
# pyCGNS.intro/Manual

***Release 4.0.1***

**Marc Poinot**

September 29, 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 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)

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

TRA, the Translater (pyCRAB), a set of translators from/to various formats.

**MAP, the Mapper, new in v4 gives basic load/save** function from/to *CGNS/SIDS* and *CGNS/HDF5*.

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

**WRA, the Wrapper (pyCGNS), is a *CGNS/MLL* and** *CGNS/ADF* Python wrapping.

**PAT, the PatterMaker, a full *CGNS/SIDS* patterns using** the *CGNS/Python* mapping.

**NAV, the Navigater (pyS7), a graphical browser that can** handle *CGNS/Python*, *CGNS/HDF5* and *CGNS/ADF* file formats.

## 1.2 Quick start

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)
```

The pyCGNS python package module now includes:

## 1.3 MAPper

MAP (new in v4)

The implementation of the SIDS-to-Python mapping. Provides functions for load/save SIDS/HDF5 from/to CGNStree.py

## 1.4 WRApper

WRA (formerly was pyCGNS)

The Python wrapper for ADF and MLL libraries.

## 1.5 PATternMaker

PAT (formerly was pyCGNS/cgnslib)

A set of functions to create/read/write/modify CGNS/SIDS patterns compilants with SIDS-to-Python.

## 1.6 NAVigater

NAV (formerly was pyS7)

A graphical browser for CGNS trees (ADF/HDF5/Python). Allows tree construction by means of copy/paste and patterns.

## 1.7 DATatracer

DAT (formerly was pyDAX)

A set of command-line tools that use DBMS services for SIDS/HDF5 files.

**Warning:** This module is experimental and should NOT be used for your applications.

## 1.8 VALidater

VAL (formerly was pyC5)

It is a comand-line tool for CGNS tree verification. It uses an XML grammar.

**Warning:** This module is experimental and should NOT be used for your applications.

## 1.9 TRAnslater

TRA (formerly was pyCRAB)

It is a set translators from/to various formats, these can be used as functions in your own application or as command-line tools.

**Warning:** This module is experimental and should NOT be used for your applications.

## 1.10 Build and Install

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, depending on what you want to build and istall in *pyCGNS*, you may 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* which is another *CGNS/HDF5* compliant implementation.

### 1.10.1 Required libraries

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

### 1.10.2 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)

### 1.10.3 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.

### 1.10.4 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.

### 1.10.5 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']
```

## 1.11 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.

### 1.11.1 Bug fixes

### 1.11.2 Todo

## 1.12 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
- NAV : PAT MAP (WRA)

### 1.12.1 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-0.8)

### 1.12.2 MAP depends

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

### 1.12.3 WRA depends

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

- *genindex*
- *search*