EFDA

EUROPEAN FUSION DEVELOPMENT AGREEMENT

Task Force
INTEGRATED TOKAMAK MODELLING

23/04/2009

ISIP tools training

F. Imbeaux for the ISIP team

TF Leader : P. Strand,
Deputies: L-G. Eriksson, R. Coelho, M. Romanelli

EFDA CSU Contact Person: D. Kalupin

https://portal.efda-itm.eu/portal/authsec/portal/itm/ISIP
isip@mail.efda-itm.eu

Introduction

- Thank you for coming
- Objectives of the session :
  - Understand the various ISIP tools
  - Practice them on the Gateway
  - Answer to questions / help
- This session has been organised for the ITM users, we try to make it as interactive as possible

Practical informations

- Lunch :
  - we are going to the cantine by bus at 12h40
  - coming back by bus at 13h30
  - Stéphane has a card to pay for your lunch, group with him before paying
- Going back to Aix : use buses at 18h20 Monday and Tuesday, 16h20 on Wednesday
- Tuesday and Wednesday : since you have your pass :
  - Pass through control
  - Take first bus in the line, goes directly to IRFM

Expected schedule (1)

- Challenge : participants have heterogeneous level
  - First introduction to the ITM world
  - Experienced users who are waiting for Kepler improvements
- Monday morning : slides
  - Welcome and Overview
  - Data organisation : CPOs, public and private data storage
  - Data communication : Universal Access Layer
  - Kepler : main features
- Monday afternoon
  - Connection to Gateway, brief description of the Portal
  - Creating private data storage
  - Working with test programs and the UAL
  - People who feel that they already know all this can have individual chats with Jacqueline, Philippe, Bernard and myself

Expected schedule (2)

- Tuesday : KEPLER practice + interactive practice / discussions
- Wednesday :
  - Experimental data chain
  - Interactive practice/discussions
  - Feedback on the ISIP tools : perspectives and to do's

