Adapting codes to IMAS
IMAS integration from WP Code Development activities in 2020-2021

Pär Strand, and the WPCD team

CHALMERS

ETS

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.

Outline

Pre-amble (setting the stage)

Integration levels – additional requirements

Immersing in the IMAS data ecology

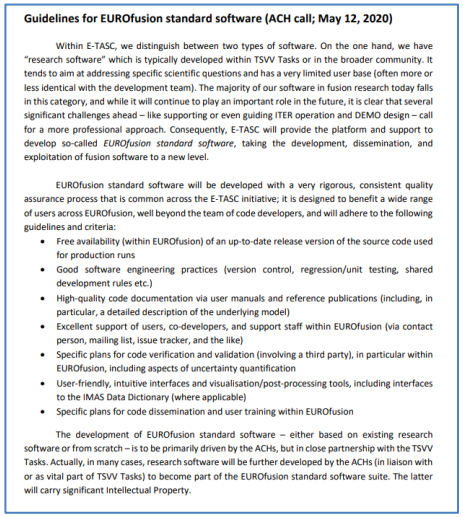
- MHD reconstruction and stability
- EDGE turbulence example
- ETS
- AMNS or how to store largish common data
- Simple Fortran examples of actor integration

My apologies to the large number of contrubutions to WP CD that I am glancing over or just not mentioning…

Pre-amble

The e-Tasc activities are expected to accelerate the scientific output, enhance the software engineering quality and promote a data centric research activity (open and FAIR) →

- Higher level of software engineering
- Code (Products) available to EUROFusion users
- Running on EUROfusion platforms
- Integrated in the IMAS ecology
- With an easy to use GUI on top



Guidelines for EUROfusion standard software (ACH call; May 12, 2020)

Within E-TASC, we distinguish between two types of software. On the one hand, we have "research software" which is typically developed within TSVV Tasks or in the broader community. It tends to aim at addressing specific scientific questions and has a very limited user base (often more or less identical with the development team). The majority of our software in fusion research today falls in this category, and while it will continue to play an important role in the future, it is clear that several significant challenges ahead – like supporting or even guiding ITER operation and DEMO design – call for a more professional approach. Consequently, E-TASC will provide the platform and support to develop so-called *EUROfusion standard software*, taking the development, dissemination, and exploitation of fusion software to a new level.

EUROfusion standard software will be developed with a very rigorous, consistent quality assurance process that is common across the E-TASC initiative; it is designed to benefit a wide range of users across EUROfusion, well beyond the team of code developers, and will adhere to the following guidelines and criteria:

- Free availability (within EUROfusion) of an up-to-date release version of the source code used for production runs
- Good software engineering practices (version control, regression/unit testing, shared development rules etc.)
- High-quality code documentation via user manuals and reference publications (including, in particular, a detailed description of the underlying model)
- Excellent support of users, co-developers, and support staff within EUROfusion (via contact person, mailing list, issue tracker, and the like)
- Specific plans for code verification and validation (involving a third party), in particular within EUROfusion, including aspects of uncertainty quantification
- User-friendly, intuitive interfaces and visualisation/post-processing tools, including interfaces to the IMAS Data Dictionary (where applicable)
- Specific plans for code dissemination and user training within EUROfusion

The development of EUROfusion standard software – either based on existing research software or from scratch – is to be primarily driven by the ACHs, but in close partnership with the TSVV Tasks. Actually, in many cases, research software will be further developed by the ACHs (in liaison with or as vital part of TSVV Tasks) to become part of the EUROfusion standard software suite. The latter will carry significant Intellectual Property.

Requirements towards open data and increased sharing within the EUROfusion community may (<u>will</u>) add additional concerns:

- Provencance capture
- Documentation of
  - Software
  - Data collections

I believe that the IMASification is not the most challenging part for the TSVVs in this!

However, it is a key component in achieving the etasc goals (cmp Frederics talk)

Integration levels

The different TSVVs have a little bit of different character – the bulk (not all) of them focus on developing large scale (mainly monolithic?) codes targeting HPC applications.

Different strategies evolving on different timescales will likely have to be devised (and within a TSVV more than one approach could or should co-exist).

Preconceived conception?

IMAS Integration levels (I am iterating some of Frederics points this is not accidental but intentional)

- ~~None~~
- Post and pre-processor for IMAS data
- Integrated read/write routines
- Full framework integration
- (libraries/storage)

Use cases will vary and require different resources and (multiple?) approaches

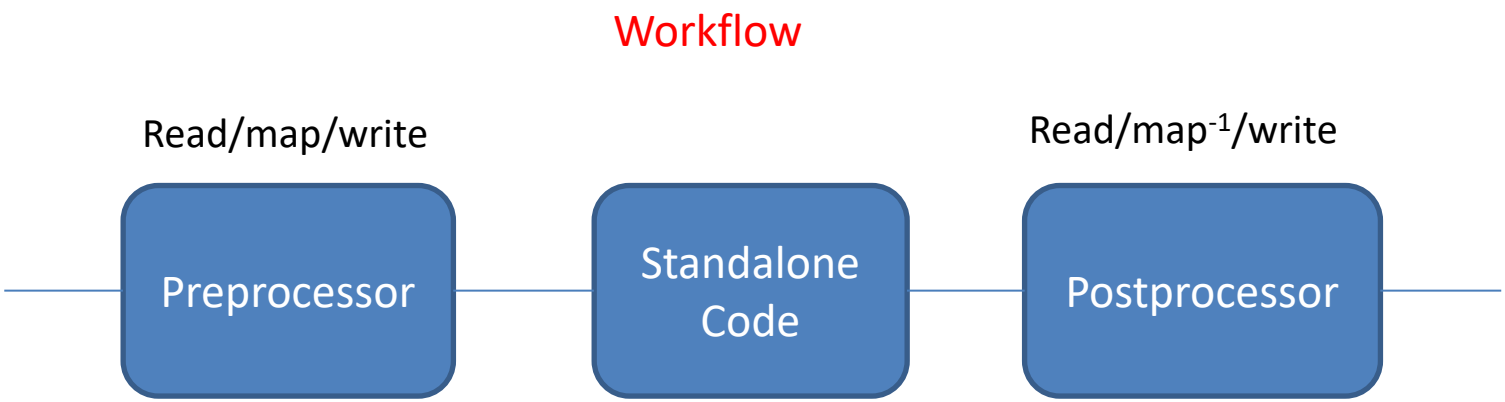
Integration levels

There is no magic bullet – you will need to remap, interpolate and adapt the IDS and other inputs/outputs to your code!

However once done it is largely reusable in all three cases:

IMAS Integration levels

- Pre and post processing
- Integrated read/write
- Framework integration



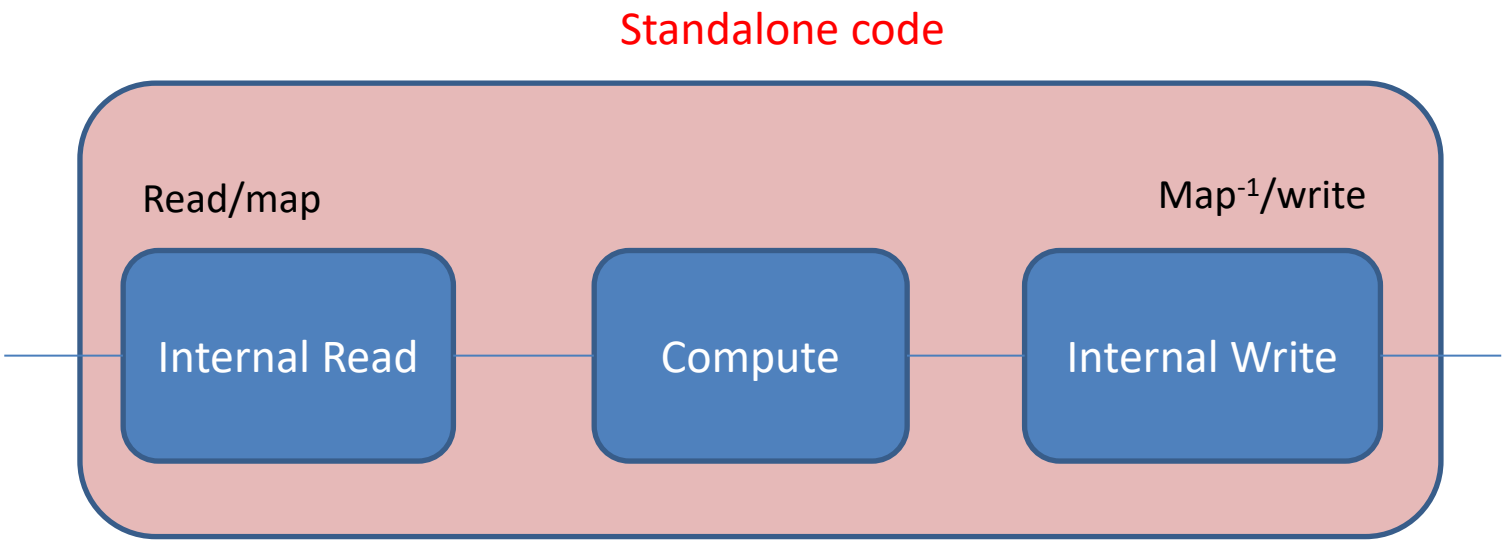
Integration levels

There is no magic bullet – you will need to remap, interpolate and adapt the IDS and other inputs/outputs to your code!

However once done it is largely reusable in all three cases:

IMAS Integration levels

- Pre and post processing
- Integrated read/write
- Framework integration



Integration levels

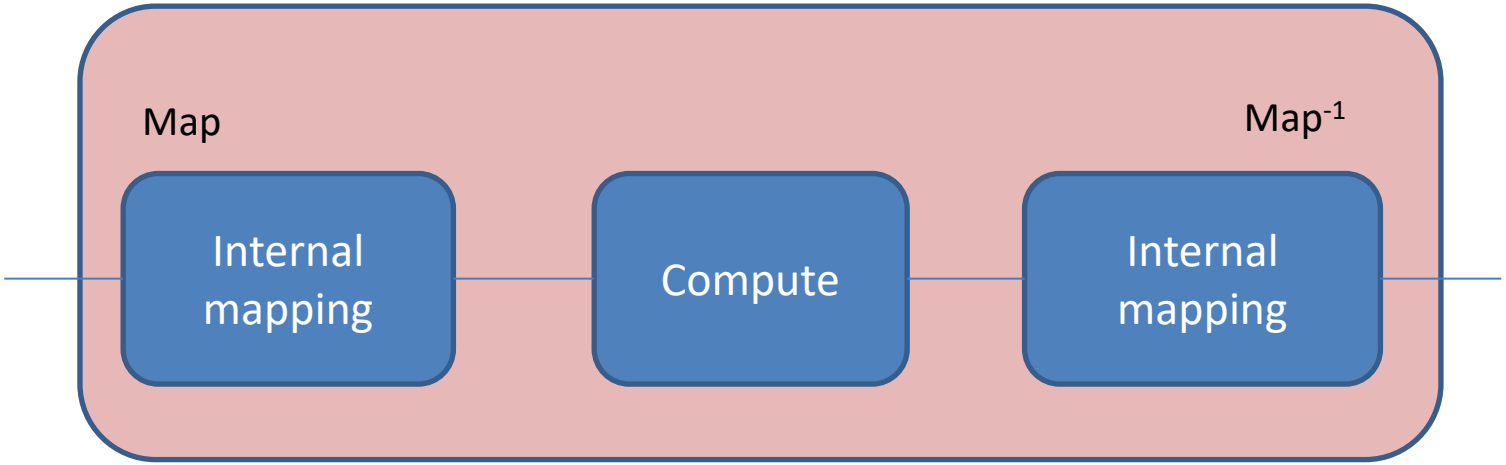
There is no magic bullet – you will need to remap, interpolate and adapt the IDS and other inputs/outputs to your code!

