pyCGNS.intro/Manual

Release 4.0.1

Marc Poinot

December 01, 2010

CONTENTS

pyCGNS is a Python package for the CGNS standard. The package gathers various tools and libraries for end-users and Python application developpers. The main object of pyCGNS is to provide the application developpers with a Python interface to *CGNS/SIDS*, the data model. The *MAP* and *PAT* modules are dedicated to this goal: map the *CGNS/SIDS* data model to a Python implementation. The *WRA* module is a wrapper on *CGNS/MLL* and is not the important module of pyCGNS. One could achieve such a wrapper using SWIG for example. This is the reason why the use of *CGNS/MLL* is optional.

The *CGNS/SIDS* data model has a mapping the *HDF5* for file storage. The *MAP* module uses CHLone instead of *CGNS/MLL* to map its *CGNS/Python* trees to *HDF5*.

The package also uses numpy and hdf5 you should install before pyCGNS.

CHAPTER

ONE

CONTENTS

1.1 About pyCGNS

1.1.1 Package contents

The pyCGNS Python module is a collection of 7 modules around the CGNS standard. Before v4, these modules were independent Python modules with more or less dependancies to each other. We gather all of them to have a common build/install/doc and test process, moreover this insures a better consistency between them.

The pyCGNS module now includes (former package names)

MAP, the Mapper, new in v4 gives basic load/save function from/to *CGNS/SIDS* and *CGNS/HDF5*. This very sinple module is able to read/write GCNS/HDF5 files and translate them to CGNS/Python. This is the main feature of pyCGNS v4.0.

PAT, the PatterMaker, a full *CGNS/SIDS* patterns using the *CGNS/Python* mapping. This is pure python module, it creates and modify CGNS/Python trees without the help of any HDF5 or even ADf calls.

NAV, the Navigater (pyS7), a graphical browser that can handle *CGNS/Python*, *CGNS/HDF5* and *CGNS/ADF* file formats. It is slightly different to *adfviewer* because it actually a tree editor, you can copy/cut/paste CGNS/Python trees and quickly draft or modify your CGNS tree.

WRA, the Wrapper (pyCGNS), is a *CGNS/MLL* and *CGNS/ADF* Python wrapping. All the CGNS/MLL functions are mapped to their Python clone.

VAL,the Validater (pyC5), an XML grammar based validation of a *CGNS/Python* tree, for example produced using *MAP* or *PAT*. *Unsuable in v4.0*

TRA, the Translater (pyCRAB), a set of translators from/to various formats. *Unsuable in v4.0*

DAT, the DataTracer (pyDAX), some DBMS services for *CGNS/HDF5* files. *Unsuable in v4.0*

1.1.2 Quick start

Loading a CGNS/HDF file with MAP

The *CGNS.MAP* module implements the *CGNS/Python* mapping. You can load/save a *CGNS/HDF5* file using the simple *MAP* functions (below, the >>> string is the python interpreter prompt):

```
 >>>import CGNS.MAP
 >>>(tree,links)=CGNS.MAP.load("./001Disk.hdf",CGNS.MAP.S2P_FOLLOWLINKS)
 >>>print tree
 ['CGNSTree', None, [['CGNSLibraryVersion',array([ 2.4000001],dtype=float32),
 [], 'CGNSLibraryVersion_t'], ['Disk', array([3, 3], dtype=int32),
[['.Solver#Command', ...
```

Now `tree` is a Python list with the whole‘‘ ./001Disk.hdf‘‘ CGNS tree into, with the data structure as described in *SIDS-to-Python*.

Using PAT to modify a CGNS tree

The previously loaded *CGNS/Python* tree is modified using plain Python functions and types. The *CGNS.APP* module contains utilities, we use the `getNodeByPath` function which returns a *CGNS/Python* node with the target tree and the target node path as parameters.:

```
import CGNS.APP.path_utils as U

node=U.getNodeByPath("/Disk/zone1/ZoneBC/ext1/PointRange",tree)
```

The returned node is a list of 4 Python values, the name (a string), the value of the node (a *Numpy* array), the list of children and the CGNS type of the node (string).