ISIP website

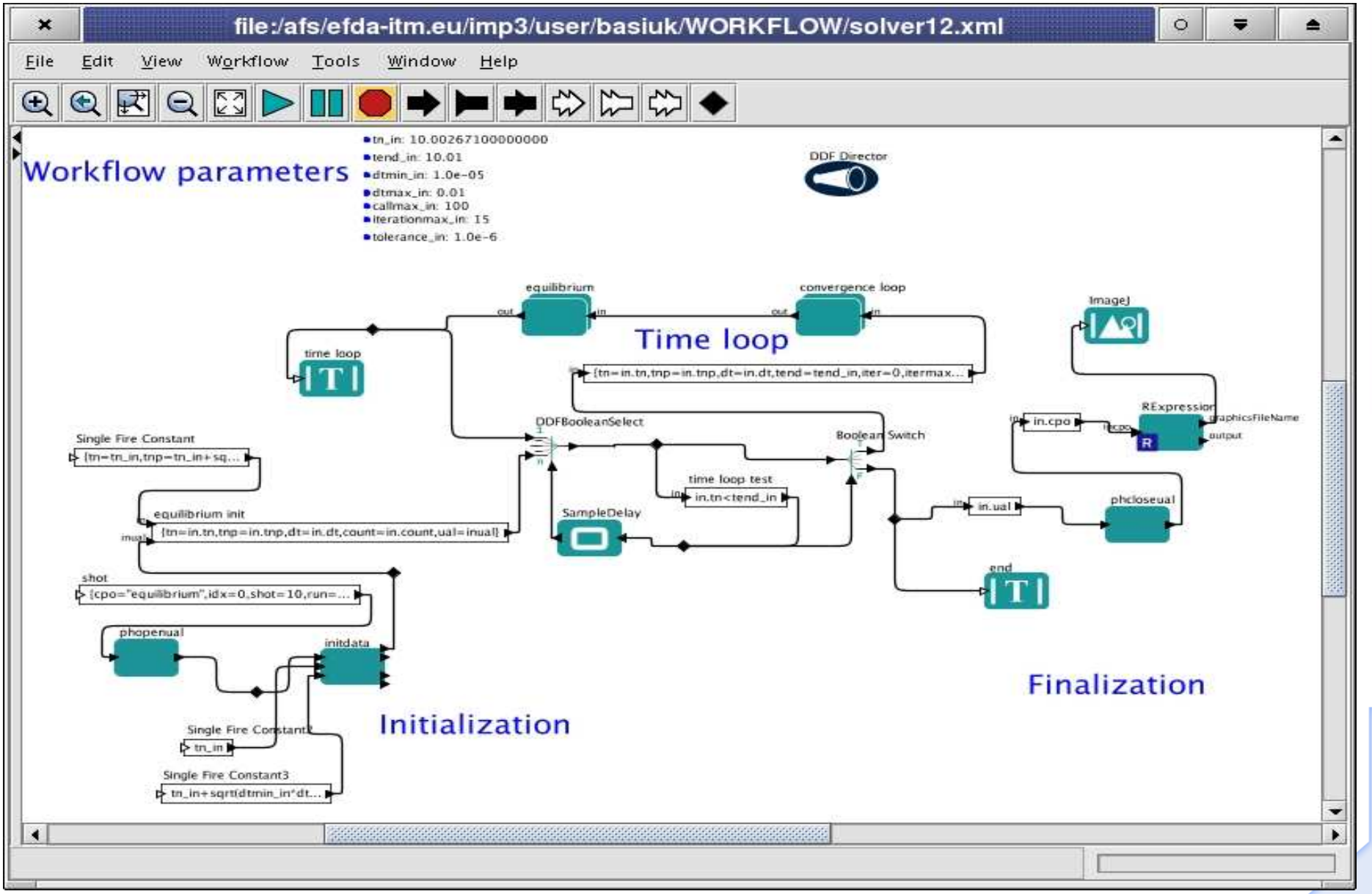
- The ISIP web page contains useful documentation and examples
- https://portal.efda-itm.eu/portal/authsec/portal/itm/ISIP
- We are trying to set up a Gforge-based hotline for ITM users
- Meanwhile, email to isip@mail.efda-itm.eu when you face a problem

Overview of the tools

Overview

- Data Structure : Physics data organised in Consistent Physical Objects (CPO) : standard of communication between physics modules. Thought in view of modular and flexible workflow design
  - coded as XML schemas, a series of XSL transforms turn them into HTML documentation, language specific libraries, machine description file, …
- UAL : Communication library between physics modules : allows to exchange CPOs
  - Available in Fortran, C++, Java, Matlab, Python
  - Generated dynamically from the data structure
  - Uses MDS+ as backend (HDF5 option exists but through MDS+ software)
  - Read/Write from/to disk (default) or memory
- KEPLER : design and runs workflows, with wrapped physics modules as elementary "acting units"
  - Integrated platform developed in San Diego, used by some of the US FSP
  - Drag and drop physical modules « actors » to create workflows

