem for CFD workflow platforms. If a tool copies or if it changes the name of one of the files used in a CGNS tree, the actual name of the link, embedded as a string, may not refer to the correct name. This will lead to a CGNS library failure.

B. Other storage layer requirements

Storage layer requirements for CGNS are short enough to allow it to be mapped to many existing storage systems. The SIDS does not assume data management system features such as concurrency, access control, transactional, or querying. Even the parallel ability feature, widely used in the CFD community, is not taken into account. The only feature that could be related to data management system is the `link` feature described in the previous section.

The ADF interface for an *in-memory* CGNS tree prototype has been developed. It creates and manipulates a CGNS conformant tree built in the process memory. The memory allocations can be done in order to insure a contiguous memory zone that can be transferred via a memory buffer based system like MPI. A parallel code can exchange MPI buffers and access these buffers through an ADF interface, thus insuring the transfer of the *information* with the *data*. There is no loss of semantics due to the marshalling of the CFD data into arrays of floats or lists of integers.

The system can also be used for other in-memory data use, like shared memory between two processes, or even in order to access to the same memory zone in one process with two different programming languages. This last option is the one used to transfer a CGNS in-memory tree that is manipulated both by a C++ solver and its Python wrapper.

C. Mapping to HDF5

As mentioned earlier, the ADF layer could be replaced without impact on the application. The HDF5 system is a good candidate for ADF replacement. In the process of building the CGNS CFD standard, it has been the practice to define a minimal storage layer and to focus on the high level interface first. Now CGNS is stable enough to insure that there is no dependancy between the application level and the storage level. The Steering Committee is currently considering changing the storage layer in order to have a more widely-used basis.

A first prototype has been developed in which the mapping has been done in the N-1 interface of the ADF. In other words, the N+1 interface (all ADF calls) is unchanged, but the calls done in the ADF library in order to build the actual nodes on disk are exchanged with HDF5 calls. The ADF node to HDF5 node mapping is the following:

ADF node	HDF5 node
name	group name+ attribute
label	attribute
data type	dataset:datatype
number of dims	dataset:dataspace <rank>
dim values	dataset:dataspace <dimensions>
data	dataset: <contents>

The prototype has been enhanced by the CGNS Steering Committee, which is now considering releasing the next version of the CGNS library with the HDF5 mapping. This mapping would then be the official mapping for the physical storage layer.

D. Mapping to a DBMS

We also see CGNS as a *long term archival* and experimental data storage mechanism. It is possible to map data structures to DBMS (*DataBase Management System*), using SQL translators to a relational DB schema for instance.²⁰

With a relational DBMS, for example, the binary large objects (blob) mechanism is used. Then the whole file is stored as a `VARCHAR`, which is a very large binary string. The semantic contents of the file is extracted at database insertion time and is used to populate attributes. It is worth mentioning that this content may only be a *part* of the expected semantic content of the result. Using such a system, we have the RDBMS added value: the contents of the file are now available for querying. For example, one can ask for *all configurations with M6 wing in Euler with scalar dissipation*. Such a query is made using the SQL statements:

```
select b.id,b.Date,f.Zone,f.Name,f.Dimensions
from f as FlowSolutionField,
     b as Base,
     e as FlowEquationSetInfoTable
where b.targetDescriptor='M6'
and f.Solution='FlowSolution#EndOfRun'
and e.GoverningEquations='Euler'
and e.ComputationDissipation='Scalar'
and b.id=e.id and b.id=f.id
order by b.Date
```

Moreover, the transactional system is available, together with access control, remote access, etc. (all services which are not the job of the CGNS standard but which are mandatory for a large simulation workbench).

In a large CFD workflow process, we can compare the data life cycle and its associated processes to a PDM²¹ system. Such a system insures version management, dependancies control and actual workflow for data state change. In that case, *metadata* have to be added to the initial CGNS information. Examples include the owner of the file, the confidentiality level, a creation date, or a checksum. There is no agreement about these *metadata* in the CGNS Steering Committee; this topic can be defined in the *profile* we mentioned previously in this paper.