However once done it is largely reusable in all three cases:

IMAS Integration levels

- Pre and post processing
- Integrated read/write
- Framework integration

Framework integration – "code → subroutine"



The "framework" manages all the input outputs in IDSs.
NB: "Framework" here is loosely defined: Kepler, Python framework under development, or a driver code in any of the supported languages (fortran, c/c++,…)

Integration levels

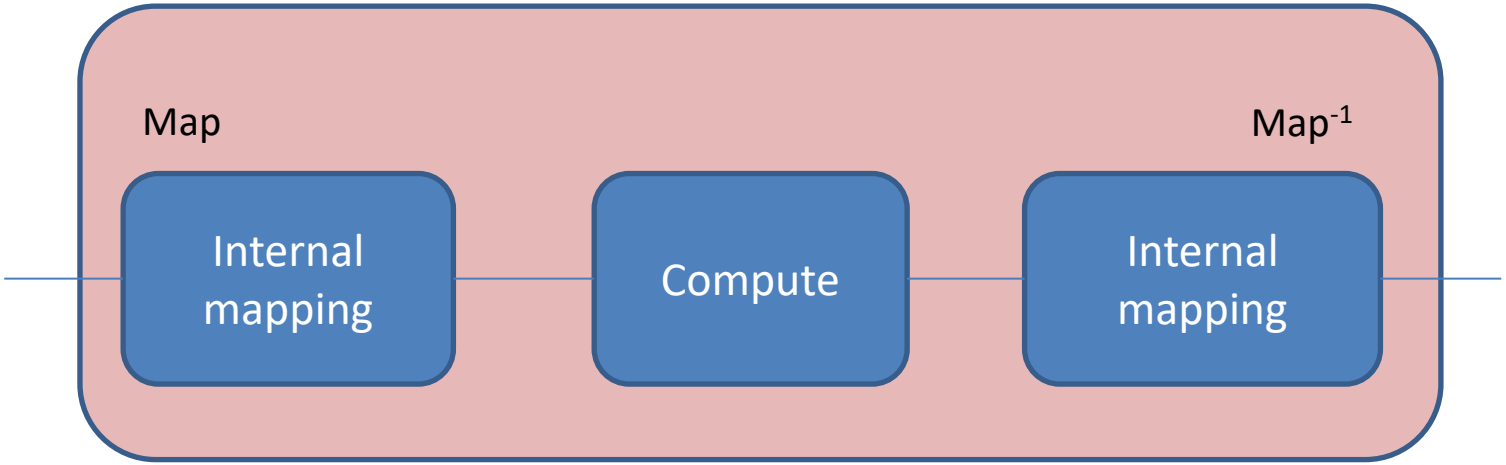
There is no magic bullet – you will need to remap, interpolate and adapt the IDS and other inputs/outputs to your code!

However once done it is largely reusable in all three cases:

IMAS Integration levels

- Pre and post processing
- Integrated read/write
- Framework integration

Framework integration – "code → subroutine"



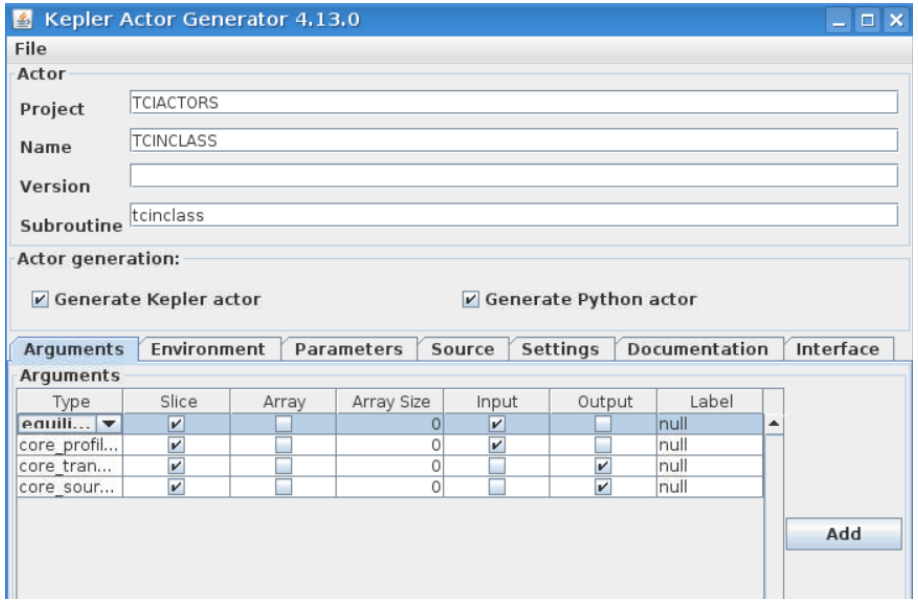
Key to development are the standardised interfaces that allows for a high level of abstraction and automation! WP CD ended up with all of the physics components available on this level.

Standardised interfaces - FC2K

Fortran example - IDS appear as derived types

subroutine name(input_IDSs, output_IDSs, code_params, outputFlag, diagnosticInfo)

- list of input IDS
- list of output IDS
- Code params (derived type)
  (xsd schema, xml input)
- OutputFlag, diagnosticInfo
  (optional but recommended)



Example from TCI (transport code interfaces) used in ETS: Inputs, outputs, [optional]

Subroutine TCINCLASS (equilibrium, core_profiles, core_transport, core_sources, code_params, [flag], [message])

FC2K is an "actor" generator for Kepler and python based frameworks. ITER is working on a replacement with updated interfaces: iWrap cf Wednesday

Interface test program Fortran

```
Program RunNclass

   ! Access IDS schemas and routine
   use IDS_schemas
   use IDS_routines
   ! get access to XML read tools
   use xml_file_reader

   implicit none

   ! --- Inputs ---
   type(ids_equilibrium)      :: eq                        ! equilibrium ids
   type(ids_core_profiles)    :: coreprof                  ! core_profiles ids
   type(ids_parameters_input) :: code_parameters   ! xml data and
                                                           ! Xsd schema

   ! --- Output ---
   type(ids_core_transport)   :: coretransp             ! core_transport
   type(ids_core_sources)     :: coresource             ! core_sources
   integer                    :: output_flag
   character(len=:), pointer :: output_message

   ! --- Local data --
   integer :: idx                                          ! IMAS data handle

   !  --- Data shot definitions ---
   character(len=5) :: treename='ids', version=¨'3', device='iter'
   character(len=8) :: user='g2past'
   integer :: shot=5, run=2, runout=999, interpol=1
   integer :: refrun=0, refshot=0 ! backwards compatibiity ignore
   real(kind(1.0d0)) :: time=200.0D0

   ! --- Code parameters – input files and definitions ---
   character(len=512) :: xmlInput='input.xml', (
                         xmlSchema='schema.xsd'
```

```
!  --- Read in data ---

   call IMAS_open_ENV( treename, shot, run, idx, USER, device, version )
   if( idx < 0 ) then
      write(6,"(a)") "ERROR: unable to open database."
      stop
   endif

   call IDS_get_slice( idx, "Core_profiles", coreprof, time, interpol )
   call IDS_get_slice( idx, "Equilibrium", eq, time, interpol )
   call IMAS_close( idx )


   ! --- Read in code parameters to code_param derived type --
   call fill_param( code_parameters%parameters_value, &
                    code_parameters%schema, &
                    code_parameters%parameters_default, &
                     xmlInput,  xmlSchema, XMLInput)
   !---------------------------------------------------------------------
   ! Run the model
   call tciNCLASS( eq, coreprof, coretransp, coresource, code_parameters, &
                                   output_flag, output_message )
   !---------------------------------------------------------------------

   ! --- Store the output with a copy of the input data ---
   call IMAS_create_env( treename, shot, runout, refshot, refrun, idx, &
                                 USER, device, version )

   call IDS_put( idx, "Core_profiles", coreprof )
   call IDS_put( idx, "Equilibrium", eq )
   call IDS_put( idx, "Core_transport", coretransp )
   call IDS_put( idx, "Core_sources", coresource )
   call IMAS_close( idx )

   !  ---  clean up data storage ---
   call ids_deallocate( coreprof )
   call ids_deallocate( eq )
   call ids_deallocate( coretransp )
   call ids_deallocate( coresource )


end program
```

Main usage is within framework structures though

Interface test program Fortran

```
Program RunNclass

   ! Access IDS schemas and routine
   use IDS_schemas
   use IDS_routines
   ! get access to XML read tools
   use xml_file_reader

   implicit none

   ! --- Inputs ---
   type(ids_equilibrium)      :: eq                        ! equilibrium ids
   type(ids_core_profiles)    :: coreprof                  ! core_profiles ids
   type(ids_parameters_input) :: code_parameters   ! xml data and
                                                           ! Xsd schema

   ! --- Output ---
   type(ids_core_transport)   :: coretransp             ! core_transport
   type(ids_core_sources)     :: coresource             ! core_sources
   integer                    :: output_flag
   character(len=:), pointer :: output_message

   ! --- Local data --
   integer :: idx                                          ! IMAS data handle

   !  --- Data shot definitions ---
   character(len=5) :: treename='ids', version='¨'3', device='iter'
   character(len=8) :: user='g2past'
   integer :: shot=5, run=2, runout=999, interpol=1
   integer :: refrun=0, refshot=0 ! backwards compatibiity ignore
   real(kind(1.0d0)) :: time=200.0D0

   ! --- Code parameters – input files and definitions ---
   character(len=512) :: xmlInput='input.xml', (
                         xmlSchema='schema.xsd'
```

```
!  --- Read in data ---

   call IMAS_open_ENV( treename, shot, run, idx, USER, device, version )
   if( idx < 0 ) then
      write(6,"(a)") "ERROR: unable to open database."
      stop
   endif

   call IDS_get_slice( idx, "Core_profiles", coreprof, time, interpol )
   call IDS_get_slice( idx, "Equilibrium", eq, time, interpol )
   call IMAS_close( idx )


   ! --- Read in code parameters to code_param derived type --
   call fill_param( code_parameters%parameters_value, &
                    code_parameters%schema, &
                    code_parameters%parameters_default, &
                     xmlInput,  xmlSchema, XMLInput)
   !---------------------------------------------------------------------
   ! Run the model
   call tciNCLASS( eq, coreprof, coretransp, coresource, code_parameters, &
                                   output_flag, output_message )
   !---------------------------------------------------------------------

   ! --- Store the output with a copy of the input data ---
   call IMAS_create_env( treename, shot, runout, refshot, refrun, idx, &
                                 USER, device, version )

   call IDS_put( idx, "Core_profiles", coreprof )
   call IDS_put( idx, "Equilibrium", eq )
   call IDS_put( idx, "Core_transport", coretransp )
   call IDS_put( idx, "Core_sources", coresource )
   call IMAS_close( idx )

   !  ---  clean up data storage ---
   call ids_deallocate( coreprof )
   call ids_deallocate( eq )
   call ids_deallocate( coretransp )
   call ids_deallocate( coresource )


end program
```

Not filing some mandatory info here for brevity “code…”, data source…

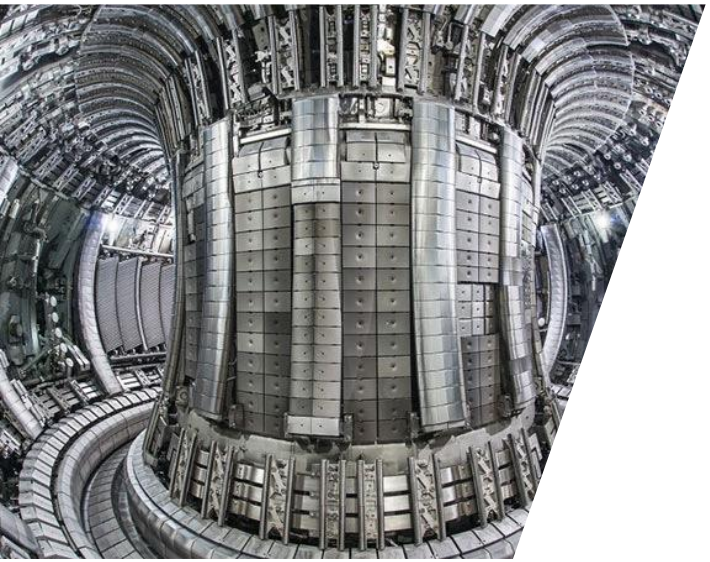
Main usage is within framework structures though

IMAS “gotchas”

Some general observations and advice:

- Be COCOS aware!
  - Know your coordinate conventions and how they relate to the IMAS definition (Immutable to cocos=11 in IMAS, likely different for your code)
- Be Change aware!
  - IMAS is evolving and non-backwards compatible changes may occur: there are discussion on moving from IMAS DD v3 to IMAS DD v4, change of integration support software (actor generation,…)
- Know your input sources!
  - Validate your inputs – IMAS IDS are not required to be complete nor internally consistent or even coherent for a single input set
- Select your integration approach or framework based on the granularity of your application(s)
- Managing your code specific parameters is likely more challenging than you would expect…
- In complex workflows data “ownership” may be an issue: if different physics codes updates the same IDS - who has preference and/or how do you merge conflicting data?