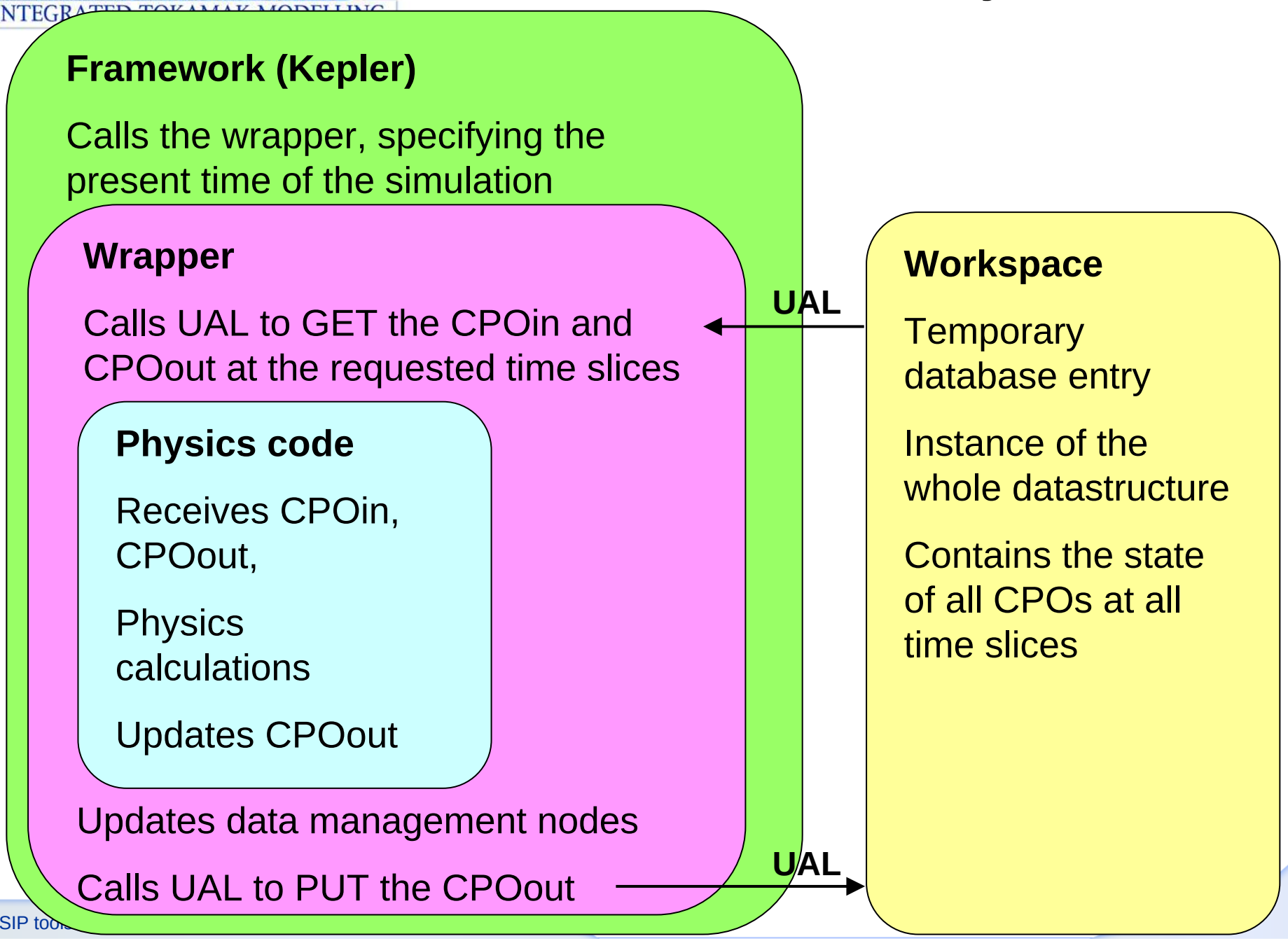
KEPLER


file:/afs/efda-itm.eu/imp3/user/basiuk/WORKFLOW/solver12.xml
File Edit View Workflow Tools Window Help
Workflow parameters
tn_in: 10.002671000000000
tend_in: 10.01
dtmin_in: 1.0e-05
dtmax_in: 0.01
callmax_in: 100
iterationmax_in: 15
tolerance_in: 1.0e-6
DDF Director
equilibrium
convergence loop
Time loop
time loop
{tn=in.tn,tnp=in.tnp,dt=in.dt,tend=tend_in,iter=0,itermax...
ImageJ
RExpression
graphicsFileName
output
in.cpo
DDFBooleanSelect
Boolean Switch
Single Fire Constant
{tn=tn_in,tnp=tn_in+sq...
time loop test
in.tn<tend_in
equilibrium init
{tn=in.tn,tnp=in.tnp,dt=in.dt,count=in.count,ual=inual}
SampleDelay
in.ual
phcloseual
shot
{cpo="equilibrium",idx=0,shot=10,run=...
end
phopenual
initdata
Finalization
Initialization
Single Fire Constant1
tn_in
Single Fire Constant3
tn_in+sqrt(dtmin_in*dt...