IV. Examples of CGNS usage in the CFD workflow

We are currently in the process of developing a methodology by which projects can apply the practical aspects of a standard like CGNS. Next, we define several specific examples of projects and computer codes that either already use CGNS or else are in the process of switching over to using it. These examples serve to illustrate the above-mentioned benefits of a public standard like CGNS.

A. Turbomachinery: A fan optimisation process

In this case, CGNS is mainly seen as the means to having an *interface specification*. Three separate teams are writing their specifications. The data structure and semantics are well defined and a part of the task can be done as a third party software, with a contract based on the CGNS specification. One can define its expected input, its output, and then “delta” translations can be solved on a standard basis. See Fig. 4. Like any other common formalism, such as UML (*Unified Modelling Language*) for software engineering, the SIDS representation “closes” the specification within the CFD scope, and it can be a basis for relationships with non-CFD contractors. For example, the grid generator passes the CGNS data to a tool that defines boundary conditions. A formal contract, not proprietary, can be accomplished between these two entities with respect to the actual boundary conditions that each particular fluid solver can handle.

In this case the fluid solver is not taking into account dimensional data. This data normalization is left up to pre- and post-processors. The CGNS data structure embeds the dimensional information, ignored by

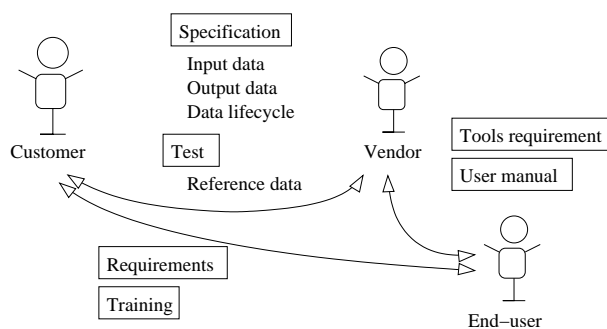


Figure 4. Specification exchange using CGNS

the solver but found and used by the post processor. Codes are adding and using data in a structural and semantic frame, which should be consistent at every stage of the process.

The ISO standard makes it possible to build a long term process. This means that specifications can be very stable and would not be subject to changes. Changing platforms will not change data specification. The stability is possible because the core CFD data that have been defined in CGNS are quite stable. These have been specified by end-users and people with a very broad knowledge of CFD. The parts that could lead to changes are extendable (turbulence models, boundary conditions, etc.).

The customer can define the relationship its platform will have with *the rest of the world*. Using the ISO standard insures that the interfaces of the platform will be stable and available through a public reference.

B. Helicopter: Blade deformation

This example emphasizes the use of CGNS as a *software bus* for code coupling. The current SIDS does not have structure deformation capabilities. However, through the use of the extensions mechanisms of CGNS, it will be possible to eventually extend SIDS with “FFT” data that defines the Fourier data related to the blade deformation. See Fig. 5. Once it becomes an official part of the standard, such data would be 100% CGNS compliant, and would avoid multiple standards for data transfer because a single formalism would be used for the entire CFD code interface.

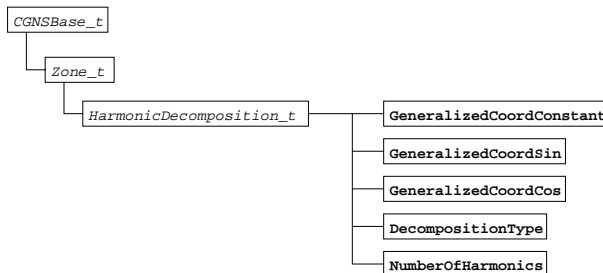


Figure 5. A possible extension dedicated to harmonic decomposition

The helicopter grid is quite large, and does not change with the computation. Therefore, this project makes use of “links,” a mechanism built into CGNS to connect different files and/or to connect different nodes of the same file. It is a way to re-use part of a CGNS file (typically the grid), without having to re-write it again along with every solution file. See Fig. 6. Thus, in the helicopter example, links are used to share the grid from many separate results files. It is worth noting that the link mechanism is a way of referencing one tree from another one, and this type of referencing is a very important step in the concept known as “object identity.” This “object identity” insures that a “real” object, such as a grid, is not duplicated but is actually shared by users.