Understand your use case and in particular if it breaks new ground in terms of IMAS data dictionary, use and or storage you may need to check and validate your chosen approach.



I will show some example of use from WPCD together with a brief comment on data availability

Demonstrate different aspects of integration as discussed before

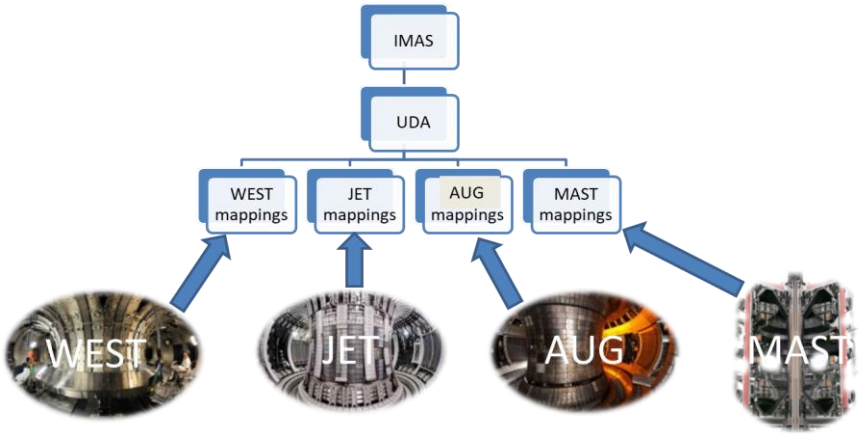
- Different integration levels
- GUI elements

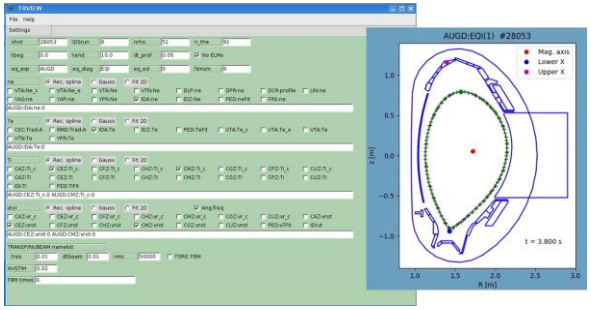
- Largely shown in Kepler framework
  - MHD equilibrium and reconstruction workflows
  - Edge turbulence WF + synthteticdiagnostics
  - ETS
  - AMNS

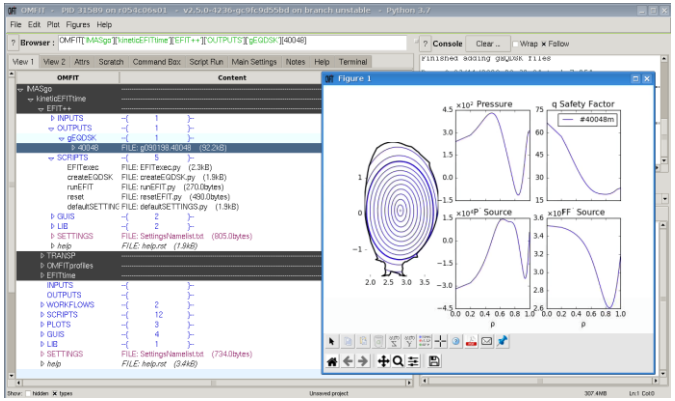
Data access - getting data from experiments

Two approaches:

- UDA (In principle available but yet to be fully established as a general tool, Open data access etc will likely be built on this)
- Bespoke toolset used to map data from experiments.
  - Exp2itm, Trview (for AUG data), readAUG, IMASgo! (Omfit plugin), TCV2IDS,…







ITER could be an incubator for the IMAS/UDA paradigm. In general, little interest from experiments otherwise to engage.

Data available in IMAS form

IDS Name	JET	TCV	AUG	MAST	WEST
iron_core					
magnetics					
mse					
pf_active					
tf					
thomson_scattering					
wall					
core_profiles					
equilibrium					
nbi					
ic_antennnas					
ec_antennnas					
core_sources					

Colour	Meaning
	Data not relevant for this machine
	Data is missing
	Data mapping in development
	Data available
	Data validated as input of EWE-2 and EWE-3 workflows

- Initial experimental input datasets provided for “all” EUROfusion machines
- - Iterative process with workflow owners to test / extend the datasets as required
- Alternates to UDA to process native data and map them in IMAS/IDS have been developed to target specific workflows:
  - TRVIEW
  - IMASgo
  - TCV2IDS
  - ReadAUG

These aspects of the is not continued from WPCD and are largely not covered within ACHs.

Edge turbulence WF

Welcome to the HESEL EDGE TURBULENCE WORKFLOW v1.0

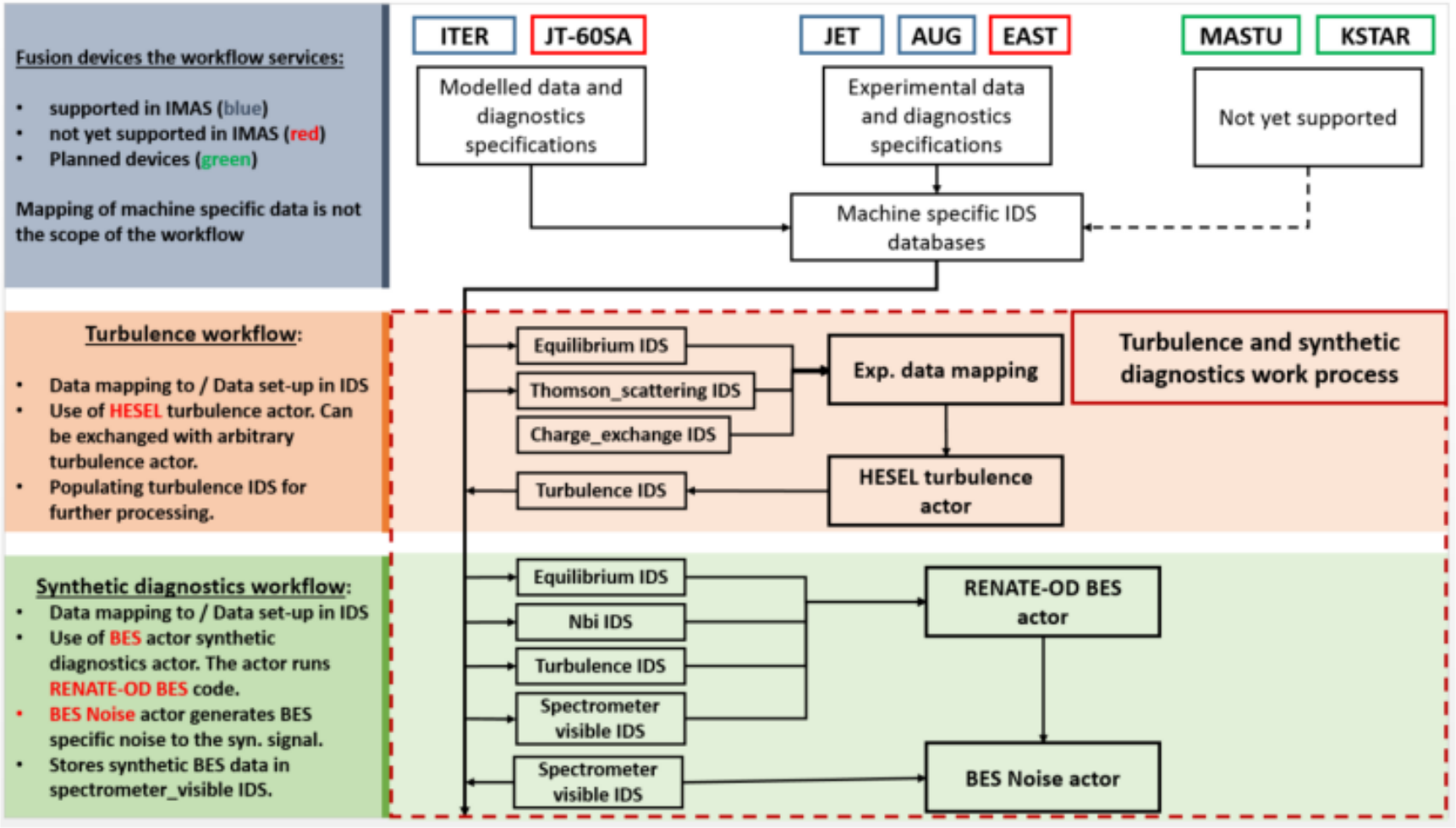
Features :
- Reads AUG shotfile database (equilibrium, edge_profiles)
- Maps the experimental fit data to (R,Z) line of sight of thomson_scattering and charge_exchange
- Evaluates SOL turbulence with HESEL code

Running the workflow :
- Set imasdb parameters in the START actor
- Set l.o.s. for virtual thomson_scattering and charge_exchange in MAP_EXP_DATA actor
- Merged equilibrium+edge_profiles+thomson_scattering+charge_exchange+turbulence are collected.

CREDITS
Workflow + data management scripts : Rui Coelho
HESEL actor : Anders Nielsen

Figure 1 The turbulence workflows, HESEL_edge_workflow, with the included Python mapper and turbulence actor, HESEL

Edge turbulence WF with synthetic diagnostics



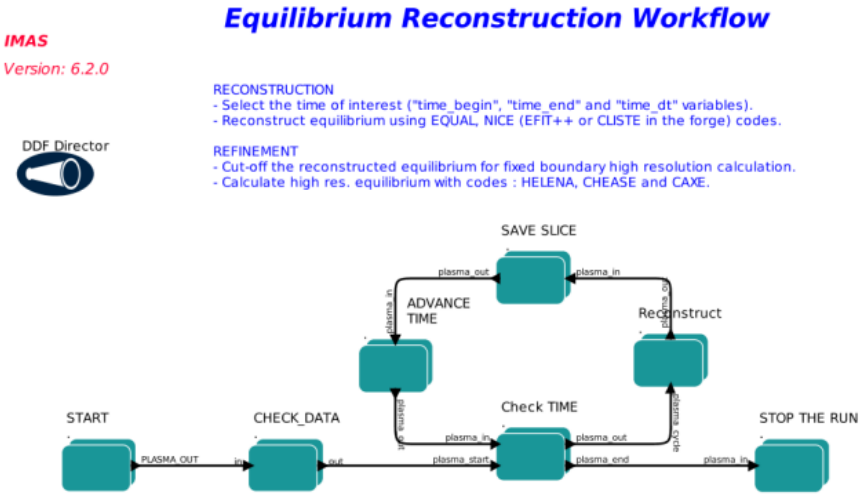
Turbulence WPCD workflows including synthetic diagnostics to compare modelled data of SOL filamentary transport to experimental data in collaboration with WPMST1 and WPJET1. The two workflows are used independently or in sequence reading data from databases.

MHD equilibrium and Stability

Reconstruction workflow

with or without kinetic constraints (default codes EQUAL, NICE).

