Integration of the source code in IMAS

Dmitriy Yadykin, Bartek Palak and Poznan ACH team

This work has been carried out within the framework of the EUROfusion Consortium and has received funding from the Euratom research and training programme 2014-2018 under grant agreement No 633053. The views and opinions expressed herein do not necessarily reflect those of the European Commission.

Different levels of integration

Two major levels of the source code integration in the IMAS infrastructure are foreseen (some deviations could of course appear)

- **integrate the code limiting the integration to the I/O data management**
- integrate the code to connect it to the other codes (workflow structures)

Two main integration platforms are foreseen/supported:

- Docker (take-away option)
- **Gateway** (centralized option)

General integration strategy is the same for both platforms, practical details could be different.

Integration check list

- Code runs standalone
- Environment is configured for the code to use IMAS
- Code can communicate with IDSs
- Code internal variables are converted to/from IDS format
- Code runs in the IMAS framework

Code runs standalone

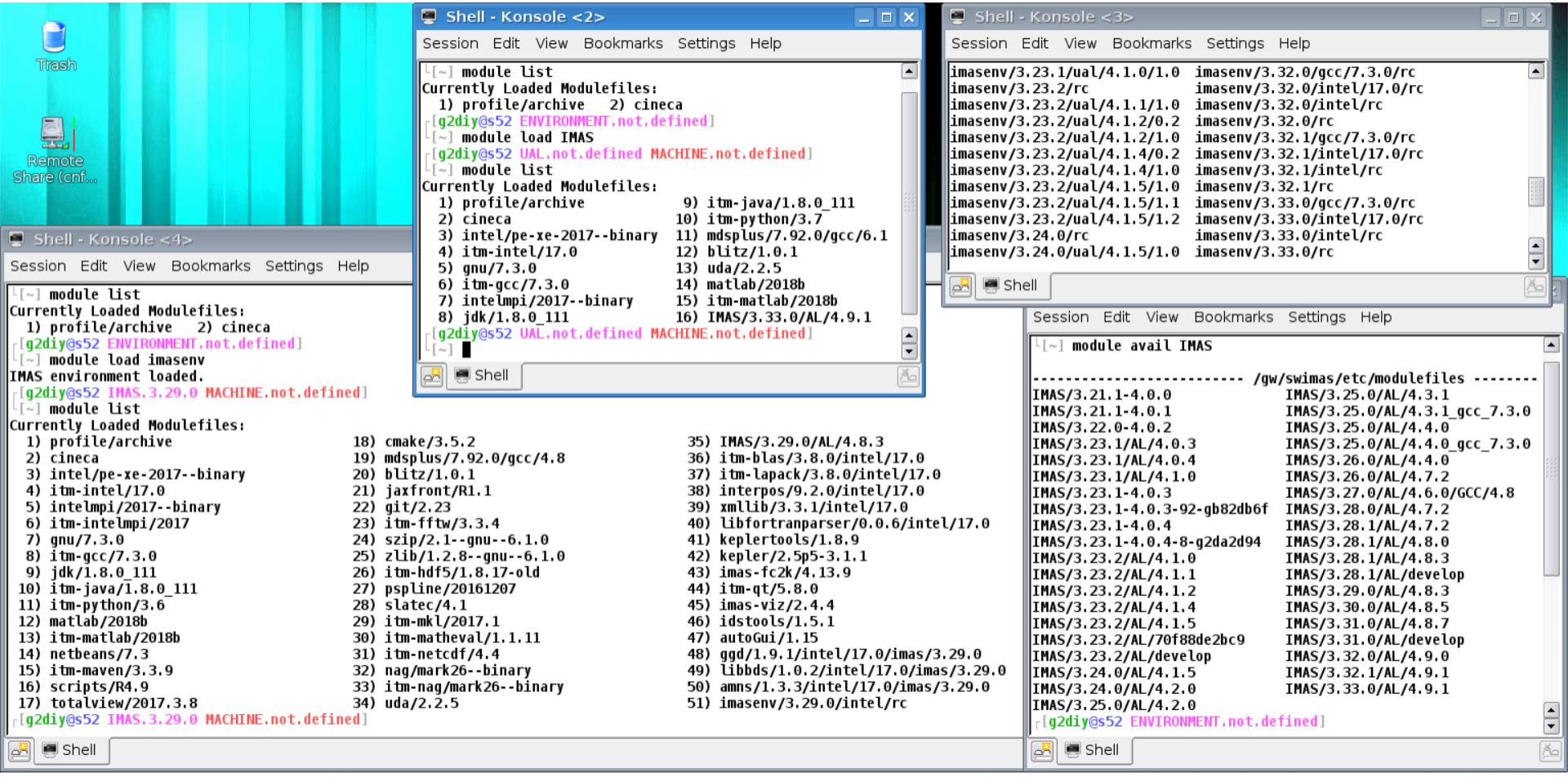
- Code can be compiled and executed using available compiler options
- All libraries required by the code are available and can be loaded (via modules)

Software resources available on the Gateway:
https://wiki.eufus.eu/doku.php?id=namespace:software_resources

Environment is configured

- Preconfigured set of modules (**imasenv**) exists that includes all frequently used modules/libraries
  - imasenv is coupled to/based on the particular IMAS module
  - imasenv exists in two ‘complier oriented’ versions: gnu and intel
  - imasenv includes both dependent on IMAS and independent on IMAS modules
- Modules that are not listed in imasenv can be loaded manually (if they are not conflicting with imasenv modules)
- User’s data structure should be configured beforehand in order to be able to read/write data in IDS format.

Examples of the environment modules



Code can communicate with IDSs

- Data stored in IDSs and to be read by the code or data produced by the code and to be put in IDSs are managed by the High Level Interface (HLI) routines (more on this see the tutorial from Monday: https://docs.psnc.pl/pages/viewpage.action?pageId=70879101).
- Communication chain (open->get->(process)->create ->put->close) can be developed by the user or can be generated automatically in IWrap
- It is recommended to separate communication routine (wrapper) if created by the user from the main body of the code
- For the examples of the wrapper routines in different languages see https://docs.psnc.pl/pages/viewpage.action?pageId=24088283

Internal variables are converted

- There are number of conventions used in IDSs that should be taken into account when converting the internal code variables; one example is the COCOS number used (11) but there are more
- Conventions are explicitly stated in the Data Dictionary documentation (reachable from the Gateway by typing ***dd_doc*** in the command line after configuring environment)
- Time handling of the dynamical quantities in IDS can be done differently: homogeneous time (1) vs inhomogeneous time (0)
- ‘Conventional’ grid to store data is ‘flux’ grid (with rho_tor_norm used as the ‘base’), but options available to store data on complex grid (general grid description)
- Conversion process is not easily generalized, and need to be perfomed in ‘code by code’ fasion

Code runs in IMAS framework

- Relevant modules are loaded in the interface routines (wrapper, converter)
  - python - *import imas*
  - Fortran - *use ids_shemas*, *use ids_routines*
  - C++ - *include UALClasses.h*
- In case Makefile is used to compile the source code (Fortran, C++), it should be updated to include relevant options (in both compilation and linkage steps). It is strongly recommended to use pkg-config tool for this

```
F90=ifort
COPTS = -g `pkg-config imas-ifort --cflags`
LIBS=`pkg-config imas-ifort --libs`
```

```
CXX=g++
COPTS = -g `pkg-config imas-cpp --cflags`
LIBS =`pkg-config imas-cpp --libs`
```