# Design of a potential new Python HLI

## Design of POC: IMASPy

K.L. van de Plassche[1]

[1] *DIFFER, PO Box 6336, 5600 HH Eindhoven, The Netherlands*

June 24, 2020

## Underlying rational for this design

- As in IMAS-3087
  - The UAL Python interface should be used by non-experts in IMAS, but experts in Python
  - Wrapping the UAL Python interface should be as thin as possible (a la IMASviz), not completely re-worked (a la OMAS)
  - We want the IMAS Python API as first level entrance into IMAS for Python users
- Which opens the questions (end of presentation)
  - IF we go to thinner wrapping, who will change the wrapping codes? (IMASViz, pyAL, OMAS, H&CD workflow, JINTRAC workflow)
  - IF we go to multi-dev, who is going to support? (WPCD, CPT, ITER, community)
  - IF we go to a redesign who will be responsible? (The one redesigning? Some common group?)

## Leading design choices: Extendability and openness

As ’going to multi-dev’ and ’attract (python) devs’ were main points, following design guidelines were taken:

- Easy to install; no imas installation needed
- Pythonic; Object-oriented design
- Verbose; Give verbose feedback to API consumer
- Maintainable; Lightweight, not-so-fancy codebase
- Unofficial version; Design as drop-in, do not break existing API

As there is no API specification so:

- Use AL user-guide as specification
- Use IMASViz and H&CD workflow as leading use-cases

## Caveat: Explicitly not taken into account

Caveat: This is an outsider, Python-developer view, so:

- No similarity to other (Fortran, cpp) interfaces
- Move from code-gen to dynamic-structures
- Do not take into account non-python devs might maintain this

These might increase maintainer burden!

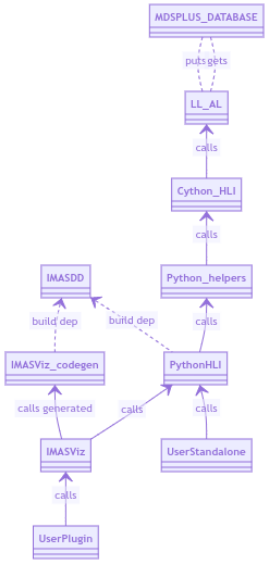
## Extra wishes by myself

The following decisions are not leading, but used if they don't conflict

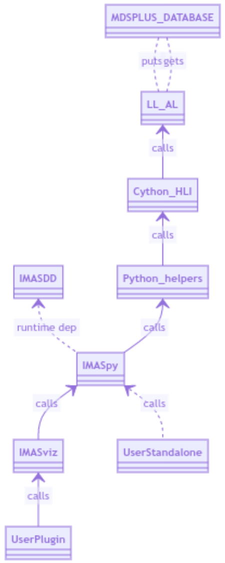
- Stay close to Data Dictionary; use XML directly, no conversion of format
- Stay lightweight; As little dependencies as possible
- Mirror internal structure of Python HLI; Even non-user-facing API should look similar
- Reuse as much as possible existing codebase

## An outsider understanding: IMASViz

Now:

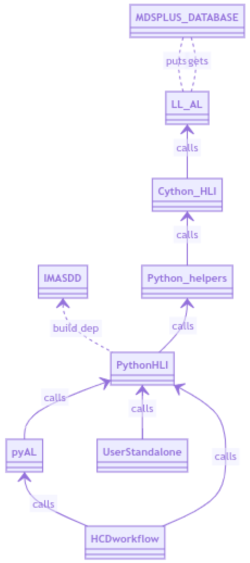


Plan:

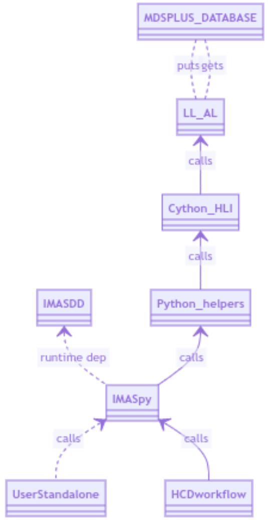


# An outsider understanding: H&CD workflow

Now:



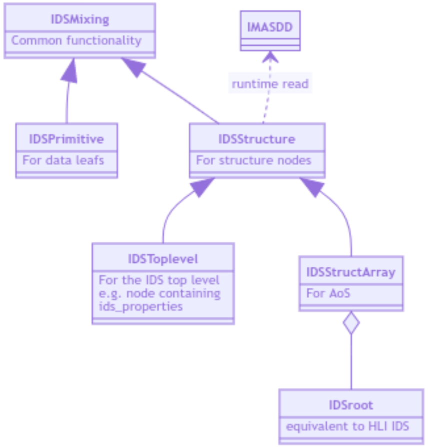
Plan:



## IMASPy design: structure (user facing)

Decouple *Structure* from *Data Dictionary* by recursive design, read XML at

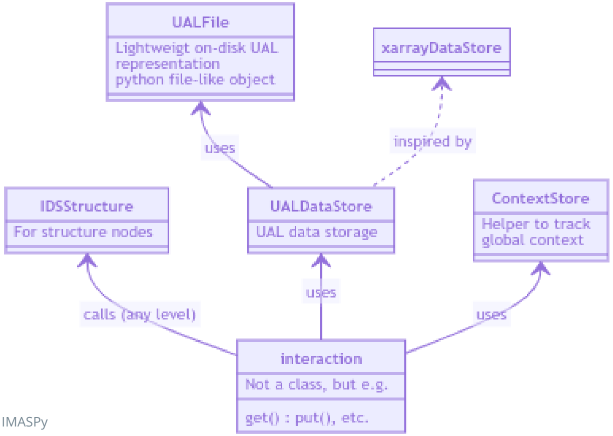
```
imas_entry = imas.ids(shot, run_in, xml_path=idsdef, verbosity=2)
```



# IMASPy design: Access layer

Decouple *Structure* from *AL* by recursive design, connect to `AL-LL` at

`imas_entry = imas.ids(shot, run_in, ual_version='4.8.0')`



## IMASPy design: Installing

- IMASPy can be installed standalone, no IMAS needed
- To have the DD, need an `IMASDef.xml` during runtime (in some accessible folder)
- To interact with AL, need at install time (or in some publicly accessible folder)
  - Need access to AL sources at install time, e.g. `git.iter.org/imas/access-layer.git`
  - Need access to compilers and possible Cython at install time

## To discuss for next steps

- CPT is interested in this design, but not able to support. Will keep close view and possible switch support from old HLI to IMASPy if users demand it
- IO is interested in this design, but with support on voluntary basis (thanks O. Hoenen and S. Pinches!)

So in the long term:

- I made this in my free time, who wants to join?
- IMASPy is 'drop-in' designed, who is willing to try? (note: Only small set of features implemented yet!)

Practically:

- As not supported by CPT, IMASPy will exist in parallel to Python HLI
- Get started here: `https://gitlab.com/imaspy-dev/imaspy`

## Backup