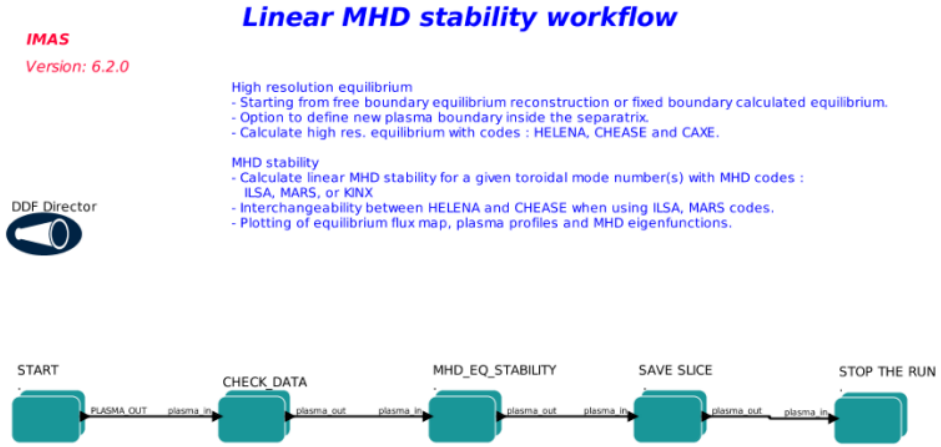
Cyclic workflow requiring (simple) control structure. Loops over time



Stability workflow as a single timepoint DAG for potential inclusion in ETS or reconstructions WF.

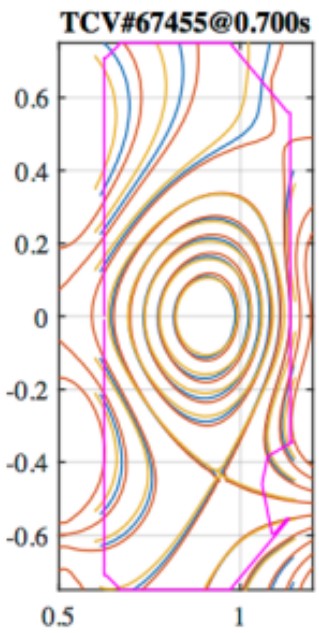
Alternate chains:

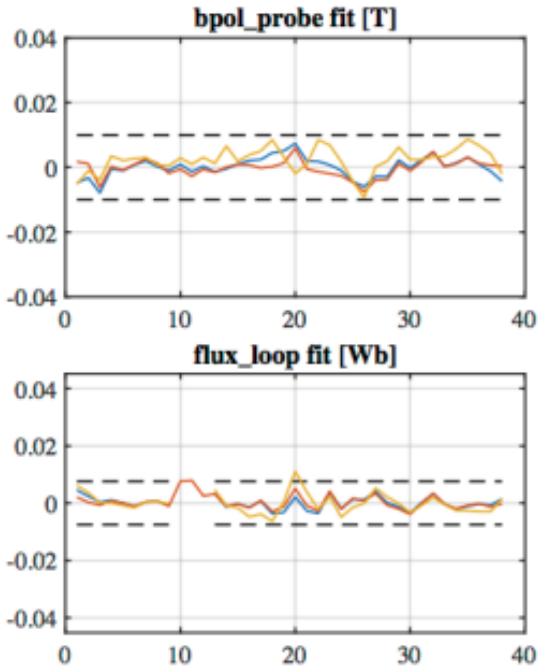
- Helena/Chease w ILSA/MARS
- Caxe w Kinx

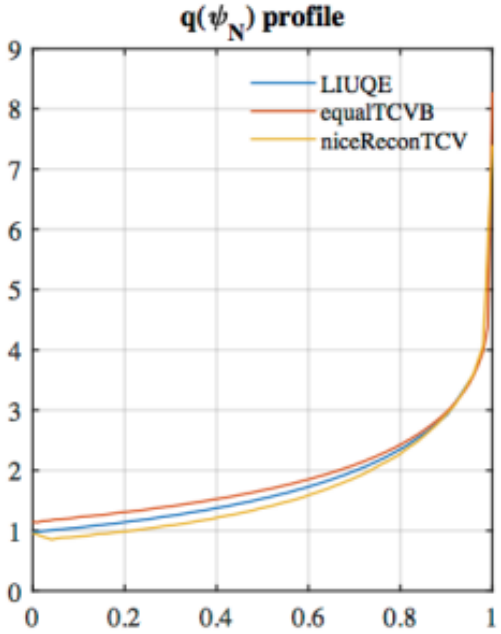


equilibrium reconstruction

A negative triangularity case with separatrix, TCV



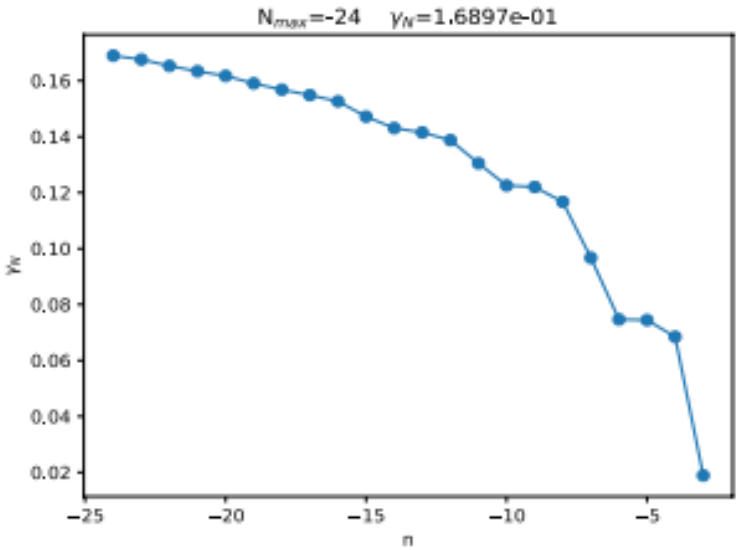


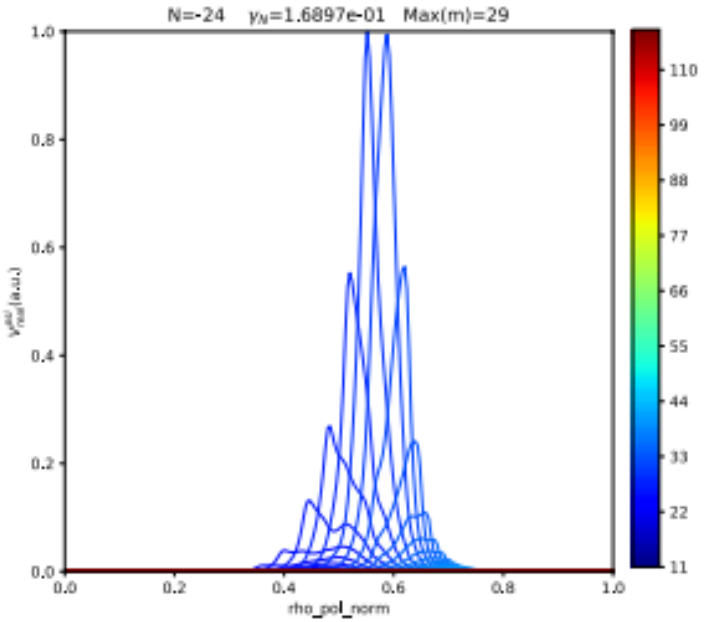


Good agreement is found between the several codes (LIUQE – the in-house tool at TCV, EQUAL and NICE). Comparison of flux contours, magnetic probe and flux loop reconstruction errors and obtained |q| profiles for the codes LIUQE, EQUAL and NICE using data from TCV discharge 67455 at t = 0.7s. Black dashed lines indicate the absolute error for the different signals

stability workflow

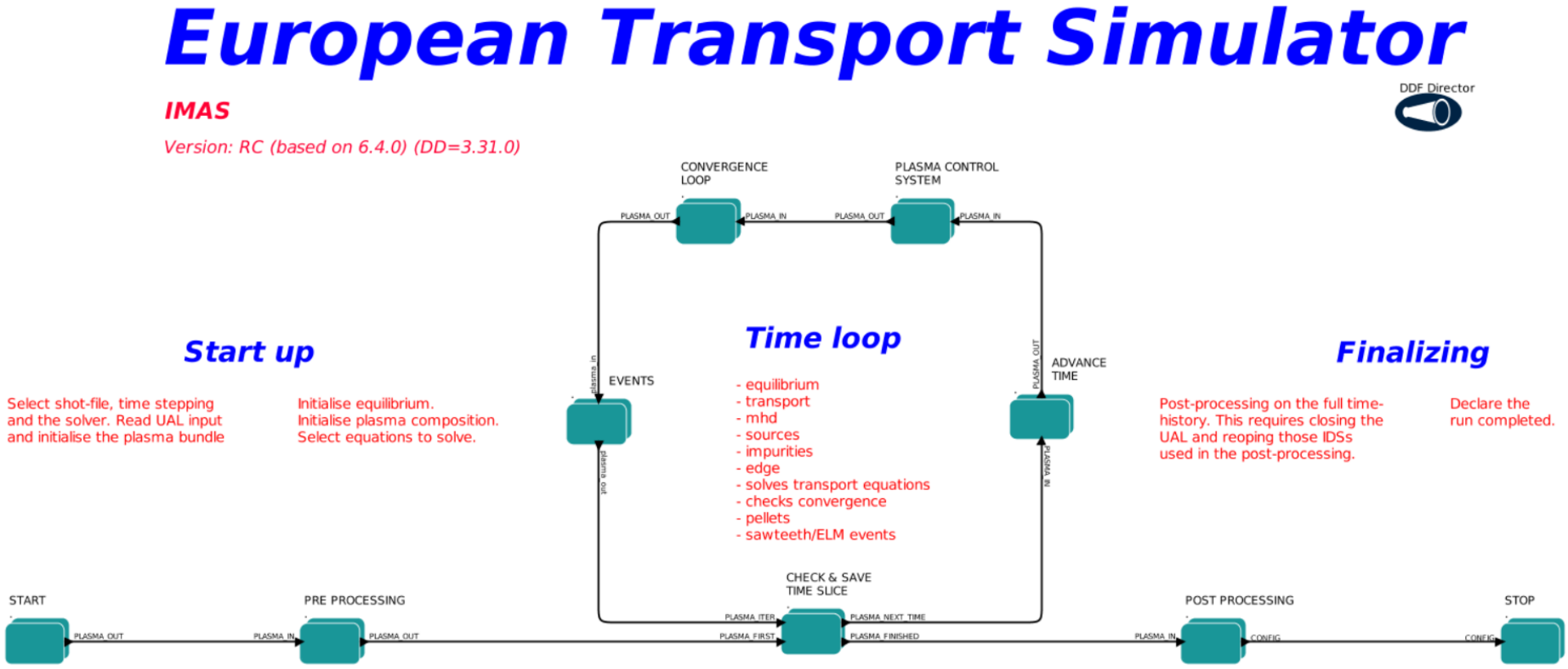
Application to JT-60SA, Scenario 4 with data mapped from eqdsk file, (hybrid scenario with internal transport barrier in ion energy channel). The scenario is characterised by internal infernal like modes where the ITB is located (as found in analysis).





Normalised growth rate scaling with toroidal mode number (left) and radial velocity component of the most unstable mode (right) are shown for the CDBM equilibrium with fast ion pressure included.