CPO transfer in KEPLER

- KEPLER is nice to link actors, but does NOT know about CPOs
  - Arrows going from one actor to the other represent a CPO transfer
  - This CPO transfer is done by the « wrapper », not by KEPLER
- Physics modules are subroutines expecting CPOs as input and output
  - No need to use the UAL in a physics module, except for testing purposes
  - Physics modules must be wrapped in a layer that deals with the UAL calls
  - The wrapping is done automatically by the FC2K interface
  - Wrapped physics modules are called « actors » and are usable by KEPLER

Outside the Physics module



Status

- The fundamental tools are up and running
  - Data structure
  - Data storage organisation
  - UAL in Fortran, C++, Java, Python, Matlab
- Still difficulties with the wrapping tool « FC2K »
  - A prototype exists, with several limitations
  - Work is ongoing to remove these limitations and address complex workflows
  - Essentially a problem of ressources, no blocking feature

Preliminar conclusions

- You do not need to learn all ITM software and subtleties
- We try to provide you with the main useful information for
  - Physics module providers (focus on CPO fine structure)
  - KEPLER users (how to build a workflow)
  - More advanced users / testers : UAL calls, wrapper concept
  - (Experimental data providers)

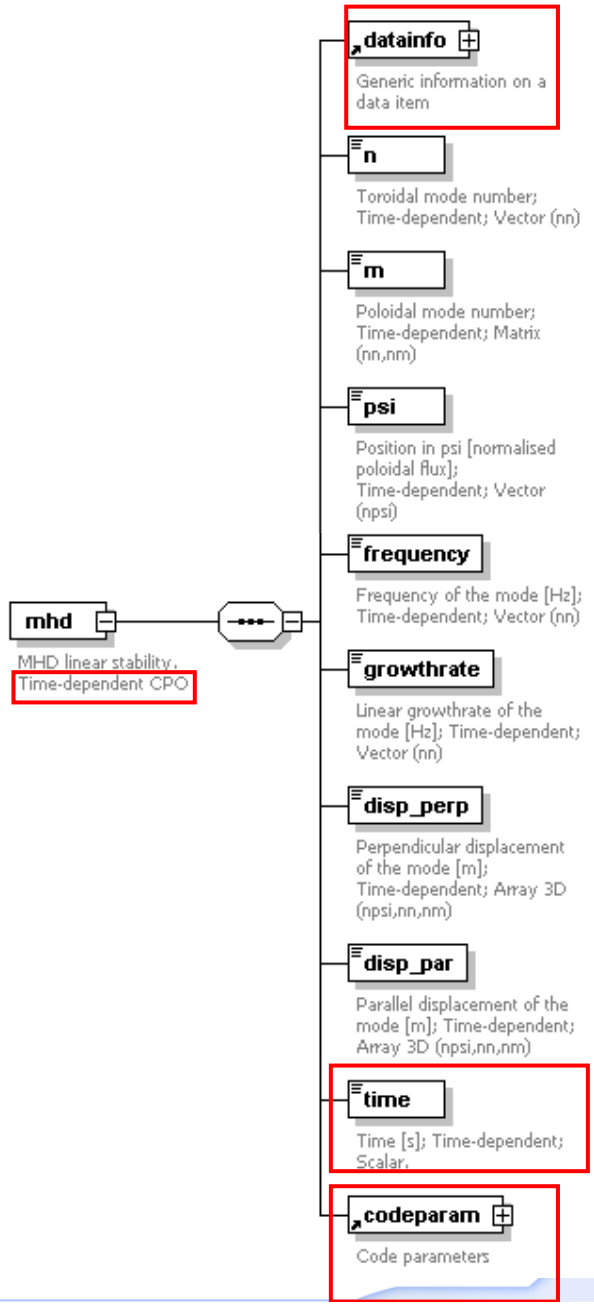
- Most examples in this presentation are given for Fortran