The current MLL and ADF libraries in CGNS are written using both C and Fortran languages. Code coupling can imply many codes, coming from different teams and with different ways of managing their own

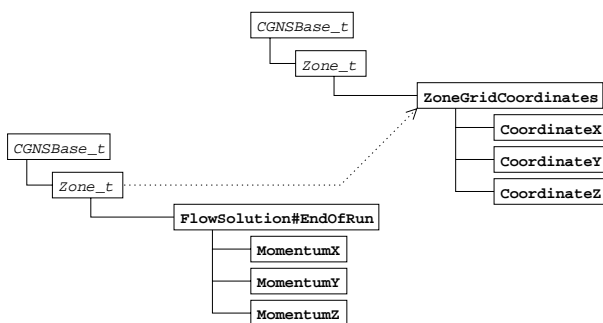


Figure 6. Example use of links

interface. Using MLL or ADF makes it possible for each code to use the API the way they want, while still being compliant.

Several solvers are used for this helicopter blade deformation coupling example. Two of these codes, the *HOST* solver and the *Tecplot* visualizer, are using text files for the communication. These two codes are encapsulated in order to use and provide CGNS binary files, instead of their proprietary format. Then, they can be connected to the software bus, and they can communicate with any other CGNS compliant application. See Fig. 7. The fluid solver uses CGNS as well. Some data have to be archived; these are actual input data or the result data requested by the user. Some other data are temporary data, which are not archived but destroyed at the end of each step. Even these temporary files are CGNS.

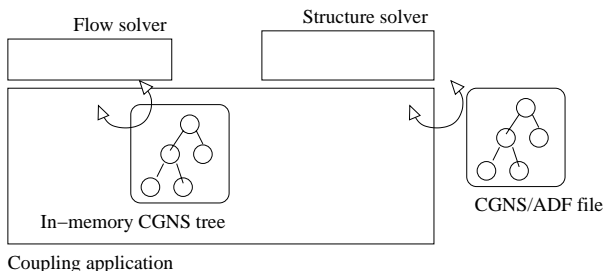


Figure 7. In memory tree exchange

C. CFL3D: General Aerodynamic Flow Solver

CFL3D is a CFD code developed at NASA, and widely used by U.S. industry.²² This code uses CGNS primarily as a long term archival mechanism. It makes use of the mid-level library (MLL) that is an integral part of the CGNS system. Users currently have the option of replacing CFL3D’s code-specific restart file with a CGNS file. There are many potential advantages to using the CGNS file:

1. When CGNS becomes an ISO standard, previously-run cases will already be in CGNS ISO format.
2. Because CGNS files are automatically date-stamped and allow user-defined Descriptor nodes, they are more self-descriptive than their predecessor. For example, in CFL3D, the CGNS files contain copies of all input files as Descriptor nodes, so in the future it will be easy to reconstruct how any particular case was run.
3. CGNS itself contains the (optional) capability to describe the equation sets used, as well as nondimensionalizations, reference states, and convergence histories. CFL3D takes advantage of most available CGNS options, and as a result provides a very complete description of each particular case run, all contained in a single CGNS file.

4. If a postprocessor software is able to read CGNS files, then such software can operate directly on CFL3D's CGNS-based restart file. In other words, it is not necessary to create additional files for use with postprocessing software. One file does it all!

A part of an example tree from a CFL3D file is shown in Fig. 8. At the top layer, the `ConvergenceHistory_t` node contains residual, lift, drag, and moment history information (along with other integrated forces and convergence-type quantities). The `ReferenceState_t` node contains reference quantities such as Mach number, Reynolds number, and general flow field quantities such as density, pressure, and velocities at the reference state. The `Descriptor_t` nodes (which can be unlimited in number and can also appear anywhere in the CGNS tree) can contain the CFL3D input file(s) or any other information desired. Under the `Zone_t` node one finds the nodes most often seen in CGNS files: `GridCoordinates`, `ZoneGridConnectivity`, `ZoneBC`, and `FlowSolution`. Additionally, the figure shows the `FlowEquationSet_t` node, under which resides specific information about the equations solved by CFL3D. This information can be useful to anyone who uses this data, and it also makes the data more archivally complete.