ETS

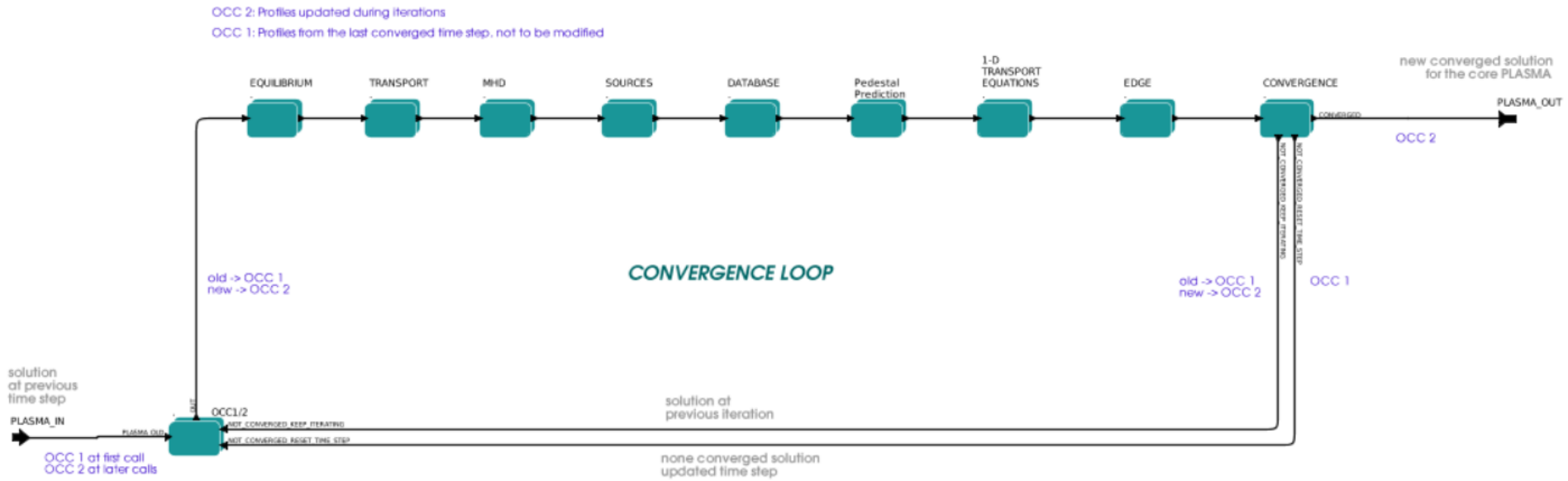


Complex workflow with more than 40 physics modules (some alternative physics Implementations) + significant number of support and mapper functions.

Integrated framework: Java intepreter runs MoML (xml) based wf description actors In fortran, c, and python

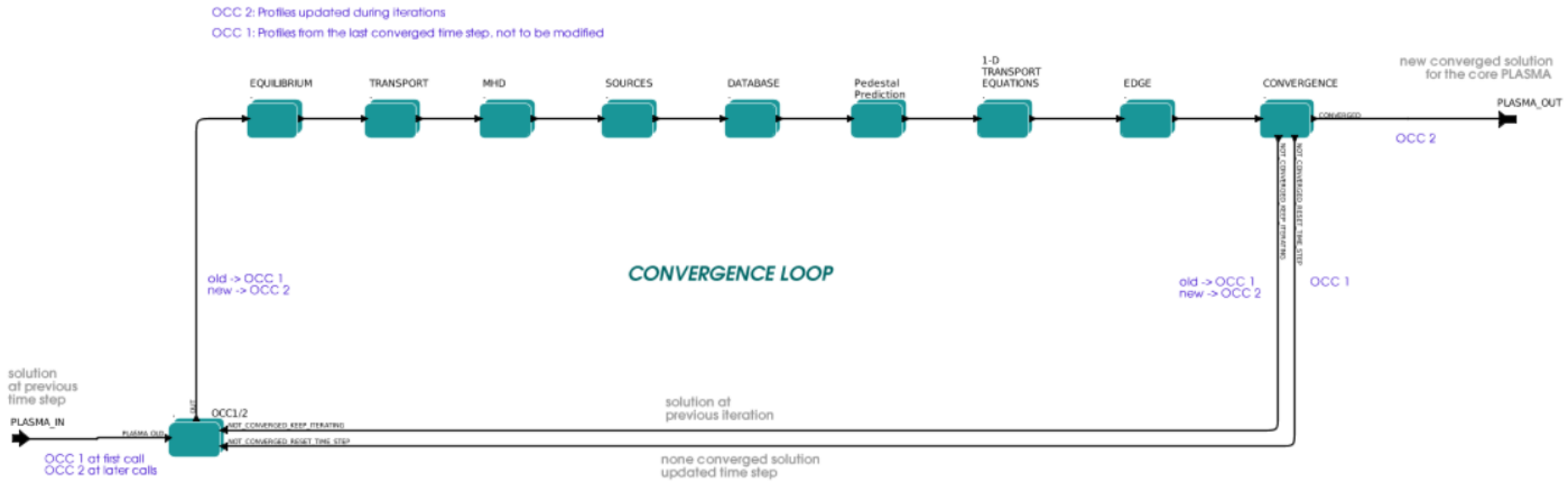
ETS interfaces - GUI

.



Two interfaces available: Kepler Canvas interface (above) which provides an excellent overview of software relations including configurations (and has a Powerful shortcut system for expert users). Input/settings as part of WF.

Bulk of the matter: ETS physics modules



Equlibrium
Static
Interpretative
Chease
Helena
GKMHD

Transport	
Database	MMM
Analytical	RITM
GLF23	
Weiland	CDBM
EDWM	BgB
TGLF	Neowes
QLK	Neos
NCLASS	
NEO	

Sources	
Database	Cyrano
Analytical	Lion
BBNBI	Nbisim2
Nemo	Risk
AFSI	Spot
Fusion_sources	Ascot4
GRAY	StixRedist
GENRAY	FoPla
Torayfom	Pion
Torbeam	Iccoup
runaway indicator	Impurities
Runaway fluid	Neutrals

Pedestal
PENN

Solvers
FEM
Progronka

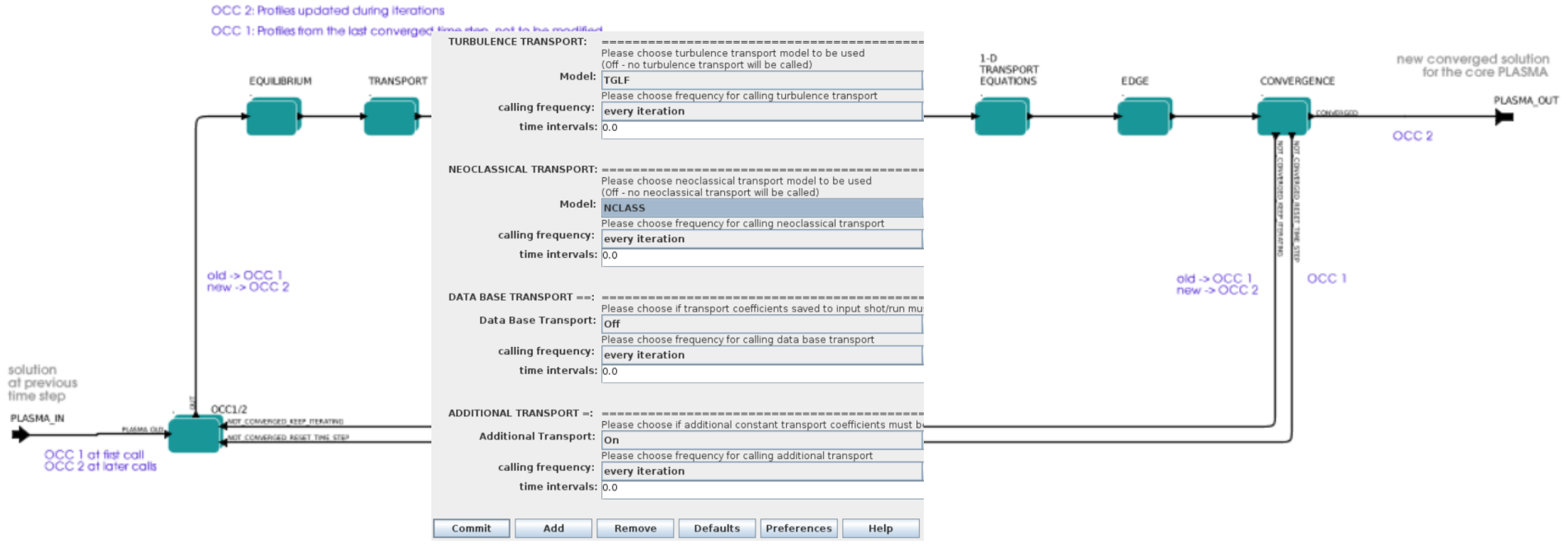
Edge
CEC
Solpsz1

In short: state of the art set of physics modules in a robust and highly configurable framework

A few models remains to be ported From ETSv5.

ETS interfaces - GUI

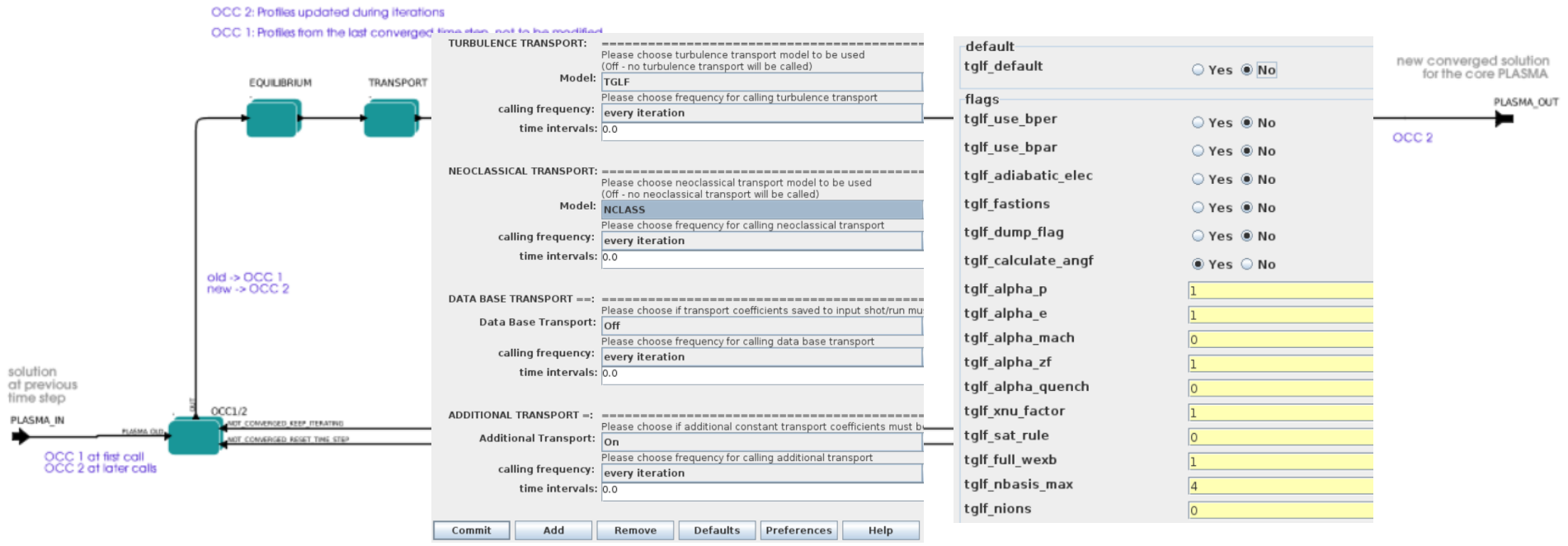
.



Two interfaces available: Kepler Canvas interface (above) which provides an excellent overview of software relations including configurations (and has a Powerful shortcut system for expert users). Input/settings as part of WF.