Consistent Physical Objects

- All physics data are organised in CPOs
- Each upgrade of the ITM data structure is released with a version : e.g. 4.06d, 4.07a
  - 4.06d is presently the default version, mainly IMP1+IMP3
  - 4.07a has just been installed, contains many new parts for IMP2 - IMP5
- Have a look to the ISIP webpage / Data Structure to find the documentation on the data structure

CPO structure

- CPOs have a structure with many signals below
  - Substructures are frequent
  - Fine structure depends on the physics
  - All physics signals related to a CPO are there
- Each CPO has its own time array (if time-dependent)
- Each CPO has a bookkeeping sub-structure (datainfo)
- Code-specific parameters are in codeparam



CPO declaration

- Use CPO library

  use euitm_schemas

- Fortran declaration for a complete time-dependent CPO and the module works with multiple time slices :

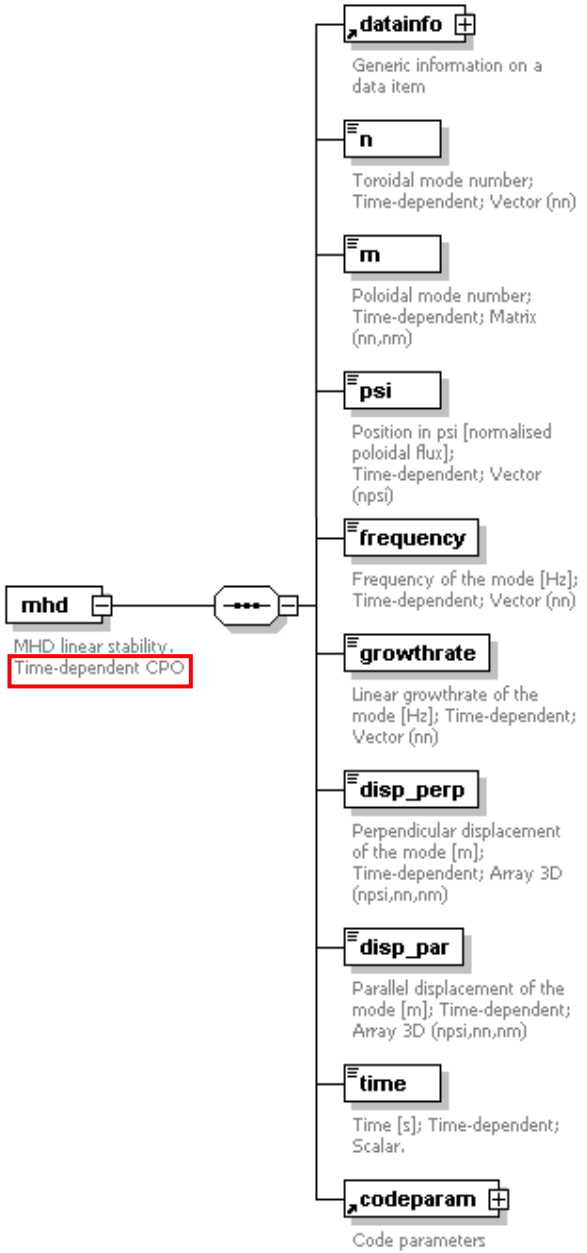
  type (type_mhd),pointer :: cpotest(:) => null()

  allocate cpotest(ntime)

- If the physics module works on a single time slice (or the CPO is not time-dependent) :

  type (type_mhd) :: cpotest

- See UAL documentation on the ISIP web page / UAL for the syntax in the other languages



Individual signal description

- Name
- Definition
- Units
- Dimensionality
- Time-dependent or not
- To put a value in a signal :

  allocate(cpotest%disp_par(npsi,nn,nm))

  cpotest%disp_par = myvalue
- For a CPO with multiple time slices, do for each time slice :

  allocate(cpotest(i)%disp_par(npsi,nn,nm)) ! Array size are NOT time-dependent

  cpotest(i)%disp_par = myvalue

Variable allocation (Fortran)