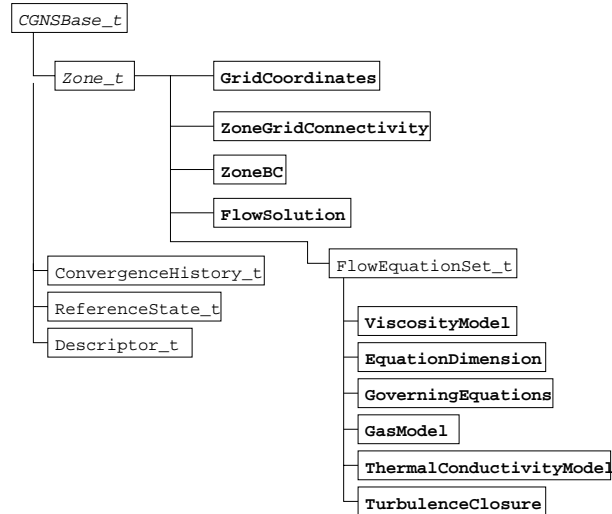


Figure 8. Part of a typical CGNS file structure from CFL3D

An example of a CFD workflow for which CGNS proved to be advantageous was the application of CFL3D to the 2nd AIAA CFD Drag Prediction Workshop.²³ The workshop organizers supplied several grids for use by the participants, including complex multiple-zone grids with 1-to-1 connectivity. For example, the medium-sized wing-body grid contained 27 zones with 103 connectivity patches (logically rectangular regions where one zone connected in a 1-to-1 fashion with another zone) and 223 BC patches. The medium-sized wing-body-nacelle-pylon grid contained 58 zones with 306 connectivity patches and 590 BC patches.

One of the disadvantages of the “usual” type of grid file (such as PLOT3D format, for example), is that only grid points are given. All connectivity and BC information, which was known when the grid was first generated, is lost and must be regenerated subsequently. With CGNS, however, this information is retained and can save a significant amount of time for the problem set-up. For the Drag Prediction Workshop cases, the above-mentioned grids were given in CGNS format, so setting up the cases to run in CFL3D took almost no time at all. Without CGNS, figuring out and re-typing all connectivity and BC information would have taken a long time and also would have introduced a significant probability of introducing errors.

D. Wind

The Wind code,²⁴ developed and supported by the NPARC alliance can utilize CGNS for all internal and external data storage and communication. This allows a single copy of CFD information to be used throughout the design process, eliminating multiple copies, data translation and other data handling bottlenecks.

The CFD process starts with the geometry, usually in IGES format, which is read in to the grid generation program. Thanks to CGNS we are able to use a number of commercial tools such as Gridgen that can be used

to create the grid and output directed into a CGNS file. This file contains the grid and boundary condition information needed by the flow solver. The Wind flow solver is run on this input grid file and writes the solution data into the CGNS file. This same data is also used if the flow solver needs to be restarted, thus is also the restart file. Finally the grid and solution are read into the commercial post-processing tool Fieldview for final analysis. There are a number of commercial post-processing tools that can be used on the CGNS grid and solution files.

The use of CGNS has allowed sharing CFD data between sites in Boeing and government customers and eliminates the deficiencies of the old PLOT3D format. We are still in the process of adding CGNS support to various legacy tools to further streamline the process. It is also being used as the basis of a framework for multi-disciplinary analysis.

V. Conclusions

CGNS is becoming recognized as a world-wide standard for CFD data storage. It is also in the process of becoming an international ISO standard. In this paper, we outlined several practical aspects of using the CGNS standard for CFD applications. First, the importance of having a common data standard in today's complex working environment was pointed out. Then, the methodology of CGNS was described in terms of layers of interoperability. The key layer to CGNS is the SIDS layer, which defines the intellectual content of the file. Lower layers currently utilize ADF, but this storage mechanism can easily be replaced by another (such as HDF5) without compromising the essential aspects inherent to CGNS. Several practical issues were then discussed, including compliance, profiling, and prototyping. Finally, several examples from current projects and codes that use (or plan to use) CGNS were given, in order to illustrate several different specific benefits obtainable by using the standard.