ETS interfaces – code parameters

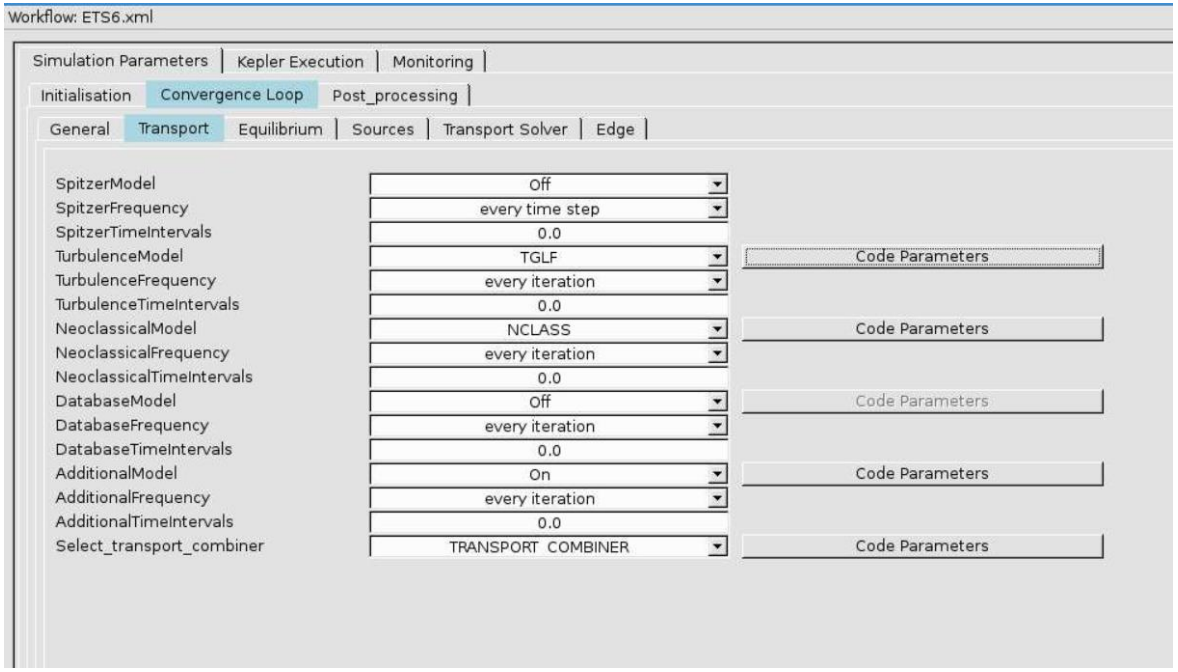
.

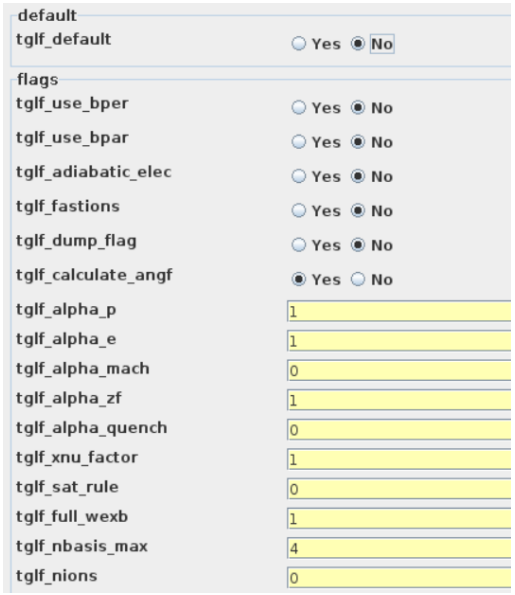


Two interfaces available: Kepler Canvas interface (above) which provides an excellent overview of software relations including configurations (and has a Powerful shortcut system for expert users). Code parameters explicitly available

ETS interfaces - autoGui

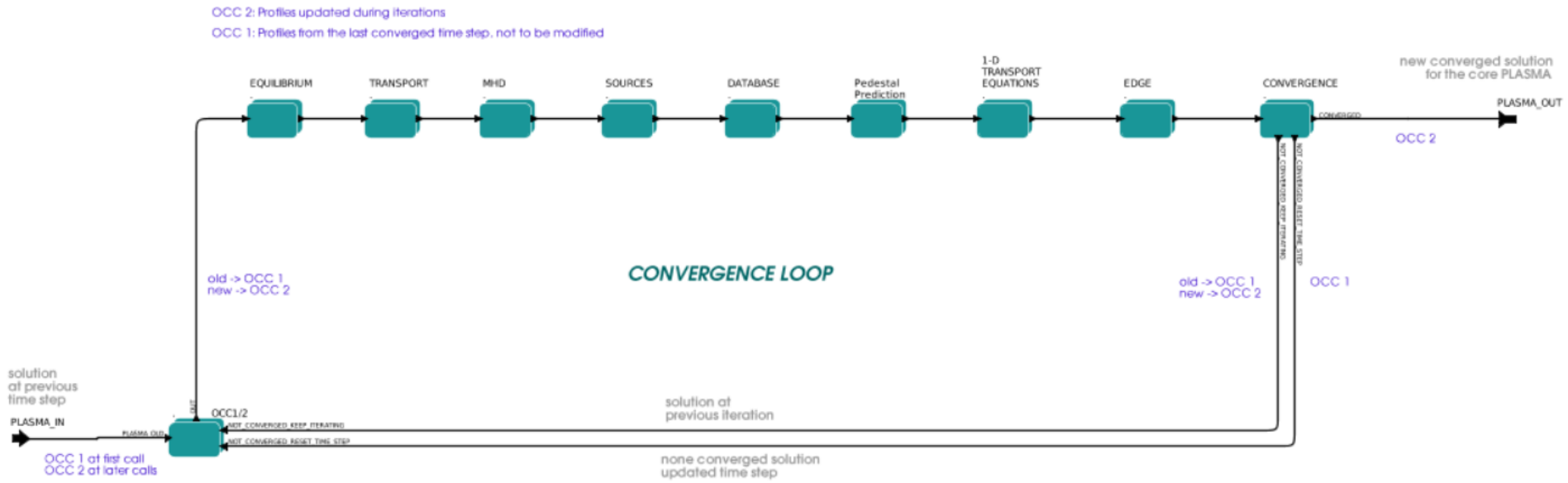
The autoGui is automagically built form the loaded workflow and provides a flatter view of ETS and its settings and in addition has added features of launching and monitoring jobs on the (gateway) cluster.





The autoGui also allows parameter files be saved, shared and distributed between users - outside of training and developments work the autoGui is the recommended Frontend tool.

Bulk of the matter: ETS physics modules



Equlibrium
Static
Interpretative
Chease
Helena
GKMHD

Transport	
Database	MMM
Analytical	RITM
GLF23	
Weiland	CDBM
EDWM	BgB
TGLF	Neowes
QLK	Neos
NCLASS	
NEO	

Sources	
Database	Cyrano
Analytical	Lion
BBNBI	Nbisim2
Nemo	Risk
AFSI	Spot
Fusion_sources	Ascot4
GRAY	StixRedist
GENRAY	FoPla
Torayfom	Pion
Torbeam	Iccoup
runaway indicator	Impurities
Runaway fluid	Neutrals

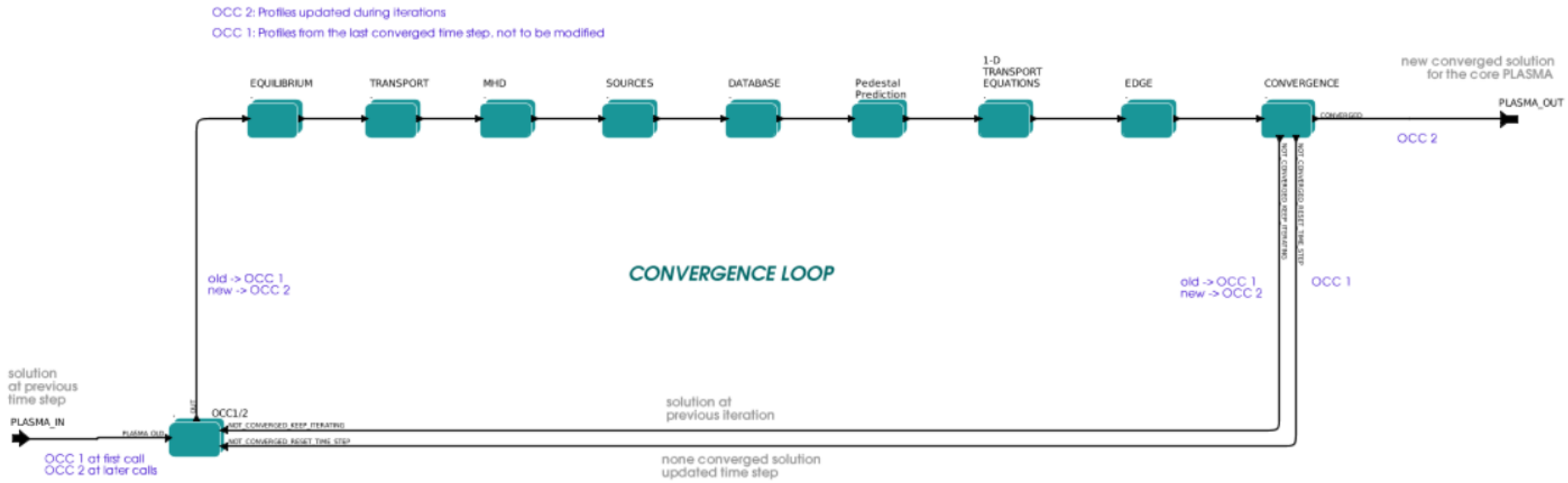
Pedestal
PENN

Solvers
FEM
Progronka

Edge
CEC
Solpsz1

The range of modules available gives a user some flexbility to either ”Zoom” to a specific physics aspect and/or vary fidelity from ”scoping” to ”advanced” through the selection of modules.

Bulk of the matter: ETS physics modules



Equlibrium
Static
Interpretative
Chease
Helena
GKMHD

Transport	
Database	MMM
Analytical	RITM
GLF23	
Weiland	CDBM
EDWM	BgB
TGLF	Neowes
QLK	Neos
NCLASS	
NEO	

Sources	
Database	Cyrano
Analytical	Lion
BBNBI	Nbisim2
Nemo	Risk
AFSI	Spot
Fusion_sources	Ascot4
GRAY	StixRedist
GENRAY	FoPla
Torayfom	Pion
Torbeam	Iccoup
runaway indicator	Impurities
Runaway fluid	Neutrals

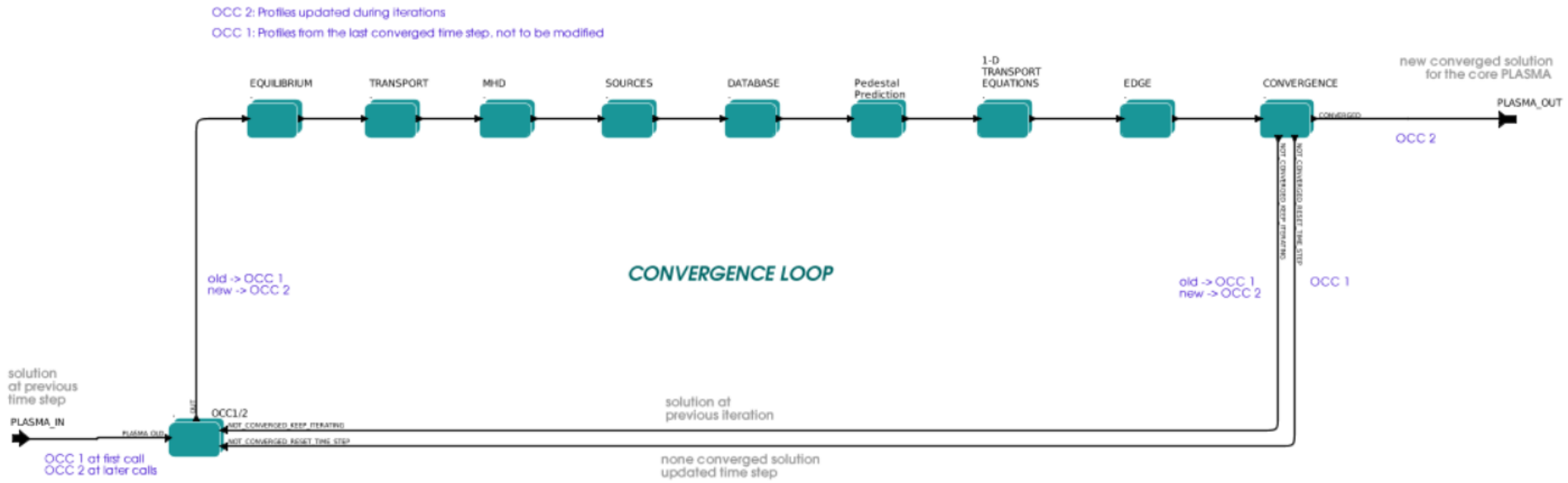
Pedestal
PENN

Solvers
FEM
Progronka

Edge
CEC
Solpsz1

Some lingering general issues in relation to IMAS implementation:
Where to store ML network data (files not allowed…)
(Affects QLKNN; PENN)