- Possible ITM types are : strings, vectors of strings, double precision real, integers, arrays of doubles or integers
- CPOs with multiple time slices must be allocated
  - allocate cpotest(ntime)
  - Use euitm_deallocate to deallocate a CPO neatly (but not a CPOin nor a CPOout !!
  - Use euitm_copy to copy neatly a CPO to another CPO variable (to use such a function, you must have « use euitm_routines » together with « use euitm_schemas »)
- All ITM arrays are allocatable
  - Must be allocated at all time slices. Array size is not time-dependent
  - String length in string vectors is limiter to 132 (but the size of the vector can be as large as you wish)
- Fortran strings are allocatable : character(len=132), dimension(:), pointer
  - Must be allocated : to represent a long string of 200 characters, allocate to size (2) and fill each index with the two parts of the string

```
allocate(cpotest(1)%codeparam%parameters(2))
cpotest(1)%codeparam%parameters = ‘First part of my string’
cpotest(1)%codeparam%parameters = ‘Second part of my string’
```

Empty signals

- CPOs are quite complete and detailed; many signals can be left empty
  - Exception : mandatory to fill the « time » signal of time-dependent CPOs
- By convention, empty signals are coded in the following way :
  - Empty allocatable (all arrays) are unallocated (test with « unassociated »)
  - Empty reals are initialised as : -9.0D40
  - Empty integers are initialised as : -999999999

Time-independent signals in time-dependent CPOs

- Time-dependent CPOs may contain time independent signals (e.g. codeparam/codename)
  - When you receive a CPO from the UAL (GET), the same value is copied to all time slices

    cpotest(i)%codeparam%codename has the same value for all time slices i
  - When you prepare a CPO as output of your code, it is sufficient to fill only the first time slice for time-independent signals. Only the first time slice will be used by the UAL

    cpotest(1)%codeparam%codename = ‘my_mhd_code’

Physics module

- To be integrated in the ITM platform, physics modules simply need interfacing to the ITM data format :
  - physics code receive CPOs as input and output :

```
Subroutine Physics_module(CPOin1,….,CPOinN, CPOout1,…,
    CPOoutM)

   use euitm_schemas     ! contains the type definitions of all CPOs
```

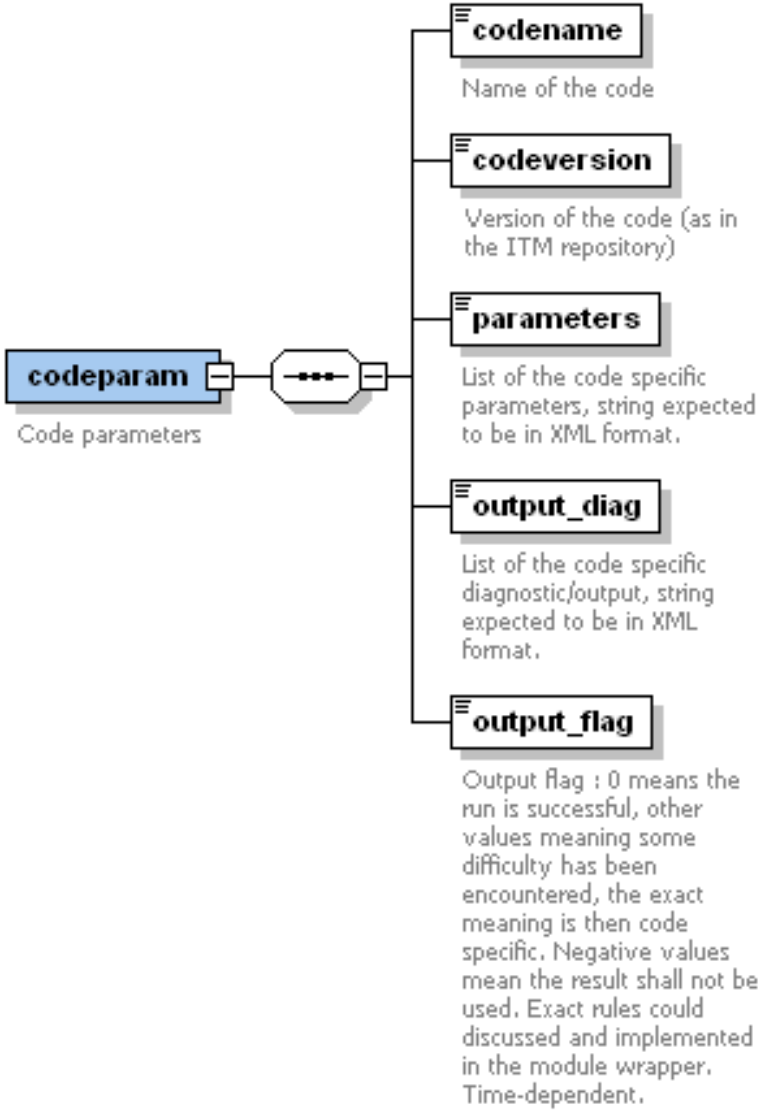
- The IMP code developer can start interfacing his code to the ITM format only by knowing the format of the CPOs. This documentation is available on the ITM website.
  - mapping the CPO data to the internal physics code variables is done by simple instructions like :

```
my_iplasma = my_equilibrium%global_param%i_plasma
```