We pointed out that the CGNS standard defines far more than a file format. It has key features that are known in other software areas, and are required for CFD workflow open systems:

1. *Specification.* The specification is CFD information, and the data is well defined. It insures a common reference, with as little ambiguity as possible.
2. *Consistency.* The entire tree-based structure is consistent, and includes the capability to record grids, equations, dimensions, units, BCs, time-dependent data, and a significant amount of other information.
3. *Usability.* The standard comes with software libraries and tools.
4. *Archival.* The storage layer is platform independent and does not assume high level system mechanisms such as parallel or access control.

References

- ¹Stolte, E., Alonso, G., "Efficient Exploration of Large Scientific Databases", Proceedings of the 28th VLDB conference, HongKong, China, 2002
- ²Foster, I., Voeckler, J., Wilde, M., Zhao, Y., "Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation", Proceedings of the 14th Conference on Scientific and Statistical Database Management, Edinburgh, Scotland, July 2002.
- ³<http://www.globus.org>
- ⁴Data Interoperability: Standardization or Mediation, S. Renner, A. Rosenthal, J. Scarano. Renner, S. A., Rosenthal, A. S., Scarano, G. J. "Data Interoperability: Standardization or Mediation", IEEE Metadata Workshop, Silver Spring, MD, April 1996
- ⁵Legensky, S. M., Edwards, D. E., Bush, R. H., Poirier, D. M. A., Rumsey, C. L., Cosner, R. R., and Towne, C. E., "CFD General Notation System (CGNS): Status and Future Directions", AIAA Paper 2002-0752, January 2002.
- ⁶CGNS Team, "The CFD General Notation System, Standard Interface Data Structures", AIAA R-101-2002, 2002.
- ⁷Poirier, D. M. A., Bush, R. H., Cosner, R. R., Rumsey, C. L., McCarthy, D., "Advances in the CGNS Database Standard for Aerodynamics and CFD", AIAA Paper 2000-0681, January 2000.
- ⁸Poirier, D. M. A., Allmaras, S., McCarthy, D., Smith, M., and Enomoto, F., "The CGNS System", AIAA Paper 98-3007, June 1998.
- ⁹Rumsey, C. L., Poirier, D. M. A., Bush, R. H., and Towne, C. E., "A User's Guide to CGNS", NASA/TM-2001-211236, October 2001.
- ¹⁰Keves, B. W., "Open Systems Formal Evaluation Process", USENIX, Systems Administration (LISA VII) Conference, Monterey, CA, November 1993.

- ¹¹Zimmerman, H., "OSI Reference Model—The ISO Model of Architecture for Open Systems Interconnection", IEEE Transactions on Communications, COM-28(4): 425-432, April 1980.
- ¹²CGNS Team, "The ADF User's Guide", May 1997.
- ¹³Abrial, J.-R., Z ref. "Data Semantics", IFIP TC2 Working Conference on Data Base Management, 1974
- ¹⁴PCTE interfaces: Supporting tools in software-engineering environments. Ian Thomas. IEEE Software, 6(6):15-23, November 1989.
- ¹⁵ECMA, "The meaning of conformance to standards", ECMA TR/18, September 1983.
- ¹⁶<http://hdf.ncsa.uiuc.edu/HDF5>
- ¹⁷<http://www.relaxng.org>
- ¹⁸Cambier, L., Gazaix, M., "elsA : An Efficient Object-Oriented Solution to CFD Complexity", AIAA Paper 2002-0108, January 2002.
- ¹⁹Poinot, M., PyCGNS User Manual, v2.0, Feb. 2004, <http://elsa.onera.fr/CGNS/releases>
- ²⁰<http://www.cscs.ch/projects/ESTEDI/results.html>
- ²¹Estublier, J., Favre, J.-M., Morat, P., Toward SCM/PDM Integration?, SCM8, Brussels, July 20th - 21st, 1998. In LNCS 1439, Springer Verlag.
- ²²Krist S. L., Biedron R. T., and Rumsey C. L., "CFL3D User's Manual (Version 5.0)", NASA TM-1998-208444, June 1998.
- ²³<http://ad-www.larc.nasa.gov/tsab/cfdlarc/aiaa-dpw/index.html>
- ²⁴Nelson, C. C. and Power, G. D., "CHSSI Project CFD-7: The NPARC Alliance Flow Simulation System," AIAA Paper 2001-0594, January 2001.