Bulk of the matter: ETS physics modules



Equlibrium
Static
Interpretative
Chease
Helena
GKMHD

Transport	
Database	MMM
Analytical	RITM
GLF23	
Weiland	CDBM
EDWM	BgB
TGLF	Neowes
QLK	Neos
NCLASS	
NEO	

Sources	
Database	Cyrano
Analytical	Lion
BBNBI	Nbisim2
Nemo	Risk
AFSI	Spot
Fusion_sources	Ascot4
GRAY	StixRedist
GENRAY	FoPla
Torayfom	Pion
Torbeam	Iccoup
runaway indicator	Impurities
Runaway fluid	Neutrals

Pedestal
PENN

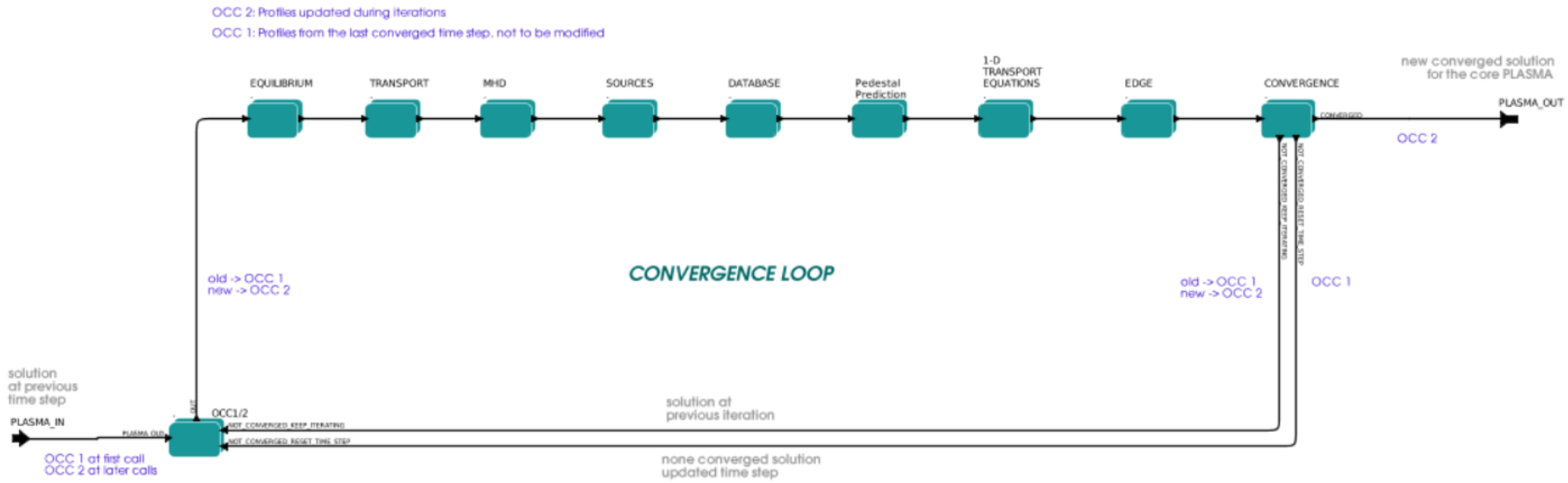
Solvers
FEM
Progronka

Edge
CEC
Solpsz1

Fast Ion physics is included in the equilibrium and the sources, restricted to dilution only in transport modules - largely by code owners recommendations.

Transport and H&CD well presented before will only discuss other and new aspects today.

Bulk of the matter: ETS physics modules



Equlibrium
Static
Interpretative
Chease
Helena
GKMHD

Transport	
Database	MMM
Analytical	RITM
GLF23	
Weiland	CDBM
EDWM	BgB
TGLF	Neowes
QLK	Neos
NCLASS	
NEO	

Sources	
Database	Cyrano
Analytical	Lion
BBNBI	Nbisim2
Nemo	Risk
AFSI	Spot
Fusion_sources	Ascot4
GRAY	StixRedist
GENRAY	FoPla
Torayfom	Pion
Torbeam	Iccoup
runaway indicator	Impurities
Runaway fluid	Neutrals

Pedestal
PENN

Solvers
FEM
Progronka

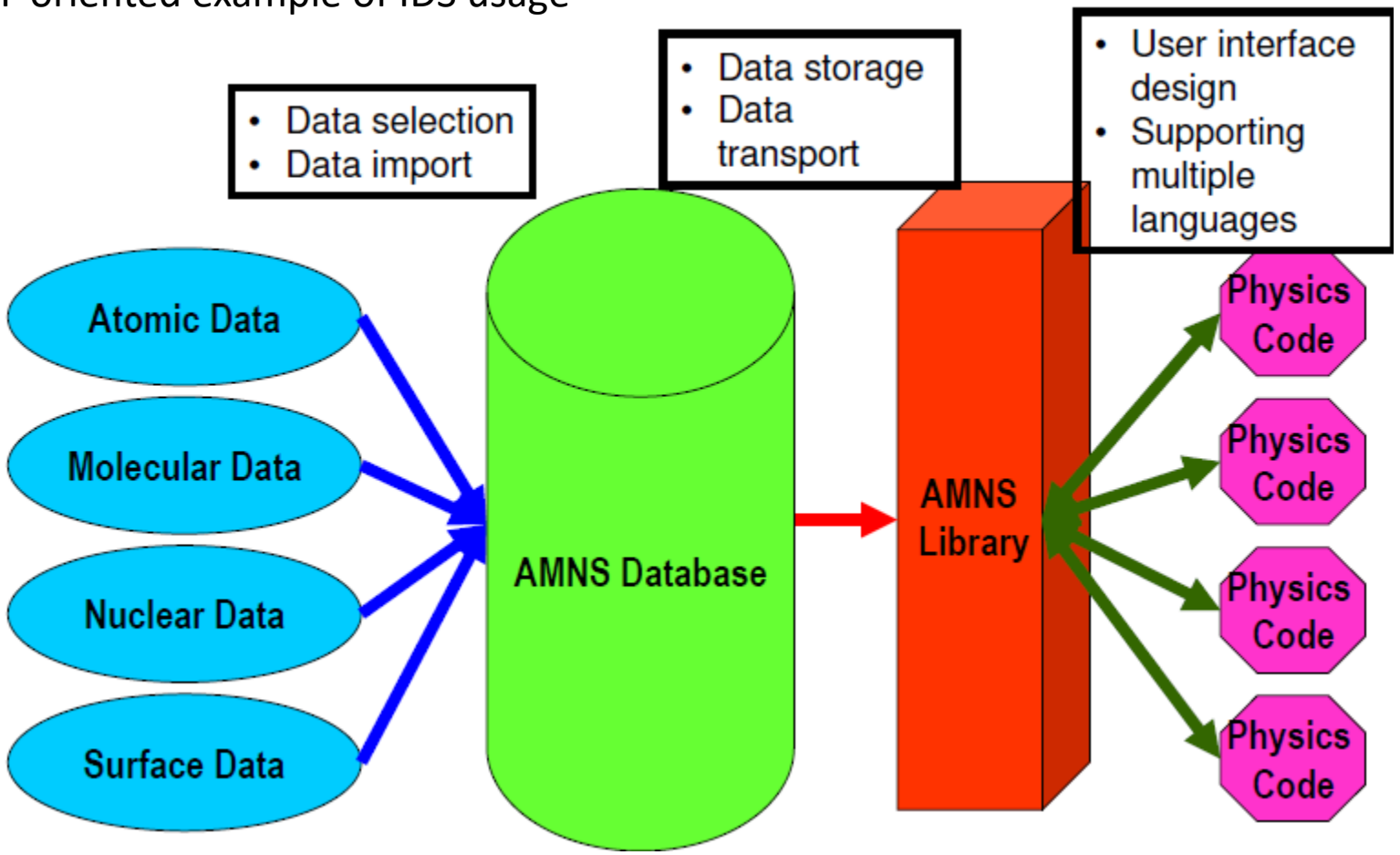
Edge
CEC
Solpsz1

ETS be used for both
- Interpretative and
- Predictive simulations!

Set up for JET; AUG, TCV; DEMO; MAST; ITER; WEST; JT-60SA as well as K-star and DIII-D

AMNS schematics

None WF oriented example of IDS usage

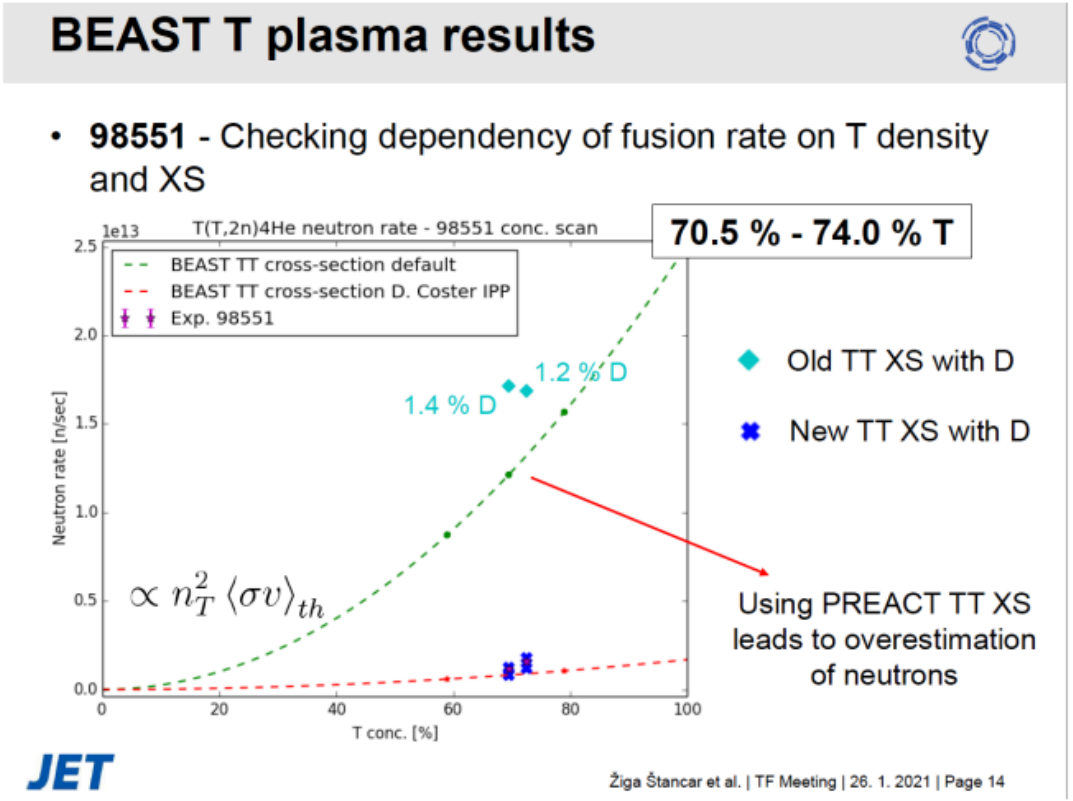


AMNS data (up to 6D) is stored in IMAS db and serve user requests through library calls. Used throughout several applications needing AMN(s) data. It is not clear what the plans are for the future maintenance of the AMNS library and the curation of future AMNS data.

Template for network data for ML applications?

AMNS data and interfaces in IMAS

Calculation of new T-T cross-section data tested at JET:



In previous work done as part of WPCD and JET activities, the existing nuclear data for T-T reactions was examined. Since the available data disagreed, a new parameterisation was prepared based on ENDF data.

