Guide to Python/CGNS

Marc Poinot
ONERA/DSNA/CS2A

pyCGNS v4.0.1
NAV v1.0.0
June 2009

ONERA
BP 72, 29 Avenue de la Division Leclerc
F-92322 Châtillon Cedex
FRANCE

http://www.onera.fr

FOREWORDS

The pyCGNS Python package is a set of tools for the CGNS standard. The tools are all released in the pyCGNS package but you can install and use each tool in a separate way, as far as dependancies for this tool allow you to do so.

CGNS.WRA	**Wrapper** A Python wrap to so-called midlevel library and adf library. No dependancy to other pyCGNS modules, but it requires the CGNS and HDF5 libraries. *Formerly was pyCGNS.MLL and pyCGNS.ADF*
CGNS.MAP	**Mapper** Implements the SIDS-to-python mapping. No dependancy to other pyCGNS modules, but it requires the HDF5 library. *Formerly was CGNS.utils.saveAsADF and loadAsADF*
CGNS.PAT	**PatterMaker** A set of functions to create or update complete SIDS sub-trees. No dependancy to other pyCGNS modules. *Formerly was pyCGNS.cgnslib*
CGNS.NAV	**Navigater** A Tk based GUI for Python/CGNS trees. Requires CGNS.MAP, CGNS.PAT, TkTreeControl and TkinterTreectrl. *Formerly was pyS7*
CGNS.VAL	**Validater** A customizable Python/CGNS tree parser to check tree compliance to SIDS or to user-defined rules. Requires CGNS.MAP *Formerly was pyC5*
CGNS.DAT	**DataTracer** A RDBMS-based application to store and manage set of CGNS files. Requires CGNS.MAP and a RDBMS. *Formerly was pyDAX*
CGNS.APP	**Applicater** A set of examples, libraries, tools and demos. Requires CGNS.MAP and CGNS.PAT *Formerly was pyCRAB*

CGNS Standard

The Computational Fluid Dynamics community has a standard for data model definition and disk storage: CGNS (CFD General Notation System), it is an AIAA recommanded practice and it is in the process of being an ISO standard in the frame of ISO/STEP.

> △ In this document, we refer to the so-called official CGNS documents and libraries when we use the .org suffix. For example, the CGNS.org documentation means '*the official CGNS documentation or library or tools you can find on the web site www.cgns.org*'.

The reference document for CGNS is the so-called SIDS, which stands for *Standard Interface Data Structure*. The reference version is described in the document AIAA-R101A-2006 publicly available on the CGNS web site.

About this document

The *pyCGNS-NAV* manual is, at the same time, a reference document and a users' guide. As we are supporting the CGNS standard, many informations related to the standard are not detailled here. The user should read the CGNS/SIDS to have more complete information about the CGNS data structures.

Each pyCGNS module has its own documentation, their may refer to each other.

The required modules and library versions, used by pyCGNS, can be detected at run-time using the pyCGNSconfig module. These are updated during the installation process.

1 QUICK START

The CGNS.NAV tool is a CGNS/Python tree browser. See the CGNS.MAP documentation to know about the CGNS/Python mapping. CGNS.NAV can browse CGNS/HDF5, CGNS/ADF and CGNS/Python trees but in all cases the data on the disk is translated as CGNS/Python and all CGNS.NAV operation are using CGNS/Python trees. In the examples we are using here, the files are HDF5, ADF and Python on the disk, the archival type is either selected when you open a file or when you save it.

To start CGNS.NAV, just type CGNS.NAV in your shell command line. Please refer to the TroubleShootings section of this document if this command fails.

1.1 Control window

CGNS.NAV has a set of main features which have their own view. For exemple, the *Tree view* is the CGNS/Python tree browsing window, the *Operate view* gathers services about querying a tree, the *Pattern view* shows the existing CGNS/SIDS patterns... Most of these views are associated to a CGNS/Python tree or even a sub-tree. This could lead to a large number or views on the screen, the control window is here to help you to manage all these windows.

When you start CGNS.NAV without any command line argument, the control window appears alone. The control window shows all the views you have on all the trees CGNS.NAV has read. It is the main window of CGNS.NAV, you have it when you start and you use it to leave CGNS.NAV and close all exsiting CGNS.NAV windows.

The menu bar contains only icons, the main part of the control window is the list of views, which is empty in that case. The action you can perform on the control view are:

Create a new tree, opens an empty Tree view where you are supposed to create a new tree from scratch or by copying parts of existing trees.

Open an existing tree, a file selection window is open, a new tree view is open with your selected tree. The way the tree is actually parsed depends on the options you have set.

Opens the pattern selection view. You can use existing patterns (see also CGNS.PAT) or you can create your own set of patterns.

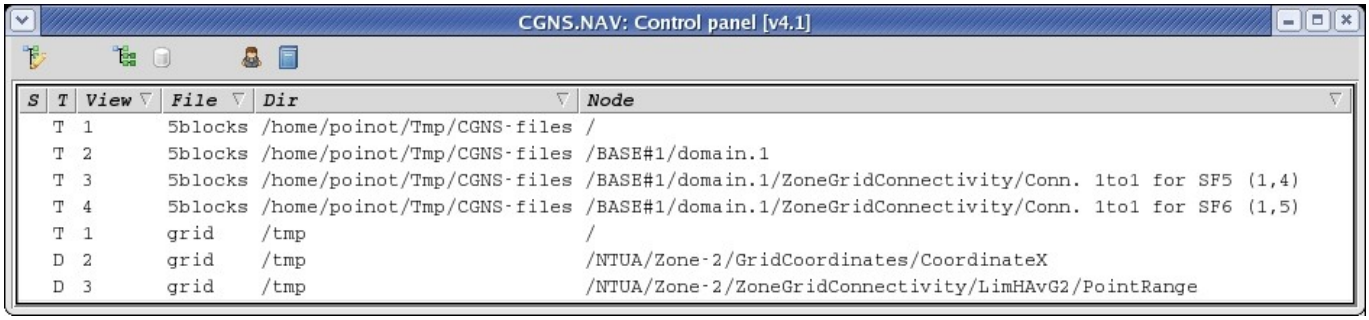
The user options view. The options window is used to set some specific CGNS.NAV behaviour, such as 'open recursively a tree' or 'follow links'...

A little book which stands for help. The help starts a web browser on these document you are reading now.

The first step is to open or create a tree. The *Control view* shows the views you have: a *Tree view* on a Python/CGNS tree or a Data view on a specific node (a `DataArray_t` node most of the time). We suppose now you have such views and we see how to manage them.

1.2 View list

The Control view main window gives an id for each Tree or Data view you have. The list is ordered using this id, this is the first integer.



1.3 Command line options

2 TREE VIEW

The Tree view is the most used view.

2.1 View overview

.

3 TREE VIEW

The Tree view is the most used view.

3.1 View overview

.Tree view

The Tree view is the most used view.

4 OPERATE VIEW

NAV uses CGNS/Python trees and proposes to operate Python actions on these.

4.1 Single file translation

A single file is required.

4.2 Python and numpy API remarks

The numpy API should be initialized before any use.

4.3 Embedding SIDStoPython

Your application can add the SIDStoPython file.