- At this stage, no knowledge of the UAL nor KEPLER is needed

Code-specific parameters

- The codeparam subtree contains information on the code specific parameters
- To be filled by the physics module for traceability
  - Codename, codeversion (obvious)
  - Parameters : list of code-specific parameters (use XML format and parser, see C. Konz talk
  - Output_diag ; list of code-specific diagnostic/output, in the same XML format
  - Output_flag : 0 if run successful, <0 values mean the result should not be used
- For the moment, there is no management of the code parameters by Kepler : the physics module should read them directly from a file



Database

General organisation of data storage on the Gateway

- Input/output of the simulations are read/written by the UAL at specific places on the Gateway
  - Public database contains validated simulations, test data, machine descriptions. Everybody can read from it but NOT write to it
  - Each Gateway user has a private database at his name, in his account. This database must be used for writing (e.g. the results of a simulation). All Users can read from it.
- The ISIP web page / Database contains
  - information on how to create and use the ITM databases
  - information on public data (contents of the public database)

Data entries

- A data entry is an instance of the ITM data structure
  - Contains a instance of all CPOs
  - Some CPOs can have multiple occurrences in a single data entry (e.g. to represent multiple equilibria with different resolutions, multiple source terms for core transport, …)
  - A data entry should be a consistent dataset : input or output of an integrated simulation (try to stick to this rule when building your workflows)

Data entries

- A data entry is described by 4 characteristics :
  - User : either « public » (refers to the public database) or the username to refer to a private DB
  - Machine : name of the tokamak, can be arbitrary (e.g. « test » for testing purposes
  - Shot : shot number
    - Can be arbitrary for testing purposes
    - Real shot number of corresponds to an experiment
    - Machine descriptions are stored under shot = 0
  - Run : simulation number, can represent various versions of a dataset, or multiple simulations done for the same (user,machine,shot)
- Inside a data entry, a CPO is referred to by its occurrence number
  - Occurrence 0 is equivalent to no occurrence

Data entries

- Before using Kepler or the UAL, you must specify in which database (user, machine) you wish to work
- This is done by running a script that sets environment variables
  - Create_itm_data_env : is needed the first time to create a private database for (user, machine, data version)
  - Set_itm_data_env : for use of UAL only. One must specify (user, machine, data version)
  - ITMv1 : sets all environment variables for Kepler and UAL. User is set to the private database of the current user, since this is the only one where it is possible to write the results of the simulations. One must specify (machine, data version)
- Functions will be provided to change database during a workflow – these are being tested …
  - Meanwhile you have to copy input entries to your private database