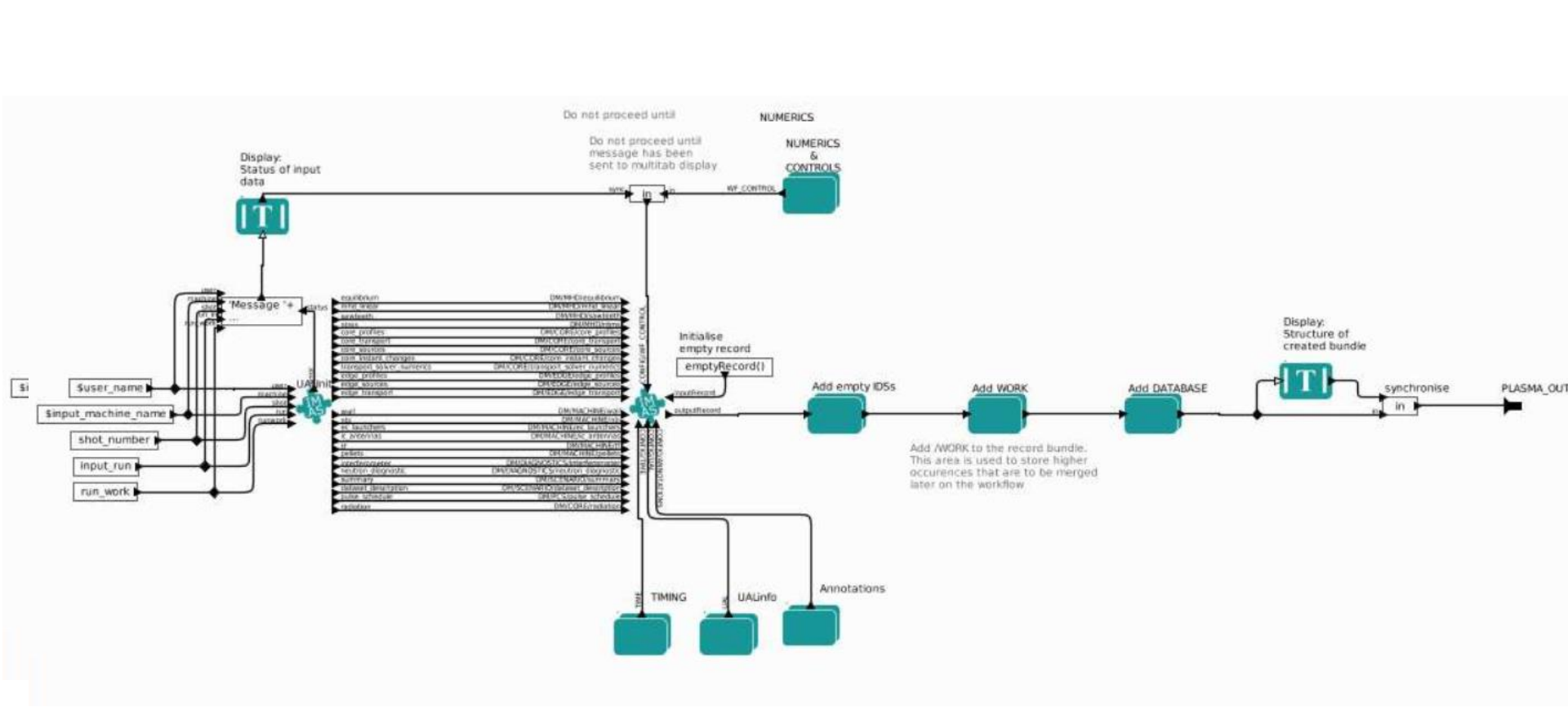
New results from JET suggest that this data is in better agreement with the experimental measurements than the previous data used in TRANSP.

Almost at the end…

Want to discuss?

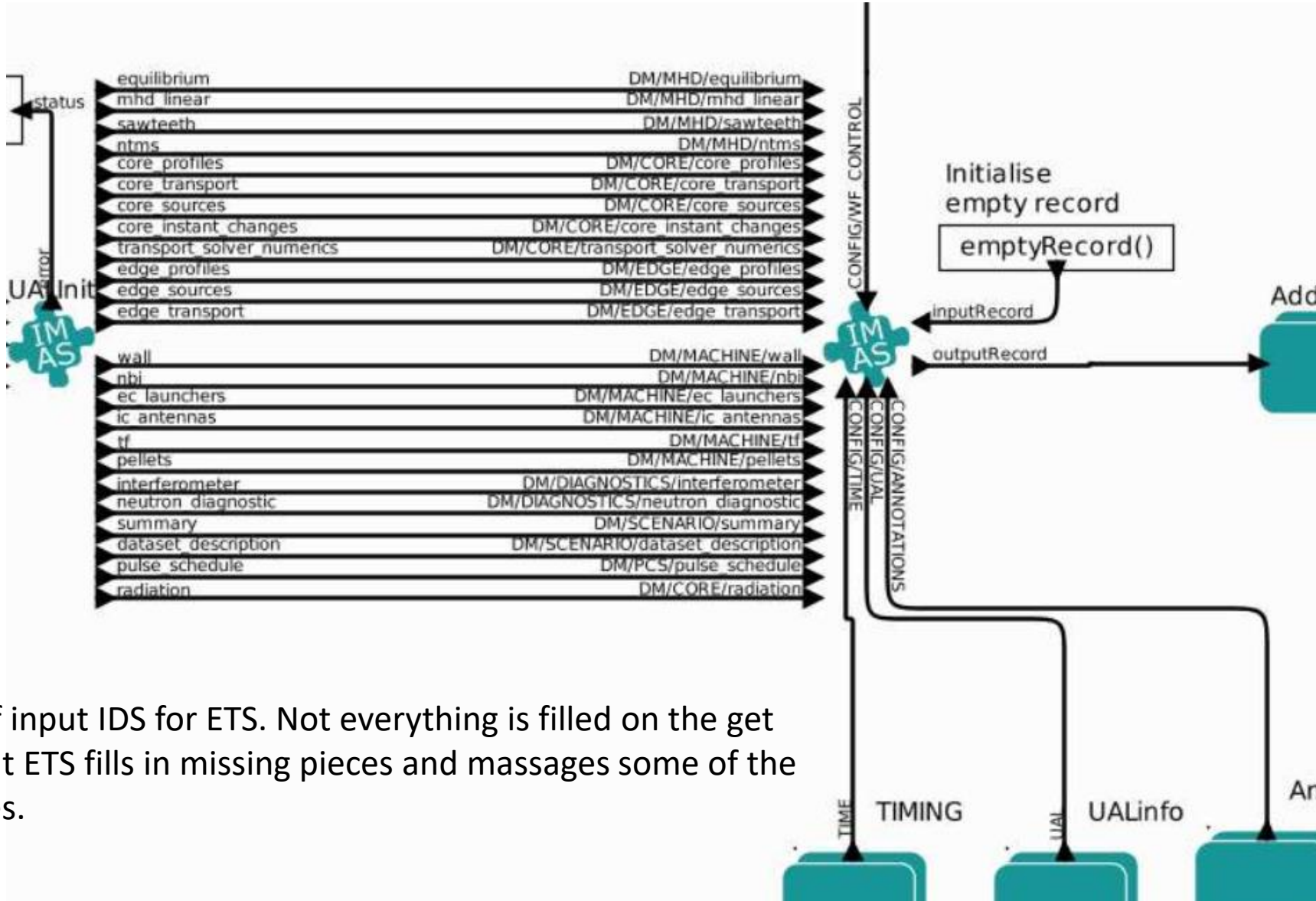
Par.strand@chalmers.se

Plasma bundle – creation in ETS



Plasma bundle is a record of the active IDS. It is not carrying the data but references to the relevant data entries. Data is imported (from memory cache or from Database) at the point of the launch of the actor.

Plasma bundle - input IDS

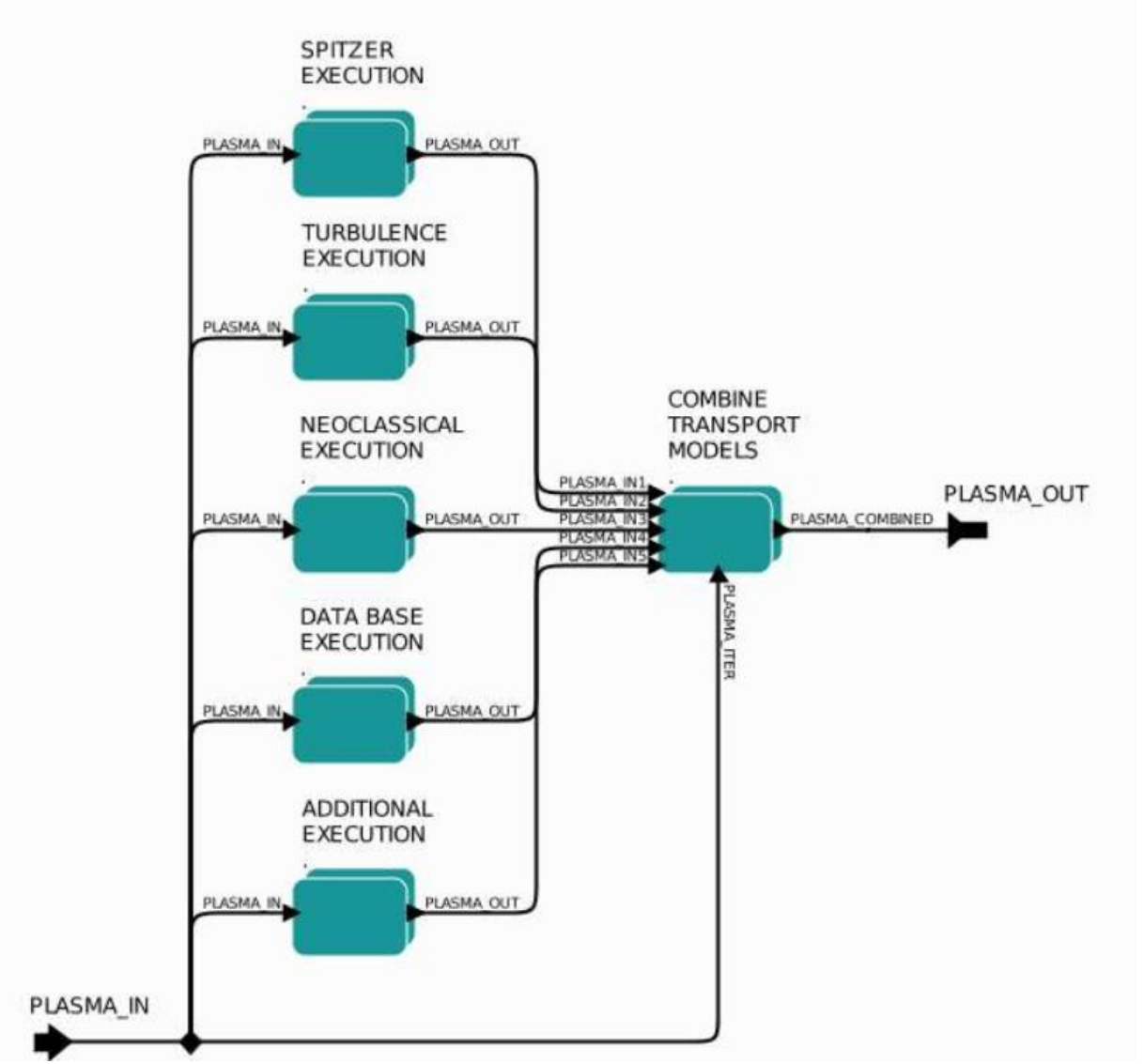


List of input IDS for ETS. Not everything is filled on the get Go but ETS fills in missing pieces and massages some of the entries.

Occurrences

Each of the different transport channels provides a plasma bundle with references to different transport IDS output with data in model(1)

The different occurrences are merged in transport combiner to a single version which has an array of models



Now really at the end….

Want to discuss?

Par.strand@chalmers.se