Using PAT to create a CGNS tree

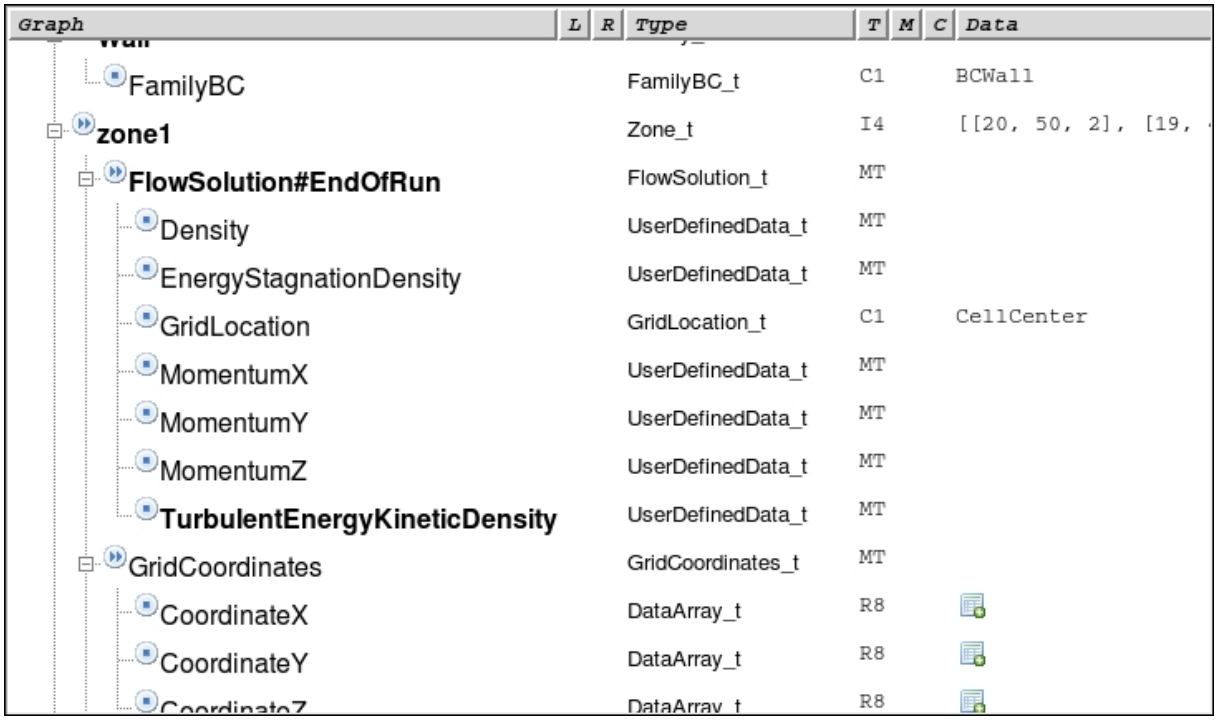
We want to create a *CGNS/HDF5* file with a simple *base* and a *reference state*:

```
import CGNS.MAP
import CGNS.PAT

T=CGNS.PAT.newCGNS()
CGNS.PAT.newBase(T,'Test Case 01',3,3) # name, physical dim, topological dim
CGNS.PAT.newSimulationType(T)
CGNS.PAT.newReferenceState(T)
```

Browsing your CGNS tree with NAV

The CGNS.NAV tool is a CGNS tree brower. You start it typing `CGNS.NAV` and you can open several views on your file.



Re-using your CGNS/MLL scripts

If you really want to use CGNS/MLL, for example if you have a large toolbox with old pyCGNS scripts, you can import CGNS.WRA but you have to change your imports:

```
import CGNS.WRA.wrapper

filesol=CGNS.WRA.wrapper.pyCGNS(sname,CGNS.WRA.wrapper.MODE_WRITE)
filesol.basewrite('Base',3,3)
filesol.close()
```

1.2 Build and Install

1.2.1 Required libraries

The first step of the installation is to make sure you have the required libraries. The mandatory libs are *Python*, *numpy*, *HDF5* and *CHLone*. Then, if you wan to build *CGNS.WRA* (we recommend to do so), you need *libcgns*.

Warning: **OUPS !** you mean I don't need *libcgns* for the *CGNS/Python* mapping ?
NO you don't, *CGNS* is a data model (so-called *CGNS/SIDS*) and some mapping definitions of this model (such as *CGNS/HDF* for example). *pyCGNS* uses *'CHLone <http://chlone.sourceforge.net>'_* which is another *CGNS/HDF5* compliant implementation.

- Python (starting from v2.4)
- numpy (v1.1 +)
- hdf5 (v1.8.5 +)
- CHLone (v0.4 +)
- tktreectrl (v2.2 +)

1.2.2 Optional libraries

The so-called *mid-level* library is not mandatory, the *WRA module* is the only one to have dependancies on. The *NAV module* also uses *CGNS/MLL* as optional if you want to be able to read *CGNS/ADF* files (see *File formats*)

- CGNS/MLL (libcgns) (starting v3.0)

1.2.3 Installation process

Once you have these installed you can proceed with pyCGNS. You go into the top directory and you edit the `pyCGNSconfig.py.in` (see *Configuration file contents*). You have to set the correct paths and various values such as directory search libs or flags. Then you run:

```
python setup.py build
```

and then:

```
python setup.py install
```

or:

```
python setup.py install --prefix=/local/tools/installation
```

All the modules of the pyCGNS package are installed and you can now proceed with tutorial examples.

1.2.4 Single module installation

You can ask for a single module installation:

```
python setup.py build --single-module=MAP
python setup.py install
```

You have to check that this installation doesn't overwrite an existing installation with the other pyCGNS modules.

1.2.5 Configuration file contents

The `pyCGNSconfig_user.py` should work with no modification if you have a standard installation. All you have to declare is the directory in which we can find *Python/numpy/hdf5/CHLone/cgns* libraries.

If you have specific installations you can change some paths/flags for each external library: *hdf5*, *numpy*, *CGNS/MLL* and *CHLone*. The configuration file is a Python file, it is imported after the default configuration. The changes you make in the configuration file will overwrite the defaults:

```
# --- stuff to add for HDF5

#HDF5_VERSION          = ''
HDF5_PATH_INCLUDES    = ['/home/myself/hdf5/include']
HDF5_PATH_LIBRARIES   = ['/home/myself/hdf5/lib']
#HDF5_LINK_LIBRARIES   = []
#HDF5_EXTRA_ARGS       = []
```

To avoid overwriting, use Python to update the config:

```
# --- stuff to add for HDF5

#HDF5_VERSION          = ''
HDF5_PATH_INCLUDES    = ['/home/myself/hdf5/include']
HDF5_PATH_LIBRARIES   = ['/home/myself/hdf5/lib']
#HDF5_LINK_LIBRARIES   = []
HDF5_EXTRA_ARGS       = HDF5_EXTRA_ARGS + ['-DMYFLAG']
```

Release Notes

> Many changes in this v4 release, you can only use MAP, WRA, PAT and NAV. The other modules, VAL, TRA and DAT are present for archival/development purpose but you should NOT use them.
>
> Please go to *http://www.python-science.org/projects/pyCGNS* to have the version/tickets list.

Module dependancies

The pyCGNS modules have dependancies with their brothers. The list below gives you the required modules (or optional) for each of them.

- MAP : None
- PAT : MAP
- WRA : PAT MAP
- APP : PAT MAP
- NAV : PAT MAP APP (WRA)

1.2.6 NAV depends

The *TkTreectrl* module is required. You first need to install *tktreectrl* (last version tested is *tktreectrl-2.2.3*) and *TkinterTreectrl* to map it to Python (last version tested is *TkinterTreectrl-1.0*). You may have to add the *tktreectrl* path into a *TCLLIBPATH* shell variable (maybe you should add a *LD_LIBRARY_PATH* as well).

1.2.7 MAP depends

The *CHLone* librarie is required and thus *HDF5* is required.

1.2.8 WRA depends

CGNS/MLL and *CGNS/ADF* libraries are required.

- *MAP Index*
- *PAT Index*
- *NAV Index*
- *WRA Index*
- *search*